WebSphere TX Accelerator

The WebSphere TX Accelerator provides a full function connector to WebSphere TX (WTX) for DataStage that executes on top of the Java Integration stage. Throughout this document, this component shall be named the 'WTX Connector'. Highlights of the connector's functionality include:

- Works with WebSphere TX 8.2 onwards
- Supports any number of input links and output links which override cards in the WTX map
- Executes a map either per wave or per input row
- Support reject links when executing a map per row
- Supports delimited or fixed format WTX type trees with configurable syntax
- Supports passing an entire card object of any structural complexity via a single column
- Can map DataStage to WTX types either in binary or string forms.
- Supports importing of WTX type trees to DataStage table definitions
- Supports importing of DataStage table definitions to WTX type trees

Properties

The WTX Connector has the following properties. These can be set by either directly editing the "Custom properties" property in the Java Integration stage, or by pressing the 'Configure' button in the Java Integration stage editor. All of the properties are optional unless otherwise stated.

Stage properties:

Label	Property	Description
Map file	MapFile	The full path for the TX map file. This property is required.
Log directory	WorkDir	The path of the directory where log, trace and backup files will be created.
Resource file	ResourceFile	The path of the resource file.
Map trace	MapTrace	Turns map trace on of off. Valid values are 'On' and 'Off'. This value overrides whatever was defined in the compiled map.
Map audit	MapAudit	Turns map trace on of off. Valid values are 'On' and 'Off'. This value overrides whatever was defined in the compiled map.
Run each	RunEach	Specifies whether to run the map at the end of the wave, or to run the map for each input row. Valid values are 'Wave' and 'Row'.
Fail on warning	FailOnWarning	Fail the job if the map returns a warning.
Debug	Debug	If set to 'true' additional job log messages are produced.
Trace file	TraceFile	Specifies the name of a trace file to which trace information

		will be written. This should be disabled when in production since it can create large files.

Link properties:

Label	Property	Description
Card number	Card	The number of the card in the map (offset from 1). This is a required property.
Charset	Charset	The character set for string data. If not set the platform default is used.
Use strings	UseStrings	If 'true' all columns will be transferred as strings.
Delimiter	Delimiter	The column delimiter.
Delimiter location	Location	The location of the column delimiter. It must be one of 'Infix', 'Prefix' or 'Postfix'.
Terminator	Terminator	The row terminator.
Release character	ReleaseChar	The release character.
Include columns	Columns	For an input link, specifies which columns to pass to the input card of the map. Permitted values are 'All' and 'Selected'

The Java Integration stage properties must be set as follows:

Usage > Java > Classpath – This must include the WTXStage.jar file built from this WTX Sample. It must also include the dtxpi.jar from the WebSphere TX installation directory

Usage > User class – Must be set to com.ibm.is.cc.javastage.wtx.WTXStage

Usage > Use user-defined function – Must be set to No

Additionally the PATH or LIBPATH environment variable must include the WebSphere TX installation directory.

Compatible Type Trees

There are 2 ways to exchange data between WTX and DataStage:

- Map DataStage columns to WTX items
- Pass the entire card object as a single VarChar or VarBinary column (or similar types).

When mapping DataStage columns to WTX items, the WTX Connector only supports type trees that use a flat structure. WTX type trees can support much more complex structures than flat data, but cards with such complex structure cannot be overridden by DataStage link data.

It is common for a WTX group to contain sub-groups, which do not affect the data hierarchy. For example, a group 'FullName' with components 'First' and 'Last' may be included in the parent group. As long as the type syntax of these inner groups is consistent with that of the parent group then such trees can be supported.

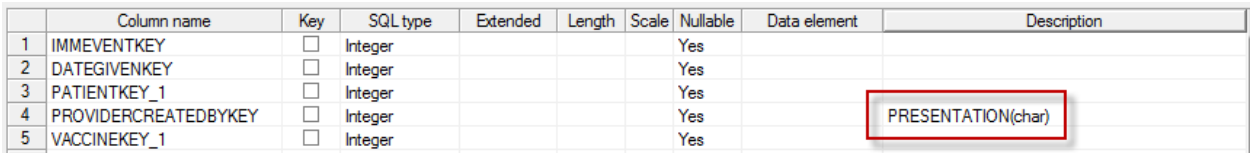
To use existing WTX maps with DataStage, it may be necessary to create a wrapper map that has a flat structure for input or output, and which maps from the flat structure to the more complex structure of the map to be invoked.

When transferring a single column value to the map, the specific column to be transferred can be selected by setting the link property 'Include columns' to 'Selected' and then specifying '[wtx-include]' in the description field of the column to be mapped. See section *Selecting which columns to map* for more details.

Data Types

Each input or output can be configured to pass data as either binary data or 'string' data. This is determined by the 'Use strings' property on the card, which by default is set to 'False', meaning that data will be transferred in a binary form.

The 'Use strings' property determines whether every column in the link is transferred as a binary or string value. To control this behavior on a column-by-column basis, the presentation can be specified by using specifiers "PRESENTATION(binary)" and "PRESENTATION(char)" in the description of the column. This string can be anywhere in the description text, so additional description can be specified. For example, if all integer columns were binary in the WTX type tree apart from PROVIDERCREATEDBYKEY, we would specify Use strings = False and then add PRESENTATION(char) to this one column:



The mapping between DataStage types and WTX types when 'Use strings' is false is as follows:

DataStage Type	WebSphere TX Type
BigInt	Number, Character, Integer
Binary	Text, Binary
Bit	Binary, Integer, Size=4
Char	Text, Character
Date	Date & Time, Character, Format = {CCYY-MM-DD}
Decimal	Number, Character, Decimal
Double	Binary, Float, Size=8
Float	Binary, Float, Size=4
Integer	Binary, Integer, Size=8
LongNVarChar	Text, Character
LongVarBinary	Text, Binary
LongVarChar	Text, Character
NChar	Text, Character
Real	Binary, Float, Size=4
SmallInt	Binary, Integer, Size=4
Time	Date & Time, Character, Format = {HH24:MM:SS[.0-6]}

Timestamp	Date & Time, Character, Format = {CCYY-MM-DD} {HH24:MM:SS[.0-6]}
TinyInt	Binary, Integer, Size=2
VarBinary	Text, Binary
VarChar	Text, Character

If 'Use strings' is set to true, then the type mappings are as follows:

DataStage Type	WebSphere TX Type
BigInt	Number, Character, Integer
Binary	Text, Character. Data is passed as hex-pairs.
Bit	Number, Character, Integer
Char	Text, Character
Date	Date & Time, Character, Format = {CCYY-MM-DD}
Decimal	Number, Character, Decimal
Double	Number, Character, Decimal
Float	Number, Character, Decimal
Integer	Number, Character, Integer
LongNVarChar	Text, Character
LongVarBinary	Text, Character. Data is passed as hex-pairs.
LongVarChar	Text, Character
NChar	Text, Character
Real	Number, Character, Decimal
SmallInt	Number, Character, Integer
Time	Date & Time, Character, Format = {HH24:MM:SS[.0-6]}
Timestamp	Date & Time, Character, Format = {CCYY-MM-DD} {HH24:MM:SS[.0-6]}
TinyInt	Number, Character, Integer
VarBinary	Text, Character. Data is passed as hex-pairs.
VarChar	Text, Character

Link/Card Options

A link to or from the WTX Connector corresponds to a card of the map that is executed by the stage. The properties for the link determine how the DataStage data is passed to or from the card. The properties should be configured in the following way:

- **Card number** - This specifies the card in the map which is being overridden by the link data. Any card which is not overridden will use the source/target defined in the map, which could be a file or any WTX adapter. Note that cards are numbered from 1.

- **Use strings** - determines how binary data types are passed to/from the map. The tables above detail the specific type mappings. Note that this setting applies to all types in the link/card. The value of this property can be overridden for a specific property by using PRESENTATION() in the description, as described above. Another approach if a particular column needs to be passed as a string value, whereas others do not, then use a Char or VarChar type for that column rather than a binary format (e.g. Integer, Double, etc).

- **Charset** - determines the character set to use when converting between Java String type and bytes sent to/from the map. These are IANA type names, the set of which is determined by each JVM. Consult the documentation for java.lang.String to see what is supported. Common values are "UTF-8", "UTF-16", "UTF-16BE" and "UTF-16LE". If Charset is not specified then the system default will be used. In many cases, the default is the most appropriate Charset to use. If the corresponding WTX type has the charset defined as 'Native' then do not specify Charset in the link properties.

- **Delimiter** and **Delimiter location** - these define the delimiter character or string to be used to delimit columns/group components. The location is one of either 'Prefix', 'Postfix' or 'Infix' and determines whether a delimiter precedes the first column (Prefix), follows the last column (Postfix) or lies only between columns (Infix). The delimiter and its location should correspond to the type syntax of the WTX group.

- **Terminator** - defines the group terminator, and should correspond to the terminator defined for the WTX Group.

- **Release character** - defines the group terminator, and should correspond to the terminator defined for the WTX Group.

Special literals can be specified for delimiter, terminator or release character:

Literal	Meaning
[NL]	Platform-specific newline
[NLW]	Windows newline (0x0D0A)
[NLU]	Unix newline (0x0A)
[LF]	Linefeed
[CR]	Carriage return
[HT]	Horizontal tab (tab)
[NULL]	Null
[SP]	Space
[WSP]	Whitespace

Selecting which columns to map

If there are columns in an input link which do not correspond to components of the type tree's group, they can be excluded from the data passed to the input card of the map. Columns which are not passed to the map can be included in an output link, and the value of the output column will be transferred from the input column. This provides a mechanism to pass additional columns through the stage.

This behavior is controlled by the 'Include columns' property of the input card. This has values 'All' or 'Selected'. It defaults to 'All'. There are 2 ways to select which columns to pass to the map:

1) Specify 'All' for 'Include columns', and then specify the string `[wtx-exclude]` in the description field of any columns which you do not wish to pass to the map.

2) Specify 'Selected' for 'Include columns', and then specify the string `[wtx-include]` in the description field of any columns which you do wish to pass to the map.

The former is most convenient when passing most columns to the map, and the latter is most convenient when passing a small number of columns to the map. In some cases, the entire card data is provided via a single column. In such a case, the latter method is the most convenient way to specify which column to provide to the map.

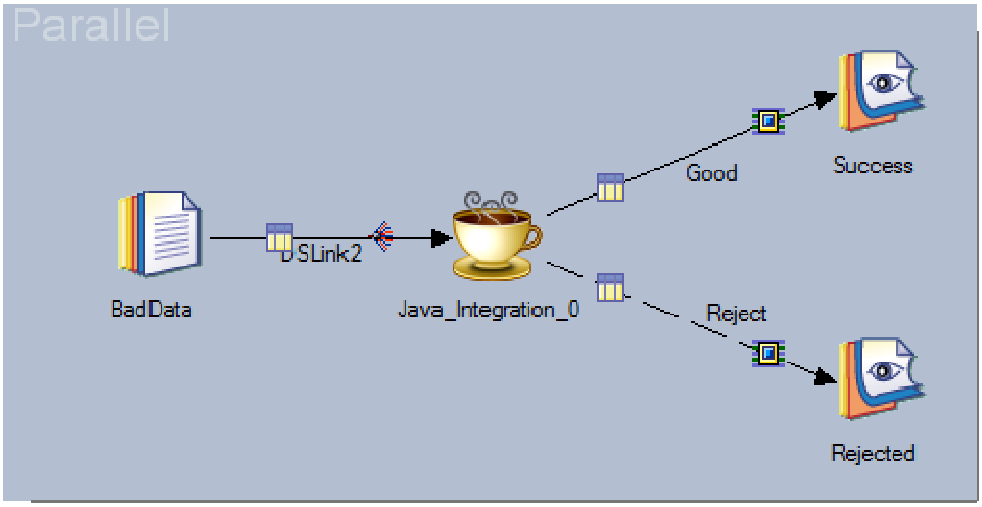
The strings `[wtx-include]` and `[wtx-include]` can appear anywhere in the description attribute.

If a column is excluded from the first input link, and a column with the same name appears in any of the output links, the column will be automatically excluded from the output link (will not be retrieved from the data returned from the output card), and the value of the column will be passed from input to output. If multiple input rows are consumed when running the map (i.e. the 'Run each' property is set to 'Wave'), then the value for the input column from the last row will be transferred to the output link.

Run modes

The WTX map can be run either for each wave or for each row of data on an input link. The behavior is controlled via the 'Run each' property which can be set to either 'Wave' (the default) or 'Row'. Run each row can only be used for jobs that have a single input link. When set to run each wave, the WTX Connector will append data from each incoming row, appending the terminator for the input link after each row. Once it has built data for all input links, it will then run the WTX map.

When running a map for each row, reject links may be used to reject input data to a reject link. The reject link can carry the map status code and map error message.



Tracing and Debugging

The most difficult aspect of using the WTX Connector is ensuring that the definitions of the link correspond to those of the WTX type tree used in the cards. The data types have to align (as per the tables given earlier in this document), and delimiter, terminator and release characters have to be correspond. When they are not matched correctly, the map will either fail to parse inputs, issue warnings on input parsing, or the WTX Connector will not be able to parse the output data. Trace files should be used to determine the cause of such problems.

Trace and log files are produced both by the WTX map and by the WTX Connector. The map audit and trace set up via the map settings apply when running the map via the WTX Connector. The location of these files shall be the directory specified via the 'Log directory' property. The trace file will be named "<mapname>.mtr" and the audit log "<mapname>.log"

Stage properties 'Debug' and 'Trace file' can be used to provide additional information from the WTX Connector. Enabling the Debug property provides additional log messages in the job log. Specifying a 'Trace file' produces a file that produces information similar to the WTX map trace file. It shows the objects that are being built for inputs or parsed in outputs:

```
Processing input link 1 (card 2)

Processing row 0.
Column 0 emp_fname:
   Value=Work A.
Column 1 emp_lname:
   Value=Holic
Input link 1 (card 2) produced 42 bytes.

Processing output link 0 (card 3)
Contains 47 bytes.

Processing row 0.
Column 0 employee_emp_fname:
   Offset 0: Looking for delimiter '|'.
   Offset 20: Delimiter found.
   Value=Work A.
Column 1 employee_emp_lname:
   Offset 21: Looking for delimiter '|'.
   Offset 41: Delimiter found.
   Value=Holic
Column 2 total_hours:
   Offset 42: Looking for delimiter '
'.
   Delimiter not found.
   Value=90.45
```

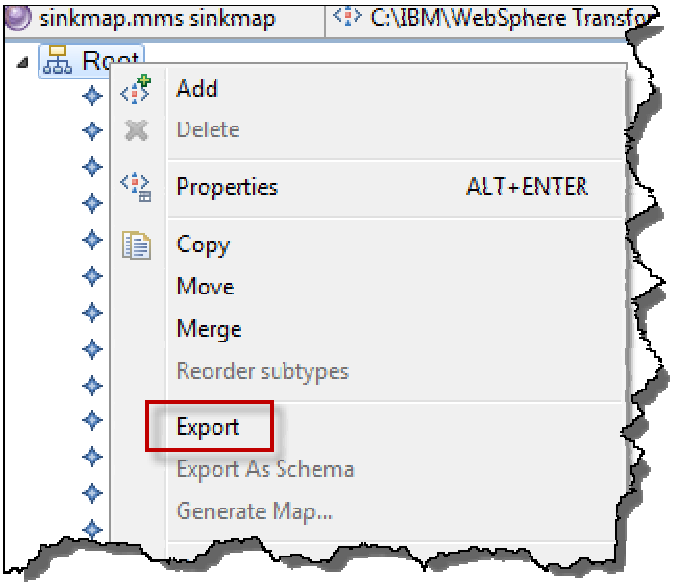
By using both the map trace files together with the Connector's trace files, you will be able to determine mismatches between type trees and link definitions.

Importing type trees into DataStage

If you have an existing WTX map that you wish to invoke from the WTX Connector, then the type trees for the cards of the map should be imported into DataStage table definitions.

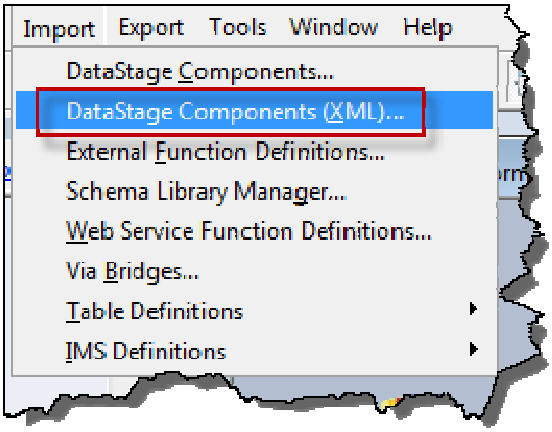
The steps to import type trees into DataStage are:

1) From within WTX Design Studio, export the type tree to an MTS file



2) Run the TDBuilder tool (part of this accelerator code) to create an XML representation of a DataStage table definition from the MTS file

3) From DataStage Designer, import the XML file to create the table definition.



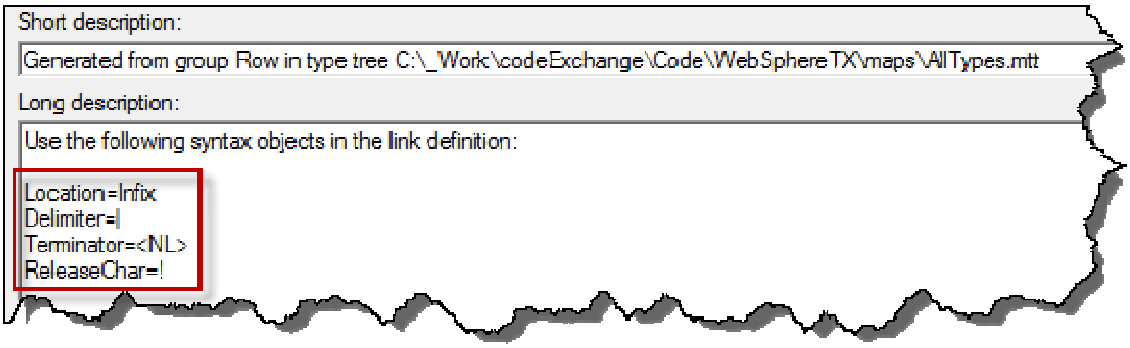
The syntax for TDBuilder is:

```
TDBuilder <mts-file> <xml-file> <tabledef-name> <group-name> [<group-path>]
```

The arguments are:

- **mts-file** - the full or relative path to the input MTS file
- **xml-file** - the full or relative path to the file that will be created by TDBuilder
- **tabledef-name** - the name of the table definition. The table definition will be created in the 'Table Definitions' folder in the repository tree.
- **group-name** - the name of the group that represents a 'row' in the type tree.
- **group-path** - if the group-name is not unique within the type tree, the required group can be identified by specifying the path. The value should be specified in double quotes since it will typically contain spaced.

The long description property of the generated table definition contains the syntax objects that should be used in the link definition where this type tree is used. These properties can be cut from here and pasted into the link properties:



If the type tree contains decimal columns, these will be represented as Decimal types in the link, but their scale and precision will not be set automatically. Before running a job that uses these table definitions be sure to specify scale and precision to appropriate values. Failure to do so will result in runtime errors.

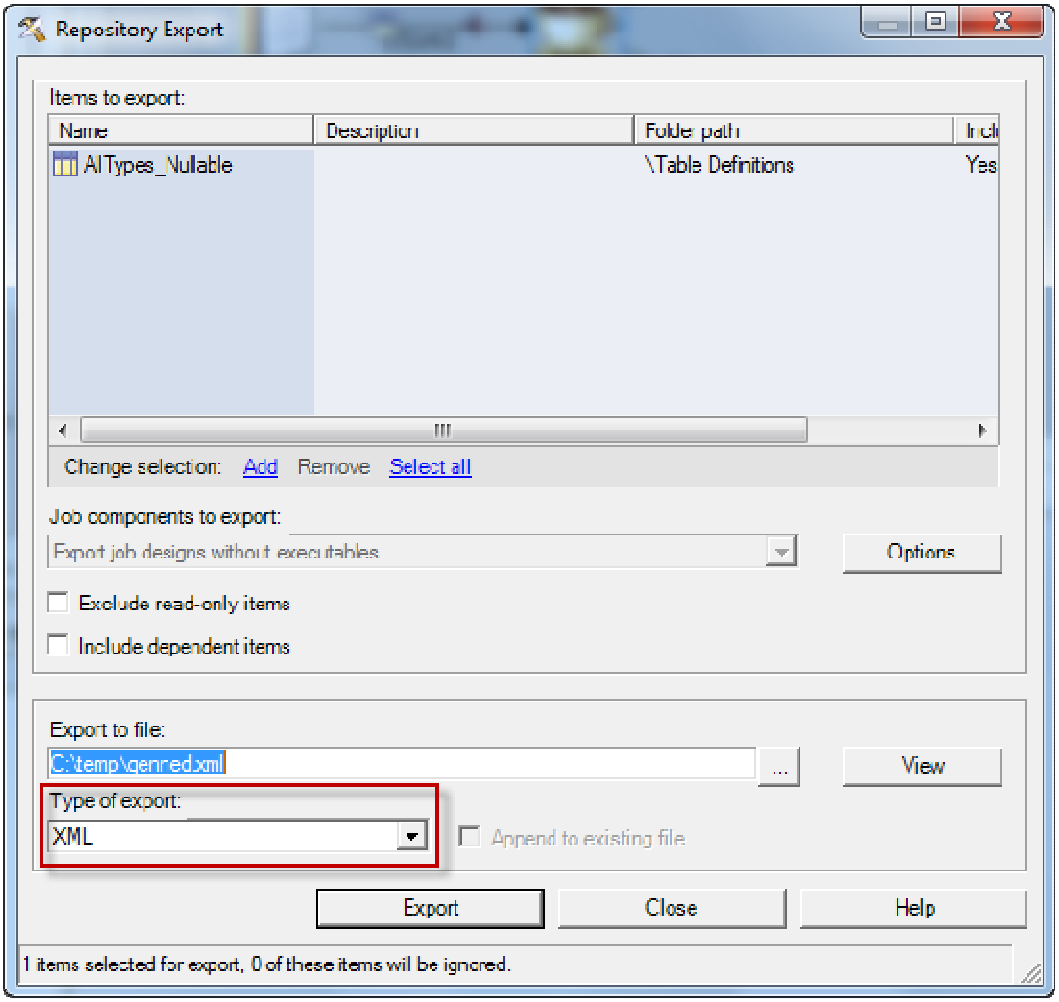
Note: There is a bug in Design Studio as of version 8.4 regarding 8 byte integers. Trying to export a type tree that contains an 8 byte binary integer will fail with *'Export failed'*. To workaround this problem, temporarily change the size to 4 bytes and then modify the generated table definition back to an Integer type.

Importing table definitions into WebSphere TX

If you have an existing DataStage job, and want to create a WTX map to perform some processing, then type trees corresponding to the link definitions should be created.

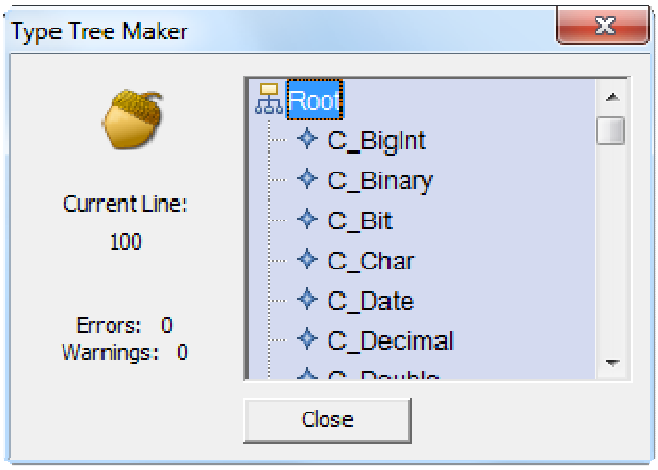
The steps to import DataStage table definitions are:

1) Export the table definition as XML



2) Run the MTSBuilder tool (part of this accelerator code) to create an MTS representation of a WTX type tree from the XML file

3) Run Type Tree Maker to create an MTT file from the MTS file

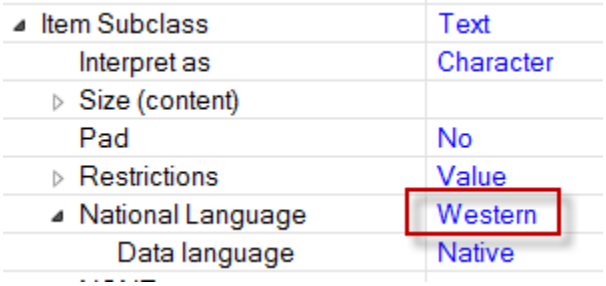


The syntax for MTSBuilder is:

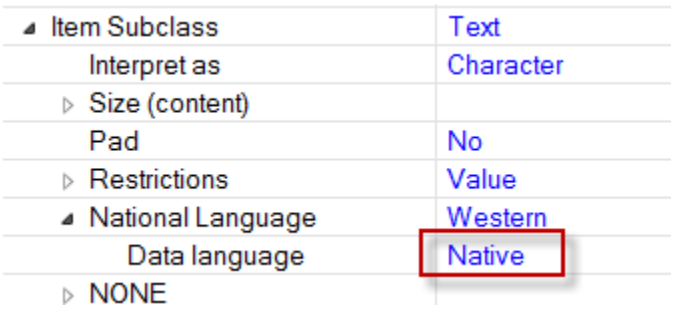
```
MTSBuilder <xml-export-file> <mts-file> <mtt-file> [-L <language>]
           [-C <charset>] [-S]
```

The arguments are:

- **xml-export-file** - the full or relative path to the input XML file
- **mts-file** - the full or relative path to the MTS file that will be created by this command
- **mtt-file** - the full path to the name of the MTT file that will be created when Type Tree Maker is run to process the MTS file.
- **language** - the value to be used for the National Language property of text items. If not specified it defaults to Western



- **charset** - the value to be used for the Data Language property of text items. If not specified it defaults to Native



- -S - if this option is specified all of the generated items will be textual representations of the type (i.e. the item property 'Interpret as' will be Character rather than Binary). By default MTSBuilder will create binary types for binary, integers, floating decimals.

The generated tree will include a group named "Document" which contains some number of "Row" groups. The Document group should be the type selected in the card definition. The Row group is created with the following syntax:

- Infix delimited
- Delimiter = |
- Terminator = <NL>
- Release character = !

This can be manually changed in the WTX Design Studio if so desired.

Note: There is a bug in Type Tree Maker as of version 8.4 regarding 8 byte integers. Type Tree Maker will report an error "*Invalid syntax in command line*." when it encounters the 8 byte integer. To workaround this problem, edit the MTS file to change the length to 4 bytes. After importing, edit the generated type tree to change the size back to 8 bytes.

Performance Tips

The following tips can be used to optimize performance. To get the best performance:

- turn off tracing in the WTX map
- turn off the 'Trace file' option in the WTX Connector
- use binary representations for binary, integer and floating decimals wherever possible
- pick delimiter and terminator characters that are not likely to be present in the data, so that you do not need to define a release character. If a release character is specified for input and outputs the data will be traversed to look for delimiter characters. If no release character is specified this extra processing will be skipped.

The WebSphere TX Accelerator Code

The code provided for the runtime functionality is comprised of 6 classes in the com.ibm.is.cc.javastage.wtx package. These are:

WTXStage - the main processing code for the WTX Connector

WTXMapRunner - wraps the WTX API to provide methods to set up and invoke a WTX map

WTXMapExecutionResults - a container for the results of running a job

WTXCard - a container for card/link properties

WTXInputCard - extends WTXCard to ad input card/link specific properties

WTXOutputCard - extends WTXCard to ad output card/link specific properties

Some notes on the **WTXStage** class:

- _userPropertyDefinitions provides the definitions of the user properties that are returned by method getUserPropertyDefinitions
- getCapabilities() specifies that the stage supports a single input or output link
- validateConfiguration() is called both at design time and at runtime to validate the links and properties. This method sets up the class member variables according to the user property values.
- initialize() loads the map and overrides the inputs and outputs
- terminate() unloads and closes the map
- process() is the entry point for the main processing logic. It calls processInputLink() on each input link to build the data to pass to the WTX map. It then executes the map, and parses the data streams returned to the output links.

The code provided for the design time functionality is comprised of 6 classes in the com.ibm.is.cc.javastage.wtxmeta package. These are:

- **MTSBuilder -** the MTSBuilder tool. This incorporates a parser to parse the DataStage XML.
- **TDBuilder** - the TDBuilder tool.
- **MTSParser** - used by TDBuilder, this class parses MTS files and creates WTXGroup and WTXItem objects.
- **WTXGroup** - represents a TX group

- **WTXItem** - represents a TX item
- **ColumnDef** - represents a DataStage column. Both MTSBuilder and TDBuilder create instances of this

Directory Structure

The directory structure of the sample contains the following subdirectories and files:

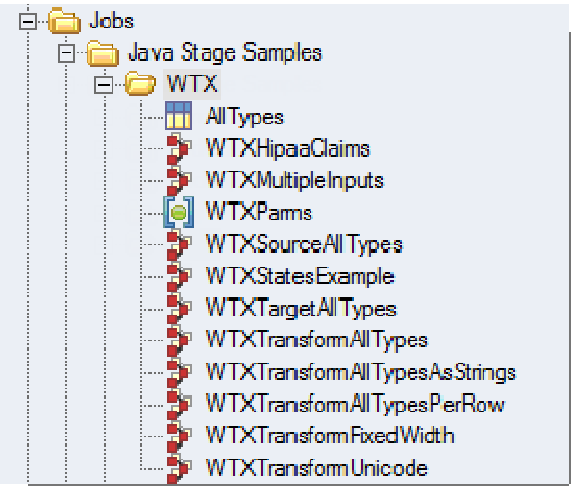
- src – the Java source code
- jobs – sample DataStage jobs
- maps - sample WTX maps
- doc - contains this document
- build.xml - ant build script
- MTSBuilder.bat - command file to run MTSBuilder class
- TDBuilder.bat - command file to run TDBuilder class

Building the Sample

The root directory of the sample provides an ant script called build.xml. To build the sample enter the command 'ant build' in the root directory. This will create a jar file in the jars subdirectory called WTXStage.jar. The path to the jar file must be specified in the Usage > Java > Classpath property. The ant file contains a property called 'wtx.dir' that should be modified to match your environment.

Sample Jobs

The jobs directory contains a number of sample jobs that demonstrate the functionality of the WTX connector. The jobs provided are these:



The default value of job parameters in the WTXParms parameter set should be modified to match your environment.

The purpose of these jobs and the map they run is:

Job	Map	Description
WTXHipaaClaims	hipaa_837p_5010_flat	Demonstrates using WTX to parse and extract claims data from a HIPAA document.
WTXMultipleInputs	sinkmap	This job uses the sinkmap example (see examples/general/map/sinkmap in the WTX directory) to demonstrate providing multiple inputs to a map from multiple input links.
WTXStatesExample	master	This job uses the master example (see examples/general/states in the WTX directory)
WTXSourceAllTypes	MapAllTypes	Uses the WTX Connector as a source of data.
WTXTargetAllTypes	MapAllTypes	Uses the WTX Connector as a target.
WTXTransformAllTypes	MapAllTypes	Uses the WTX Connector to transform data. This example uses all possible column data types.
WTXTransformAllTypesAsStrings	MapAllTypesAsStrings	Demonstrates transferring data between DataStage and WTX as string types.

WTXTransformAllTypesPerRow	MapBadData	Demonstrates invoking a map for each row of data. The data contains rows that have invalid dates causing the WTX map for those rows to fail, and for the bad data to be sent to the reject link.
WTXTransformFixedWidth	MapFixedWidth	Demonstrates using a fixed-width group, rather than delimited.
WTXTransformUnicode	MapUnicode	Demonstrates passing UTF-16 data by specifying the Charset property on the links.