



# M3 Enterprise Collaborator Partner Admin Tool User Guide

Version 11.4.2.0

Published May 6, 2014

**Copyright © 2014 Infor. All rights reserved.**

### **Important Notices**

The material contained in this publication (including any supplementary information) constitutes and contains confidential and proprietary information of Infor.

By gaining access to the attached, you acknowledge and agree that the material (including any modification, translation or adaptation of the material) and all copyright, trade secrets and all other right, title and interest therein, are the sole property of Infor and that you shall not gain right, title or interest in the material (including any modification, translation or adaptation of the material) by virtue of your review thereof other than the non-exclusive right to use the material solely in connection with and the furtherance of your license and use of software made available to your company from Infor pursuant to a separate agreement, the terms of which separate agreement shall govern your use of this material and all supplemental related materials ("Purpose").

In addition, by accessing the enclosed material, you acknowledge and agree that you are required to maintain such material in strict confidence and that your use of such material is limited to the Purpose described above. Although Infor has taken due care to ensure that the material included in this publication is accurate and complete, Infor cannot warrant that the information contained in this publication is complete, does not contain typographical or other errors, or will meet your specific requirements. As such, Infor does not assume and hereby disclaims all liability, consequential or otherwise, for any loss or damage to any person or entity which is caused by or relates to errors or omissions in this publication (including any supplementary information), whether such errors or omissions result from negligence, accident or any other cause.

Without limitation, U.S. export control laws and other applicable export and import laws govern your use of this material and you will neither export or re-export, directly or indirectly, this material nor any related materials or supplemental information in violation of such laws, or use such materials for any purpose prohibited by such laws.

### **Trademark Acknowledgements**

The word and design marks set forth herein are trademarks and/or registered trademarks of Infor and/or related affiliates and subsidiaries. All rights reserved. All other company, product, trade or service names referenced may be registered trademarks or trademarks of their respective owners.

### **Publication Information**

Release: 11.4.2.0

Publication date: May 6, 2014

Document Number: MECPATUG\_11.4.2.0\_W\_01

---

## Version Log

The version log describes the changes between versions of this document.

Part Number	Release Date	Description
MECPATUG-1142UWA	201405	Republish. Updated the header record format in <a href="#">FTP Scripts overview</a>
MECPATUG-1142UWA	201312	Updated for version 11.4.2.0 Added new topics: Deleting Agreement or Agreement Group <a href="#">Deleting Agreements or Agreement group</a> Schema Location <a href="#">Schema Location</a> Error Suppression <a href="#">Error Suppression Overview</a> , <a href="#">Error Suppression tasks</a> Updated Web Service Definitions <a href="#">Web Service Overview</a> Reorganized topics on Process Steps <a href="#">Process Steps</a>
MECPATUG-1141UWA	201308	Updated for version 11.4.1.0 Added the following: <a href="#">Importing Agreements</a> , <a href="#">Exporting Agreements</a> , <a href="#">Company/ Division Override</a> , <a href="#">Import/Export Agreement Override Concepts</a> Added CONO/DIVI <a href="#">API Fields Definition</a>
MECPATUG-114UWA	201305	Updated to version 11.4.0.0
MECPATUG-104UWA	201305	Updated to version 10.4.0.0
MECPATUG-103UWA	201208	Updated for version 10.3.0.0 Added new topics: MECDataTranslator <a href="#">Data Translator Overview</a> , <a href="#">Using MEC Data Translator</a> DB References <a href="#">Database Reference Overview</a> , <a href="#">Setting Transaction Isolation Level</a> DAF updates <a href="#">MIME Type</a> , <a href="#">Option to Remove Document/Image File</a> , <a href="#">Option to Archive self(.rcv)</a> , API <a href="#">API Clean Up and Validation</a> Channel and Service Scheduling <a href="#">Channel and Service Scheduling Overview</a>

Part Number	Release Date	Description
		<p>Ordered Page, Variation ID, and MBM Identifier</p> <p>Run On Host, added a note where Run On Host field is applicable.</p>
MECPATUG-92UWA	201201	<p>Updated for version 9.2.0.0</p> <p>Added new and updated existing topics based on the following:</p> <p>DAF Archive:</p> <p>ADC-DAF: <a href="#">DAF Overview</a>, <a href="#">DAF Fields Definition</a>, <a href="#">Creating DAF Connection and Basic Data</a>, <a href="#">Creating DAF Archive Process</a></p> <p>Web Service Channel: <a href="#">Web Service Overview</a>, <a href="#">Web Service Definitions Fields</a>, <a href="#">Uploading and viewing a WSDL file</a>, <a href="#">Modifying Web Service Definitions</a>, <a href="#">Exporting Web Service Definitions</a>, and <a href="#">WebServicesSyncIn Protocol Type</a></p> <p>Moved Schedules topic to MEC Admin Guide</p> <p>ION Channels: <a href="#">IONDbIn</a>, <a href="#">IONDBOut</a></p> <p>BOD: , <a href="#">Polling Type Channel Properties</a> in IONDBIn</p>
MECPATUG-91UWA	201105	<p>Updated for version 9.1.4.0</p> <p>Updated version references, terminologies, and procedures.</p> <p>Added the following new topics:</p> <p><a href="#">Split Redetect</a>, <a href="#">Advanced PA Tasks</a> , , <a href="#">FTP Scripts overview</a>, <a href="#">FTP Script Fields Definition</a>, <a href="#">Creating FTP Script</a>, <a href="#">Deleting and Editing FTP Script</a>, <a href="#">Viewing and Exporting FTP Script</a>, <a href="#">MvxNGInRouter</a>, <a href="#">HTTPSyncIn</a> , , <a href="#">HTTPSISIn</a> , , <a href="#">FTPScriptPollIn</a>, <a href="#">Empty Namespace</a>, and <a href="#">Namespace in output file</a></p> <p><a href="#">Deleting and Editing Subscriptions</a>, <a href="#">Adding a Subscription</a>, <a href="#">EventHub Overview</a>, <a href="#">FTP Script</a>, and <a href="#">Message Processes</a></p> <p>Updated the following topics:</p> <p><a href="#">Communication Channel Types</a>, <a href="#">Polling Type Channel Properties</a>, <a href="#">Server Type Channel Properties</a>, and <a href="#">Send Channel Properties</a></p>
MECPATUG-91UWA	201005	<p>Updated for version 9.1.3.1</p> <p>Updated with a note in every affected polling channel</p> <p>Added <a href="#">MvxNGInRouter</a></p> <p>Added channel user access rights.</p>

---

Part Number	Release Date	Description
MECPATUG-9130UWA	200910	Added channel notes in:
		Added new chapter: <a href="#">Advanced PA Tasks</a>
		Integrated PATUG and ComPlugInUG
MECPATUG-9130UWA	200905	Updated Properties for MSMQIn and MSMQ Protocol
		Updated Namespaces, and Properties for FTPPollin2

---

# Contents

<b>Chapter 1: Partner Admin Tool Overview.....</b>	<b>15</b>
About this Guide.....	15
Partner Admin Tool Overview.....	16
Partner Admin Tool Information Model.....	17
Uses of Partner Admin Tool.....	18
 <b>Chapter 2: Getting Started with Partner Admin Tool.....</b>	 <b>19</b>
Starting Partner Admin Tool.....	19
Configuring the Database Settings.....	19
Database Fields Definition.....	20
Setting up a database connection.....	21
Deleting a Database Connection.....	22
Priority Settings in MEC.....	22
 <b>Chapter 3: Communication Channels.....</b>	 <b>24</b>
Communication Channel Overview.....	24
Communication Channel Types.....	25
Setting Up Receive Channels .....	25
Polling Type Channel Properties.....	26
Server Type Channel Properties.....	35
WebServicesSyncIn Protocol Type .....	41
Adding Receive Channel.....	44
Viewing and Editing Receive Channel.....	45

---

Deleting a Receive Channel.....	45
Setting Up Send Channels.....	46
Send Channel Properties.....	46
Creating New Send Protocol .....	52
Creating New Send Channel.....	52
Managing Send Channels.....	53
Modifying Filenaming Settings.....	54
File Naming Default Classes.....	54
Filenaming User Defined Classes.....	56
M3 API.....	56
M3 API Overview.....	57
API Clean Up and Validation.....	57
API Fields Definition.....	57
Adding API Reference.....	59
Managing API Reference.....	60
Company/ Division Override.....	60
FTP Script.....	61
FTP Scripts overview.....	61
FTP Script Fields Definition.....	62
Creating FTP Script.....	63
Deleting and Editing FTP Script.....	63
Viewing and Exporting FTP Script.....	63
Web Services.....	64
Web Service Overview.....	64
Web Service Definitions Fields.....	65
Uploading and viewing a WSDL file.....	66

Modifying Web Service Definitions.....	67
Exporting Web Service Definitions.....	68
DAF Archiving.....	69
DAF Overview.....	69
DAF Fields Definition.....	70
MIME Type.....	71
Creating DAF Connection and Basic Data.....	71
Databases.....	73
Database Reference Overview.....	73
Setting Transaction Isolation Level.....	75
MECEventHub.....	75
EventHub Overview.....	76
EventData Messages Overview.....	76
Adding a Subscription.....	78
Deleting and Editing Subscriptions.....	78
MEC Schedules.....	79
Channel and Service Scheduling Overview.....	79
Data Translator.....	81
Data Translator Overview.....	81
Using MEC Data Translator.....	83
Creating Applications Object.....	85
Creating Partner Objects.....	86
Creating Translation Data Object.....	87
<b>Chapter 4: Message Detection.....</b>	<b>90</b>
Detections Overview.....	90



---

Detections.....	91
Uses of Detections.....	91
Adding a new Detection.....	92
Executing XML Detection Using the MBM Target Group.....	92
Executing XML Detection Using the MVX Target Group.....	92
Executing Flat Detection Using Target Groups.....	94
Target and Target Groups Overview.....	94
XML Targets.....	95
Creating XML Targets.....	95
Managing XML Targets.....	95
Exporting XML Targets.....	97
Viewing XML Tree.....	98
Importing XML Targets.....	99
XML Target Group.....	99
Creating XML Target Group.....	99
Managing XML Target Group.....	100
Flat File Targets.....	100
Creating Flat File Targets.....	101
Viewing Flat File Targets.....	101
Managing Flat File Targets.....	102
Flat File Target Group.....	102
Creating Flat Target Group.....	103
Managing Flat Target Group.....	103
Moving Targets Between Areas.....	104
Performing Sanity Check on Target Groups.....	104
Target Groups and Shadowing.....	104

Shadowing Overview.....	105
Shadowing in Flat File Targets.....	106
Shadowing and Channel Detection.....	107
<b>Chapter 5: Message Processes.....</b>	<b>108</b>
XML Processing.....	108
Creating new XSLT Definitions.....	108
Managing XSLT Definitions.....	109
Exporting XSLT Definitions.....	109
Unpublishing XML Mappings.....	110
Apply Envelopes.....	111
Envelope Objects Overview.....	111
Importing a New Envelope Template.....	112
Managing Existing Envelope.....	112
Exporting an Envelope Template.....	113
Managing Flat File Definitions.....	114
Creating DAF Archive Process.....	114
Option to Remove Document/Image File.....	115
Option to Archive self(.rcv).....	116
Modify Flat Record.....	117
Modify Flat Records.....	117
Property Records.....	118
Strip / Replacement.....	119
String Literals.....	120
Examples.....	120
Split Redetect.....	123

---

File Splitting Overview.....	123
SplitRedetect Parallel Run.....	124
Batch File Splitter.....	125
Large File Splitter.....	125
File Splitter Types and Properties.....	127
Batch File Splitter properties definition.....	127
Large File Splitter properties definition.....	128
Splitting Batch Files.....	129
Splitting Large Files.....	130
Splitting on N:th(property) Occurrence.....	131
Send Webservice.....	132
Using Send Web Service in Agreement.....	133
<b>Chapter 6: Partner Agreement.....</b>	<b>134</b>
Partner Agreement Overview.....	134
Creating Partner Agreement Folder Structure.....	134
Creating New Agreement .....	135
Setting up a process flow.....	136
Viewing Agreements.....	136
Import/Export Agreement Override Concepts.....	137
Importing Agreements.....	141
Exporting Agreements.....	142
Deleting Agreements or Agreement group.....	142
<b>Chapter 7: Process Steps.....</b>	<b>144</b>
Regular Process Steps.....	145

---

Archive.....	146
DAF Archive.....	146
Check Order.....	147
Transform Process Steps.....	148
Modify Flat Records.....	149
Re-Detect.....	150
Split Redetect.....	150
Reroute.....	150
Send Web Service .....	150
SNA Send.....	151
Validate.....	152
Error Handling Process Steps.....	152
Create ConfirmBOD.....	152
Retrieve MBM Identifier.....	153
Process Steps Applicable to both Regular and Error Handling.....	154
Apply Envelope.....	155
Outbound MBM Status.....	155
SEND.....	156
XML Transform.....	156
XSL Transform.....	157
<b>Chapter 8: Error Handling.....</b>	<b>158</b>
Process Flow Error Handling.....	158
Error Handling.....	158
Process Levels.....	158
Handling Error Mails.....	159

---

Error Message in a single email.....	159
Error Suppression Overview.....	160
Error Suppression tasks.....	162
Handling Receive Channel Errors.....	163
Handling Receive Channel Errors.....	164
 <b>Chapter 9: Advanced PA Tasks .....</b>	<b>165</b>
Advanced PA tasks.....	165
Defining File Name Extensions.....	165
Managing File Name Extensions.....	165
Defining Character Encoding.....	166
Managing Character Encoding.....	166
Defining Populating Targets.....	167
Namespaces.....	168
Namespace Overview.....	168
Namespaces in XML Detection.....	169
Namespaces in XML Transform Process.....	172
Common Upgrade Scenarios Affecting Namespaces.....	173
Schema Location.....	175
 <b>Chapter 10: Writing a Custom Channel.....</b>	<b>178</b>
Writing a Custom Receive Channel.....	178
Custom Receive Communication Channel Overview .....	178
Creating an Incoming Communication Channel.....	179
Server-Side Implementation for Incoming Communication.....	180
Adding Receive Protocol.....	182

Configuring New Incoming Communication Channel.....	183
Writing a Custom Send Channel.....	183
Custom Send Communication Channel Overview.....	183
Creating an Outgoing Communication Channel.....	184
Server-Side Implementation for Outgoing Communication.....	184
Configuring New Outgoing Communication Channel.....	186
Configuring MEC with HTTPS Communication and Similar Protocols.....	187
<b>Appendix A: Sample Incoming Channel.....</b>	<b>188</b>
Sample Code - SampleIncommingChannel.package .....	188
<b>Appendix B: Sample Outgoing Channel.....</b>	<b>192</b>
Sample Code for HTTP Server Configuration File.....	192
Sample Code for HTTPSOut Class.....	193
Sample Code for HTTPSOutPanel.....	196
<b>Appendix C: Create ConfirmBOD process specifics.....</b>	<b>201</b>
ConfirmBOD.....	201

This chapter provides an overview of the functions and usage of Partner Administration Tool.

- ["About this Guide" on page 15](#)
- ["Partner Admin Tool Overview" on page 16](#)
- ["Partner Admin Tool Information Model" on page 17](#)
- ["Uses of Partner Admin Tool" on page 18](#)

## About this Guide

This user guide describes how to use and manage the Partner Administration (PA) client tool and gives you a general understanding on the relationship between a partner agreement information model and the PA client tool.

For more information about MEC, see the *M3 Enterprise Collaborator Server and Client Tools Installation Guide*.

## Users of this guide

This user guide is intended for use by Partner Admin tool business consultants, application engineers, and Java developers to help them define collaboration and agreements with business partners, customers, suppliers, banks, and others. This user guide can also help them define internal business applications like Product Lifecycle Management (PLM) system. The collaboration and agreements can be available for use through MEC.

This user guide is not intended to be used as a complete self-study guide on how to learn to use the tool and all its concepts. It is expected of the reader to have sufficient back-ground knowledge acquired elsewhere.

## Knowledge prerequisites

To use this product, you should be knowledgeable and highly experienced in the following areas:

- Strong understanding of ION Grid, MEC and its prerequisites.

- Communication protocols and regular expression (regex) settings.
- Programming concepts, such as functions and input/output parameters, loops, and execution flow control.
- XML concepts such as elements, attributes, namespaces, schemas, XSL, and XPath.
- Java programming for the customer extensions, such as channel development, fine naming, and detections.

## Where to Go for Additional Information

For information on:	See:
Installing MEC server and client tools	<i>M3 Enterprise Collaborator Server and Client Tools Installation Guide.</i>
Infor applications	Infor product documentation on Infor Xtreme and online Help.
Monitoring agreements and message status	<i>M3 Enterprise Collaborator Administration Guide.</i>
Using your third-party consumer product	The corresponding vendor documentation for your third party product.

## Partner Admin Tool Overview

The Partner Administrator (PA) tool is a component within the M3 Enterprise Collaborator (MEC) product. PA tool enables the user to define and manage the partner agreement information needed by MEC to send and receive business messages between you and your partners. The data used for the process are stored in a MEC database. PA provides a core of services for controlling different specific tasks to support partner administration.

Here is a list of tasks in Partner Admin tool:

- Server Communication channels
- Detections
- Envelopes
- Flat Definitions
- FTP Scripts
- Web Service Definitions
- EventHub Subscriptions
- XML Mappings



- XSLT Definitions
- Data Translation
- DAF Basic Data
- Advanced PA tasks
- Error Suppression

## Partner Admin Tool Information Model

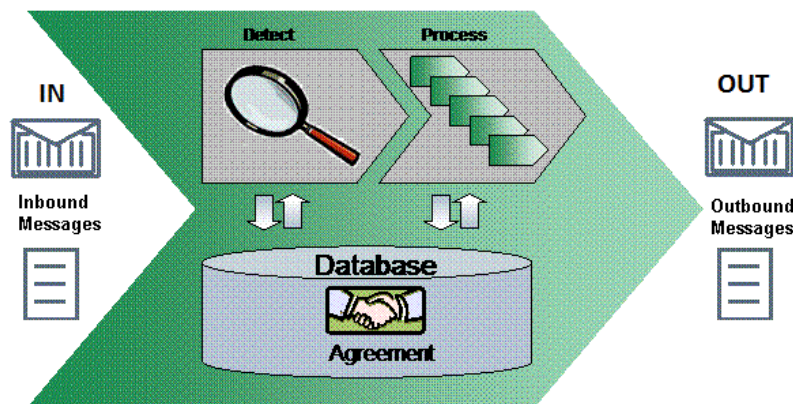
The information defined in PA tool is called a partner agreement. MEC uses these information to send and receive business messages between you and your partners. The agreements are organized according to type and MEC identifies which message is received, from what sending party, how to process the message, which transport media to use, and the routing method to the proper partner.

Agreements are used to send (outbound) and receive (inbound) a message to one or multiple partners. An agreement must be created for every message that is sent (outbound) and received (inbound). For an example, you can have several installations of business servers connected to a single MEC and you will send one message to several different partners. You can also have more than one installation of M3 on a server.

The following overview shows MEC in runtime.

- **Detect** - A received message is identified and detected. The detection rules are defined in the Partner Admin Tool.
- **Process** - When a message is identified and detected, MEC will apply the process steps created in the Partner Admin Tool to execute the agreement.

Figure 1. MEC in runtime.



### Note:

In MEC versions earlier than 2.0, only XML files are detected by MEC and the conversion of flat file format to XML format is a required process step.

Starting from MEC version 2.0 to the latest, the Flat and XML files are detected and managed in the same way. But, if you will apply a mapping, file transformation to XML is still necessary.

For more information about mapping, see the *M3 Enterprise Collaborator ION Mapper Tool User Guide*.

## Uses of Partner Admin Tool

- Communication - manages inbound, outbound, and M3 API communications settings.
- Detections - manages MEC flexible message detection settings.
- Envelopes - manages the user-defined message envelopes that MEC can apply to outbound messages.
- Flat Definitions - manages all flat definitions in MEC database.
- XML Mappings - manages all published mappings.
- XSLT Definitions - manages all XSLT Definitions.
- Advanced - manages advanced settings to extend MEC's functionality.
- Agreement - manages all your partners and agreements.
- Error Suppression - suppresses errors by regular expression (regex)

- ["Starting Partner Admin Tool" on page 19](#)
- ["Configuring the Database Settings" on page 19](#)

## Starting Partner Admin Tool

- 1 On Windows Start menu, navigate to the M3 Enterprise Collaborator<version> program.
- 2 Expand the view and select Partner Administrator.  
The Partner Admin Tool window opens.
- 3 Start by managing the database settings.  
For more information, see ["Setting up a database connection" on page 21](#).  
For more information, see ["Database Fields Definition" on page 20](#).
- 4 Save the settings and exit the tool.

## Configuring the Database Settings

- ["Database Fields Definition" on page 20](#)
- ["Setting up a database connection" on page 21](#)
- ["Deleting a Database Connection" on page 22](#)
- ["Priority Settings in MEC" on page 22](#)

## Database Fields Definition

Refer to the following table of fields description and default value when configuring the database settings and Communication Databases in Partner Admin Tool.

Field	Definition
Name	Type a descriptive name for this database.
DB URL	<p>The URL for the JDBC connection string.</p> <ul style="list-style-type: none"> <li>If the database is configured with the default port number and not installed as a named instance, the URL format is :  <code>jdbc:sqlserver://[host]:[port number]</code> </li> <li>If the database is installed as a named instance, the format is any of the following:  <code>jdbc:sqlserver://[host]:[port number]\[instance name]</code>  <code>jdbc:sqlserver://[host]\[instance name]</code> </li> </ul>
DB Name	Type the database name.
Schema	<p>Type the schema to be used.</p> <p>Default value - <i>dbo</i>.</p>
User	Type a case sensitive database user name to access the database.
Password	Type a case sensitive database password for the user.
Driver	<p>Type the JDBC driver class to use. Configured by default as:</p> <p><code>com.microsoft.sqlserver.jdbc.SQLServerDriver</code></p>
Tr. Isolation	<p>Select the isolation level for this transaction.</p> <ul style="list-style-type: none"> <li>TRANSACTION_SERIALIZABLE</li> <li>UNKNOWN_TRANSACTION_ISOLATION</li> <li>TRANSACTION_NONE</li> <li>TRANSACTION_READ_COMMITTED</li> <li>TRANSACTION_READ_UNCOMMITTED</li> <li>TRANSACTION_REPEATABLE_READ</li> <li>TRANSACTION_SERIALIZABLE</li> </ul> <p><b>Note:</b> This field is used only in database Communication channels. To access the page, select Manage &gt; Communication, then access the Databases tab.</p>

## Setting up a database connection

Use this procedure to create new, configure existing, and switch between, or remove an existing database connection.

**Important:** In Partner Admin Tool, one database configuration must always be present.

- 1 On Partner Admin Tool menu, select File > Database Settings...
- 2 On the Database settings window, click New.
- 3 On Create new database item window, consider the database fields.  
For more information, see "[Database Fields Definition](#)" on page 20.
- 4 Click OK to save the new setting.
- 5 Click **Test** to verify the completeness of your new database settings.
- 6 Click OK to finish.

The new database setting is now listed in the table. If prompted for error, take note of the error to correct and click OK to continue. Then, go back to your database fields setting and correct the error.

- 7 You can now perform any of the following:

Task	Steps
To select a database to use	<ul style="list-style-type: none"> <li>• Select a configuration from the list and click <b>Use</b>.</li> </ul> <p>An asterisk appears under the column "Use" to indicate a database in use.</p>
To edit an existing configuration	<ol style="list-style-type: none"> <li>a Select a configuration from the list and click <b>Edit</b>. The Edit existing database item dialog opens.</li> <li>b Update the field values. For more information, see "<a href="#">Database Fields Definition</a>" on page 20.</li> <li>c Click Test to verify the completeness of your new configuration. If prompted for error message, take note of the error to correct and click OK to continue.</li> </ol>

Task	Steps
To remove a database connection	<p><b>Important:</b></p> <ul style="list-style-type: none"><li>• Delete has no undo function. Verify the version and name of a database before you proceed with delete.</li><li>• You cannot disconnect a database setting that is in use.</li></ul> <p><b>a</b> Select a database connection to delete and click <b>Delete</b>.</p> <p><b>b</b> At the confirmation prompt, click Yes.</p>

**8** Close and exit.

## Deleting a Database Connection

**Important:**

- This procedure will remove only the connection to a database and not the database in its location.
- Delete has no undo function. Verify the version and name of a database setting before you delete.
- You cannot delete a database setting that is in use.

- 1** On Partner Admin Tool menu, select File > Database Settings.
- 2** On the Database settings window, select a database connection and click Delete.
- 3** At the confirmation prompt, click Yes to delete the database connection.
- 4** Close and exit.

## Priority Settings in MEC

### Priority property

Integer. Value '1' is set as high priority.

- The priority property is available in a channel and in a partner agreement.
- The priority is used to compute where a message is placed in a queue and is computed (internally) with the formula which also includes the timestamp.

Technically, even if a message is set to priority '1', but a priority '3' message has been queued ahead of priority 1 message, the priority '3' message will be executed first.

- The Priority property is different from another property - Ordered. In a multi-worker thread setup (which is the standard), the priority does not guarantee that priority 1 message gets to finish execution first than lower priority messages. The priority guarantees only the order inside the queues but multiple worker threads can get from queue in succession.

## **Ordered property**

Value 0 or 1.

- Value of 1 or 0. The ordered priority guarantees that a message is processed one after the other based on an Order ID. Order ID is defined internally using the channel name and/or the File Splitting process.
- An ordered channel ABC has a different set of ordered messages from ordered channel DEF.
- File Splitting process has an ordered option if the user wants to handle processing of sub messages sequentially.

For more information, see the chapter on File Splitting. "[Split Redetect](#)" on page 123

## **Check Order process**

- The check order process uses the ordering functionality. It defines an order ID based on an XPath value from the received message. For example, Related messages should have a unique ID which will be used to order them against.

- ["Communication Channel Overview" on page 24](#)
- ["Communication Channel Types" on page 25](#)
- ["Setting Up Receive Channels " on page 25](#)
- ["Setting Up Send Channels" on page 46](#)
- ["M3 API" on page 56](#)
- ["FTP Script" on page 61](#)
- ["Web Services" on page 64](#)
- ["DAF Archiving" on page 69](#)
- ["Databases" on page 73](#)
- ["MECEventHub" on page 75](#)
- ["MEC Schedules" on page 79](#)
- ["Data Translator" on page 81](#)

## Communication Channel Overview

Settings for inbound, outbound, and M3 Application Program Interface (API) communications are initially defined when you set up MEC. Later on, you can add properties such as folder names, port numbers, time-out values, and queue settings for MEC receive channels.

When a receive channel is enabled, MEC will "listen" for messages on that channel. When a message is received it is detected to get the correct agreement to use. You can also set similar properties for outbound addresses that are used in outbound agreements.



## Communication Channel Types

Files are transferred through data connections between hosts using different channel types for different functions to perform.

### Send Channel

Send channel allows the sending host to communicate to a storage device in the receiving host. Several send channels can support both synchronous and asynchronous message delivery types.

### Receive Channel

Receive channel allows the receiving host to communicate from a storage device in the sending host. There are two receive communication channel types, the Server and the Polling type.

#### Server type receive channel

Server is a passive channel type that waits for a connection from external resources for incoming communication.

When MEC is busy,

- **MvxNGIn** returns a busy code back to M3.
- **MvxNGInRouter** will not return a busy code, but it will persist messages into the local file system.

**Note:** Server type channel load should be managed external to MEC.

For more information, see "[Server Type Channel Properties](#)" on page 35.

#### Polling type receive channel

Polling is a proactive channel type that regularly communicates with external resources based on your defined time interval.

MEC engine is enhanced to provide utilization check based on the current load that MEC is processing. If MEC has a full load, polling channels may skip a poll.

For more information, see "[Polling Type Channel Properties](#)" on page 26.

## Setting Up Receive Channels

- "[Polling Type Channel Properties](#)" on page 26
- "[Server Type Channel Properties](#)" on page 35
- "[WebServicesSyncIn Protocol Type](#)" on page 41
- "[Adding Receive Channel](#)" on page 44

- ["Viewing and Editing Receive Channel" on page 45](#)
- ["Deleting a Receive Channel" on page 45](#)

## Polling Type Channel Properties

This section describes the fields and default value of different polling type receive channel protocol properties. Use these to define the communication protocols needed in setting up receive communication channels.

- [Properties for applicable channels](#)
- [MSMQIn](#)
- [MQSeriesIn](#)
- [FTPPollin](#)
- [FTPPollin2](#)
- [FTPScriptPollin](#)
- [EventHub Subscriber/MECEventHubSubscriber](#)
- [IONDbIn](#)

### Properties for applicable channels

The properties: **Ordered**, **Priority**, and **DetectionOverride**, are present in all applicable channels in this section. With the default value of "3", these properties set the Prioritization of messages received into the channel.

### RunOnHost for Disk channels

In MEC version x.4.2.0, **RunOnHost** channel property is explicitly available to **OrderAck** and **EventHub** channels. For other channels, customize the protocol in PA Advanced menu to be able to set and use **RunOnHost**.

**Note:** Running **DiskIn/DiskOut** in central folder is not always equal to running the same channel on a **RunOnHost** property, regardless of whether the value is the same host as with central.

- If the folder value of **DiskIn/DiskOut** is absolute - both scenarios will resolve to the same folder in the file system.
- If the folder value is relative - the central node will resolve the value to the Central File root while the local process node will resolve the value to its internal directory in the grid installation.

## DiskIn

**Note:** Polling channels DiskIn, FTPPollIn, FTPPollIn2, MSMQIn, and MQSeriesIn will poll only if there is space in the queue. That is, when the receive queue length is less than the maximum length hint.

Property	Default Value	Description
BatchSize	20	Sets the maximum number of messages to process in each run.
SetVariationId	0	Sets the option to include variation ID into the manifest. Set to Off (0) by default.
DetectionOverride	0	Indicates if a channel is fixed to a particular detection group. (1-Channel, 2-XML, 3-Flat)
Encoding	UTF-8	Encoding used for received files. This is required for flat files.
Folder	Input	<p><b>Important:</b> Add read/write access to Grid Agent logon user. Default user is "Local Service".</p> <ul style="list-style-type: none"> <li>If the location is absolute, the DiskIn and DiskOut folder locations/paths are resolved in the Central File Folder host machine.</li> <li>If the location is relative, the DiskIn and DiskOut folder locations/paths are resolved in the Central File Folder share.</li> </ul> <p><i>Data</i> = {a...z   A...Z   0...9}*. The folder to poll.</p>
Ordered	0	Indicates if a channel processes messages in an ordered way. (1-ordered or 0-not ordered)
PollIntervall	1000	The poll interval in milliseconds. <i>Data</i> = unsigned integer.
StopTimeOut	0	<p>Number of milliseconds.</p> <p>Sets the waiting time for MEC during a stop before force terminating the channel. The default value 0, disables this feature.</p>

## MQSeriesIn

**Note:** Polling channels DiskIn, FTPPollIn, FTPPollIn2, MSMQIn, and MQSeriesIn will poll only if there is space in the queue. That is, when the receive queue length is less than the maximum length hint.

Property	Default Value	Description
ByteOrderMark	0	1 or 0, usually 0. Mandatory.
Channel		Channel name on the MQSeries server. Mandatory.
ConnectionRetries	10	An integer that is > 0.  Sets the number of times that plug-in will try to reconnect to MQSeries, if an error occurs.
ConnectionRetryWait	10000	Number of milliseconds between connection retry
Convert	0	MQSeriesInConfig.setConvert (true   false)  Applications do not regularly use this request because data conversion is also applied when a data is retrieved from the message buffer.  Use this to request for the application data to be converted and conformed to the <b>characterSet</b> and encoding attributes of <b>MQMessage</b> before the data is copied into the message buffer.  Mandatory.
DetectionOverride	0	Indicates if a channel is fixed to a particular detection group.  (1-Channel, 2-XML, 3-Flat)
Encoding	UTF-8	Encoding used for received files.
Host		Name of the <b>MQSeries</b> server host. Mandatory.
Manager		Name of the queue manager to connect to. Mandatory.
MQQueueWait	1000	Number of milliseconds.  Sets the MQ <b>GET</b> operation waiting time for a message to arrive on the queue.  If no message arrives within the set time, plug-in is given a chance to shut down before the queue is polled again.

Property	Default Value	Description
Port	1414	Port number used by MQSeries. Must be an integer >1024 and <65536. Mandatory.
Queue	MBMIn	Name of the MQSeries queue used. Mandatory.
Ordered	0	Indicates if a channel processes messages in an ordered way. (1-ordered or 0-not ordered)
ReceiveQueueWait	1000	Number of milliseconds. If MEC's internal queue is full, this sets the waiting time for MQ plug-in before trying to receive messages again.
StopTimeOut	0	Number of milliseconds. Sets the waiting time for MEC during a stop before force terminating the channel. The default value 0, disables this feature.

## MSMQIn

**Before you start** You must install MSMQ Windows components for the MSMQ functionality.

**Important:** When configuring inbound or outbound MSMQ channels, the property configurations must match both ways. If a wrong combination is configured, messages can disappear.

**Note:** Polling channels DiskIn, FTPPollIn, FTPPollIn2, MSMQIn, and MQSeriesIn will poll only if there is space in the queue. That is, when the receive queue length is less than the maximum length hint.

Property	Default Value	Description
Batch Size	10	Sets the maximum number of message to process in every run.
DetectionOverride	0	Indicates if a channel is fixed to a particular detection group. (1-Channel, 2-XML, 3-Flat)

Property	Default Value	Description
Encoding	UTF-8	Encoding used for received files.
Ordered	0	Indicates if a channel processes messages in an ordered way. (1-ordered or 0-not ordered)
PollInterval	1000	Sets the poll interval in milliseconds. Data= unsigned integer.
Queue		Indicates the queue name (generated GUID). Mandatory.
StopTimeOut	0	Number of milliseconds Sets the MEC waiting time during a stop before force terminating the channel. The default value 0, disables this feature.
Transactional	1	Indicates if a transaction will be used. 1=on, 0=off. Mandatory.
Use direct format	0	Indicates if direct format or path name will be used. 1=direct format, 0=path name.

## FTPPollin

Polls an external FTP server

**Note:** Polling channels DiskIn, FTPPollIn, FTPPollIn2, MSMQIn, and MQSeriesIn will poll only if there is space in the queue. That is, when the receive queue length is less than the maximum length hint.

Property	Default Value	Description
DetectionOverride	0	Indicates if a channel is fixed to a particular detection group. (1-Channel, 2-XML, 3-Flat)
Encoding	UTF-8	Encoding used for received files. This is required for flat files.
Host		Remote FTP server name.

Property	Default Value	Description
Ordered	0	Indicates if a channel processes messages in an ordered way. (1-ordered or 0-not ordered)
Password		User password.
PollInterval	1000	Poll interval in milliseconds. Data= unsigned integer.
Port	21	Server port number.
ReceiveDir		Folder name on the remote server to poll.
ReadTimeOut	5000	Number of milliseconds Sets the MEC waiting time for FTP server to respond.
StopTimeOut	0	Number of milliseconds. Sets the waiting time for MEC during a stop before force terminating the channel. The default value 0, disables this feature.
User		The user name to access the server.

## FTPPollin2

This protocol is similar to FTPPollin, but with the additional flag to toggle between active and passive modes.

**Note:** Polling channels DiskIn, FTPPollIn, FTPPollIn2, MSMQIn, and MQSeriesIn will poll only if there is space in the queue. That is, when the receive queue length is less than the maximum length hint.

Property	Default Value	Description
ConnectionRetryCount	0	Sets the number of times that the channel will try to reconnect to FTP Server.
DetectionOverride	0	Indicates if a channel is fixed to a particular detection group. (1-Channel, 2-XML, 3-Flat)
Encoding	UTF-8	Encoding used for received files. This is required for flat files. For example: UTF-8

Property	Default Value	Description
Host		Name of the remote FTP server.
Ordered	0	Indicates if a channel process messages in an ordered way. (1-ordered or 0-not ordered)
Password		The user password.
PollInterval	1000	Poll interval in milliseconds. Data is unsigned integer.
Port	21	Server port number. Data= unsigned integer.
ReadTimeOut	5000	Number of milliseconds. Sets the MEC waiting time for the FTP server to respond.
ReceiveDir		Folder name in the remote server to poll.
RetryInterval	5000	Number of milliseconds. Sets the waiting time for MEC before reconnecting to the FTP Server.
StopTimeOut	0	Number of milliseconds. Sets the waiting time for MEC during a stop before force terminating the channel. The default value 0, disables this feature.
UseActiveMode	1	Indicates the use of FTP active (1) or passive (0) mode.
User		User name to access the server.

## FTPScriptPollIn

Property	Default Value	Description
DetectionOverride	0	Indicates if a channel is fixed to a particular detection group. (1-Channel, 2-XML, 3-Flat).



Property	Default Value	Description
Encoding		Encoding used for received files. This is required for flat files. For example: UTF-8
Host		Remote FTP server name.
Ordered	0	Indicates if a channel process messages in an ordered manner. (1-ordered or 0-not ordered)
Password		User password.
PollInterval		Number in milliseconds for poll interval.
Port	8080	Server port number.
ReadTimeOut	5000	Number of milliseconds that MEC waits for the FTP server to respond. Mandatory.
RecordSeparator		Record separator to use for prefixed header.
ScriptFile		Number of files to execute.
User		User name to access the server.

## EventHub Subscriber/MECEventHubSubscriber

The MECEventHubSubscriber channel is used to receive events/messages that are published by another application through the Event Hub application. To be able to use MECEventHubSubscriber, you need to add subscriptions. Subscriptions are predicates indicating that a subscriber is to receive a particular event.

For more information, see the *EventHub* topic in *ION Grid Administration Guide*.

**Important:** MEC and EventHub must be running on the same Grid.

Property	Default Value	Description
SetVariationId	0	Sets the option to include variation ID into the manifest. Set to Off (0) by default.
Ordered	0	Indicates if a channel processes messages in an ordered manner. (1-ordered or 0-not ordered)

Property	Default Value	Description
PersistFlag	1	Activates persistence in the EventHub. (1-persist or 0-no persistence)
RunOnHost	Any	Sets to which host to run the channel. The default value "any" indicates that the channel can run on any host. See the note below.
StopTimeOut	0	Number of milliseconds.  Sets the waiting time for MEC during a stop before force terminating the channel. The default value 0, disables this feature.

## IONDbIn

This channel polls for ION messages from the specified `ConnectionURL`.

Property	Default Value	Description
BatchSize	50	Number of messages to read in every inbox poll.
BodTypes		A comma separated list of BodTypes to handle in this instance
ConnectionUrl	path to jdbc	The JDBC connection url. For example: <code>jdbc:sqlserver://localhost:1433; databaseName=&lt;value&gt;</code>
DelayTime	10000	Time between inbox polls in milliseconds.
DriverClass		The JDBC driver class. For example: <code>com.microsoft.sqlserver.jdbc. SQLServerDriver</code>
Password		Password to access the database.
UserName		User name to access the database.

## Server Type Channel Properties

This section describes the fields and default value of different server type receive channel protocol properties. Use these to define the communication protocol to use when setting up receive communication channels.

- [HTTPIn](#)
- [FTPIn](#)
- [MvxNGIn](#)
- [MvxTGIn](#)
- [MvxNGInRouter](#)
- [HTTPSIn](#)
- [HTTPSISIn](#)

### Applicable to all channel properties

The properties: Ordered, Priority, and DetectionOverride, are present on all applicable channels in this section. With the default value of "3", these properties set the Prioritization of messages received into the channel.

### HTTPIn

This channel generates an acknowledgment received response.

Property	Default Value	Description
DetectionOverride	0	Indicates if a channel is fixed to a particular detection group (1-Channel, 2-XML, 3-Flat)
Encoding	UTF-8	Encoding to use for received files.
MaxSubThreads	5	Sets the limit of receiving threads to avoid MEC overload. Data = unsigned integer.
Ordered	0	Indicates if a channel processes messages in an ordered way. (1-ordered or 0-not ordered).
Port	8000	Mandatory computer port number. Data = unsigned integer.

Property	Default Value	Description
StopTimeOut	0	Number of milliseconds.  Sets the waiting time for MEC during a stop before force terminating the channel. The default value 0, disables this feature.
ReadTimeOut	5000	Number of milliseconds  Sets the MEC waiting time for FTP server to respond.

## FTPIn

Property	Default Value	Description
DetectionOverride	0	Indicates if a channel is fixed to a particular detection group.  (1-Channel, 2-XML, 3-Flat)
Encoding	Latin 1	Encoding to use for received files.
MaxSubThreads	5	Sets the limit of receiving threads to avoid MEC overload.  Data= unsigned integer.
Ordered	0	Indicates if a channel processes messages in an ordered way.  (1-ordered or 0-not ordered).
Port	21	FTP channel port number.  Data= unsigned integer.
StopTimeOut	0	Number of milliseconds.  Sets the waiting time for MEC during a stop before force terminating the channel. The default value 0, disables this feature.
TimeOut	1000	Number of milliseconds.  Sets the timeout for the FTP service.

## MvxNGIn

This protocol is used for MBM initiator XML files sent from M3 Java.

**Note:** If MEC is busy, MvxNGIn returns a busy code back to M3. MvxNGInRouter does not return busy code, it persists messages into the local file system.

Property	Default Value	Description
ByteOrderMark	0	Indicates if a UTF-16LE byte order mark should be prefixed to the received message. (1   0)
DetectionOverride	0	Indicates if a channel is fixed to a particular detection group. (1-Channel, 2-XML, 3-Flat)
Encoding	UTF-8	Encoding to use for received files. This is required for flat files.
MaxSubThreads	5	Sets the limit of receiving threads to avoid MEC overload.  Data= unsigned integer.
Ordered	0	Indicates if a channel processes messages in an ordered way. (1-ordered or 0-not ordered).
Port	6010	MvxNGIn channel port number.
ReadTimeOut	5000	Number of milliseconds.  Sets the MEC waiting time for M3 to respond.
StopTimeOut	0	Number of milliseconds.  Sets the waiting time for MEC during a stop before force terminating the channel. The default value 0, disables this feature.

## MvxTGIn

This protocol is used for MBM initiator XML files sent from M3 RPG.

Property	Default Value	Description
ByteOrderMark	0	Indicates if a UTF-16LE byte order mark should be prefixed to the received message. (1 or 0)
DetectionOverride	0	Indicates if a channel is fixed to a particular detection group (1-Channel, 2-XML, 3-Flat)
Encoding	UTF-8	Encoding to use for received files. This is required for flat files.
MaxSubThreads	5	Sets the limit of receiving threads to avoid MEC overload. Data= unsigned integer.
Ordered	0	Indicates if a channel processes messages in an ordered way. (1-ordered or 0-not ordered).
Port	6011	MvxNGIn channel port number. <i>Data</i> = the computer port.
ReadTimeOut	5000	Number of milliseconds. Sets the MEC waiting time for M3 to respond.
StopTimeOut	0	Number of milliseconds. Sets the waiting time for MEC during a stop before force terminating the channel. The default value 0, disables this feature.

## MvxNGInRouter

This protocol is used for MBM initiator XML files sent from M3 Java.

Property	Default Value	Description
ByteOrderMark	0	Indicates if a UTF-16LE byte order mark should be prefixed to the received message. (1 or 0)
DestinationFolder	routerInput	Local directory where received files are temporarily stored before sending to MEC for processing.

Property	Default Value	Description
DetectionOverride	0	Indicates if a channel is fixed to a particular detection group. (1-Channel, 2-XML, 3-Flat)
Encoding	UTF-8	Encoding to use for received files. This is required for flat files.
MaxSubThreads	5	Sets the limit of receiving threads to avoid MEC overload. Data= unsigned integer.
Ordered	0	Indicates if a channel processes messages in an ordered way. (1-ordered or 0-not ordered).
Port	6010	MvxNGIn channel port number.
ReadTimeOut	5000	Number of milliseconds. Sets the MEC waiting time for M3 to respond.
StopTimeOut	0	Number of milliseconds. Sets the waiting time for MEC during a stop before force terminating the channel. The default value 0, disables this feature.

## HTTPSyncIn

HTTPSyncIn channel is used to receive messages through HTTP. It acts as an HTTP listener waiting for connections from clients.

When using HTTPSync channel, the agreements must have a Send process with the HTTPSync protocol. After setting an HTTPSync channel, define an error handling flow so that custom error responses can be reported back to the requestor.

**Note:** Long processing of messages can cause the host to terminate the connection due to a timeout.

Property	Default Value	Description
DetectionOverride	0	Indicates if a channel is fixed to a particular detection group (1-Channel, 2-XML, 3-Flat).

Property	Default Value	Description
Encoding		Encoding to use for received files. This is required for flat files.  For example: UTF-8
MaxSubThreads	5	Sets the limit of receiving threads to avoid MEC overload.  Data= unsigned integer.
Ordered	0	Indicates if a channel processes messages in an ordered manner.  (1-ordered or 0-not ordered).
Port	8080	HTTP channel port number.
ReadTimeOut	5000	Number of milliseconds.  Sets the MEC waiting time for a client to respond.
StopTimeOut	0	Number of milliseconds.  Sets the waiting time for MEC during a stop before force terminating the channel. The default value 0, disables this feature.

## HTTPSISIn

This is a custom HTTP channel for Service Information System (SIS).

Property	Default Value	Description
DetectionOverride	0	Indicates if a channel is fixed to a particular detection group  (1-Channel, 2-XML, 3-Flat).
Encoding		Encoding to use for received files. This is required for flat files.  For example: UTF-8
MaxSubThreads	5	Sets the limit of receiving threads to avoid MEC overload.



Property	Default Value	Description
Ordered	0	Indicates if a channel process messages in an ordered manner. (1-ordered or 0-not ordered).
Port	8080	HTTP channel port number.
ReadTimeOut	5000	Number of milliseconds. Sets the MEC waiting time for a client to respond.
StopTimeOut	0	Number of milliseconds. Sets the waiting time for MEC during a stop before force terminating the channel. The default value 0, disables this feature.

## WebServicesSyncIn Protocol Type

Following is a table of field definitions for your reference when managing Web Services Security parameters.

Property	Default Value	Description
Keystore		File location of the SSL keystore. Required, if URL uses HTTPS protocol.
Storepass		Password of the SSL keystore. Required, if URL uses HTTPS protocol.
URL		Base URL that will be used to publish webservises. Required, if protocols https, keystore, and storepass will be applied. Otherwise, ignored.
User		User name for the username token identification of inbound messages. Required, if the username token and callback handler class is empty.

Property	Default Value	Description
Password		<p>Password for the username token identification of inbound messages.</p> <p>Required, if username token is 1 (true) and callback handler class is empty. This means, the property is required if both are true.</p>
InPwdCallbackClass		<p><b>WSS4J</b> callback handler class to handle the retrieval of passwords for inbound messages.</p> <p>Optional.</p> <p>If left blank, the User and Password properties will be required.</p>
InPasswordType	Valid values are Text and Digest.	<p>Password encoding for username token of inbound messages.</p> <p>Required, if you will use a username token for inbound messages.</p>
InApplyUsername		Performs a <b>Username</b> Token identification for inbound messages.
InApplyTimestamp		Adds timestamp to the security header for inbound messages.
InApplySignature		Adds signature to inbound messages.
InApplyEncryption		Adds encryption to inbound messages.
InSignaturePropFile		<p>Name of the crypto property file to use for SOAP Signature of inbound messages. File must be in the application classpath.</p> <p>Required, if applying a signature for inbound messages.</p>
InSignatureKeyIdentifier	<p>Values are DirectReference and IssuerSerial.</p> <p>Default is: IssuerSerial</p>	<p>Defines which key identifier type to use for inbound messages.</p> <p>Used only when applying signature for inbound messages.</p> <p>Optional.</p>
InDecryptionPropFile		<p>Name of the crypto property file to use for SOAP Decryption of inbound messages. File must be in the application classpath.</p> <p>Required, if applying decryption for inbound messages.</p>

Property	Default Value	Description
InEncryptionKeyIdentifier	Values are X509KeyIdentifier, DirectReference, Thumbprint, SKIKeyIdentifier, EmbeddedKeyName, and IssueSerial.  Default is: IssuerSerial	Defines which key identifier type to use for inbound messages.  Used only when applying encryption for inbound messages  Optional.
InAliasPassword		Private key alias password for encryption.  Required, if InApplyEncryption is true and if InPwdCallbackClass is empty.
OutApplyUsername		Performs UsernameToken identification for outbound messages.
OutApplyTimestamp		Adds timestamp to the security header for outbound messages.
OutApplySignature		Adds signature to outbound messages.
OutApplyEncryption		Adds encryption to outbound messages.
OutUser		User name for username token identification and/or the alias name used for signature/ encryption of outbound messages.  Required, if username token and callback handler class is empty, or if signature and/or encryption is applied.
OutPassword		Password for username token identification of outbound messages.  Required, if username token and callback handler class is empty.
OutPwdCallbackClass		<b>WSS4J</b> callback handler class to handle the retrieval of passwords for outbound messages.  Optional. If left blank, the User and Password properties will be required.
OutPasswordType	Valid values are Text and Digest.	Password encoding for username token of outbound messages.

Property	Default Value	Description
OutSignaturePropFile		Name of the crypto property file to use for SOAP Signature of outbound messages. File must be in the application classpath.  Required, if applying signature for outbound messages.
OutSignatureKeyIdentifier	Values are DirectReference and IssuerSerial.	Defines which key identifier type to use for inbound messages.  Optional. Used only when applying signature for inbound messages.
OutEncryptionPropFile		Name of the crypto property file to use for SOAP Decryption of outbound messages. File must be in the application classpath.  Required, if applying encryption for outbound messages.
OutAliasPassword		Private key alias password for encryption/ signature.  Required, if outbound password callback handler class is empty.
OutSignatureParts		Parameter to define which parts of the outbound message shall be <b>signed</b> .  Optional.
OutEncryptionParts		Parameter to define which parts of the outbound message shall be <b>encrypted</b> .  Optional.

## Adding Receive Channel

Use this procedure to define the valid channels to receive messages.

- 1 On Partner Admin Tool menu, click Manage > Communication.
- 2 Go to Receive tab and click New.
- 3 On Create new receive channel object window, perform the following:
  - a Type a unique channel name,

- b** Select a protocol to use, and
  - c** Click on the Values column and add the protocol values.
- 4** Click OK to store the receive channel in MEC database.

Each protocol has its own set of defined properties, for example, a property name, value, and description. Fields with asterisk (\*) are mandatory, these fields must have an associated default value or a user added value.

For more information on receive channel protocol properties, see the following:

["Polling Type Channel Properties"](#) on page 26

["Server Type Channel Properties"](#) on page 35

## Viewing and Editing Receive Channel

Use this procedure to view and modify the properties of a receive channel.

**Important:** Restart MEC after modifying an agreement or communication channel for changes to take effect.

- 1** On Partner Admin Tool menu, click Manage > Communication.
- 2** Click on Receive tab and select a channel in the list.  
The properties for the selected channel are shown.
- 3** Click Edit.
- 4** On edit existing receive channel object window, modify the field values.
- 5** Click OK to save your changes.
- 6** In the lower pane listing, select **Enabled** across the modified object to activate and MEC will poll for messages on that channel.

Clear the **Enabled** check box to disable a receive channel.

For more information on receive channel protocol properties, see the following:

["Polling Type Channel Properties"](#) on page 26

["Server Type Channel Properties"](#) on page 35

## Deleting a Receive Channel

Use this procedure to remove a receive channel.

**Important:** Restart MEC after you have made changes to an agreement or communication channel for changes to take effect.

- 1 On Partner Admin Tool menu, click Manage > Communication.
- 2 Click on Receive tab and select the channel to remove.
- 3 Click Delete.
- 4 At the prompts, click Yes to delete the receive channel from MEC database.

## Setting Up Send Channels

- ["Send Channel Properties" on page 46](#)
- ["Creating New Send Protocol " on page 52](#)
- ["Creating New Send Channel" on page 52](#)
- ["Managing Send Channels" on page 53](#)
- ["Modifying Filenaming Settings" on page 54](#)
- ["File Naming Default Classes" on page 54](#)
- ["Filenaming User Defined Classes" on page 56](#)

## Send Channel Properties

This section describes the fields and default value of different send channel protocols. Refer to these lists when you define the send communication protocols to use in send communication channels.

**Note:** Server type channels have a property called Host. This property defines which host the channel will run on the Grid. You can use either an IP address or a host name.

- [DISK](#)
- [FTP](#)
- [HTTP](#)
- [WebSphere MQ \(MQSeries\)](#)
- [MSMQ](#)
- [FTPScript](#)
- [HTTPSync](#)

- [IONDBOut](#)

## DISK

The disk protocol writes the message to a disk. You need to change several parameters to be able to put the file in a desired location and with a specific name.

### Note:

- Add full control access to the folder on the Central File Location host for the GridAgent logon Infor ION Grid Bootstrap.

For more information on BootStrap service, see *ION GRID Administration Guide*.

- If the location is absolute, the DiskIn and DiskOut folder locations/paths are resolved in the Central File System host machine.
- If the location is relative, the DiskIn and DiskOut folder locations/paths are resolved in the Central File System share.

Property groups	Description
Naming	Name and description for this channel.
Basic Configuration	Includes the output folder, file name, and file extension.
Unique File Name	For more information about Unique File Name, see the disk protocol.

## FTP

**Important:** After you set the FTP, send a test message with the new configuration parameters. Send Test Message uses an empty manifest. As a result, using the MEC\_InfixManifest unique file name class option will return a blank text when retrieving manifest properties.

Property groups	Description
Naming	Name and description for this channel.
Basic Configuration	Includes the host, port, URL path, file name, and file extension.  The host, port, and URL path together will form a correct URL. The URL is the path from the FTP Root to the folder that you will connect to.  For the File Name, refer to the documentation on disk protocol.
Unique File Name	For more information about Unique File Name, see the disk protocol.

Property groups	Description
User Configuration	Sets user access to Anonymous, or specify a user name/ password.

## HTTP

**Important:** After you set the HTTP, send a test message to verify the new configuration parameters.

Property groups	Description
Naming	Channel name and description.
Basic Configuration	<p>Includes the host, port, URL path, and content type.</p> <p>The host, port, and URL path all together will form a correct URL. The URL is the path from HTTPRoot to the folder that you will connect to.</p> <p>Use the HTTP content type header to specify the content type returned, for example, a web browser. When an internet client (a browser) accesses a remote document using the current HTTP protocol (1.0 or 1.1), the server will inform the client what content type is returned based on the HTTP content-type header details.</p> <p>This is an authoritative information according to HTTP standards.</p>
User Configuration	Sets user access to Anonymous, or specify a user name/ password.

## WebSphere MQ (MQSeries)

Correlation ID is automatically managed in the message when you use MQ protocol.

For a detailed description of MQ parameters, see the MQ manual.

**Important:** After you set the WebSphere MQ (MQSeries), send a test message with the new configuration parameters.

Properties	Description
Naming	Channel name and description.
Queue	Adds the name of the MQ message queue where you will send messages.



Properties	Description
Host	Adds the address to the MQ server.
Channel	Adds the name of the MQ channel to use.
Port	Adds the MQ port to use.
Queue Manager	Adds the name of the MQ Queue Manager.
Format	See the MQ manual for the meaning of different formats.
Character Set	Specifies the coded character set identifier of character data in a message.

## MSMQ

MSMQ protocol window contains the queue name and the message label.

For a detailed description of MSMQ parameters, see the MSMQ manual.

Property groups	Description
Naming	Channel name and description.
Queue Name	Adds the name of the MSMQ queue where to send a message.
Message Label	Adds the label to a message.
Path name and Direct format	Defines the queue name format for this configuration.
Use transaction	Allows for transaction defined MSMQ connection.

**Important:** When configuring inbound or outbound MSMQ channels, ensure that the property configurations match both ways. If a wrong combination of inbound or outbound MSMQ channels is configured, messages can disappear.

Destination queues can be set to public or private, transactional or non-transactional, or a combination. In MEC, when distributed message queuing is created, it specifies the destination queue setting (public, private, transactional, non-transactional, or a combination).

The following table shows the distributed message processing in MEC Send(Out).

	Public	Private
Transactional	<ul style="list-style-type: none"><li>• Path name and Direct Format works for local and remote</li><li>• Works for Send and Receive (in and out channel)</li><li>• No support for remote Transactional Receive</li><li>• Works for remote Non-Transactional Receive</li></ul>	<ul style="list-style-type: none"><li>• Path name and Direct Format works for local</li><li>• No support for remote private transactional queue</li></ul>
Non-transactional	<ul style="list-style-type: none"><li>• Path name and Direct Format works for local and remote</li><li>• Works for Send and Receive (in and out channel)</li><li>• No support for remote Non-Transactional Receive</li><li>• Works for remote Non-Transactional Receive</li></ul>	<ul style="list-style-type: none"><li>• Direct Format works for local and remote</li><li>• No support for remote private non-transactional queue</li><li>• Works for Send and Receive (in and out channel)</li></ul>

**Note:** In MSMQ 3.0, if a message is sent to a non-transactional queue within a transaction, or a transactional queue outside of a transaction, the send operation succeeds but the message is not delivered to the destination queue.

## FTPScript

Property groups	Description
Channel Configuration	Type the name, description, and then select <b>FTPScript</b> protocol.
Basic Configuration	Defines the host, port, send file name. Select the filename extension type (xml, txt, or zip).
Unique File name	Select this to make the file name unique and then select a unique file name class from the list.
User Configuration	Sets user access to Anonymous, or specify a user name/ password.

**Note:** You can access the agreement's Send process for additional FTPScript protocol settings.

You can select the FTP script to use, specify the values to the EDITABLE properties, and view the FTP scripts.

## HTTPSync

Use HTTPSync protocol to reply to messages received through the HTTPSyncIn receive channel.

When using HTTPSync channel, the agreements must have a Send process with the HTTPSync protocol. After you define your HTTPSync, define an error handling process flow for custom error responses to be reported back to the requestor.

**Important:** Long processing of messages could cause the host to terminate the connection due to a timeout.

Property groups	Description
Channel Configuration	Type the name, description, and then select <b>HTTPSyncSend</b> protocol.
Response Configuration	Defines the content type.  Select <b>Keep connection alive</b> to maintain the communication with the client.

## IONDBOut

Property groups	Description
Channel Configuration	Type the name, description, and then select <b>IONDbOut</b> protocol.
Basic Configuration	<p>Defines the JDBC driver class and Connection URI.</p> <ul style="list-style-type: none"> <li>For an example of JDBC driver class: <code>com.microsoft.sqlserver.jsdb.SQLServerDriver</code></li> <li>For an example of Connection URI: <code>jdbc:sqlserver://localhost:1433;databaseName=&lt;value&gt;</code></li> </ul> <p>Type the username and password.</p> <p><b>Important:</b> Click Test Connection to test the basic configuration parameters.</p>

Property groups	Description
ION Outbox Configuration	Define the source logical and tenant identification, for example: <ul style="list-style-type: none"><li>From Logical Id: <code>lid://infor.m3be.&lt;value&gt;</code></li><li>Tenant id: <code>infor</code></li></ul> Type the message priority order, for example, <code>4</code> .

## Creating New Send Protocol

**Note:** All Java classes and supported character encoding used for outbound classes must be defined in Advanced Channel settings.

If you want to use a protocol that is not supported as a standard in MEC, define your own Java classes.

- 1 On Partner Admin Tool menu, click Manage > Advanced.
- 2 Go to Send Protocols tab, then click New.
- 3 On Create new send protocol object window, consider the following fields:

<b>Protocol</b>	Type a descriptive name for this send protocol.
<b>Send class</b>	Type a Java send class to use, for example: <code>com.intentia.ec.communication.DiskOut</code>
<b>UI class</b>	Type a Java UI class to use, for example: <code>com.intentia.ec.partneradmin.swt.manage.DiskOutPanel</code>

- 4 Click OK to store the new send protocol in the MEC database.

## Creating New Send Channel

- 1 On Partner Admin Tool menu, click Manage > Communication.
- 2 Go to Send tab, then click New.
- 3 On Create new send object window, consider the Channel configuration fields:

<b>Name</b>	Type a name for this new channel.
<b>Description</b>	Type a brief description.

---

<b>Protocol</b>	Select a protocol to use.
-----------------	---------------------------

**Note:** The DISK protocol writes the message to a disk. To put the file in a desired location with a specific name you need to change several parameters.

For more information, see "[Modifying Filenaming Settings](#)" on page 54.

#### 4 Consider the Basic configuration settings.

**Note:** The standard MEC setting requires for an output folder and a file extension. MEC then constructs a unique filename by using the internal `UUID` for the message and appending the chosen extension `UUID.ext`.

<b>Output folder</b>	Type the output folder name.
----------------------	------------------------------

<b>File name</b>	Type a name for this file.
------------------	----------------------------

<b>File extension</b>	Select a file extension type to use (XML, txt, zip).
-----------------------	--

For example, a simple setting with an `xml` file extension will have this filename:

`0a65c7fee1-4f42-3190-e849-920ae4d170.xml`

#### 5 Click OK to save the new send channel in the MEC database.

## Managing Send Channels

Use this procedure to view, edit, or remove the properties of existing send channels.

- 1 On Partner Admin Tool menu, click Manage > Communication.
- 2 Access the Send tab, then select a send channel from the list.
- 3 Perform any of the following:

Task	Steps
To view	<ul style="list-style-type: none"> <li>• Rest on a channel in the list to display its property information.</li> </ul>
To edit	<ol style="list-style-type: none"> <li>a Click Edit.</li> <li>b On Edit existing routing object, modify the Channel configuration, basic configuration, and Unique file name information.</li> <li>c Click OK to save your changes.</li> </ol>

Task	Steps
To delete	<p><b>Important:</b> Routings in use cannot be deleted.</p> <p><b>a</b> Click delete.</p> <p><b>b</b> At the prompts, click Yes to delete.</p>

- 4 Close and exit the application.

## Modifying Filenaming Settings

Use this procedure to activate the advanced setting to be able to construct a unique file name.

**Important:** Do not make any changes to these settings if you do not have the proper experience.

- 1 On Partner Admin Tool menu, click Manage > Communication > Send tab.
- 2 Select a channel from the list and click Edit.
- 3 On Edit existing routing object window, select **Make File name unique**.
- 4 On Class to make file name unique field, select a class to use.

For an example of advanced setting using **MEC\_PrefixUUID**, the file name will be:

**0a65c7fee1-4f42-3190-e849-920ae4d170.AnyText.xml**

**Note:** The default list for MEC classes contains: MEC\_PrefixUUID, MEC\_InfixUUID, MEC\_SuffixUUID, and MEC\_InfixManifest.

To expand the list with user-defined classes, go to Manage > Advanced > File Names.

For more information on MEC default classes, see "[File Naming Default Classes](#)" on page 54

- 5 Click OK to save the modified send channel in the MEC database.

## File Naming Default Classes

The advanced setting lists the following MEC default classes.

MEC Default Classes	Definition
MEC_Prefix	<p>The <b>MEC_PrefixUUID</b> class adds the internal UUID in front of (prefix) the file name.</p> <p>File name result: <b>UUIDAnyText.ext</b></p>
MEC_Infix	<p>The <b>MEC_InfixUUID</b> class inserts the internal UUID in the position indicated by "(UUID)" in the file name field.</p> <p>File name result: <b>AnyUUIDText.ext</b></p> <p><b>Note:</b> The parenthesis must be included in the file name field.</p> <p>If the indicator (UUID) is not found, an exception will be thrown.</p>
MEC_Suffix	<p>The <b>MEC_suffixUUID</b> class appends the internal UUID after the file name.</p> <p>File name result: <b>AnyTextUUID.ext</b></p>
MEC_InfixManifest	<p>The <b>MEC_InfixManifest</b> class allows for manifest data in the generated file name.</p> <p>When you set up the send object, the manifest item name is case sensitive and must be enclosed in parenthesis.</p> <p>The manifest item name, including the parentheses, will be replaced by the manifest item's value at runtime. If the manifest item does not exist, an empty string ("" ) will be used as value.</p> <p>You can use one or more manifest items in a file name. If the generated file name is blank an error will be generated.</p> <p>File name format: ([any text]"("manifest item)")...[any text"</p> <p><b>Note:</b> If you type a dash ("-") for the file extension in the send object setting, it will create files without extension.</p>

Examples for **MEC\_InfixManifest**:

<i>UUID.xml</i>	<p>Generated File name:</p> <p><b>627674dfe9-1245-9f8b-9344-d8582dc4fd.xml</b></p> <p>The manifest's UUID is used.</p>
-----------------	--

**Prefix** *cmn:agreement\_UUID\_*  
**Suffix.xml**

Generated File name:

**Prefix\_Test\_20\_ce820edf05-a749-b896-13bd-f9d409ac60\_Suffix.xml**

The name of the agreement is Test\_20.

---

*com:fileName*

Generated File name:

**MBM\_Test\_20\_in.xml**

The generated file extension is included in the manifest item value and the generated file name is equal to the received file's name.

---

If you put this Java code into a user defined function in a map:

```
setManifestInfo("map:FileName", "MyFileNameFromTheMap.xml");
```

You must do the following:

- set the file name to: `"(map:FileName)"`
- set the file extension to: `"-"` ,

The resulting file name will be: `"MyFileNameFromTheMap.xml"`.

You can now set the complete file name or part of the file name from a map.

---

## Filenaming User Defined Classes

You can also construct your own file naming classes. Any such class must implement the `IUniqueFileName` interface. When a class is built, the class is added to the Manage, Advanced, and File Names tabs. When a class is inserted into the system, it is treated by MEC in the same way as the other classes provided by MEC.

## M3 API

- ["M3 API Overview" on page 57](#)
- ["API Clean Up and Validation" on page 57](#)
- ["API Fields Definition" on page 57](#)
- ["Adding API Reference" on page 59](#)
- ["Managing API Reference" on page 60](#)



- ["Company/ Division Override" on page 60](#)

## M3 API Overview

Application Programming Interfaces (APIs) are routines that validate records, display record related information, and create or update files in the Infor database. In an API reference you define the data needed by MEC to call M3 APIs. The defined data is used in a mapping to call one or more M3 APIs.

## API Clean Up and Validation

### Proper API clean up and close

MEC allow various methods of handling API/MI programs and getting only the first data on an MI returning multiple records. As a result, unsent data remain on TCP buffers of the BE and system resources are consumed. Now, when closing API, a warning in `releaseAPICaller` is displayed. Then, MEC will read the remaining data in the BE to clean it up and to save on system resources.

### API checks and validation

Every MI connection in a pool that MEC will reuse is checked or verified first using the `GetServerTime` command. This verification method consumes time and resources. Now, MEC and BE performances can be improved by skipping check or validation based on a given time frame.

To define the time frame, open `ec.properties` and add a numeric value in the property `MvxAPI.Pool.Connection.CheckAPI.Time`. The default value 0 (zero) means to always perform check or validate.

For example, set the property value to 10, for MEC to check or validate APIs not used within 10 seconds.

```
MvxAPI.Pool.Connection.CheckAPI.Time=10
```

**Note:** LEPD v2 (LPD CAT project) runs +60h on big files with `checkAPI` consuming a fair amount of this time. Control the `check` property and run a test to see a fifty percent improvement in performance.

## API Fields Definition

Field	Description
Name	Name of the API reference.

Field	Description
Host	Name of the server where the M3 BE Server runs.
Port	<p>M3 BE Server port number that will be used to connect to M3 Interface (MI) programs.</p> <p>The port number is in the M3 property file <code>boot.batchdispatcher.port</code>. This property file can be viewed in M3 server view.</p> <p>Default port number for M3 Java is 26800.</p> <p>For M3 RPG, the port number is defined during the FPW (Floating Protocol Wrapper) setup. For more information on FPW, see the FPW documentation on the net.</p>
Encoding	<p>Select the character encoding that this particular API connection will use to communicate with M3. New encodings can be added in Manage &gt; Advanced &gt; Character Encodings tab.</p> <p>Default encoding for M3 Java is UCS-2 (UTF-16 with no BOM).</p> <p>Default encoding for M3 RPG is Latin1.</p>
M3 user	An M3 user for the API communication. This M3 user will establish the environment (M3 database), company and division, and other details to be used for the API calls.
Password	The M3 user password.
Force proxy	<p>Select this to specify the proxy connection that will be used.</p> <p>You may need to use a proxy connection because of firewalls and NATed networks settings.</p>
Company	<p>Type a numeric value, maximum of three digits.</p> <p>Use this to override the M3 API Company value.</p>
Division	<p>If the Company field has a value, use this field to assign a name for the Division.</p> <p>Use this to override the M3 API Division value.</p>
Import Use	Select an M3 API from the list then click <b>Use</b> . A numeric value 1 across an M3 API indicates that it is in use. When in use, it enables importing of agreements. You can not import agreements on an M3 API that is not in use.

## Adding API Reference

- 1 On Partner Admin Tool menu, click Manage > Communication > M3 API.
- 2 Click New.
- 3 On Create new api reference form, consider the following fields.

<b>Name</b>	Name of the API reference.
<b>Host</b>	The name of the server where the M3 BE Server runs.
<b>Port</b>	<p>The M3 BE Server port number that will be used to connect to M3 Interface (MI) programs.</p> <p>The port number is in the M3 property file <code>boot.batchdispatcher.port</code>. This property file can be viewed in M3 server view.</p> <p>The default port number for M3 Java is 26800.</p> <p>For M3 RPG, the port number is defined during the FPW (Floating Protocol Wrapper) setup. For more information on FPW, see the FPW documentation on the net.</p>
<b>Encoding</b>	<p>Select the character encoding that this particular API connection will use to communicate with M3. New encodings can be added in Manage &gt; Advanced &gt; Character Encodings tab.</p> <p>The default encoding for M3 Java is UCS-2 (UTF-16 with no BOM).</p> <p>The default encoding for M3 RPG is Latin1.</p>
<b>M3 user</b>	An M3 user for the API communication. This M3 user will establish the environment (M3 database), company and division, and other details to be used for the API calls.
<b>Password</b>	The M3 user password.
<b>Force proxy</b>	Select this to specify the proxy connection that will be used. You may need to use a proxy connection because of firewalls and NATed networks settings.
<b>Company</b>	<p>Type a numeric value, maximum of three digits.</p> <p>Use this to override the M3 API Company value.</p>
<b>Division</b>	<p>If the Company field has a value, use this field to assign a name for the Division.</p> <p>Use this to override the M3 API Division value.</p>
<b>Import Use</b>	Select an M3 API from the list then click Use. A numeric value 1 across an M3 API indicates that it is in use. When in use, it enables importing of agreements. You can not import agreements on an M3 API that is not in use.

- 4 Click OK to store the API reference in MEC database.

## Managing API Reference

Use this procedure to view the contents, modify the properties, or remove the API reference from the MEC database.

- 1 On Partner Admin Tool menu, select Manage > Communication > M3 API.
- 2 Move the mouse pointer over an API reference to display its contents.
- 3 You can perform any of the following:

Task	Steps
To edit	<ol style="list-style-type: none"><li>a Click Edit.</li><li>b On the Edit existing API reference window, modify the field values.  For API field definitions, refer to "<a href="#">API Fields Definition</a>" on page 57</li><li>c Click OK.</li></ol>
To delete	<ol style="list-style-type: none"><li>a Click Delete.</li><li>b At the prompts, click Yes.</li></ol>
To enable import	<ol style="list-style-type: none"><li>a Select an existing M3 API.</li><li>b Click Use.  A numeric value 1 under the column Import Use, across an M3 API indicates that it is enabled for import.</li></ol>

- 4 Close and exit the application.

## Company/ Division Override

When CONO/DIVI is defined in multiple places, evaluation shall be done in increasing order of priority, that is:

- user associated CONO / DIVI,
- API-Ref,
- Folder control properties,
- agreement control properties,
- Mapping

Where the top priority is "Mapping" and the least is "user associated CONO/DIVI". That is, if set, Mapping will override all the other location settings.

## FTP Script

- ["FTP Scripts overview" on page 61](#)
- ["FTP Script Fields Definition" on page 62](#)
- ["Creating FTP Script" on page 63](#)
- ["Deleting and Editing FTP Script" on page 63](#)
- ["Viewing and Exporting FTP Script" on page 63](#)

## FTP Scripts overview

FTP Scripts applies only to FTPScriptPollIn and FTPScript outgoing channels. Polling is always for a single file and FTP Scripts are uploaded in the Partner Admin Tool. These scripts support raw commands except for **PORT** with the addition of **GET** and **PUT** commands.

The Header record is appended with the following format to received messages in FTP Script channel:

1 - 6 = \$\$ADD

7 - 20 = Sender Reference

21 - 34 = Application Reference

35 – 54 = User Reference

55 - 80 = Service Reference

### Script file types and process order

There are two script file types, the single file and the sequence file. Single file contains only a single script. While Sequence File contains one or more comma delimited single script types.

Polling scripts are read as they are and there is no conversion of editable properties and manifest attributes. The class instances are interpreted as Send scripts. Entries in polling scripts can be edited, or deleted, even when used by an incoming FTP script channel. Although, if used by an outgoing channel, the scripts can neither be edited nor deleted.

### Supported Dynamic data types

FTP Scripts are uploaded on the Partner Admin Tool FTP Scripts page. It supports the following three dynamic data types:

## Manifest

Value: `%+MANIFEST:any manifest item+%`

For example:

`%+MANIFEST:ManifestConstants.UUID+%` or `%+MANIFEST:UUID+%`

## Class

Value: `%+CLASS:any class that implements IEnvelopeTemplateClass+%`

For example:

`%+CLASS:com.intentia.ec.server.envelope.BizTalkDate+%`

## Editable

Value: `%+EDITABLE:any property name+%`

For example:

`%+EDITABLE:URI+%`

The two other dynamic data types not supported are: `%ENCODING%` and `%THE_PAYLOAD%`

**Note:** The data types are not configurable if used for PollIn.

## FTP Script Fields Definition

Following is the FTPScriptPollIn definitions to use when creating, editing, deleting, and exporting FTPScript.

Property	Description
Name	Type a descriptive name.
Description	Type a brief description.
Encoding	Select an encoding to use. For example: UTF-8
Single Script Type	Select this option if you are using a single script file. Otherwise define a comma delimited script names.
FTP Script Type	Click to browse to the folder location of your script file to use.

## Creating FTP Script

Use this procedure to define the FTP script you will use.

- 1 On Partner Admin Tool menu, select Manage > FTP Scripts.
- 2 On FTP Scripts window, click New.
- 3 On create new FTP script window, update the fields data.  
For more information, see [FTP Script Fields Definition](#).
- 4 Click OK.

Your created script will be listed up on the FTP Scripts list.

## Deleting and Editing FTP Script

Use this procedure to modify the details or to delete an FTP Script.

- 1 On Partner Admin Tool menu, select Manage > FTP Scripts.
- 2 On the FTP Scripts window, select an FTP Script and perform any of the following:

Task	Steps
To edit	<ol style="list-style-type: none"><li>a Click Edit.</li><li>b On the Edit existing FTP script object window, modify the value fields to your specification.</li><li>c Click OK.</li></ol>
To delete	<ol style="list-style-type: none"><li>a Click Delete.</li><li>b At the prompts, click Yes.</li></ol>

- 3 Click Close and exit.

## Viewing and Exporting FTP Script

Use this procedure to view and then export FTP Scripts from MEC.

- 1 On Partner Admin Tool menu, select Manage > FTP Scripts.

- 2 On the FTP Scripts window, select a template.
- 3 Click View Template. The contents of the script is displayed.
- 4 Click Export Template.
- 5 On the Windows Save As dialog, browse to the folder location where you want to export the script.
- 6 Click Save and exit.

## Web Services

- ["Web Service Overview" on page 64](#)
- ["Web Service Definitions Fields" on page 65](#)
- ["Uploading and viewing a WSDL file" on page 66](#)
- ["Modifying Web Service Definitions" on page 67](#)
- ["Exporting Web Service Definitions" on page 68](#)

## Web Service Overview

Web Service is a self-contained, self-describing application component that communicates with other applications using open protocols. It allows different applications and platforms to link their data and reuse the application-components.

MEC can act as a web service server. A MEC administrator defines the server settings and web service definitions through Partner Administrator Tool then the payloads of the received web service invocations are extracted and then sent for detection. In order to generate an appropriate response, add Send Web Service process step in the process flow.

A basic Web Service platform consists of XML+HTTP. XML is the part that is used to code and decode data. SOAP encodes the information in Web Service request and response messages before sending them over HTTP.

A Web Service Definition Language (WSDL) file contains an XML coding of network services as collections of communication endpoints. This schema is a documentation of distributed systems and required details in applications communication which can be used in message exchange.

## Security

Web Service security settings and parameters are configurable in MEC. For message-level security, we use XML and Web Services Security (XWSSecurity) where the information is contained within the SOAP messages or attachments. Applications are secured by signing, verifying, encrypting, or decrypting



parts of SOAP messages and attachments. The message sender can associate security tokens with the message allowing for security claims, like sender username and password.

## Directory structure

You can upload multiple WSDL files and schemas in one or several different directories per web service definition. Although, a WSDL file can only reuse schemas contained within the same main directory where the WSDL belongs.

If your main WSDL includes tag references to other files, ensure that the schema referenced and WSDL files are contained within the same directory.

## Limitations

- A WSDL file or a schema can only reference schemas contained within the same root directory where the main WSDL belongs, or within subdirectories of the root directory.
- Referencing other files past the root directory of the main WSDL using `../.. /someSchema.xsd` in an import tag is not supported.

## Related topics

For more information, see the following topics:

- Web Service Definitions fields, [Web Service Definitions Fields](#).
- Process Flow step, see Send Web Service
- Channel Protocols, see WebServiceSynchOut

## Web Service Definitions Fields

The table here describes the fields in Web Service Definitions for Receive Channel objects.

In Partner Admin tool menu, select Manage > Web Service Definitions. Create a new, or select an existing WSDL file to update.

In the **Basic** pane, create a new, or select a Web Service Definitions to edit.

**Note:** Except for the name and description fields, all the rest are auto-filled with data based on your uploaded wsdl file. When editing these open text fields, ensure that the information corresponds to your selected wsdl file details.

After creating a new or editing an existing Definition, select a receive channel to use from the list of available channels in the **Channel Assignments** pane.

Field	Description
Name	Type a descriptive name for this definition.
Description	Optional. Type a brief description for this WSD.
Service Namespace	WSDL Service namespace to use.
Service Name	Required WSDL Service name. For example: <pre>&lt;wsdl:service name="PartStoreRequiredService"&gt;   &lt;wsdl:port name="PartStoreRequiredServiceSOAP11port_ http"     binding="ns5:PartStoreRequiredServiceSOAP11Binding"&gt;      &lt;soap:address       location="http://midtierws.cat.com:80/webapp/ services/PartStoreRequiredService" /&gt;     &lt;/wsdl:port&gt;   &lt;/wsdl:service&gt;</pre>
Port Namespace	WSDL Port namespace. an abstract set of operations supported by one or more endpoints.
Port name	Required WSDL port name, for example, see this excerpt from Service Name above. <pre>&lt;wsdl:port name="PartStoreRequiredServiceSOAP11port_ http"   binding="ns5:PartStoreRequiredServiceSOAP11Binding"&gt;</pre>
Definition file	You can import new, or edit existing, definition files. For more information, see <a href="#">Exporting Web Service Definitions</a> and <a href="#">Modifying Web Service Definitions</a>

## Uploading and viewing a WSDL file

Use this procedure to upload on PA tool a predefined WSDL file.

**Note:** Uploads are limited to \*.wsdl file types only.

When uploading schema definitions, a copy of the source schema is stored in the DB.

For more information on WSDL directory structure, see "[Directory structure](#)" on page 65

### To upload

- 1 On Partner Admin Tool menu, select Manage > Web Service Definitions.
- 2 Click **New** and type a name for this definition. A Description is optional.
- 3 Click **Select Definition** and browse to the folder location of the main WSDL file to upload.

**Note:** Except for the name and description fields, all the rest are auto-filled with data based on your uploaded .wsdl file. When editing these open text fields, ensure that the information corresponds to your selected .wsdl file details.

- 4 Click OK.
- 5 Verify that the new definition is listed on the Web Service Definitions window.

### To view

- 1 Select a definition to use.
- 2 Click **View Definition**. The contents of your selected WSDL file are displayed on a new window.
- 3 Close the window when finished.

## Modifying Web Service Definitions

Use this procedure to modify the fields or contents of a definition file. On PA menu, select Manage > Web Service Definitions and complete the tasks here:

### To edit the fields

- 1 Select a web service definition to use and click **Edit**.
- 2 On Edit existing Web Service definition object window, modify the field values based on the corresponding WSDL file definition.
- 3 Assign a channel to use for this definition.
- 4 Click OK when finished.

## To update the contents

**Important:** Upload has no undo function. Ensure that you have a back up of the current definition before you run Update. Use Export to create a back up.

If deleted, explicitly update the current with the original file source to recover the original file format.

- 1 Select a definition to use and click **Edit**.
- 2 Click **List Schemas**. A list of available schemas is displayed.

**Note:** The list displays copies of schemas or secondary WSDLs referenced by the main WSDL.

- 3 Select a schema definition file that you want to change. Click **Update**.
- 4 Browse to the folder location of a schema definition file to update this with.

When you click **Open**, the contents of your selected schema file in step 4, will overwrite the contents of your selected schema file in step 3.

- 5 Close and exit when finished.

## Exporting Web Service Definitions

Use this procedure to create a copy of WSDL file definitions in MEC. On PA menu, select Manage > Web Service Definitions and complete the tasks here:

### To export a definition file

- 1 Select a web service definition to use.
- 2 Click **Export Definition** and browse to the folder location where to save this definition.
- 3 Click Save when finished.

### To export a schema

- 1 Select a web service definition to use.
- 2 Click Edit.
- 3 Click List Schemas.
- 4 Select a schema from the list and click Export.
- 5 Browser to the folder location where to save this schema.

**6** Close and exit when finished.

**Note:** After this procedure, you must create a Send Process (configured WSSyncOut) channel for Agreements coming from **WebServiceSyncIn** channel.

For more information, see the chapter on Communication Channels.

## DAF Archiving

- ["DAF Overview" on page 69](#)
- ["DAF Fields Definition" on page 70](#)
- ["MIME Type" on page 71](#)
- ["Creating DAF Connection and Basic Data" on page 71](#)

## DAF Overview

Different applications can be used to automate the conversion of original documents (hard copy, e-mail, facsimile) to another format (xml, image file format, edi or pdf file) ready for archiving. Document archiving is a legal requirement and it allow users to revisit the document for later use or for processing.

The Document Archive Foundation (DAF) is a tool used for storing, sharing, reusing, and archiving digital content. It helps reduce the time and cost of document search. With DAF, customers can store business documents connected to Infor M3 objects and make them available in the Infor user interface.

MEC provides an integration workflow (M3 API/mappings) and an archiving solution through DAF for document or image files. In this integration workflow, MEC can process the document or image files and retrieve the digitized images for DAF archive process. The corresponding document or image linked to the received message is automatically archived in DAF and a copy of the document remains in the sender application. The archived file attributes can be updated in MEC DAF.

Here is an example procedure in integration workflow:

**1** Create DAF connections.

Do this from Manage > Communications.

**2** Create the Basic Data for the connection.

Do this from Manage > DAF Basic Data.

**3** Use DAF Archive process step.

Do this from Manage > Agreement View tab > Processes tab.

## Release lock on received file

Document files after a DAF Archive process remains locked in MEC. When a document is locked it is not accessible, not possible to delete, and MEC cannot reuse it. Now, after a DAF Archive process, MEC releases its lock on a document file and make it accessible to other applications, including MEC.

## Option to remove document/image file

An option to remove the document file from the sender application is now added.

In Configuration for DAF Archive window, select "Update Only" and "Delete the document file". The location of the document files defined in 'Specify File Location' or 'File location Xpath' fields will then be cleared.

## Option to achive self(.rcv)

An option to archive self(.rcv) is now added.

In Configuration for DAF Archive window, leave blank the 'Specify File Location' or 'File location Xpath' fields to enable DAF to archive itself and not require for a link to the corresponding document or image.

## MIME type

The MIME type of the message or document file for DAF archiving is automatically detected through its file extension as listed in the table below.

Received file extension	MIME type
<b>.tif</b>	image/tiff
<b>.png</b>	image/png
<b>.jpeg</b>	image/jpeg
<b>.pdf</b>	application/pdf
All others file extensions, for example <b>.txt</b> or <b>.rcv</b>	text/xml

## DAF Fields Definition

The following table details the DAF (Document Archive Foundation) fields definition. These fields are used for connecting to DAF.

Field	Definition
Name	Type a descriptive name for this connection.
User Name	Type a DAF user name for this connection.
Rest URL	Type the HTTP URL where the DAF application is located.
Auth Mode	Select the authentication mode which the DAF service is configured with.

## MIME Type

The MIME type of the message or document for archiving in DAF is automatically detected through its file extension as detailed in the table below.

Received file extension	MIME type
.tif	image/tiff
.png	image/png
.jpeg	image/jpeg
.pdf	application/pdf
All others file extensions, for example .txt or .rcv	text/xml

## Creating DAF Connection and Basic Data

Use these procedures to create a DAF connection and the Basic Data needed before you can use "DAF Archive" process step in your message.

**Important:** MEC now validates the connection to DAF(DB) and then reuses a working connection. If the connection is not working, MEC will automatically close that connection and then get a new one to avoid connection failure when trying to send data to DAF.

### ☐ Create DAF Connection

**Before you start** Ensure that DAF Environment is loaded and running.

- \_\_\_1 On Partner Admin Tool menu, click Manage > Communications.
- \_\_\_2 Click on DAF (Document Archive Foundation) tab and click New.
- \_\_\_3 On Create New DAF Connection window, update the field values.  
For more information, see "[DAF Fields Definition](#)" on page 70.
- \_\_\_4 Click OK.
- \_\_\_5 Verify that your created DAF connection is now listed in the DAF (Document Archive Foundation) tab.

### **Create DAF Basic Data**

**Before you start** Ensure that a DAF connection is created in Manage > Communications.

- \_\_\_1 On Partner Admin Tool menu, click Manage > DAF Basic Data.
- \_\_\_2 Click New.
- \_\_\_3 On Create new DAF Basic Data window, consider the following fields:

<b>DAF Connections</b>	Select a connection from the list.
<b>Document Type</b>	Select a document type from the list.
<b>Relation(File type)</b>	Select a relation from the list. This is a dynamic list corresponding to your selected Document Type.
- \_\_\_4 Click OK.
- \_\_\_5 Update the field values for the **XPath** and **Default Value**.
- \_\_\_6 Click OK.
- \_\_\_7 Verify that your created DAF Basic Data is now listed in the DAF Basic Data tab.
- \_\_\_8 Perform any of the following:

Task	Steps
To Edit	<ol style="list-style-type: none"><li><b>a</b> Select a DAF Basic data from the table list.</li><li><b>b</b> Click Edit. The Edit existing DAF Basic Data window is displayed.</li><li><b>c</b> Modify or update the field values and click OK when completed.</li></ol>

---



Task	Steps
To Delete	<ol style="list-style-type: none"> <li><b>a</b> Select a DAF Basic data from the table list.</li> <li><b>b</b> Click Delete.</li> </ol> <p><b>Important:</b> Delete has no undo option. Ensure that the DAF basic data you are about to delete is no longer needed.</p> <ol style="list-style-type: none"> <li><b>c</b> On the delete confirmation prompt, click YES.</li> </ol>
To Check for updates...	Click on this to check for updates and verify that the Basic data is properly synchronized.

## Databases

- ["Database Reference Overview" on page 73](#)
- ["Setting Transaction Isolation Level" on page 75](#)

## Database Reference Overview

When defining database connection in Partner Admin the configuration must be the same between Partner Admin and Mapping Manager. The Database structure or metadata (tables, columns, stored procedures) must be the same for design time (creating), generation, and runtime (executing).

In Mapping Manager, published XML mappings may contain both M3 API and database calls. In Partner Admin, before you use a published mapping, you can check whether it uses M3API or database reference or not. From the menu, select Manage > XML Mappings. Published mappings are listed with the corresponding boolean value "true" or "false" under the column M3API or database calls.

### Transaction isolation level overview

Transaction isolation levels are used to define the type of protection acquired on read operations. The different isolation levels provide different controls on the concurrency effects of modifications made by other transactions.

To set the transaction isolation level for your database connection, use the Partner Admin tool. By default, MEC has set the transaction isolation level to "UNKNOWN\_TRANSACTION\_ISOLATION".

You can only set one transaction isolation level for your database connection at a time and it will remain in effect until you change the isolation level. You can change a transaction from one isolation level to another during the transaction. When changed, the type of protection on read operation will depend on the rules of the isolation level at the time the resource was read, for example:

- Before the change - protection rules of the previous level applies.
- After the change - protection rules of the new level applies.

In Mapping Manager, each mapping uses only one Database Connection object that is used throughout the life of the mapping. Database transaction within a mapping is defined when you start a database call, and a transaction completes when you invoke database commit or rollback. Except if you set isolation levels within stored procedures.

## Transaction isolation level options

Following is a list of transaction isolation level options:

Transaction Isolation level	Details
UNKNOWN_TRANSACTION_ISOLATION	MEC Database default level.
TRANSACTION_NONE	No rules of protection applied.
TRANSACTION_READ_COMMITTED	<ul style="list-style-type: none"><li>• Statements cannot read data that has been modified but not committed by other transactions. This level prevents dirty reads.</li></ul>
TRANSACTION_READ_UNCOMMITTED	<p>This level does not prevent other transactions from modifying data read by the current transaction.</p> <ul style="list-style-type: none"><li>• Statements can read uncommitted modifications, also called dirty reads. Data values can change and rows can appear or disappear in the data set before the transaction completes.</li><li>• This is the least restrictive isolation level where transactions are isolated to ensure that corrupted data is not read.</li></ul>
TRANSACTION_REPEATABLE_READ	<p>This level prevents other transactions from modifying any rows that have been read by the current transaction until the current transaction completes.</p> <ul style="list-style-type: none"><li>• Statements cannot read data that has been modified, but not committed by other transactions</li></ul>

Transaction Isolation level	Details
TRANSACTION_SERIALIZABLE	<p>This is the highest isolation level where transactions are totally isolated from one another.</p> <ul style="list-style-type: none"> <li>• Statements cannot read data that has been modified, but not yet committed by other transactions.</li> <li>• No other transactions can modify data that has been read by the current transaction until the current transaction completes.</li> <li>• Other transactions cannot insert new rows with key values within the range of keys read by any statements in the current transaction until the current transaction completes.</li> </ul>

## Setting Transaction Isolation Level

**Before you start** Create a mapping with database calls and publish the mapping.

**1** On Partner Admin Tool menu, select Manage > Communication.

**2** Go to Databases tab.

A table list of your defined database connection is displayed.

**3** Select a database connection to set the transaction isolation level.

**4** Click Edit.

The Edit existing database item window is displayed.

**5** On the **Tr.Isolation** field, select the isolation level for this database connection.

By default, MEC has set the transaction isolation level to "UNKNOWN\_TRANSACTION\_ISOLATION".  
["Database Reference Overview"](#) on page 73

**Tip:** You can click Test, to verify the connection to your selected database item.

**6** Click OK.

## MECEventHub

- ["EventHub Overview"](#) on page 76

- ["EventData Messages Overview" on page 76](#)
- ["Adding a Subscription" on page 78](#)
- ["Deleting and Editing Subscriptions" on page 78](#)

## EventHub Overview

The Event Hub is a generic Grid extension for sending events between Infor applications. It is a publisher-subscriber application framework that allows publisher applications to expose historical data to subscriber applications that are interested in receiving this data.

In the EventHub page there are two views for all subscriptions. The Subscriber view that shows the subscriptions as created by the subscribers, and the Publisher view that shows the subscriptions as propagated to the publishers.

MEC can act as a subscriber application through the MECEventHubSubscriber channel. You can create multiple MECEventHubSubscriber channels.

**Important:** When you subscribe to MECEventHubSubscriber channels, ensure that EventHub and MEC are running on the same grid and that each channel has one subscription, at the least.

**Note:** MEC version 9.2.0.0 is compatible with Event Hub v1.0.4.

MEC version 10.3.0.0 is compatible with Event Hub starting from v1.3.1 to the latest.

Definition of terms:

- **Event-** A discreet unit of historical data shared by an application that may be relevant to other applications or "Subscriber".
- **Publisher-** An application that need to publish events.
- **Subscriber-** An application that need to receive events that are published by another application or "Publisher".
- **Subscription-** A predicate indicating that a subscriber is to receive a particular event.

## EventData Messages Overview

MEC can act as a subscriber application and will receive **EventData** messages through the MECEventHubSubscriber channel.

The **EventData** message will be in an XML format following the schema below:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:jxb="http://java.sun.com/xml/ns/jaxb"
  jxb:version="2.0">
```

```

<xsd:element name="EventData" type="EventData" />
<xsd:complexType name="EventData" />
<xsd:sequence>
  <xsd:element ref="Publisher" />
  <xsd:element ref="DocumentName" />
  <xsd:element ref="Operation" />
  <xsd:element ref="TrackingId" />
  <xsd:element ref="SentTimestamp" />

  <xsd:element name="Document" type="DocumentType" />
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="DocumentType" />
<xsd:sequence>
  <xsd:element name="ElementData" type="ElementDataType" minOccurs="0" maxOccurs="unbounded" />
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ElementDataType" />
<xsd:sequence>
  <xsd:element ref="Name" />
  <xsd:element ref="Value" minOccurs="0" />
  <xsd:element ref="OldValue" minOccurs="0" />
</xsd:sequence>
</xsd:complexType>
<xsd:element name="Publisher" type="xsd:string" />
<xsd:element name="DocumentName" type="xsd:string" />
<xsd:element name="Operation" type="xsd:string" />

<xsd:element name="TrackingId" type="UUID" />
<xsd:element name="SentTimestamp" type="xsd:dateTime" />
<xsd:element name="Name" type="xsd:string" />
<xsd:element name="Value" type="xsd:string" />
<xsd:element name="OldValue" type="xsd:string" />
<xsd:simpleType name="UUID">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}" />
  </xsd:restriction>
</xsd:simpleType>
</xsd:schema>

```

Notice that you will have problem distinguishing the value from **ElementData** element because the node/element is unbounded, for example, **/EventData/Document/ElementData/Name**.

For MEC to be able to differentiate or detect elements on their ordered or numbered sequence, specify the two XPath predicates below as XML targets. Predicates are used to find a specific node or a node that contains a specific value. Predicates are always embedded in square brackets.

The following table lists the path expressions with predicates and its corresponding result. This applies only to XPath options of all XML targets.

Path Expression	Result
<b>/bookstore/book[1]</b>	Selects the first book element that is the child of the bookstore element.
<b>/bookstore/book[last()]</b>	Selects the last book element that is the child of the bookstore element.

**Note:** The element sequence for **ElementData** is fixed per document name. Check the publisher implementation for correct element sequence.

## Adding a Subscription

Perform this procedure to add a subscription for MECEventHubSubscriber channel in MEC.

**Important:** After you create a subscription, edit and assign it to a MECEventHubSubscriber channel. For more information, see "[Deleting and Editing Subscriptions](#)" on page 78.

- 1 On Partner Admin Tool menu, select Manage > EventHub Subscriptions > New.
- 2 On create new EventHub subscription window, consider the following fields:

<b>Name</b>	Type a descriptive name.
<b>Description</b>	Type a brief description.
<b>Subscription</b>	Type a predicate indicating that a subscriber is to receive a particular event.  Subscription format:  <code>&lt;Publisher&gt;":"&lt;DocumentName&gt;[":"[&lt;operation char&gt;]":["("true" "false")"][":"[&lt;priority&gt;]]]</code>  For example: <code>M3:MITMAS:C::P1</code>

Variable	Definition and example
<code>&lt;operation char&gt;</code>	<code>:C U D S X F Q R</code>
<code>&lt;priority&gt;</code>	<code>P1 P2 P3</code> Event priorities take place in the Event Hub's out queue for the MEC subscriber.
<code>[("true"   "false")]</code>	The <code>[("true"   "false")]</code> part for guaranteed event sequence will soon be implemented.

**Note:** This is currently not supported in EventHub.

- 3 Click OK.
- 4 Verify that the EventHub Subscription you created is now in the list, then close the window.
- 5 Close the window and exit.

## Deleting and Editing Subscriptions

Use this procedure to modify the details or to delete a MECEventHubSubscriber channel.

- 1 On Partner Admin Tool menu, select Manage > EventHub Subscriptions.
- 2 On the EventHub Subscriptions window, select a MECEventHubSubscriber then perform any of the following:

Task	Steps
To edit	<ol style="list-style-type: none"> <li>a Click Edit.</li> <li>b On the Edit existing EventHub subscription window, modify the field values to your specification.  You can assign this Subscription to their respective MECEventHubSubscriber channels.</li> <li>c Click OK.</li> </ol>
To delete	<ol style="list-style-type: none"> <li>a Click Delete.</li> <li>b At the prompts, click Yes.</li> </ol>

- 3 Close the window and exit.

## MEC Schedules

- ["Channel and Service Scheduling Overview" on page 79](#)

## Channel and Service Scheduling Overview

### Channels

The channel states are controlled by MEC, and in earlier versions when you restart MEC all the channels are restarted along with MEC. Now, the channel schedules survive MEC restart. That is, when you invoke a MEC restart, the channels scheduled to pause remain on pause. To supersede the scheduled channel pause, manually invoke a channel pause or resume.

### Services

In earlier versions, MEC does not maintain the status of the services and it does not support offline or online scheduling. That is, when MEC Windows Service is stopped or started the schedules do not change. Now, when you restart MECServer Service, all services scheduled to be **"ONLINE"** will be

started, and all services scheduled to be "**OFFLINE**" will be stopped, disregarding the current Windows Service state. To supersede the scheduled Service stop/start, manually invoke a start/stop for that service.

## Sample scenarios

Scenario	Setting/Behavior
Offline schedule is every Saturday 1PM and runs for 1 minute (this is set in PA)	<ul style="list-style-type: none"><li>• Default DiskIn - set to Pause/Resume</li><li>• Service ABC - set to Stop/Start</li></ul>
Case 1 (Normal scenario): MEC is running	<ul style="list-style-type: none"><li>• Default DiskIn - will Pause 1PM and Resume 1:01PM</li><li>• Service ABC - will Stop 1PM and Resume 1:01PM</li></ul>
Case 2: MEC is shutdown at 12noon and started at 2PM manually	<ul style="list-style-type: none"><li>• Default DiskIn - is started (default behavior)</li><li>• Service ABC - will be started by MEC (because based on schedule, service should be started)</li></ul>
Case 3: MEC is shutdown 10 seconds after 1PM and is manually started at 2PM.	<ul style="list-style-type: none"><li>• Default DiskIn - is Paused just before the shutdown because of the schedule<ul style="list-style-type: none"><li>• On startup Default DiskIn will be started (default behavior)</li></ul></li><li>• Service ABC - will be stopped just before shutdown because of the schedule<ul style="list-style-type: none"><li>• On startup Service ABC will be started by MEC</li></ul></li></ul>
Case 4: MEC is shutdown at 12noon and started 30 seconds after 1PM	<ul style="list-style-type: none"><li>• On startup Default DiskIn - will not start because based on schedule it is still offline<ul style="list-style-type: none"><li>• Default DiskIn will start at 1:01PM on schedule</li></ul></li><li>• On startup Service ABC - will be stopped because based on schedule it is still offline<ul style="list-style-type: none"><li>• Service ABC will start at 1:01PM on schedule</li></ul></li></ul>
Case 5: Default Disk is (Manually) Paused and Service ABC is (Manually) stopped at 12noon	<ul style="list-style-type: none"><li>• Default DiskIn - is set to Resume at 1:01PM</li><li>• Service ABC - is set to Start at 1:01PM</li></ul> <p><b>Important:</b> MEC does not keep track of manual pause/stop.</p>



# Data Translator

- ["Data Translator Overview" on page 81](#)
- ["Using MEC Data Translator" on page 83](#)
- ["Creating Applications Object" on page 85](#)
- ["Creating Partner Objects" on page 86](#)
- ["Creating Translation Data Object" on page 87](#)

## Data Translator Overview

String data going to (outbound), or received from (inbound) a partner can be translated or converted from one form to another. It is called MEC data translation. It applies only to string data and not to the numeric values carried by the string data. For example, the received string data "KG" can be translated to "kg".

MEC data translator uses a class called `MECDataTranslator` that is defined in design time and, executed and monitored in runtime. Here are the basic knowledge required to be able to use `MECDataTranslator`:

- Java knowledge to be able to define data translations in Mapping Manager.
- Grid and grid application to control and monitor `MECDataTranslator`.

### Design time

The required details for data translation are defined in Mapping Manager and Partner Admin tools.

- In MEC Mapping Manager:
  - 1 Create a mapping with Java functions using `MECDataTranslator` class in your mappings.
  - 2 Save and Publish your mapping.

- In MEC Partner Admin:

**Note:** If you do not have a mapping with `MECDataTranslator` class to start with, an error message is displayed. When this happens, click OK to proceed. The error prompt will stop only when a defined `MECDataTranslator` class is detected.

- Define an Application for Data Translation.
- Define a Partner to use this translation.
- Define the Translation Data.

## Runtime

After the data translation design is completed, you can now control and monitor the MECDataTranslator.

- In Grid MEC application: Execute and remotely control MECDataTranslator.
- In MEC web Monitor UI: Monitor the progress of MECDataTranslator.

**Note:** Translations have expiry date. By default, you can add translations as an adHoc application property in grid MEC application. For more information on adHoc property settings, see the *M3 Enterprise Collaborator Administration Guide*.

## How data is translated in MECDataTranslator

Briefly, here is how MECDataTranslator checks for data to translate:

- 1 First, it will try to find a translation value for the given partner and application context.
- 2 If there is no match, it will try to find a translation for blank (generic) partner and top application context.
- 3 If there is no match, it will use the non-translated original value.

For more information, see "[Using MEC Data Translator](#)" on page 83.

## Data translation remote control

In Grid MEC application, you can remotely control loading of data for translation through the Utilities page.

Select the grid node where your MEC Application is installed. Right-click ION Grid <version> > Application > Manage Application > Utilities.

In the DataTranslator Remote Control page, here are the available options:

- Reload next time - select this to load data when the data translator or MECDataTranslator utilities are used. For details of load line-up, click Check Properties.
- Reload now - select this to load translation data based on Partner Admin updates. Loaded data are automatically lined up according to its order or execution. You can clear the cache when completed and on the next runtime, data for translation will be loaded.
- Check properties - select this to display the details and load line-up of data translation.
- Dump translation data - select this to extract the cache values of your data translation and save it on a csv file type. After saving, you will be prompted to view the downloaded file, use notepad to view its contents.

## Using MEC Data Translator

Use these procedures to define the data translation details in MEC.

**Note:** Translations have expiry date. By default, you can add translations as an adHoc application property in grid MEC application. For more information on adHoc property settings, see the *M3 Enterprise Collaborator Administration Guide*.

### ☐ Complete the following tasks in Mapping Manager

\_\_\_1 Create a mapping with Java functions. Use MECDataTranslator class in your mappings.  
See the following screen shot of example mapping with Java functions:

\_\_\_2 Save and Publish your mapping.

For more information, see *M3 Enterprise Collaborator Mapping Manager User Guide*.

### ☐ In Partner Admin, set up an Agreement using your published mapping

**Note:** If you do not have a mapping with MECDataTranslator class to start with, an error message is displayed. When this happens, click OK to proceed. The error prompt will stop only when a defined MECDataTranslator class is detected.

### ☐ Define an Application

\_\_\_1 Select Manage > Data Translation > Applications tab.

\_\_\_2 Click New and create the following record:

Field	Value
Logical Id	<code>lid://infor.m3be.test</code>
Tenant ID	330 For Logical ID: <code>lid://infor.m3be.test</code>
Accounting Entity ID	<b>AAA</b> For Logical ID: <code>lid://infor.m3be.test</code> and Tenant ID: 330

Field	Value
Location ID	<b>F01-W01</b>  For Logical ID: <b>lid://infor.m3be.test</b> and Tenant ID: <b>330</b> and Accounting ID: <b>AAA</b>

### ☐ Define the Partners

\_\_\_1 Select Manage > Data Translation > Partners tab.

\_\_\_2 Click New and create the following Partner record:

**Note:** The Partner values must be cross-referenced with the Group in Partner Agreements View Partner Admin Tool

**Name** Type a descriptive name for this Partner.

**Description** Type a brief description for this Partner.

\_\_\_3 Click OK.

Your new Partner Object details is listed in the table.

### ☐ Define the Partners

\_\_\_1 Select Manage > Data Translation > Partners tab.

\_\_\_2 Click New and create the following Partner record:

**Note:** The Partner values must be cross-referenced with the Group in Partner Agreements View Partner Admin Tool

**Name** Type a descriptive name for this Partner.

**Description** Type a brief description for this Partner.

\_\_\_3 Click OK.

Your new Partner Object details is listed in the table.

### ☐ Reload MECDataTranslator

\_\_\_ In Grid MEC application, go to Utilities page to reload MECData Translator.

Back in Partner Admin, see the fields populated with details from your published mapping.

**Note:** If your mapping has no detailed application type values, Partner Admin will follow this hierarchy when looking for values.

- Will look up in "Location\_ID"
- If not found, will look up in "Accounting Entity ID" ,
- If not found, will look up in "Tenant ID" ,
- If not found, will look up in "Logical ID" ,

If any of the application\_ID above is found, then it will look for a matching partner(partner\_id), Else, it will look into <default partner>, else it will return the same string.

## Creating Applications Object

Use these procedures to manage your Applications Objects in **Data Translation**. For more information, see "[Data Translator Overview](#)" on page 81.

**Before you start** You should have a mapping with data translation.

### ☐ Create Applications Object

- 1 On Data Translations window, go to the Applications tab.
- 2 Click New.  
The Create new applications context object window is displayed.
- 3 Select an Application context Type to use:

**Note:** Additional fields, and available definitions created for those fields, will become available depending on your selected Application context type.

Type	Description
LogicalId	<p>This is the Logical ID for the application. You must use a variable or a hard coded string.</p> <p>For example: <code>lid://infor.m3be.test</code></p> <p><b>Important:</b> This Id must match the Logical Id used in a Mapping.</p>
TenantId	For example: 330
AccountingEntityId	For example: AAA

Type	Description
LocationId	For example: F01-W01

\_\_\_4 On the Description field, type a brief description for this application context.

\_\_\_5 Click OK.

Your new Partner Object, with details, is listed in the table. Repeat the process until all application context types are completed.

### ☐ **Edit Applications Object**

\_\_\_1 Select an Application Context Object and click **Edit**.

The Edit existing application context object window is displayed.

\_\_\_2 Update the fields.

\_\_\_3 Click OK.

### ☐ **Delete Applications Object**

#### **Important:**

- Delete has no undo function. Verify the details of the Application Object before you proceed with delete.
- You cannot delete an Application Object (parent) if there is already a corresponding Translation Data (dependent) linked to it.

\_\_\_1 Select an Application Object and click **Delete**.

\_\_\_2 At the delete confirmation prompt, click Yes.

## Creating Partner Objects

Use these procedures to manage your Partner Objects in Data Translation. For more information, see "[Data Translator Overview](#)" on page 81.

**Note:** If you do not have a mapping with MEC Data Translation, an error message is displayed. Click OK and proceed.

### ❑ Create Partner Object

- \_\_\_1 On Data Translations window, go to the Partners tab.
- \_\_\_2 Click New.
- \_\_\_3 On Create new partner window, consider the following fields:

**Note:** The Partner values must be cross-referenced with the Group in Partner Agreements View Partner Admin Tool

**Name** Type a descriptive name for this partner.

**Description** Type a brief description.

- \_\_\_4 Click OK.  
Your new Partner Object, with details, is listed in the table.

### ❑ Edit Partner Object

- \_\_\_1 Select a Partner Object and click **Edit**.  
The Edit existing partner window is displayed.
- \_\_\_2 Update the Partner Object fields.
- \_\_\_3 Click OK.

### ❑ Delete Partner Object

**Important:**

- Delete has no undo function. Verify the details of the Partner Object before you proceed with delete.
- You cannot delete a Partner if there is already a corresponding Translation Data linked to it.

- \_\_\_1 Select a Partner Object and click **Delete**.
- \_\_\_2 At the delete confirmation prompt, click Yes.

## Creating Translation Data Object

Use these procedures to manage your Translation Data Objects in **Data Translation**. For more information, see "[Data Translator Overview](#)" on page 81.

**Before you start** You should have a mapping with data translation.

## ☐ Create Translation Data Object

\_\_\_1 On Data Translations window, go to the Translation Data tab.

The Data Translation window is displayed.

\_\_\_2 Consider the following fields:

<b>Name</b>	Select a Mapping with DataTranslation.
-------------	--

<b>Version</b>	Select a version to use for this Mapping.
----------------	---

\_\_\_3 In the XML Mapping Grid, select the Mapping.

\_\_\_4 In Application Object Grid, select any Logical Id.

\_\_\_5 In the Translation Object Grid, select any Message Standard.

This selection enables the New button.

\_\_\_6 Click New.

The Create new translation data window is displayed.

\_\_\_7 Consider the following fields:

<b>Partner</b>	Select a Partner, for example, <i>PartnerInBound</i>
----------------	--

**Note:** If you select an empty Partner, it will be considered as an Agreement wide configuration.

<b>Application value</b>	Type a string value.
--------------------------	----------------------

<b>Partner value</b>	Type a string value.
----------------------	----------------------

**Note:** This value must correspond with the declared Application value in the Mapping that you will use for Data Translation.

<b>Description</b>	Type brief description for this Translation Data.
--------------------	---

\_\_\_8 Click OK.

Your Translation Data Object is created and the Partner Object is displayed in Partners window.

## ☐ Edit Translation Data Object

\_\_\_1 In XML Mapping Grid, select a Mapping.

\_\_\_2 In Application Object Grid, select any Logical Id.

\_\_\_3 In the Translation Object Grid, select any Message Standard.

This will populate the Translation Data Grid for you to be able to select a Translation data.



- \_\_\_4 Click **Edit**.  
The Edit existing Partner Object window is displayed.
- \_\_\_5 Update the fields: Partner, Application value, Partner value, and Description.
- \_\_\_6 Click OK.

#### ☐ **Delete Translation Data Object**

**Important:** Delete has no undo function. Verify the details of the Application Object before you proceed with delete.

- \_\_\_1 In XML Mapping Grid, select a Mapping.
- \_\_\_2 In Application Object Grid, select any Logical Id.
- \_\_\_3 In the Translation Object Grid, select any Message Standard.
- \_\_\_4 In Translation Data Grid, select a Translation data.
- \_\_\_5 Click **Delete**.
- \_\_\_6 At the delete confirmation prompt, click Yes.

- ["Detections Overview" on page 90](#)
- ["Detections" on page 91](#)
- ["Target and Target Groups Overview" on page 94](#)
- ["XML Targets" on page 95](#)
- ["XML Target Group" on page 99](#)
- ["Flat File Targets" on page 100](#)
- ["Flat File Target Group" on page 102](#)
- ["Moving Targets Between Areas" on page 104](#)
- ["Performing Sanity Check on Target Groups" on page 104](#)
- ["Target Groups and Shadowing" on page 104](#)

## Detections Overview

Detections allow you to set the flexible message detection of MEC. There are several important objects that makes up the detection part. One of which is the Message Detection. It defines the kind of data that MEC can detect. By default, only XML and flat file messages can be detected.

Detections contain one or more target groups. Each target group contains one or more targets. A target points out a unique path to the data in the receiving message.

## Detections Order

When MEC receives a message, it starts to detect the message according to the detections and target groups in the detection order list. The detection order list is processed from top to bottom. For example: MEC will start to look at the received message and see if it fits the MBM target group in the XML detection. If it does not fit, MEC continues to see if the message fits the MVX\_S0\_R0 target group and so on, until it finds a matching target group for the message. If a matching target group is not found, the message fails and the status is set to "detection failed" for the message.

In the area for Unused Targets, all the detections and the defined target groups for that detection are shown.

In the area for Targets for Selected Groups, all the detections and target groups that you selected from the Unused Target area are shown. MEC uses this detection order in the Processes tab and in the Detection step.

**Important:** When you create your target groups and place them in the detection order, the problem of shadowing can arise.

When creating message agreements ensure that all field definitions are completed. If only the "Name" field was provided, an exception error may occur resulting to the message staying in "Detect" state. If this error occurs, go back to the agreement and complete the field definitions

**Need More Details? Check out the following concepts:**

- For a complete understanding of the complications of shadowing, see "Target Groups and Shadowing"
- For more information on Message Detection, see "[Creating New Agreement](#)" on page 135

## Detections

- "[Uses of Detections](#)" on page 91
- "[Adding a new Detection](#)" on page 92
- "[Executing XML Detection Using the MBM Target Group](#)" on page 92
- "[Executing XML Detection Using the MVX Target Group](#)" on page 92
- "[Executing Flat Detection Using Target Groups](#)" on page 94

## Uses of Detections

When MEC receives a message, it starts to detect the message according to the detections and target groups in the detection order list. The detection order list is processed from top to bottom. For example: MEC will start to look at the received message and see if it fits the MBM target group in the XML detection. If it does not fit, MEC continues to see if the message fits the MVX\_S0\_R0 target group and so on, until it finds a matching target group for the message. If a matching target group is not found, the message fails and the status is set to "detection failed" for the message.

In the area for Unused Targets, all the detections and the defined target groups for that detection are shown. In the area for Targets for Selected Groups, all the detections and target groups that you selected from the Unused Target area are shown. It is this detection order that is used by MEC in the Processes tab and in the Detection step.

## Adding a new Detection

Use this procedure to add a new detection that can be used in run time.

- 1 On Partner Admin Tool menu, click Manage > Advanced > Detections.
- 2 Click New. Type a unique name, description, a Java class, and then select a type to use.

Example of Java class:

```
com.intentia.ec.server.detection.xml.XMLDetector.
```

For an example of type: XML

- 3 Click OK to save the values.

## Executing XML Detection Using the MBM Target Group

Use this procedure to execute XML detection using the MBM target group.

- 1 On Partner Admin Tool, select an agreement from the tree view.
- 2 On the right pane, click on Detection tab.
- 3 Select Target Group MBM in the Detection tree view.

Other detection field information are displayed.

- 4 On Body element Xpath window, click MBM body XPath.

The MBM body XPath will be automatically added in the input field.

- 5 Add the target values, consider the following fields:

**Detection/env:from**      Type your source partner detail.

---

**Detection/env:to**      Type your recipient partner detail.

---

**Detection/env:URI**      Type the URI for your message and the version.

For example: `Test_100_in + 1.1 entail #Test_100_in@1.1`

- 6 Click Save to save the XML detection.

## Executing XML Detection Using the MVX Target Group

Use this procedure to execute XML detection using the MVX target group.

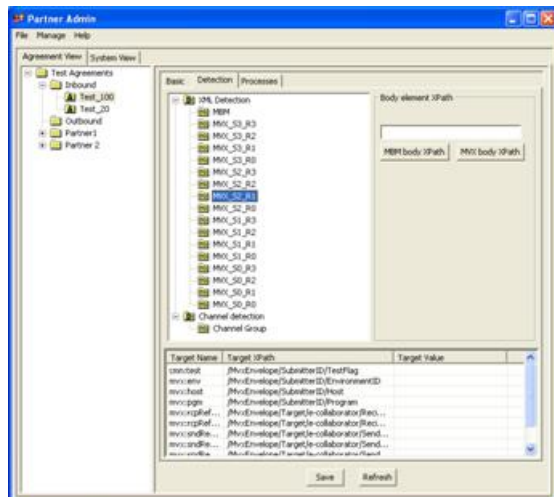
- 1 On Partner Admin Tool, select an agreement in the tree view.
- 2 On the right pane, click Detection tab.

3 On the Detection tree view, select Target Group **MVX\_X\_Y**.

The detection you choose depends on how many values you are using in previous MEC. In this example we use two sender values (**CONO** and **DIVI**) and one receive value (**CUNO**).

In this example, we will select the **MVX\_S2\_R1** target group.

Figure 2. Detecting XML using the MVX target group.



4 Click the **MVX** body XPath to automatically add in the input field.

5 Add the target values, consider the following fields:

**Detection/mvx:env** add the M3 environment, for example **MVXCENV**

**Detection/mvx:host** add the address to the M3 server, for example **SEIRD400** or an IP address.

**Detection/mvx:pgm** type the name of the M3 program that generated the initiator, for example (**OIS606PF**).

**Detection/sndRefData1** sender reference field 1, for example **CONO**.

**Detection/sndRefData2** sender reference field 2, for example **DIVI**.

**Detection/sndRefData3** sender reference field 3.

**Detection/rcpRefData1** receiver reference field, for example **CUNO**.

**Detection/rcpRefData2** receiver reference field 2.

**Detection/rcpRefData3** receiver reference field 3.

**Important:** If you leave an empty target value, the agreement will not be detected.

6 Click **Save** to save the detection.

## Executing Flat Detection Using Target Groups

Use this procedure to manage flat detection using target groups.

**Before you start** For flat detection using target group to work, you should first create a flat target group and flat targets.

- 1 On Partner Admin Tool menu, select an agreement in the tree view.
- 2 Create flat targets.
- 3 Create a flat target group.
- 4 Add the flat target group to the Targets for selected group list.
- 5 Add the flat detection to the detection order and click Save
- 6 Close the window and create a new agreement.
- 7 Fill in the basic information, and then click on Detection tab.
- 8 Select your flat target group in the Flat detection, in our example, the FLAT TG target group.
- 9 Add the target values.
- 10 Click Save to save the detection.
- 11 Click on Processes tab to set up the process flow to be executed.
- 12 Save the process flow.

**Need More Details? Check out the following concepts:**

- On how to create flat target group, see "[Creating Flat Target Group](#)" on page 103
- On how to create flat targets, see "[Creating Flat File Targets](#)" on page 101

## Target and Target Groups Overview

A target is a unique path to a single XML element, a single XML attribute, or a single position-based field in a flat file that is used for message detection. The XML element and XML attribute is defined by its absolute XPath.

A target group is a group of XML or flat targets used for message detection. There are two different target groups, the XML targets and Flat File targets. When detecting a message envelope you have one target group that corresponds to the envelope.

The XML targets included in the target group defines all elements and/or attributes in the XML envelope that are used for detection. You can use any element in the XML document for detection, however, you do not need to use an envelope.

In flat file targets you can detect on several position-based fields in one or more records. These position-based fields are defined by the flat targets included in the target group. These records are delimited by record separators such as CRLF.

## XML Targets

- ["Creating XML Targets" on page 95](#)
- ["Managing XML Targets" on page 95](#)
- ["Exporting XML Targets" on page 97](#)
- ["Viewing XML Tree" on page 98](#)
- ["Importing XML Targets" on page 99](#)

## Creating XML Targets

Use this procedure to add new XML targets in Partner Admin Tool.

- 1 On Partner Admin Tool menu, click Manage > Detection.
- 2 Click on Targets tab, click on XML tab and click New.
- 3 On Create new target window, type a unique Name and Description.
- 4 Type the XPath for the XML element or attribute (starting with "/").
- 5 Type a default Namespace URI. This is optional.
- 6 Click OK to save. Your new XML target is now listed in the XML tab contents.
- 7 Exit and close the application.

## Managing XML Targets

Use this procedure to view the details, edit the name or description, and to remove an XML target in Partner Admin Tool

- 1 On Partner Admin Tool menu, click Manage > Detection.
- 2 Click on Targets tab > XML tab.
- 3 Select a target from the list.

The fields are populated with the information of your selected target.

For Simple XML document and EventData message examples, see the section after this procedure.

Use the EventData message example for MEC to be able to distinguish or detect EventData messages supporting numbered element detection of XPath, that is `/bookstore/book[1..n]`.

- 4 Perform any of the following:

Task	Steps
To edit	<ol style="list-style-type: none"> <li><b>a</b> Click Edit group.</li> <li><b>b</b> On the Edit existing target window, modify the target name, description, path, and the default namespace URI.</li> <li><b>c</b> Click OK.</li> </ol>
To delete	<ol style="list-style-type: none"> <li><b>a</b> Click Delete.</li> <li><b>b</b> At the delete prompts, click Yes to remove the XML target from the database.</li> </ol>

**5** Click OK to save the changes.

## Simple XML document and its possible XML targets

```
<?xml version="1.0" encoding="UTF-8"?>
<Envelope>
  <Address>
    <To>ToAddress</To>
    <From>FromAddress</From>
  </Address>
  <Element Attribute="AttributeData">ElementData</Element>
  <Body/>
</Envelope>
```

The XPath `"/Envelope/Address/To"` gives the element data `"ToAddress"`.  
 The XPath `"/Envelope/Address/From"` gives the element data `"FromAddress"`.  
 The XPath `"/Envelope/Element"` gives the element data `"ElementData"`.  
 The XPath `"/Envelope/Element[@Attribute]"` gives the attribute data `"AttributeData"`.

## EventData message and its possible XML targets

Use this example for MEC to be able to distinguish or detect EventData messages supporting numbered element detection of xpath, that is `/bookstore/book[1..n]`.

**Important:** The XPath `"/a/b/c"` is equivalent to `"/a/b/c[1]"`. It is not allowed for both to be created in the same XML Target.



```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<EventData>
  <Publisher>M3</Publisher>
  <DocumentName>FGLEDX</DocumentName>
  <Operation>CREATE</Operation>
  <TrackingId>cab00804-efbd-4d8c-8af7-9e43753a84a7</TrackingId>
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<EventData>
  <Publisher>M3</Publisher>
  <DocumentName>FGLEDX</DocumentName>
  <Operation>CREATE</Operation>
  <TrackingId>cab00804-efbd-4d8c-8af7-9e43753a84a7</TrackingId>
  <SentTimestamp>2011-04-07T01:15:34.182-07:00</SentTimestamp>
  <Document>
    <ElementData>
      <Name>startProgram</Name>
      <Value>GLS040CL</Value>
    </ElementData>
    <ElementData>
      <Name>currentProgram</Name>
      <Value>GLS040</Value>
    </ElementData>
    <ElementData>
      <Name>StartTimeMillis</Name>
      <Value>1302164102126</Value>
    </ElementData>
    <ElementData>
      <Name>Job no</Name>
      <Value>679518557716169333</Value>
    </ElementData>
    <ElementData>
      <Name>Owner</Name>
      <Value>14785</Value>
    </ElementData>
    .
    .
    .
    <ElementData>
      <Name>LAST_ONE</Name>
      <Value>LAST_VALUE</Value>
    </ElementData>
  </Document>
</EventData>

```

```

The Xpath "/EventData/Document/ElementData/Name" gives the element data "startProgram"
The Xpath "/EventData/Document/ElementData/Name[1]" gives the element data "startProgram"
The Xpath "/EventData/Document/ElementData/Name[2]" gives the element data "currentProgram"
The Xpath "/EventData/Document/ElementData/Value[5]" gives the element data "14785"
The Xpath "/EventData/Document/ElementData/Name[last()]" gives the element data "LAST_ONE"
The Xpath "/EventData/Document/ElementData/Value[last()]" gives the element data "LAST_VALUE"

```

## Exporting XML Targets

Use this procedure to export available XML targets onto a new filename on a disk.

- 1 On Partner Admin Tool menu, click Manage > Detection.
- 2 Click on Targets tab > Import/Export XML tab.

All XML targets are shown under Available targets.

- 3 Select one or more XML targets from the list then click Export.
  - Shift-click - to select one or more random XML targets from the list.
  - CTRL-click - to extend an existing selection.
  - CTRL-Shift-click - to add a collection of XML targets.
- 4 On the Save as window, browse to the folder location where to export the XML and name the file.
- 5 Click Save.

## Viewing XML Tree

Use this procedure to view XML targets as a tree.

- 1 On Partner Admin Tool menu, click Manage > Detection.
- 2 Click on Targets tab > Import/Export XML tab.
- 3 Select an XML Target on the Available targets pane.
- 4 Perform any of the following and click Close to go back to the previous window.

Task	Steps
To view export tree	<ol style="list-style-type: none"> <li>a Click View Export Tree. A window shows a tree view of all the targets.</li> <li>b Click Close when finished.</li> </ol>
To view import tree	<ol style="list-style-type: none"> <li>a Specify an import target file.</li> <li>b Click View Import Tree.</li> </ol>

Message Envelope is the part of the message used for detection. Every message envelope has a different root element name and there will be one top node in the tree. However, if several envelopes or other XML elements share the same root element, then all these XML targets will be mixed under the same top node.

## Importing XML Targets

Use this procedure to import XML targets from a file or into the MEC database. When importing or exporting XML targets, you can view the tree structure of an XML targets.

**Need More Details? Check out the following concepts:**

- On how to view tree structure, see "[Viewing XML Tree](#)" on page 98

### ☐ Import XML targets from a file

- \_\_1 On Partner Admin Tool, click Manage > Detection.
- \_\_2 Click on Targets tab and click on Import/Export XML tab.
- \_\_3 Select an available XML target and click Open.
- \_\_4 On the Open window, select a file and click Open.  
The selected file path is shown in the Import targets file field.

**Note:** Existing Xpaths can not be imported.

- \_\_5 Click Import.

### ☐ Import XML targets to the MEC database

- \_\_1 On Partner Admin Tool, click Manage > Detection.
- \_\_2 Click on Targets tab and click on Import/Export XML tab.
- \_\_3 On Targets tab, click on Import/Export XML.  
The XML targets in the file are stored in the MEC database and are shown in the list of available XML targets.

## XML Target Group

- "[Creating XML Target Group](#)" on page 99
- "[Managing XML Target Group](#)" on page 100

## Creating XML Target Group

Use this procedure to create XML target group.

- 1 On Partner Admin Tool menu, click Manage > Detection.
- 2 Click on Target Groups tab, click on XML tab.
- 3 Click Create group. The Create a new Target Group window appears.
- 4 Type a group name and a description.
- 5 Click Create to store the new XML target group in the MEC database. This new target group contains targets.

## Managing XML Target Group

Use this procedure to view the details, edit the name or description, or remove an XML target group from the database.

- 1 On Partner Admin Tool menu, click Manage > Detection.
- 2 Click on Target Groups tab, click on XML tab.
- 3 On the Target Group field, select a group.  
A tree view with the XML targets included in the group in Targets for selected group is displayed.
- 4 Perform any of the following:

Task	Steps
To edit	<ol style="list-style-type: none"> <li>a Click Edit group.</li> <li>b On the Target Group dialog, modify the Target group name and the description.</li> <li>c Click Update to save the changes.</li> </ol>
To delete	<ol style="list-style-type: none"> <li>a Click Delete group.</li> <li>b At the delete prompt, click Yes to remove the XML target from the database.</li> </ol>

## Flat File Targets

- ["Creating Flat File Targets" on page 101](#)
- ["Viewing Flat File Targets" on page 101](#)

- ["Managing Flat File Targets" on page 102](#)

## Creating Flat File Targets

Use this procedure to create new flat file targets in the MEC database.

- 1 On Partner Admin Tool menu, click Manage > Detection.
- 2 Click Targets tab > Flat tab.
- 3 On Create new target form, consider the following fields:

<b>Name</b>	Type a unique flat file name.
<b>Description</b>	Type a brief description.
<b>Record Number</b>	Type a numeric record.
<b>Record separator</b>	Type a character to be used as a record separator.
<b>Start position</b>	Type a start position.
<b>End position</b>	Type an end position.

- 4 Click OK to store the flat target in the MEC database.

## Viewing Flat File Targets

Use this procedure to view the data of a flat file target

- 1 On Partner Admin Tool menu, click Manage > Detection.
- 2 Click Targets tab > Flat tab.
- 3 Select a Flat File target in the list. The data for the flat target is shown.

For an example of a flat file target view, following is a simple two flat files and some matching flat targets view.

```
123200508240000014358000000034920
456200508240000000034000000003420
456200508240000000069000000006920

Record number = 1
Record separator = \r\n
Start position = 1
End position = 3

000120050824000356821971@@525020050824@@07402005082445

Record number = 2
Record separator = @@
```

```
Start position = 1  
End position = 4
```

In this example, the Record number = 1 start and end position refer to **123**.

And for Record number = 2, the start and end position refer to **5250**.

## Managing Flat File Targets

Use this procedure to edit or delete existing flat file targets.

- 1 On Partner Admin Tool menu, click Manage > Detection.
- 2 Click Targets tab > Flat tab.
- 3 Select the flat file target.
- 4 Perform any of the following:

Task	Steps
To edit	<ol style="list-style-type: none"><li>a Click Edit.</li><li>b On Edit existing target window, modify the field information.</li><li>c Click OK.</li></ol>
To delete	<ol style="list-style-type: none"><li>a Click Delete.</li><li>b At the delete prompt, click Yes.</li></ol>

- 5 Close and exit the application.

## Flat File Target Group

- ["Creating Flat Target Group" on page 103](#)
- ["Managing Flat Target Group" on page 103](#)

## Creating Flat Target Group

Use this procedure to add a new flat target group in MEC database.

- 1 On Partner Admin Tool menu, click Manage > Detection.
- 2 Click on Target Groups tab, click on Flat tab.
- 3 Click Create group.
- 4 On the Create a new Target Group form, type a new Group name and Description.
- 5 Click Create.

The new target group is now stored in the MEC database, but it does not contain any targets.

To add flat target groups, use the edit procedure. For more information, see "[Creating Flat File Targets](#)" on page 101.

## Managing Flat Target Group

Use this procedure to view the tree structure of flat target groups, change the name and the description, or remove it from the MEC database.

- 1 On Partner Admin Tool menu, click Manage > Detection.
- 2 Click on Target Groups tab, click on Flat tab.
- 3 Select a group.

A tree view with the flat targets included in the group is displayed.

- 4 Perform any of the following:

Task	Steps
To edit	<ol style="list-style-type: none"><li>a Click Edit group.</li><li>b On Edit form, change the group name and the description.</li><li>c Click Update to save the changes.</li></ol>
To delete	<ol style="list-style-type: none"><li>a Select a group.</li><li>b Click Delete.</li><li>c Click Yes at the prompt to remove the target group from the MEC database.</li></ol>

## Moving Targets Between Areas

Use this procedure to move targets between areas.

- 1 On Partner Admin Tool menu, click File > Manage > Detection.
- 2 Click on Detection Order tab.
- 3 Select the target to move from either **Unused** or **Used** panes.
- 4 Click one of the directional buttons ">>" or "<<" to move the targets.

**Note:** To help you with problems that can occur according to the detection order preform a sanity check. For more information on how to perform sanity check, see "[Performing Sanity Check on Target Groups](#)" on page 104.

## Performing Sanity Check on Target Groups

Use this procedure to perform sanity check on your targets. Sanity check helps you with problems that can occur according to the detection order.

- 1 On Partner Admin Tool menu, click File > Manage > Detections.
- 2 Click on Detection Order tab.
- 3 Right-click on a selected target for the selected group area.
- 4 Select the Sanity check.

The sanity check will go through the detection order and issue a warning if it finds something suspicious. One of the many things sanity checks for is the occurrence of shadowing.

For more information on shadowing, see "[Shadowing Overview](#)" on page 105.

## Target Groups and Shadowing

- "[Shadowing Overview](#)" on page 105
- "[Shadowing in Flat File Targets](#)" on page 106
- "[Shadowing and Channel Detection](#)" on page 107



## Shadowing Overview

**Important:** You must understand shadowing when setting up a detection order.

Shadowing is about the theory of sets. If you have two sets and neither of the two sets is a subset of the other, there is no problem.

Figure 3. Shadowing set one and set two.



However, if one of the set is a subset of the other, there might be a problem with the two sets shadowing each other.

Figure 4. Shadowing set one as subset of set two.



For example:

- One target group (TG 1) with two paths: **A\B\C = "X"** and **D\E\F = "Y"**.
- Another target group (TG 2) with one path: **A\B\C = "X"**.

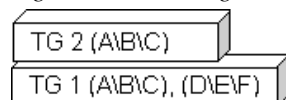
In this case the target groups will shadow each other, meaning that TG 2 is a subset of TG 1.

Now, assume that the data in the paths **A\B\C = "Alpha"** and **D\E\F = "Bravo"** is sent to MEC:

```
<A>
  <B>
    <C>Alpha<C/>
  </B>
</A>
<D>
  <E>
    <F>Bravo<F/>
  </E>
</D>
```

With the detection order:

Figure 5. Shadowing TG2 and TG1



In this example, the two paths (A\B\C) in both TG 2 and TG 1 have the same value "Alpha".

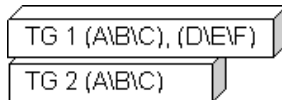
Then, MEC tries to detect the message according to this detection order. MEC will start to check if there is a match on TG 2 for the target (A\B\C = "Alpha"). MEC finds a match on this target, so it will not continue to TG 1. This is because target (A\B\C) is included in TG 2.

This is called shadowing. It is when the target groups are ordered in such a way that a larger set will no longer be used because its target values are already included in a smaller set, and placed earlier in the detection order.

## When the order is changed in pairs of target groups

Changing the order of two target groups affects the detection order of targets.

Figure 6. Changed order of target groups.



In the given example, the smaller set is placed after the larger set. MEC will check if the first target group (TG) will have a match on the targets A\B\C and D\E\F. For as long as the TG 1 will have a match on both of its targets, TG 1 will be used. But if there is no match on both targets, MEC will continue to find for a match on TG 2. MEC will only proceed in the detection order list if TG 1 does not get a match on both targets.

For as long as the first target has a match on all the targets, that match is used. This explains the particular order of MVX\_XZ\_YQ target groups.

To avoid shadowing, the target groups are ordered with the largest set first and the smallest set last.

## Shadowing in Flat File Targets

Shadowing problems can occur when creating flat file target groups.

For example, you have set up three different types of flat file target groups, TG 1, TG 2, and TG 3:

```
TG 1
Record Number: 1
Record Separator: @@
Start Position: 2
End Position: 2

TG 2:
Record Number:1
Record Separator: ff
Start Position: 2
End Position: 2

TG 3:
Record Number: 1
Record Separator:\r\n
Start Position: 2
End Position: 2
```

Now the question will be: *"Which TG will be used when MEC receives the different flat files?"*

MEC does not know what it receives and in what particular data order or message. MEC then reads any inbound message and tries to find a matching detection and targets group. When a matching detection and target group is found, MEC continues to perform the process steps in the agreement for the specific detection and target group.

In this example:

- TG 1 will be used.
- TG 1 shadows TG 2 and TG 3.
- TG 2 and TG 3 will not be used.

For more information on shadowing see, "[Shadowing Overview](#)" on page 105.

## Shadowing and Channel Detection

If you are using channel detection, it is important to consider the occurrence of shadowing in channel detections.

For example: if you are using the MBM Target Group in the XML Detection for the Test\_20 message and at the same time you have the channel detection enabled using DiskIn, when you drop the Test\_20 message on the Input folder, MEC processes the message. However, the result does not look right. It could be that your message did not match the MBM Target Group. Instead it was picked up by the channel detection.

This problem can be caused by a simple misspelling in the envelope for Test\_20 message. The error could cause the message to run past the MBM Target Group and picked up by the channel detection.

- ["XML Processing" on page 108](#)
- ["Apply Envelopes" on page 111](#)
- ["Modify Flat Record" on page 117](#)
- ["Split Redetect" on page 123](#)
- ["Send WebService" on page 132](#)

## XML Processing

- ["Creating new XSLT Definitions" on page 108](#)
- ["Managing XSLT Definitions" on page 109](#)
- ["Exporting XSLT Definitions" on page 109](#)
- ["Unpublishing XML Mappings" on page 110](#)

## Creating new XSLT Definitions

Use this procedure to create new and edit existing XSLT definitions that MEC can apply to XML messages. By default, no definitions are included.

- 1 On Partner Admin Tool menu, click Manage > XSLT Definitions > New.
- 2 On Create new XSLT definition window, type a Basic Name and Description for your new definition.
- 3 Click Select Definition to import a definition from file.
- 4 Browse to the file to add click OK.
- 5 Verify that the new definition is listed in the XSLT Definitions form.

## Managing XSLT Definitions

Use this procedure to manage the contents of your XSLT definitions. These procedure will allow you to view the details and change the contents of the XSLT definition, delete from file, or edit the details of the XSLT definitions.

- 1 On Partner Admin Tool menu, click Manage > XSLT Definitions.
- 2 On XSLT Definitions window, select a definition from the list.
- 3 Perform any of the following:

Task	Steps
To view	<ul style="list-style-type: none"><li>• Click View Definitions.</li></ul> <p>The View Envelope form appears with the definition details.</p>
To edit	<ol style="list-style-type: none"><li>a Click Edit.</li></ol> <p>The Edit existing xslt definition dialog opens.</p> <ol style="list-style-type: none"><li>b Change the Basic name and description for the definition.</li><li>c Click OK to save your changes.</li></ol> <p><b>Note:</b> To change the XSLT definition content use a text editor of your choice.</p>
To delete	<ol style="list-style-type: none"><li>a Click delete.</li><li>b A confirmation box is displayed. Click Yes to remove the definition file.</li></ol>

- 4 Close and exit the application.

## Exporting XSLT Definitions

Use this procedure to export a definition to file on the disk or to remove an existing XSLT definition from file.

- 1 On Partner Admin Tool menu, click Manage > XSLT Definitions.
- 2 Select a definition from the list.

- 3 Click Export Definition.

The Save as form appears.

- 4 Browse to the folder location and then save in a new filename.

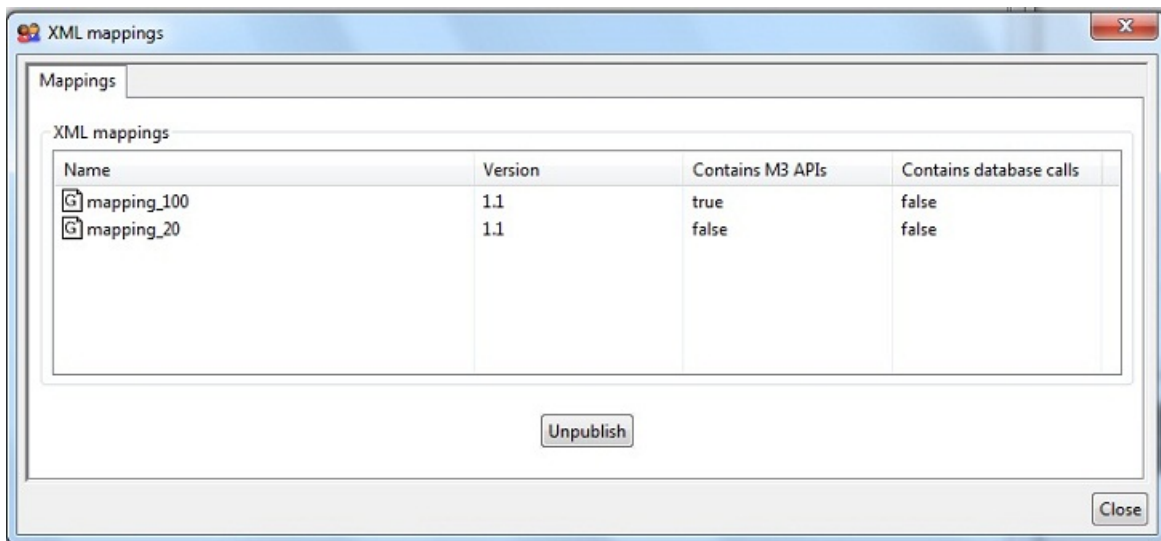
## Unpublishing XML Mappings

Use this procedure to unpublish XML mappings from MEC database.

- 1 On Partner Admin Tool menu, select Manage > XML Mappings > Mappings.

The XML mappings window is displayed. All XML mappings published to MEC database are listed.

For more information about publishing and XML Mappings, see the *M3 Enterprise Collaborator Mapping Manager Tool User Guide*.



**Note:** XML mappings has a new column that indicates if a mapping contains database calls or not.

- 2 Select the XML mapping you want to unpublish, and then click Unpublish.

**Important:** Mappings in use can not be unpublished.

- 3 At the delete prompt, click Yes to unpublish the mappings.

Unpublished mappings are removed from the list.

- 4 Close and exit.

# Apply Envelopes

This section explains the procedures in managing user-defined XML Message envelopes that MEC can apply to outbound messages.

**Important:** MBM envelope and XML Declaration are included by default.

- ["Envelope Objects Overview" on page 111](#)
- ["Importing a New Envelope Template" on page 112](#)
- ["Managing Existing Envelope" on page 112](#)
- ["Exporting an Envelope Template" on page 113](#)
- ["Managing Flat File Definitions" on page 114](#)
- ["Creating DAF Archive Process" on page 114](#)
- ["Option to Remove Document/Image File" on page 115](#)
- ["Option to Archive self\(.rcv\)" on page 116](#)

## Envelope Objects Overview

Envelopes allow users to create a text file where system and editable values are defined by start and end separators. An example of a created text file is XML. Users can create any message envelope using the text file as a template. The user-defined message envelopes that MEC can apply to outbound messages are managed in Partner Admin Tool.

**Note:** MEC is delivered with the external MBM envelope and also with the internal MEC envelope.

### System Values:

System values are set by MEC at run time. For example: the system value THE\_PAYLOAD will be replaced by the message's payload created by the MEC mapping.

For more information about mapping, see *MEC Mapping Manager User Guide*.

### Editable Values:

Editable values are set in the message's Agreement.

For more information about Agreement, see ["Partner Agreement Overview" on page 134](#).

## Importing a New Envelope Template

**Before you start** Create the XML template before importing into Partner Admin Tool. You can not create the XML template from within the Partner Admin Tool.

Use this procedure to import a new envelope template definition that can be applied on your messages.

- 1 On Partner Admin Tool menu, click Manage > Envelopes.
- 2 Click New.
- 3 On Create new envelope object window, type a Basic name, description, and select an encoding type.
- 4 Click Select Template to import an envelope template.
- 5 Browse to the folder location of the XML envelope template to import.
- 6 Click OK.

For an example of XML template envelope including only the "To" and "From" tags, their values within % are system variables used to add system values to the envelope. The message body is included in the tag: %**THE\_PAYLOAD**%. .

```
<?xml version="1.0" encoding="%+ENCODING+%" ?>
<Envelope xmlns="http://www.lawson.com/MBM"
xmlns:env="http://www.lawson.com/MBM_Envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xsi:schemaLocation="http://www.lawson.com/MBM %+EDITABLE:Schema+%">
<Header>
  <env:delivery>
    <env:to>
      <env:address>%+EDITABLE:To Address+%/</env:address>
    </env:to>
    <env:from>
      <env:address>%+EDITABLE:From Address+%/</env:address>
    </env:from>
    <env:identity>%+MANIFEST:ManifestConstants.UUID+%/</env:identity>
  </Header>
<Body>%+THE_PAYLOAD+%/</Body>
</Envelope>
```

## Managing Existing Envelope

Use this procedure to change the name, description, and encoding of an existing envelope.

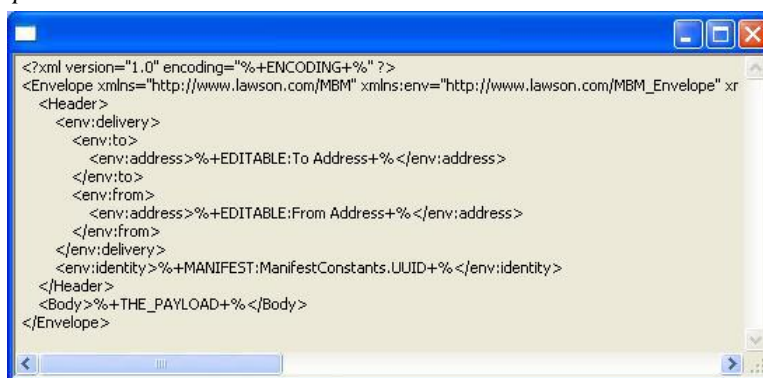
- 1 On Partner Admin Tool menu, click Manage > Envelopes.
- 2 Select an envelope.
- 3 Perform any of the following:



Task	Steps
To view	<ol style="list-style-type: none"> <li>Click View Template.</li> <li>In View envelope, the XML format content of the template is shown.  See the figure below for content details.</li> </ol>
To edit	<ol style="list-style-type: none"> <li>Click Edit.</li> <li>In Edit existing envelope object, modify the Channel configuration, basic configuration, and Unique file name information.</li> <li>Click OK.</li> </ol> <p><b>Note:</b> To change the envelope content, you must change the template file and import it using Select Template.</p> <p>For more information, see <a href="#">Importing a New Envelope Template</a></p>
To delete	<ol style="list-style-type: none"> <li>Click delete.</li> <li>At the prompts, click Yes.</li> </ol>

#### 4 Close and exit the application.

Figure 7. Envelope template content



## Exporting an Envelope Template

Use this procedure to export an existing envelope as a template.

- 1 On Partner Admin Tool menu, click Manage > Envelopes.
- 2 Select an envelope object and click Export Envelope.
- 3 On Save As window, browse to the folder location where you want to save the template file.
- 4 Assign a file name for the template and click Save.
- 5 Open the file in an editor to see what the template file looks like.

## Managing Flat File Definitions

Use this procedure to manage flat file definitions in Partner Admin Tool. This procedure allows you to enable, disable, and view the imported flat file definitions.

For more information on how to create, edit, import, and export flat file definitions, see the *Flat File Definition Tool User Guide*.

- 1 On Partner Admin Tool, click Manage > Flat Definition.
- 2 On Flat Definitions tab, select the flat definition.
- 3 Perform any of the following:

Task	Steps
To view	<ul style="list-style-type: none"> <li>Click View Definitions.</li> </ul> <p>The View Envelope window opens with the definition details.</p>
To enable flat file definitions	<ul style="list-style-type: none"> <li>Select Enabled across your selected flat file definition.</li> </ul> <p>The Edit existing <b>xslt</b> definition dialog appears.</p>
To disable flat file definitions	<ul style="list-style-type: none"> <li>Clear Enabled across your selected flat file definition.</li> </ul>

Close and exit the application.

## Creating DAF Archive Process

Use this procedure to create DAF agreements and verify that you can send messages in DAF.

**Before you start** Ensure that the following conditions exists:

- DAF Environment is running and a DAF connection is already configured in Communications menu.
- DAF Agreements are created.

## ❑ Create DAF Archive process

- \_\_\_1 Start Partner Admin Tool, and click on Process tab.
- \_\_\_2 Right-click on Selected Processes panel and select DAF Archive.  
The **Configuration for DAF Archive** window is displayed.
- \_\_\_3 Select DAF\_CONN Connection.  
On this window, consider the following fields:  

<b>Fine name Xpath</b>	Type <b>/DAF/FILENAME</b>
<b>File Location XPath</b>	Type <b>/DAF/FILELOC</b>
- \_\_\_4 Click OK, click SAVE.  
The DAF Archive Process is created and saved on MEC.

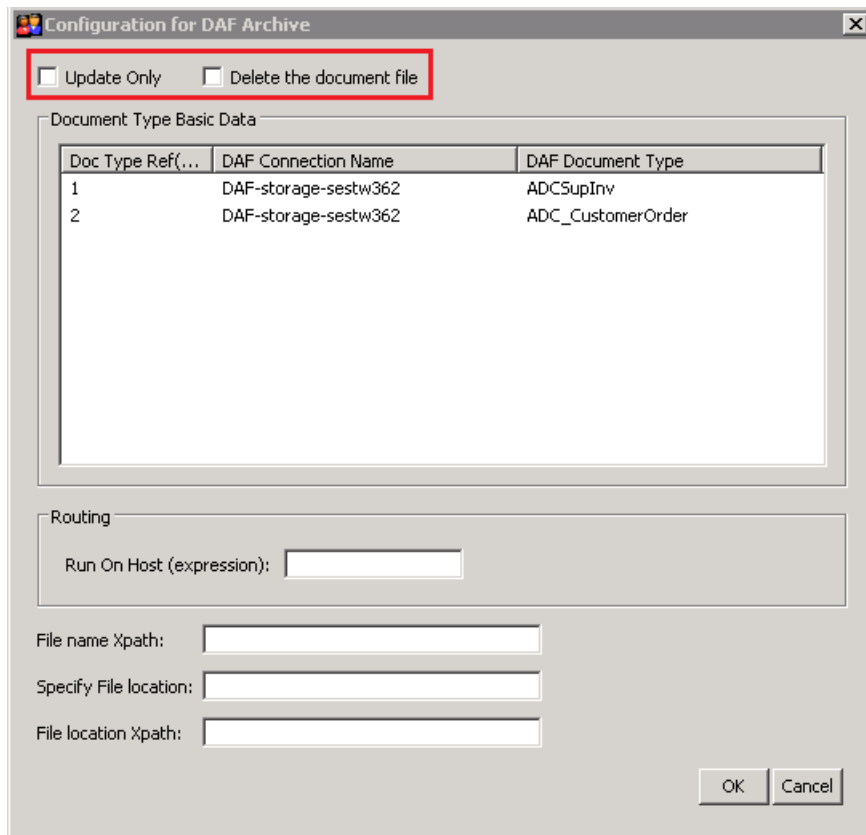
## Option to Remove Document/Image File

In earlier versions, when MEC receive a message for DAF Archive process the corresponding document or image linked to the received message is automatically archived in DAF. A copy of the document remains in the sender application.

Now, MEC added an option to remove the document file from the sender application.

- 1 Access the Configuration for DAF Archive window.
- 2 Select a document to edit.
- 3 Select **"Update Only"** and **"Delete the document file"** .

The location of the document files defined in 'Specify File Location' or 'File location XPath' fields will be cleared.

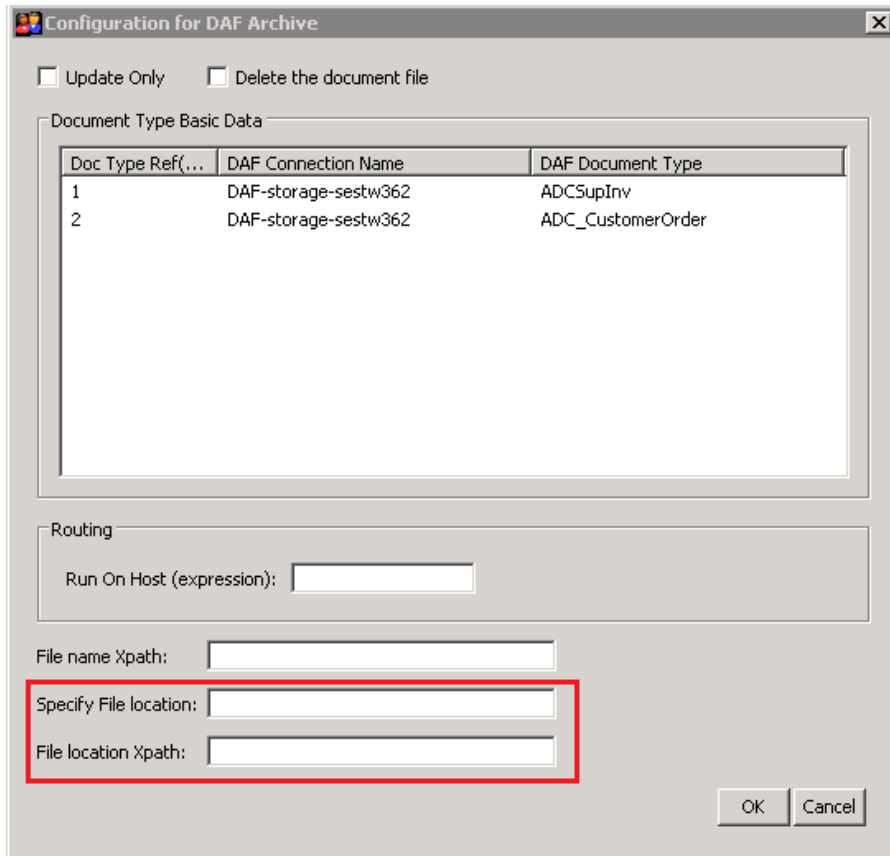


## Option to Archive self(.rcv)

In earlier versions, when MEC receive a message for DAF Archive process, it will require for a link to the corresponding document or image. An error occurs when there is no image or document link, or no file location is defined in 'Specify File Location' or 'File location Xpath' fields.

Now, MEC added an option to automatically archive in DAF received messages.

- 1 Access the Configuration for DAF Archive process window.
- 2 Select a document to edit.
- 3 Leave blank the fields **Specify File location** and **File location Xpath**.



The dialog box is titled "Configuration for DAF Archive". It contains two checkboxes at the top: "Update Only" and "Delete the document file". Below these is a section titled "Document Type Basic Data" which contains a table with three columns: "Doc Type Ref(...)", "DAF Connection Name", and "DAF Document Type". The table has two rows of data. Below the table is a "Routing" section with a label "Run On Host (expression):" and an empty text box. At the bottom, there are three text boxes: "File name Xpath:", "Specify File location:", and "File location Xpath:". The "Specify File location:" and "File location Xpath:" boxes are highlighted with a red rectangle. At the bottom right are "OK" and "Cancel" buttons.

Doc Type Ref(...)	DAF Connection Name	DAF Document Type
1	DAF-storage-sestw362	ADCSupInv
2	DAF-storage-sestw362	ADC_CustomerOrder

## Modify Flat Record

- ["Modify Flat Records" on page 117](#)
- ["Property Records" on page 118](#)
- ["Strip / Replacement" on page 119](#)
- ["String Literals" on page 120](#)
- ["Examples" on page 120](#)

## Modify Flat Records

When working on MEC flat files, there are instances when modifications are required before doing any MEC process. Following is a list of modifications that can be done:

- Removing header records
- Removing other unwanted records
- Changing record separator
- Removing "garbage" characters
- Replacing non-XML characters

Typically these modifications must be completed before the Split Redetect process or before the FLAT to XML process.

When a flat file reaches the FLAT to XML process the flat file should conform to the message specification and nothing else. All extra records, wrong record separators, "garbage" characters, and others that are created by different communication processes from the trading partner. Otherwise, these must be removed or fixed prior to converting flat file to XML.

Modifying Flat Record is a very generic process that uses regular expressions. A basic to good knowledge of regular expressions is recommended to fully benefit from this process.

## Split Redetect Process

Split Redetect is a process that splits a message based on the header, trailer, and regular expression settings, and then sends the split messages back to MEC for redetection.

## Modify Flat Record Process

The process Modify Flat Record has two properties and one or more Strip/Replacement steps that need to be configured. The properties Record Separator and Max Record Length define the records. The Strip/Replacement steps define what to do with the records, that is to strip (remove) them or to modify them in some way.

**Need More Details? Check out the following concepts:**

- For more information, see Javadoc for the class Pattern:
- <http://java.sun.com/j2se/1.5/docs/api/java/util/regex/Pattern.html>

## Property Records

The process reads records from the persisted (input) file one at a time. Each record is processed by all Strip/Replacement steps in the defined order and then written to the output file. Large files can thus be handled without risk of high memory consumption.

## Property Record Separator

The property Record Separator defines a character sequence that constitutes the record separator. A record separator must be given. You can use escape sequences for string literals in the same way as

when you hard code a string in Java. To define carriage return line feed, more commonly called newline, as the record separator you can type the value `\r\n`.

For more information about string literals see [String Literals](#).

## Property Max Record Length

The property Max Record Length defines the maximum number of characters a record can contain, including the record separator. This is also the buffer size when reading the input file. You must not set this to a very low value. However, a too high value will work but there may be a somewhat higher memory consumption for large files.

The record sent to the Strip/Replacement steps includes the record separator at the end. If the record separator is not found all currently read characters are assumed to constitute the last record in the file, without a record separator. This record will also be processed by the Strip/Replacement steps. If there are additional characters in the input file they will not be read, and will not be written to the output file.

A warning log message that the file is truncated will be given in this case. To avoid this error, it is very important to set the correct record separator and max record length.

## Strip / Replacement

You can define one or more Strip/Replacement steps that will be applied to every record in the defined order. The output from one step is the input to the next step. After the last step, the record is written to the output file.

## Record Filter

The property Record filter expression is used to specify a record filter. If a regular expression is given the complete record, including the record separator, must match the regular expression. Otherwise, the record will not be processed by this step. Note that the **dotall** mode is used: the expression "." matches any character including a line terminator. Use a blank value for no record filter.

## Record max

The property Record max count to replace is used to limit the number of records to be processed by this step. For example: if you type the value 1, this step will be applied to max one record. Use a blank value for no limit.

## Regular expression

The property Regular expression contains a regular expression for the substrings to be replaced in the record. The regular expression does not need to match the complete record. This property is mandatory.

## Replacement string

The property Replacement string contains the replacement string that will replace each substring in the record that matches the given regular expression. Remember that the resulting record, normally, must end with a record separator. You can use escape sequences for string literals in the same way as when you hard code a string in Java.

For more information about string literals, see [String Literals](#).

If the regular expression matches the complete record, including the record separator, and an empty replacement string is given, this means that the record is removed or "stripped".

## String Literals

You can use escape sequences for string literals for the properties Record Separator and Replacement string. This works just like when you hard code strings in Java; you enter `\t` for the tab character, `\n` for the line feed character, `\u001e` for the Unicode character hex 1e and others.

The following escape sequences are decoded:

<code>\b</code>	<code>\u0008: backspace BS</code>
<code>\t</code>	<code>\u0009: horizontal tab HT</code>
<code>\n</code>	<code>\u000a: line feed LF</code>
<code>\f</code>	<code>\u000c: form feed FF</code>
<code>\r</code>	<code>\u000d: carriage return CR</code>
<code>\\</code>	<code>\u005c: backslash \</code>
<code>\uhhhh</code>	<code>\u0000 to \uffff: Unicode value</code>

These are Unicode values. The actual file may use another character encoding. When reading a file, the character encoding is converted from the file's default encoding to Unicode. When writing a new file, the character encoding is converted back from file's encoding. Therefore, a binary, octal, decimal, or hexadecimal value, representing a character as given in the specification, may not match the Unicode value that represents the character since specifications normally refer to the character encoding used by the flat file. Similar functionality for string literals also exists for regular expressions. For more information, see the Javadoc.

## Examples

The record separators and record lengths below are examples.



### Strip the first record that starts with \$\$ADD:

Record Separator	\r\n
Max Record Length	100
Record filter	
Count	1
Regular expression	^\\$ \\$ADD.*\$
Replacement	

### Change the record separator \u001e to \r\n

Perform this change to avoid the need to change the flat file definition and for archived files to be more readable.

Record Separator	\u001e
Max Record Length	200
Record filter	
Count	
Regular expression	\u001e\$
Replacement	\r\n

### Strip all records that start with ISA or IEA:

Record Separator	\r\n
Max Record Length	200
Record filter	
Count	
Regular expression	^ISA.*\$ ^IEA.*\$
Replacement	

**Strip all records that are not 100 characters long, also remove "garbage" character at the end of the file:**

Record Separator	\r\n
Max Record Length	
Record filter	
Count	
Regular expression	^.{0,99}\r\n\$ ^.{101,}\r\n\$ ^\w\$
Replacement	

### Strip "garbage"

Remove garbage characters before the start tag, for example: "**anytext**=" and after the end tag to get a real XML file. This is not easy to make, however, here are possibilities:

Record Separator	<
Max Record Length	200
Record filter	^.*=<\$
Count	1
Regular expression	^[^>]*<\$
Replacement	<
Record filter	^.*[^<]\$
Count	1
Regular expression	>.*\$
Replacement	>

# Split Redetect

- ["File Splitting Overview" on page 123](#)
- ["SplitRedetect Parallel Run" on page 124](#)
- ["Batch File Splitter" on page 125](#)
- ["Large File Splitter" on page 125](#)
- ["File Splitter Types and Properties" on page 127](#)
- ["Batch File Splitter properties definition" on page 127](#)
- ["Large File Splitter properties definition" on page 128](#)
- ["Splitting Batch Files" on page 129](#)
- ["Splitting Large Files" on page 130](#)
- ["Splitting on N:th\(property\) Occurrence" on page 131](#)

## File Splitting Overview

Certain incoming files must be split into corresponding individual messages before processing. The source CEMS files before the split are also called the parent message. After the split dependent messages are created each having its own partner agreement, processing flow, and UUID.

Message processing follows a specified order method, order mode setting, and selected order manifest.

### Order method

Messages are processed in ordered or unordered method, or sequenced through the FTP communication channel with an option to include the batch message as the last item in a sequence.

Ordering allows processing of dependent messages in a manner based on how MEC queues the manifest, which is dependent on the communication channel's order method. You can select the method by setting the flag property value to 1 for ordered, or 0 for unordered. The default value is set to 0.

When MEC is restarted, or is started back up from a crash, it will continue processing messages depending on the message order mode that you have set.

### Order modes

Ordered messages are processed according to individual channel sequence. In the event of a process failure, MEC will follow the message order mode setting to continue a process. The available order mode options are below:

- **Continue on Fail** - Regardless of whether the previous message fails or completes successfully, the succeeding messages will continue processing.

- Fail Succeeding - If the previous message fails, succeeding messages are flagged as "failed".
- Wait on Fail - If the previous message fails, succeeding messages are put on "wait" status. When the previous message is verified or completed successfully through **Reprocess** or **Retry**, then processing resumes for the succeeding messages on "wait" status.

### Order manifest

- Order Split Message - select this option to process messages by split order, processing one message after another.
- Order Main Manifest After Last Split - select this option to wait for the last message in a split to complete processing before it is marked finished, and then it will go back to the main agreement for processing.

## SplitRedetect Parallel Run

MEC message processing is enhanced to allow dependent messages to run in parallel while sequencing the parent message per channel ordering. In parallel processing, the **Order Main Manifest After Last Split** option is no longer dependent on the **Order Split Message** option. That is, dependent messages now do not need to be processed in order, but the main manifest must wait until MEC has completed processing all dependent messages before the main manifest is marked complete.

To allow dependent messages to run in parallel, while sequencing main messages per channel ordering, select the options below.

- 1 Select the order manifest to use (split or main).
- 2 Select the order mode to use.

### Example scenario of parallel run:

- Ordered channel receives messages A, B, C, and D.
- A, B, C, and D have split redetect so they have dependent messages A1, A2, A3, B1, B2, B3, C1, C2, C3, and so on.
- Order Main Manifest option must be selected.

Requirement: No need to process dependent messages in order.

Result: Select Order Main Manifest option. Unselect Order Split Children.

Processing order:

```
A, A1, B, B1, C, C1, D, D1
A2, B2, C2, D2
A3, B3, C3, D3
```

## Batch File Splitter

A CEMS batch file contains headers, records, and footers. Batch files can support multiple combinations of header and footer. These combinations are configurable and are used as bases in splitting a batch file. Record separators are used in every split and you can define what character to use as a record separator, for examples, a comma, forward slash, back slash, or "\n".

When splitting a batch, headers are first detected and then the trailers. If a data has no matching header and footer, or if there is a header match but has no trailer, a unique message is created. Here are some examples:

- If a header is found on the 5th record, the first four records are considered as the first message.
- If a trailer is found, the Splitter assumes that the succeeding records are part of the trailer.
- If a batch message is received through FTPScriptPollIn channel, dependent messages will have a header based on the batch message.
- If a message is received through FTPScriptPollIn channel, after the batch split dependent messages are assigned a header with the format below:

1 - 6 = \$\$BAT

7 - 20 = Sender Reference

21 - 34 = Application Reference

35 - 84 = Service Reference 85 onwards = Record Separator

## Large File Splitter

A large file can be loaded from a DVD or any external storage on demand. It contains several records where splitting helps identify where a new record begins and where it ends. Flat file messages are created from splitting Large files based on specific sets of information for implementation of the agreement process. Split Flat file messages are processed in ordered or unordered method.

### Example of Number of Lines splitter:

Here is an example of a series of records before a split, based on the number of lines:

```
Header Records
Replacement Parts Prices
Replacement Parts Prices
Remanufactured Parts Prices
Replacement Parts Prices
Remanufactured Parts Prices
Replacement Parts Prices
Replacement Parts Prices
Remanufactured Parts Prices
Trailer
```

When a channel receives an incoming large file, the message is split based on "split after X number of rows". When it reaches the x number of lines and there is no split record, it will check the next record.

Now, if the checked record still has no split, it will not be included. But if it has a split record it will be included in the split.

Logically, here is how it works:

- 1 If the last record matches 'Remanufactured Parts Prices', the file is split from this point.
- 2 Then, if the last record matches 'Replacement Part Prices', one more record is read.
- 3 Then, if the next record matches 'Remanufactured Parts Prices' it is included, and the split starts from this point.
- 4 Then, if the next record matches 'Replacement Part Prices' it is not included, and the file splits here for the next flat message.

After a split, every flat file message created will have a pair of header and trailer based on the original data.

```
Header Record (1)
Replacement Parts Prices
Replacement Parts Prices
Trailer (1)
```

```
Header Record (2)
Replacement Parts Prices
Remanufactured Prices
Trailer (2)
```

```
Header Record (n)
Replacement Parts Prices
Replacement Parts Prices
Replacement Parts Prices
Remanufactured Prices
Trailer (n)
```

### Example of Sub-header splitter:

After a split, every message created may contain a sub-header with one or several of the following data:

```
System/Sub-system (U)
Reference Number (R)
Note (N)
Text (T)
```

#### Example: file split 1

```
Header (H1)
Sub-Header (S1)
System/Sub-system (U)
Reference Number (R)
Note (N)
Text (T)
Trailer (Z1)
```

#### Example: file split 2

```
Header (H2)
Sub-Header (S1)
System/Sub-system (U)
Reference Number (R)
Note (N)
Text (T)
Trailer (Z2)
```

## File Splitter Types and Properties

Here are four file splitter types available for CIF. You can configure the properties below, depending on your selected splitter type.

### Batch File Splitter Type

- Header and Trailer - splitter is based on the header and footer combinations. Headers are first detected and then the trailer. If a data has no matching header and footer, or if there is a header match but has no trailer, a unique message is created.

After a batch split, dependent messages are assigned a header identifiable to the received batch file.

### Large File Splitter Types

- Number of Lines - splitter is based on the specified number of lines. For more information, see "[Large File Splitter](#)" on page 125.
- Sub-Header - splitter defines how to identify a sub-header value and start position.
- Multi Sub-Header - splitter supports multiple sub-header detections as bases of file splitting. The sub-header detection is defined as a regular expression (regex).

## Batch File Splitter properties definition

Consider the following property definitions when updating Batch file Splitter fields:

Header Trailer Setting	Definition
Order split messages	Select this to process messages by split order, one message after another.
Order main manifest after last split	Select this to wait for the last message in a split to complete before it is marked finish, and then it will go back to the main agreement for processing.
Order Mode	Messages in order are processed in a sequence based on a selected mode.

Header Trailer Setting	Definition
Record Separator	Defines the character to use as a record separator. For example, a comma, backslash, forward slash, or a new line insert (\n).
Header	Defines the header record and its position in the record. If there is a header, the splitter assumes that it is the first record.
Trailer	Defines the trailer record. When a trailer is found, the Splitter assumes that all the succeeding records are part of the trailer.
Header Record Count	Defines how many records are contained in a header. Splitter supports a multiple-matching header records.
Trailer Bytes Hint	Guides the splitter where to start looking for a trailer. For example, the value 1 gives the splitter a hint to look for the trailer at the start of file.

## Large File Splitter properties definition

The Header and Trailer Settings is standard for all large file splitter modes.

Properties	Definition
Order split messages	Select this to process messages by split order, one message after another.
Order main manifest after last split	Select this to wait for the last message in a split to complete before it is marked finish, and then it will go back to the main agreement for processing.
Order Mode	Messages in order are processed in a sequence based on a selected mode.
Record Separator	Defines the character to use as a record separator. For example, a comma, backslash, forward slash, or a new line insert (\n).
Header	Defines the header record and its position in the record. If there is a header, the splitter assumes that it is the first record.
Trailer	Defines the trailer record. When a trailer is found, the Splitter assumes that all the succeeding records are part of the trailer.
Number of Lines Hint	Defines the number of lines for a split.
No Split Record	The value to use for No Split Record.



Properties	Definition
Sub-Header	Defines how to identify a sub-header with its value and start position.
Multi Sub Header	Defines a regular expression (Regex Detection) sub-header to use as a search string pattern for detection.

## Splitting Batch Files

Use this procedure to split CEMS Batch files into corresponding individual messages to be implemented as an agreement process for the batch or parent agreement. For more information, see "[File Splitting Overview](#)" on page 123.

**Before you start** Add a Split Redetect process in an Agreement.

**1** In Partner Admin Tool, edit the Split Redetect Process.

- a** Select Header and trailer, and click Edit.
- b** On Header/Trailer Split Additional Settings window, click Add and then consider the following fields:

<b>Header</b>	Type a header name. For example: <i>Header1</i>
<b>Start Position</b>	set to 1
<b>Trailer</b>	Type a trailer name. For example: <i>Trailer1</i>
<b>End Position</b>	set to 1

- c** Click Save, OK.
- d** On Configuration for Split Redetect window, select the order manifest (split or main).
- e** Select the Order mode.
- f** Type a Record Separator character to use.

**2** Click OK.

## Splitting Large Files

Large files can be split into flat files by detecting the number of lines or sub-header.

**Before you start** Complete the following tasks before you proceed with any of the flat file splitting processes:

- Create Flat File and Flat Group.
- Add a Split Redetect process in an Agreement.

### ☐ Split by detecting the number of lines

- \_\_\_1 Edit the Split Redetect Process.
- \_\_\_2 Select Number of Lines, and click Edit.
- \_\_\_3 On Number of Lines Split Additional Settings window, consider the following fields:

<b>Header</b>	Type a header name. For example: <i>Header1</i>
---------------	---

<b>Header Record Count</b>	Type a numeric value.
----------------------------	-----------------------

<b>Trailer</b>	Type a trailer name. For example: <i>Trailer1</i>
----------------	---

<b>Trailer bytes hint</b>	Type a numeric value.
---------------------------	-----------------------

<b>Number of lines hint</b>	10
-----------------------------	----

<b>No Split Record</b>	For example: <i>XSPLIT</i>
------------------------	----------------------------

<b>Start Position</b>	Set all start position to 1
-----------------------	-----------------------------

- \_\_\_4 Click Save, OK, OK.

### ☐ Split by detecting the Sub-Header

**Before you start** Add a Split Redetect process in an Agreement.

- \_\_\_1 Edit the Split Redetect Process.
- \_\_\_2 Select Sub-Header, and click Edit.
- \_\_\_3 On Sub-Header Split Additional Settings window, consider the following fields:

<b>Header</b>	Type a header name. For example: <i>Header1</i>
---------------	---

<b>Header Record Count</b>	Type a numeric value.
----------------------------	-----------------------

<b>Trailer</b>	Type a trailer name. For example: <i>Trailer1</i>
----------------	---

<b>Trailer bytes hint</b>	Type a numeric value.
---------------------------	-----------------------

<b>Sub-header</b>	SubSplit
-------------------	----------

<b>Start Position</b>	Set all start position to 1
-----------------------	-----------------------------

\_\_4 Click Save, OK, OK.

## ❑ Split by detecting Multi Sub-Header

**Before you start** Add a Split Redetect process in an Agreement.

\_\_1 Edit Split Redetect Process.

\_\_2 Select Multi Sub-Header and click Edit.

\_\_3 On Multi Sub-Header Additional Settings dialog, consider the following fields:

<b>Header</b>	Type a header name. For example: <i>Header1</i>
<b>Header Record Count</b>	Type a numeric value.
<b>Trailer</b>	Type a trailer name. For example: <i>Trailer1</i>
<b>Trailer bytes hint</b>	Type a numeric value.
<b>Start Position</b>	Set all start position to 1
<b>Regex Detection</b>	Define a regular expression as bases of split detection.

\_\_4 Click Save, OK, OK.

## ❑ Split when there is no recordtype in the file

\_\_1 Use modify with regular expression: "(.\*)\$"

\_\_2 Replace Head\n\$1.

\_\_3 Use a split redetect with the header and trailer:

<b>Header</b>	Type a header name. For example: <i>Head</i>
<b>Trailer</b>	Type the ID trailer name. For example: <i>ID</i>

\_\_4 On the children, use **modify record** and remove the **Head** before you reprocess the record.

## Splitting on N:th(property) Occurrence

In Split Redetect process for large files, messages can be split based on every *N:th* occurrence regardless of which expression will match the splitter.

1 Create a Flat File and a Flat Group.

2 Add a Split Redetect Process in the Agreement.

3 Edit the Split Redetect Process.

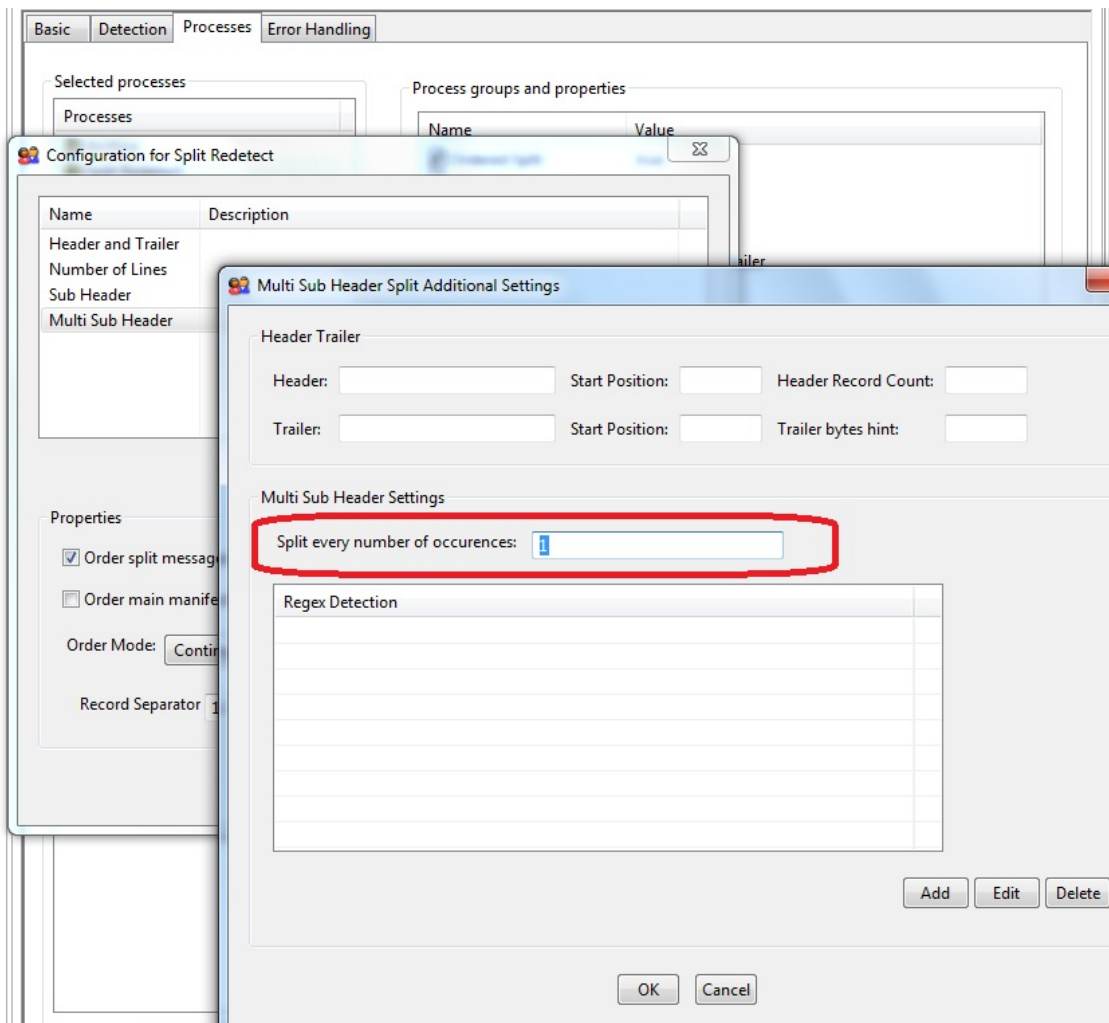
The Configuration for Split Redetect window is displayed.

- 4 Select Multi Sub Header, and click Edit.

The Multi Sub header Split Additional Settings window is displayed.

**Note:** This section is enhanced for message split based only on every *N:th* occurrence.

- 5 On **Split every number of occurrences** field, type a numeric value to set the basis of split. The default value for this field is set to "1".
- 6 Click Add, OK.



## Send WebService

- ["Using Send Web Service in Agreement" on page 133](#)

## Using Send Web Service in Agreement

Use this procedure to send the agreement messages as a web service to an external resource.

**Before you start** The messages to be used in web service must be formatted according to SOAP (Simple Object Access Protocol) specifications. Use web Service with XML Transform, XML Transform, or Apply Envelope processes.

- 1 Access the Agreement view tab. Create a new, or select an existing agreement to use.
- 2 Access the Detection tab to update the detections for this agreement.
- 3 Access the Process tab, right click in Selected processes pane, select **Send Web Service**.

The Configuration for Send Web Service dialog is displayed.

- 4 On Service Properties, update these fields:

<b>End point address</b>	Type a working http address for the end point. For example: <code>http://localhost:8080/services/Version?</code>
<b>Content type</b>	Type "text/xml"
<b>SOAP action</b>	Type a SOAP action value. For example, the path to a working end point address. <ul style="list-style-type: none"><li>• SOAP 1.1, type <code>text/xml</code></li><li>• SOAP 1.2, type <code>application/soap+xml</code></li></ul>

- 5 Select **Has Response**, if you will require for a reply.
- 6 On User Configuration, select either Anonymous user or Specify a user.

**Note:** If web service needs authentication, select Specify a user and type a username and password.

- 7 Click OK, and Save.
- 8 Verify that the Send Web Service is added in the Selected processes pane.

- "Partner Agreement Overview" on page 134
- "Creating Partner Agreement Folder Structure" on page 134
- "Creating New Agreement " on page 135
- "Setting up a process flow" on page 136
- "Viewing Agreements" on page 136
- "Import/Export Agreement Override Concepts" on page 137
- "Importing Agreements" on page 141
- "Exporting Agreements" on page 142
- "Deleting Agreements or Agreement group" on page 142

## Partner Agreement Overview

A Partner Agreement contains the agreement information between you and your partners. M3 Enterprise Collaborator uses the agreement information to send and receive business messages between you and your partners.

For a list of available predefined data used when creating an agreement, on Partner Admin Tool menu, select Manage.

## Creating Partner Agreement Folder Structure

Use this procedure to create a process flow and manage your partners and agreements in Partner Admin tool.

You can have one folder each for inbound and outbound agreements, or add new folders to set the structure of your partners and agreements. The inbound Test Agreement **Test\_100** and **Test\_20** are delivered with the package.

**Important:**

- Validate new structures for broken or duplicate agreements.
- If you need to create new groups, create the groups outside of the Test Agreement group.

- 1 Access the Agreement View tab.
- 2 Right-click on a blank space in the Partner Agreements pane, or on a node to contain your new group, select **Insert Group**.
- 3 Type a name for the new folder.

**Important:** The parent folder for your agreement is your partner. This folder name is shown as your partner agreement name in MEC Administration interface.

- 4 Create new agreements connected to the folder in the existing or new structure.  
For more information, see "[Creating New Agreement](#) " on page 135.
- 5 Save and exit.

## Creating New Agreement

Use this procedure to create new partner agreements.

- 1 Access the Agreement View tab.
- 2 Right-click on a parent folder to contain your new agreement, select **Insert Agreement**.

**Important:** The parent folder for your agreement is your partner. This folder name is shown as your partner agreement name in MEC Administration interface.

- 3 Access the Basic tab. For your new agreement, consider the following fields.

**Important:** When creating message agreements ensure that all field definitions are completed. If only the "Name" field was provided, an exception error may occur resulting to the message staying in "Detect" state. When this happens, review and complete the agreement field definitions.

<b>Name</b>	Required name for this agreement.
<b>Description</b>	Optional. A brief description for this agreement.
<b>Creator</b>	Optional. Name of agreement author.
<b>Email</b>	Optional. Email address used by this agreement.
<b>Priority</b>	Set the numeric priority level for this agreement.

- 4 Click Add to set the agreement control properties. Define a name and a value for each property.

**5** Save the new agreement.

For more information on Process Flow steps, see "[Message Processes](#)" on page 108.

## Setting up a process flow

Use this procedure to set up a process flow to run in MEC.

- 1** Access the Agreement View tab and select an agreement to use.
- 2** Access the Processes tab. In Selected processes pane, right-click on a blank space and select a process step from the list.
- 3** At the prompts, define the process details and confirm to add.

For more information on field details and available options for each process step, see "[Process Steps](#)" on page 144.

- 4** Save the Process Flow when finished.
- 5** Save the message after adding process steps.

Repeat steps 2-5 to add as many process steps as you may require to fulfill your process flow structure.

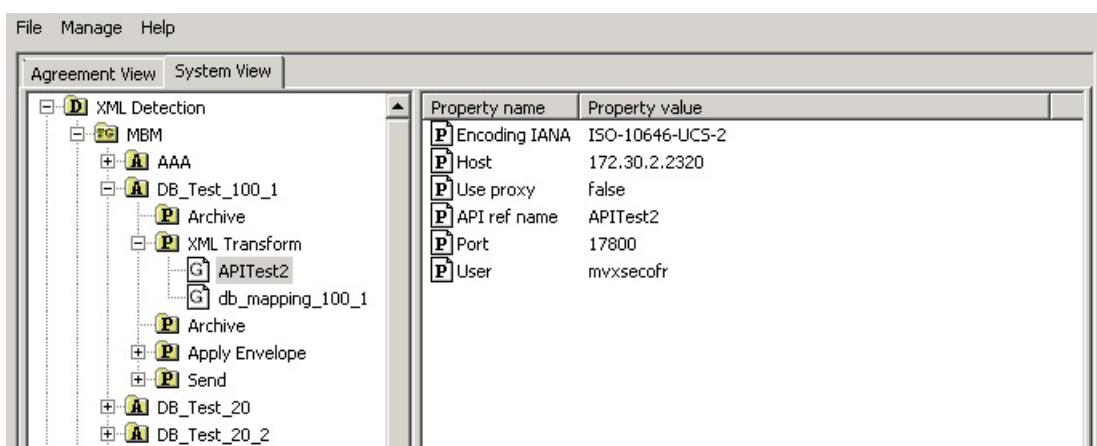
## Viewing Agreements

Use this procedure to view agreement by Detection and Target Groups

- 1** In Partner Admin Tool, access System View tab.
- 2** Expand the XML detection view to display its target groups.
- 3** Click on the node containing the agreements you want to view.
- 4** Expand the view of an agreement to show its process flow and process steps.

In the right pane, details of the property name and property value of currently selected agreement is displayed.





For more information on Message processes, see "[Message Processes](#)" on page 108.

## Import/Export Agreement Override Concepts

Agreement override allows you to change the agreement target values, adjust specific group control properties, or to conform an agreement with the business setup on its target environment. You can use the override function on a group of agreements instead of editing individual agreements. In an xml format, define the (old) values to override and the new values to override it with. Partner Admin tool refers to this set of instructions to run on imported agreements.

### Example scenario

To setup BE BODs in MEC you need to import more than 100 agreements in Partner Admin. A huge number of agreements in this group requires an update on Company Number (CONO) specific to its target environment. After importing agreements groups, you would normally edit individual agreements. You can use agreement override parametrized import/export feature to update the target values by agreement groups.

An empty xml file is in the agreement .agr file package. Rename the .agr file to .zip file, then open the xml template file.

- 1 Define the target conditions.
- 2 Define new values for the target.
- 3 Update the overrides.xml template file with details from steps 1 and 2.
- 4 Import the .agr file.

For more information, see "[Importing Agreements](#)" on page 141.

## Example Types of Override

### Example 1: TargetOverride

Objective: change the target value to "14", if the conditions of **TargetItem** in *<conditions>* has a match.

```
<AgreementOverride>
  <overrides>
    <TargetOverride>
      <conditions>
        <TargetItem>
          <name>hub:5_elementvalue01</name>
          <path>/EventData/Document/ElementData/Value[1]</path>
          <value>750</value>
          <defaultNamespace></defaultNamespace>
          <formattedXPath>/EventData/Document/ElementData/Value[1]</formattedXPath>
        </TargetItem>
        <TargetItem>
          <name>hub:4_elementname01</name>
          <path>/EventData/Document/ElementData/Name[1]</path>
          <value>CONO</value>
          <defaultNamespace></defaultNamespace>
          <formattedXPath>/EventData/Document/ElementData/Name[1]</formattedXPath>
        </TargetItem>
      </conditions>
      <name>hub:5_elementvalue01</name>
      <value>14</value>
      <mandatory>true</mandatory>
      <default>VALUE</default>
    </TargetOverride>
  </overrides>
</AgreementOverride>
```

### Example 2: AgreementGroupPropertyOverride

Objective: update the property value to "A\_1", if **PropertyItem** named "**nameA**" in the *<condition>* has a match.

```
<AgreementOverride>
  <overrides>
    <AgreementGroupPropertyOverride>
      <conditions>
        <PropertyItem>
          <key>nameA</key>
          <value>valueA</value>
        </PropertyItem>
      </conditions>
      <name>nameA</name>
      <value>valueA_1</value>
      <mandatory>true</mandatory>
      <operation>UPDATE</operation>
    </AgreementGroupPropertyOverride>
  </overrides>
</AgreementOverride>
```

### Example 3: AgreementGroupPropertyOverride

Objective: insert a new group control property [**nameB**, **valueB**], if property does not exists.

```
<AgreementOverride>
  <overrides>
```

```

    <AgreementGroupPropertyOverride>
      <conditions>
        <PropertyItem>
          <key></key>
          <value></value>
        </PropertyItem>
      </conditions>
      <name>nameB</name>
      <value>valueB</value>
      <mandatory>true</mandatory>
      <operation>INSERT</operation>
    </AgreementGroupPropertyOverride>
  </overrides>
</AgreementOverride>

```

## Example 4: Combined

Objective: Put together all three previous examples in one xml file.

```

<AgreementOverride>
  <overrides>
    <TargetOverride>
      <conditions>
        <TargetItem>
          <name>hub:5_elementvalue01</name>
          <path>/EventData/Document/ElementData/Value[1]</path>
          <value>330</value>
          <defaultNamespace></defaultNamespace>
          <formattedXPath>/EventData/Document/ElementData/Value[1]</formattedXPath>
        </TargetItem>
        <TargetItem>
          <name>hub:4_elementname01</name>
          <path>/EventData/Document/ElementData/Name[1]</path>
          <value>CONO</value>
          <defaultNamespace></defaultNamespace>
          <formattedXPath>/EventData/Document/ElementData/Name[1]</formattedXPath>
        </TargetItem>
      </conditions>
      <name>hub:5_elementvalue01</name>
      <value>14</value>
      <mandatory>true</mandatory>
      <default>VALUE</default>
    </TargetOverride>
    <AgreementGroupPropertyOverride>
      <conditions>
        <PropertyItem>
          <key>asdf</key>
          <value>asdf</value>
        </PropertyItem>
      </conditions>
      <name>nameB</name>
      <value>valueB</value>
      <mandatory>true</mandatory>
      <operation>INSERT</operation>
    </AgreementGroupPropertyOverride>
    <AgreementGroupPropertyOverride>
      <conditions>
        <PropertyItem>
          <key>nameA</key>
          <value>valueA</value>
        </PropertyItem>
      </conditions>
      <name>nameA</name>
      <value>valueA_1</value>
      <mandatory>true</mandatory>
      <operation>UPDATE</operation>
    </AgreementGroupPropertyOverride>
  </overrides>
</AgreementOverride>

```

## overrides.xml template

Here is a content overview of the xml template. Update this file to contain your override details.

```
<AgreementOverride>
  <overrides>
    <TargetOverride>
      <conditions>
        <TargetItem>
          <name>replace me</name>
          <path>replace me</path>
          <value>replace me</value>
          <defaultNamespace>replace me</defaultNamespace>
          <formattedXPath>replace me</formattedXPath>
        </TargetItem>
      </conditions>
      <name>replace me</name>
      <value>replace me</value>
      <mandatory>false</mandatory>
      <default>VALUE</default>
    </TargetOverride>
    <AgreementGroupPropertyOverride>
      <conditions>
        <PropertyItem>
          <key>replace me</key>
          <value>replace me</value>
        </PropertyItem>
      </conditions>
      <name>replace me</name>
      <value>replace me</value>
      <mandatory>false</mandatory>
      <operation>UPDATE</operation>
    </AgreementGroupPropertyOverride>
  </overrides>
</AgreementOverride>
```

## Valid overrides.xml values

- TargetOverride.mandatory = true/false
- TargetOverride. Default = SAME/VALUE
- AgreementGroupPropertyOverride.mandatory = true/false
- AgreementGroupPropertyOverride= UPDATE/INSERT

## Rules and limitations

- Keys are unique. For **AgreementGroupPropertyOverride**, insert will not occur if the key is already existing.
- No target override occurs, if there is a match and **TargetOverride.default** = SAME
- An override will not be executed, if **Mandatory** = **false**
- Multiple overrides can be included in a single overrides.xml
- **TargetOverride** can have multiple *<TargetItem>* conditions.
- **AgreementGroupPropertyOverride** can have multiple *<PropertyItem>* conditions.

- **AgreementOverride.overrides** in overrides.xml can have multiple **AgreementGroupPropertyOverride** and **TargetOverride**
- Override process order within the xml file is from top to bottom.

### When to feed the override file

You can provide the override file during agreement or agreement group export or import.

## Importing Agreements

Multiple individual, or a group of, agreements can be imported into Partner Admin from your selected folder location.

#### **Important:**

- Importing can only be done by individual users.
- FTPScriptPollin, DAF, and Data Translation are not supported.

#### **Note:** Import limitations

- External schema files referenced by a WSDL of a webservice definition entry.
- External scripts referenced by an ftp script file entry.

- 1 Access the Agreement View tab, select a folder within which to add imported agreements to.
- 2 Right-click and select **Import Agreement/group**.
- 3 At the confirmation prompt for "Apply Agreements Override", select whether agreements override will be used or not.

**Note:** If you are using an agreement override click Yes and browse to the folder location of the xml file containing the override rules to use.

- 4 Browse to the folder containing **.agr** file to import. Click Open.

An **.agr** file can contain one or several agreements.

- 5 Click Finish.

Successfully imported agreements are shown as a list of new agreements in the Agreement View tab.

- 6 To check for newly imported API Reference entry, access Manage > Communications > M3 API.

**Note:** If there are new entries, you must re-enter the password.

## Exporting Agreements

Multiple individual agreements, or agreement groups, can be exported from Partner Admin to your selected folder location. You can use export to back up agreements or agreement group. When exporting, check the trade.log file for activity details. Use Import to copy back exported agreements or agreement groups, ensure that you have created or selected to use an API Connection from PA menu, Manage > Communication > API

**Important:**

- Multiple users can run export simultaneously.
- FTPScriptPollin, DAF, and Data Translation are not supported.

**Note:** Export limitations

- External schema files referenced by a WSDL of a webservice definition entry.
- External scripts referenced by an ftp script file entry.

- 1 Access Agreement View tab. Select a folder from which to export agreements from.
- 2 Right-click and select **Export Agreement/group**.
- 3 At the confirmation prompt for "Apply Agreements Override", select whether agreements override will be used or not.

**Note:** If you are using an agreement override click Yes and browse to the folder location of the xml file containing the override rules to use.

- 4 Browse to a folder location to contain your exported .agr file. Type a new filename and click Save.

**Important:** Ensure that there are no duplicate filenames to avoid data overwrite.

- 5 Click Finish.

Successfully exported agreements are shown as new .agr files on your selected location.

## Deleting Agreements or Agreement group

You can simultaneously delete multiple individual agreements, or delete agreements by group.

**Important:** Delete has no undo function.

If you want to create a back up, use Export or Export Agreement Group before deleting agreements.

### **To delete multiple agreements**

- 1 Access Agreement View tab. Select multiple agreements to delete.
- 2 Right-click and select **Delete**.
- 3 Click Yes at the delete confirmation prompts.

### **To delete agreement group**

- 1 Access Agreement View tab. Select an agreement group folder to delete.
- 2 Right-click and select **Delete group**.
- 3 Click Yes at the delete confirmation prompts.

- ["Regular Process Steps" on page 145](#)
- ["Error Handling Process Steps" on page 152](#)
- ["Process Steps Applicable to both Regular and Error Handling" on page 154](#)

MEC executes a process flow based on the order of the process steps that you define for messages in an Agreement. There are two types of message handling process flows.

- **Regular process steps** - use this to define a regular process flow for messages.
- **Error handling process steps** - use this to handle errors encountered in regular process flow.
- **Process steps applicable to both regular and error handling** - use this to handle errors in both process steps.

For more information on Error Handling process steps, see ["Error Handling" on page 158](#)

The table here shows a complete list of available MEC process steps.

Regular Handling Process Steps	Error Handling Process Steps	Applicable to both regular and error handling process flows
Apply Envelope	Apply Envelope	Apply Envelope
Archive	Create ConfirmBOD	
Check Order		
DAF Archive		
EDCDIC/COBOL to XML		
Flat to XML		
Modify Flat		
Outbound MBM Status	Outbound MBM Status	Outbound MBM Status
Re-detect	Retrieve MBM Identifier	



<b>Regular Handling Process Steps</b>	<b>Error Handling Process Steps</b>	<b>Applicable to both regular and error handling process flows</b>
Reroute		
Send	Send	Send
Send Web Service		
SNA Send		
Split Redetect		
Validate		
XML to EBCDIC/COBOL		
XML to Flat		
XML Transform	XML Transform	XML Transform
XSL Transform	XSL Transform	XSL Transform

## Regular Process Steps

- ["Archive" on page 146](#)
- ["DAF Archive" on page 146](#)
- ["Check Order" on page 147](#)
- ["Transform Process Steps" on page 148](#)
- ["Modify Flat Records" on page 149](#)
- ["Re-Detect" on page 150](#)
- ["Split Redetect" on page 150](#)
- ["Reroute" on page 150](#)
- ["Send Web Service " on page 150](#)
- ["SNA Send" on page 151](#)
- ["Validate" on page 152](#)

Process steps are added to define a complete process flow in existing agreements. The following list of process steps are used in regular MEC process flows. On Partner Admin tool Agreement View tab,

select an Agreement to use. Then, access the **Processes** tab in the right pane to define a process flow.

**Important:** Save the message after adding a process step.

For a list of error handling process steps, see "[Error Handling Process Steps](#)" on page 152

## Archive

This regular process step archives a message in the MEC central file archive folder. Depending on the output of the preceding process step, messages will be saved in specific folders, for example:

If it fails, archive will be saved in: `/mec_central_file/archive/persistence`

if successful, archive will be saved in: `/mec_central_file/archive/doc`

All archived messages are viewable in Grid MEC Management Archive page based on message UUID.

## DAF Archive

This is a regular process step used to create an Archive to DAF (Document Archive Foundation) based on the Basic Data and xml input values. In a process flow, you can have two DAF Archive process steps. Where the first DAF Archive process contains initial reference data and the second DAF Archive process contains the updates.

When you create the second DAF Archive process step that will contain archive updates only, select **Update only**. Updates are limited to attribute values. To update the manifest values, use **XML Transform** process step.

- For the attribute value, the manifest key is preceded by "daf:" {attribute\_name}.
- For sub-entity attributes, the manifest key is: "daf:" {subEntity\_Name}\_{attribute\_name}[instance].

After a DAF Archive process MEC releases the lock on a document file to make it accessible to other applications, including MEC.

### To add DAF Archive in a process flow

#### Before you start

- You must define a DAF Agreement to use.
- DAF Environment must be running and a DAF connection is already configured.  
In PA menu, select Manage > Communications > DAF (Document Archive Foundation) tab.

- 1 Access the Agreement View tab and select a DAF agreement to use.

- 2 Right click on Process tab, select **DAF Archive**.
- 3 On the **Configuration for DAF Archive** window, select a DAF connection to use.
- 4 Select your options:

<b>Update Only</b>	Select this to update only the archive file.
--------------------	--

<b>Delete the document file</b>	Select this to remove the document file from the sender application.
---------------------------------	--

- 5 On Run On Host (expression) field, type an expression to use. For example, "cfs".  
Where "cfs" refers to the central file system where MEC\_Central node is located.

**Note:** MEC now runs on different hosts. You must add a value to Run On Host.

- 6 Leave blank the "Specify File Location" or "File location Xpath" fields to enable DAF to archive itself and not require for a link to the corresponding document, or image.

If you will use these fields, consider the following:

<b>File name Xpath</b>	Type: <code>/DAF/FILENAME</code>
------------------------	----------------------------------

<b>File Location Xpath</b>	Type: <code>/DAF/FILELOC</code>
----------------------------	---------------------------------

- 7 Click OK, Save.
- 8 Verify that DAF Archive process is created and saved on MEC.

## Check Order

Check Order is used to process related messages, or a sequence of messages. It uses the value of the primary key and variation ID to determine if messages are related and what process order to follow. Both the primary key and variation ID must be available in the XML message. Their values are extracted using Xpath. If the primary key or variation ID values are not found, the message will have an exception.

### Primary Key

A primary key can be a single Xpath, or a combination of multiple Xpaths.

- If multiple messages have the same primary key value, the messages are related.
- Then, the Variation ID value is checked to determine the process order of messages.

### Variation ID

Variation ID (VID) uses numeric value, this is the basis of process order.

- If Variation ID is empty, message will be processed.

- If Variation ID has a value, it will be compared against the values of previously processed messages within the same primary key.
- If Variation ID contains an equal or lesser value, the current message will be rejected.
- If Variation ID contains a greater value, the current message will proceed. The value is used as basis of process order.

**Important:** In MEC Management UI page you can add, edit, or remove the functionality of Variation IDs to allow processing of previously skipped messages.

For example, in "Xpath No Attribute Existing", it means that the node with the Xpath should not have an existing attribute in it.

```
<A>  
<B attri_1="someValue">XXX</B>  
<B attri_1="someValue">XXX</B>  
<B >333</B>  
</A>
```

Then, we will get **333**.

## Transform Process Steps

For more information on transform process steps, see "[XML Processing](#)" on page 108

### XML to EBCDIC/COBOL

Transforms outgoing message to an EBCDIC/COBOL encoded file according to a specific flat file definition.

- 1 Access the Processes tab. Right click on a blank space in the Selected processes pane, select **XML to EBCDIC/COBOL**.
- 2 On **Configuration for XML To EBCDIC/COBOL** window, select an enabled flat definitions to use and click OK.

To enable Flat file definitions, in PA menu select, Manage > Flat Definitions.

For more information, see the section on Setting Up Flat Definition with COBOL data types in *M3 Enterprise Collaborator Administration Guide*.

### XML to FLAT

Transforms outgoing message to a flat file according to a specific flat file definition.

- 1 Access the Processes tab. Right click on a blank space in the Selected processes pane, select **XML to FLAT**.
- 2 On **Configuration for XML To Flat** window, select an enabled flat definitions to use and click OK.

To enable Flat file definitions, in PA menu select, Manage > Flat Definitions.

## FLAT to XML

Transforms the incoming flat file message to an XML file according to a specific flat file definition.

- 1 Access the Processes tab. Right click on a blank space in the Selected processes pane, select **Flat to XML**.
- 2 On **Configuration for Flat to XML** window, select an enabled flat definitions to use and click OK.  
To enable Flat file definitions, in PA menu select Manage > Flat Definitions.

## EBCDIC/COBOL to XML

Transforms an incoming EBCDIC/COBOL coded message to an XML file according to a specific coded flat definition.

- 1 Access the Processes tab. Right click on a blank space in the Selected processes pane, select **EBCDIC/COBOL to XML**.
- 2 On **Configuration for EBCDIC/COBOL to XML** window, select an enabled flat definitions to use and click OK.

To enable Flat file definitions, in PA menu select Manage > Flat Definitions.

For more information, see "Setting Up Flat Definition with COBOL data types" in *M3 Enterprise Collaborator Administration Guide*.

## XML to EBCDIC/COBOL

Transforms the outgoing message to an EBCDIC/COBOL encoded file, according to a specific flat file definition.

- 1 Access the Processes tab. Right click on a blank space in the Selected processes pane, select **XML to EBCDIC/COBOL**.
- 2 On **Configuration for XML to EBCDIC/COBOL** window, select an enabled flat definitions to use and click OK.

To enable Flat file definitions, in PA menu select Manage > Flat Definitions.

For more information, see the section on "Setting Up Flat Definition with COBOL data types" in *M3 Enterprise Collaborator Administration Guide*.

## Modify Flat Records

Use this process to replace or remove texts from within flat messages based on regular expression patterns.

For more information, see "[Modify Flat Records](#)" on page 117.

## Re-Detect

Re-Detect is a process step equivalent to sending a message from MEC, back into MEC. When it re-detects the current message no further processing can be done on the re-detected message.

For more information, see "[Split Redetect](#)" on page 123.

## Split Redetect

Split Redetect is a process step that splits a message based on the header, trailer, and regular expression settings. Split messages are sent back to MEC for redetection.

For more information, see "[Split Redetect](#)" on page 123.

## Reroute

This process step reroutes the message body. Use this together with Apply Envelope.

## Send Web Service

Use this process step to send the agreement messages as a web service to an external resource. The message should be formatted according to SOAP specifications. MEC XML Transform, XSL Transform, or Apply Envelope processes can be used to achieve this formatting.

**Before you start** You must set up Web Services Definition in PA menu, select Manage > Web Service Definitions. For more information, see "[Web Service Definitions Fields](#)" on page 65.

- 1 Select or create a new Agreement to use.
- 2 Access the Detection tab and update the detection.
- 3 Access the Processes tab. Right click on a blank space in the Selected processes pane, select **Send Web Service**.
- 4 On the Configuration for Send Web Service window, in Service Properties group:

<b>End point address</b>	Type a working http address for the end point access, for example: <code>http://localhost:8080/services/Version?</code>
<b>Content type</b>	Type a content type, for example: for SOAP 1.1, type <code>text/xml</code> for SOAP 1.2, type <code>application/soap+xml</code>
<b>SOAP action</b>	

- 5 On User Configuration group, select to use an Anonymous user, or Specify a user.

**Note:** If web service needs authentication, select Specify a user and type a username and password.

- 6 Click OK and save.

## SNA Send

Sends a message using the LU6.2/SNA communication protocol. For more information, see the SNA Examples section in *M3 Enterprise Collaborator Administration Guide*.

- 1 Access the Processes tab. Right click on a blank space in the Selected processes pane, select **SNA Send**.

- 2 On the Configuration for SNA Send window, define the LU6.2 Conversation Characteristics.

**Partner LU name** Type the name of the Partner LU Name to use.

**Mode Name** Type a model name to use.

**Transaction Program** Type a transaction program name.

**Name**

- 3 Select a synchronization level.
- 4 Select a security type.
- 5 Type a user name and password to connect to CAT/SNA.
- 6 Select your response and retry options.
  - **Has response** - to receive a message response based on your requested data.
  - **Automatic retry** - to retry sending only if error code 17 occurs (**CM\_DEALLOCATED\_ABEND**).
- 7 Type a Run On Host (expression) to use, for example "cfs".  
Where "cfs" refers to the central file system where MEC\_Central node is located.

**Note:** MEC runs on different hosts. Ensure that RunOnHost field has a value.

- 8 Click Add to set the Prepend Record Layout in the right pane.
  - a On the Field Details window, define the name, length, and value.
  - b Save.
- 9 Click OK and save.

## Validate

This process step validates an XML file against an XML schema (XSD). The schema location must be specified in the XML message.

- 1 Access the Processes tab. Right click on a blank space in the Selected processes pane, select **Validate**.

- 2 Update the Parser Properties:

<b>External Schema Location</b>	Type the path to External Schema Location
---------------------------------	---

---

<b>No Namespace Schema Location</b>	Type the path to No Namespace Schema Location.
-------------------------------------	--

- 3 Click OK.

## Error Handling Process Steps

- ["Create ConfirmBOD" on page 152](#)
- ["Retrieve MBM Identifier" on page 153](#)

Process steps are added to define a complete process flow in existing agreements. When a regular MEC process step encounters an error, define an error handling process flow to handle the process error. On Partner Admin tool Agreement View tab, select an Agreement to use. Then, access the **Error Handling** tab in the right pane to define a process flow that will handle the process error.

**Important:** Save the message after adding an error handling process step.

For a list of regular process steps, see "[Regular Process Steps](#)" on page 145

## Create ConfirmBOD

This error handling process step creates a ConfirmBOD message to be sent to an IONDBOut. Create ConfirmBOD has no editable properties, it creates a ConfirmBOD using hard coded values, manifest data, and received (.rcv) file. The user defined XML document for ConfirmBOD is created in the format [XPath] = [value]. These Xpath values must match the Target XPath defined in Agreement Detections. When you add this process step you must also define the IONDBIn/IONDBOut channels that will poll ION messages from specified ConnectionURL.



## Values

There is a maximum of 24 elements and attributes set in ConfirmBOD and you can set some values in different ways. This is indicated by 1), 2), 3) etc.

The ConfirmBOD values can be:

- hard coded.  
For example "2.6.2"
- a value for a mapping manifest item, set in the mapping.  
For example, "map:ionFromLogicalId"
- a value for an agreement control property, set for the agreement, or one of its parent groups.  
For example "ionFromLogicalId"
- taken from the original BOD.

that is the received file, the XPath points where to fetch this value from the original BOD.

for example: `/pre:*/pre:DataArea/pre:*/pre:TenantID`

Where: "pre" is an internal namespace prefix, and asterisk (\*) means wildcard.

If a mandatory value is not found, the process will stop, throwing an exception with an error message, for example, "No FromLogicalId found"

## ConfirmBOD XMLSpecifications

Using the **Tenant ID** as an example, here is how Create ConfirmBOD process flows:

```
/ConfirmBOD/DataArea/Confirm/TenantID = 1) mapping manifest item "map:ionTenantId" 2) mapping
manifest item "map:m3beTenantID" 3) original BOD TenantID (/pre:*/pre:DataArea/pre:*/pre:TenantID)
4) exception "No TenantID element found."
```

- first, try to get the value for the manifest item "map:ionTenantId" set in a mapping.
- If a value is not found, the process tries to find a value for another manifest item set in a mapping, "map:m3beTenantID".
- If still not found, the process tries to fetch the value for the **TenantID** element in the original BOD, for example from `/SyncItemMaster/DataArea/Sync/TenantID` (namespace prefixes is omitted) if the original BOD is a SyncItemMaster.
- If still not found, the Create ConfirmBOD process will stop, throwing an exception with the message, for this example, "No TenantID element found."

## Retrieve MBM Identifier

M3 Business Messages (MBM) contains UUID as an identifier that is used for tracking in M3/BE.

M3 sends MBM messages through MvxNGIn or MvxTGIn to MEC. Inappropriate detection, or improperly configured message agreement, may cause errors. MEC refers to the MBM Identifier to update the BE

that a particular MBM message encountered an error in MEC. Then, you can set up a reply message back to M3 about the status of the MBM based on its identifier indicating that it failed to process the message. Then, Outbound MBM Status pulls up the MBM Identifier.

There are two ways to set MBM Identifier Error Handling in MEC, one through the Agreement, and the other through the Channel. Although MBM Identifier is not meant to be used in an Agreement, it can be used in error handling.

## Agreement

When checking through an Agreement, the MBM identifier must be present in the manifest.

A properly set Agreement has MBM Identifier saved in the manifest. If an Xpath value is found, the manifest is automatically populated. To verify the populated manifest, in PA menu select Manage> Advanced > Populating Targets tab.

When MBM identifier has been set in the manifest, this error handling process will retrieve the identifier. Otherwise, this error process will parse the value of the node /MvxEnvelope/MBMIdentifier to look for the MBM identifier.

## Channel

When messages pass through a channel, MEC will try to find the MBM Identifier of messages in error on a channel level.

# Process Steps Applicable to both Regular and Error Handling

- ["Apply Envelope" on page 155](#)
- ["Outbound MBM Status" on page 155](#)
- ["SEND" on page 156](#)
- ["XML Transform" on page 156](#)
- ["XSL Transform" on page 157](#)

Process steps are added to define a complete process flow in existing agreements. The following list of process steps can be used on both regular and error handling MEC process flows. On Partner Admin tool Agreement View tab, select an Agreement to use. Then, access the **Processes** tab or the **Error Handling** tab in the right pane to define your required process flow.

**Important:** Save the message after adding a regular or error handling process step.

## Apply Envelope

Use this to apply a specific envelope to your outgoing message. This process step can be used on both regular and error handling process flows.

For more information, see "[Apply Envelopes](#)" on page 111.

## Outbound MBM Status

Use this process step to send MEC status in M3 database table "CBMSTA" back to M3 for outbound messages.

You can use this in regular processing or for error handling process on an Agreement level. For pre-detection level error handling, use the Retrieve MBM Identifier process prior to calling the Outbound MBM Status process.

- When MEC updates MBM message UUID, the MEC UUID is translated to an M3 ID.
  - When a status is called, MEC checks for the original M3 UUID.
- 1 Access the Processes tab. Right click on a blank space in the Selected processes pane, select **Outbound MBM Status**.
  - 2 On the **Configuration for Outbound MBM Status** window, consider the following fields:
 

<b>API Reference</b>	Select an API reference that corresponds to the M3 environment that sent the MBM initiator.  <code>setOutboundMBMStatus</code> <code>addOutboundMBMLog</code>
<b>NOK Handling</b>	If you want the agreement processing to continue, even when MEC cannot update the message status in M3, you must set " <b>NOK handling</b> " to " <b>NOK = Ignore</b> ".
<b>Status</b>	Type a numeric value that is > 20 and <= 90. For example "2". For example, 78
<b>Message</b>	Optional. Type a message to show for this status, for example, type <code>alpha 78</code> .
  - 3 Click OK.

### For example:

- In the message field, if you typed: `Agreement=cmn:agreement`
- The manifest that will be inserted into the text will be: `Agreement=partsOrderSub-14.1`
- The message will be truncated at: 78 characters.

**Note:** If the manifest item `mvx:mbmIdentifier` does not exist, or is blank, the Outbound MBM Status process will not do anything.

## SEND

This process step sends a message using a communication protocol. When added in a process flow, you must also select a communication channel to use.

If used without another process step the incoming message will be sent as it is using the specified communication protocol.

To see the **Send** information details, in PA menu, select Manage > Communication > Send.

- 1 Access the Processes tab. Right click on a blank space in the Selected processes pane, select **Send**.
- 2 Access **Configuration for Send** window.
- 3 On Properties group, select an encoding to use, for example `Latin1`.
- 4 Select a ZIP output to send a ZIP message to the output folder, then type a name in the ZIP entry name field.

**Note:** You can select to split the ZIP output in specific file size.

- 5 Click OK.

## XML Transform

XML transform makes it possible to run a mapping and transform the body part of the XML message according to the XML mapping.

**Important:** When you add this process step, you must also select which mapping you want to run. Only published mappings can be used. To view a list of published mappings, from PA menu select Manage > XML Mappings.

- 1 Access the Processes tab. Right click on a blank space in the Selected processes pane, select **XML Transform**.
- 2 On the **Configuration for XML Transform** window, do the following:
  - a Select a published mapping.
  - b Select an API Reference to use.

- c Select a Database Reference to use.
- d Select Add Schema Location to be able to define a schema location to use.

**Note:** If left blank, the mapping output schema name will be used by default.

- 3 Click OK to save.

For more information on un/publishing mappings, see *M3 Enterprise Collaborator ION Mapper User Guide*

For more information on XML, see "[XML Processing](#)" on page 108, "[Namespaces in XML Transform Process](#)" on page 172

## XSL Transform

Transforms an XML message according to an XSLT definition.

**Important:** To display a list of available XSLT definitions, you must add XSLT Definitions from PA menu, select Manage > XSLT Definitions.

- 1 Access the Processes tab. Right click on a blank space in the Selected processes pane, select **XSL Transform**.
- 2 On the **Configuration for XSL Transform** window, select an XSLT definition to use.
- 3 Update the value of the available transformation properties.

**Note:** For more information on XSL transform property description, see section 16 (Output) in: <http://www.w3.org/TR/xslt>

- 4 Click OK.

- ["Process Flow Error Handling" on page 158](#)
- ["Handling Error Mails" on page 159](#)
- ["Handling Receive Channel Errors" on page 163](#)

## Process Flow Error Handling

### Error Handling

In a process flow, when an error occurs an error-mail is sent and the message log displays the error details. You can use error handling to define a custom process flow on the source data to support basic communication channel or agreement-specific error conditions. Error handling supports the XSL Transform, XML Mapping, Apply Envelope, and Send processes. The input to error flow is the MEC error message document.

For more information, see `TestData\Schemas\Error_Message.xsd` in the **MEC installation folder**.

### Process Levels

You can set up error handling in any of the following process levels:

#### Pre-detection level

On pre-Detection level, the error flow is defined from the point when a message is received until it is sent for detection. Detection failures are also covered on this level.

Use XSL Transform, XML Mapping, Apply Envelope, Send, and Outbound MBM Status processes.

**Note:** Use the Retrieve MBM Identifier process prior to calling the Outbound MBM Status process in pre-detection error handling.

## Agreement level

On Agreement Level, the error flow is defined when an agreement is identified from detection point onwards, until the process is completed. If there are remaining errors, Reprocess and Retry will rerun the error flow. An additional value in this level is the SyncIn channel where the actual message state is sent to the message sender. Then, you can send a custom error handling that will be sent back to the message sender.

Use the Outbound MBM Status for Agreement level error handling process.

## Handling Error Mails

- ["Error Message in a single email" on page 159](#)
- ["Error Suppression Overview" on page 160](#)
- ["Error Suppression tasks" on page 162](#)

## Error Message in a single email

Error messages are grouped as an attachment and sent as a single email per agreement. The group agreement mail setting overrides the **ErrorMail.Limit** settings. In Grid MEC Configuration Page > Edit Properties, expand the view of ErrorMail and set the following properties based on your requirement.

- **ErrorMail.Limit.TimeFrameSeconds** - sets the duration in seconds from the last successful email before MEC sends and polls thread to send mail.
- **ErrorMail.Limit.Count** - specifies the maximum number of error mail messages that the recipient will receive based on your time setting in **ErrorMail.Limit.TimeFrameSeconds**

**Important:** Override will occur only if the error mails are specific to an agreement, and if an email is enables and a recipient is defined.

In earlier MEC versions using **ec.Properties**, refer to the following setting:

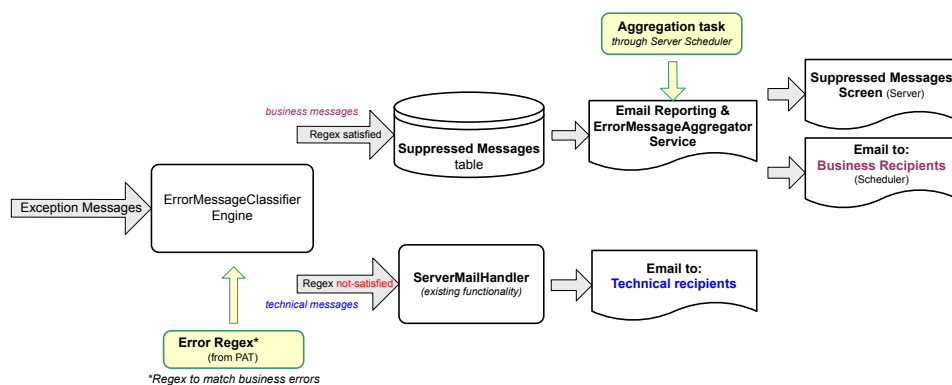
Property	Description
<b>ErrorMail.Agreement.TimeFrameSeconds</b>	Number of seconds that MEC waits before sending and polling thread to send mail. Example value: 10
<b>ErrorMail.Agreement.MaxAttachementSize</b>	Maximum size of attachment in bytes. Example value: 10000
<b>MEC.ErrorMailFolder</b>	Folder location where to save attachments. Example value: <code>archive/ErrorMail/</code>

For more information on MEC application property settings, see *M3 Enterprise Collaborator Administration Guide*.

## Error Suppression Overview

This section guides you on how to use the Error Suppression function using examples and process flow diagram.

On an overview, the diagram here shows that received exception messages are classified based on its origin, whether technical or business errors. Classification helps determine the recipient and information layout for error emails.





## Error Suppression

Error Suppression is the filtering of error messages based on Agreement specific regex.

- **Error Regex**

In Partner Admin tool, set up an Error Suppression Definition (ESD) specific to Agreement. Enable the definition applicable to the Agreement for processing. Although editable, you must first disable the definition before making any changes.

- **ErrorMessageClassifierEngine**

ErrorMessageClassifierEngine uses Error Suppression Definition as basis in determining whether to suppress a message or not

- **Technical message** - passes through the ServerMailHandler. Error messages are grouped as an attachment and sent as a single mail per agreement. This is an existing error handling flow.
- **Business message** - passes through Email Reporting and ErrorMessageAggregatorService. Errors are aggregated by parent agreement.

You define a regular expression (regex) to match business errors. In the event log, an "**INFO Message suppressed**" entry indicates an occurrence of non-technical message errors. The log is saved to a table for summary reporting (Error Aggregation). The error mail is sent following your server schedule setting. So instead of handling individual error emails, you will receive a summary report per error type.

All suppressed messages are viewable in Grid MEC Management Page > Message > Suppressed Errors Reports.

By default, there is one Error Mail suppression definition included when upgrading MEC database. From this default, find this regular expression, `NOK.{252}.*ERROR_CODE`. Edit '`ERROR_CODE`' with the specific error code from the mapping that you want to suppress in your agreement.

For more information about Regular Expression, see <http://docs.oracle.com/javase/tutorial/essential/regex/>

## Error Aggregation schedule

Error Aggregation are reports generated based on Error Suppression log table.

- Error aggregation schedules are independent from each other.

For example, if you have two schedules, A and B, schedule A can overlap its query time window with schedule B and still fetch a complete result. It does not matter if Schedule B has already generated a report of the same data (overlapped data).

- Reports from error aggregation schedule can be identified through "generated by <schedule name>".
- MEC mailer anti-spamming feature exempts mails received from error aggregation schedule.
- Aggregation task sends a summarized Error Aggregated email based on the configured CRON (scheduled time).

## Error Suppression maintenance

Maintenance tasks are set and defined in Grid MEC Management Page > Maintenance Schedule. For more information on MEC Management Page, see *M3 Enterprise Collaborator Administration User Guide*.

### Note:

- When several maintenance tasks are scheduled to run, you can select to enable or disable a maintenance from the list of schedules.
- For tasks with the same run time schedule, MEC will check the priority level setting to determine which task to run first. Priority level "1" will have the first priority.

## Clean up

- **Error Suppressed Log cleanup**

Error Suppressed Log cleanup is based on the existing Error Maintenance parameters. In Grid MEC Maintenance Schedule, when you schedule a maintenance log cleanup to delete UUIDs, the same UUIDs will be removed from both regular and error suppressed log tables.

- **Error Suppressed Document cleanup**

Error suppressed documents are archived in MEC central folder/archive/errorSuppressionReport. This archive can also be viewed through MEC Server UI from Message > SuppressedErrors Reports page.

There is only one purge maintenance task per schedule. After you add the maintenance schedule, set the details for purging error suppressed documents.

In the list of available Task Types, click the plus icon beside "Maintenance". On the Maintenance Task window, type the number of days aging for message document purging. Select to enable "Purge Error Suppressed Document".

## Error Suppression tasks

The table here provides an overview of the minimum required tasks to complete an Error Suppression process. Follow this series of tasks between PA tool and MEC Server Administration in Grid.

Step	Task	Description and tool to use
1	Use an Agreement	Create or select an Agreement to use. In PA, access the Agreement view tab.

Step	Task	Description and tool to use
2	Set up Error Suppression Definitions	User defined regular expression (regex) error suppression criteria  In PA, right click on an Agreement to use, select Error Mail Suppression.  Enable a definition to use.
3	Define error email	Enable email sending, define Host and Port properties  In grid, access the Configure Application page > Edit Properties > ErrorMail
4	Schedule error emails	In grid, access the Management Page > Schedules, click Add  Optional. Add an email recipient and include the email recipient configured in PA.
5	Add Error Aggregation tasks	Do this only after you add a schedule.  Add a task. Select "Error Aggregation" for task type.  <b>Note:</b> You must define an Error Suppression flow in Agreement to be able to add an Error Aggregation task.
6	Load messages	In grid, access the Management Page > Communication > Import Message
7	Check message status	In grid, access the Management Page > Message > Status
8	Monitor Event log	In grid, access the Management Page > Event > Log
9	View suppressed messages	In grid, access the Management Page > Server > Error Report
10	View error mails	Check out the email for these error mails.

## Handling Receive Channel Errors

- ["Handling Receive Channel Errors" on page 164](#)

## Handling Receive Channel Errors

Use these procedures for advanced error handling on detection and agreement levels.

### ☐ **Handle errors in agreement level**

- \_\_\_1** On PA menu, select Manage > Communication.
- \_\_\_2** Access Receive tab and select an existing receive channel.  
The field details about your selected channel is displayed on the top part of the window.
- \_\_\_3** Click Error.  
The Edit existing receive channel error handling properties window appears.
- \_\_\_4** On Selected Processes dialog box, edit or delete processes for this channel.
- \_\_\_5** On Process groups and properties dialog box, update the name and corresponding value.
- \_\_\_6** Click Save to store all the changes. Click Refresh to update the data.

### ☐ **Handle errors in pre-agreement level**

- \_\_\_1** Access Agreement View tab in the left pane and select an agreement.
- \_\_\_2** Access Error Handling tab in the right pane to define the custom error flow.  
Add from a list of available error processes.
- \_\_\_3** On Selected Processes dialog box, edit or delete processes for this channel.
- \_\_\_4** On Process groups and properties dialog box, update the name and corresponding value.
- \_\_\_5** Click Save to store all the changes. Click Refresh to update the data.

- ["Advanced PA tasks" on page 165](#)
- ["Namespaces" on page 168](#)

## Advanced PA tasks

### Defining File Name Extensions

Use this procedure to define all the file name extensions to be used when setting up the communication channels.

- 1 On Partner Admin Tool menu, click Manage > Advanced.
- 2 Click on File names tab.
- 3 Click New.
- 4 On Create a new unique file name object form, type a unique name and a Java class to use.  
For example: `com.intentia.ec.communication.UniqueFileNameInfix`
- 5 Click OK to save the new file name extension.

### Managing File Name Extensions

Use this procedure to modify details or remove File Names from MEC database.

- 1 On Partner Admin Tool menu, click Manage > Advanced.
- 2 Click on File names tab.
- 3 Select a file name.

**4** Perform any of the following:

Task	Steps
To edit	<ul style="list-style-type: none"><li><b>a</b> Click Edit.</li><li><b>b</b> On Edit existing fine name object window, modify the name and java class.</li><li><b>c</b> Click OK.</li></ul>
To delete	<ul style="list-style-type: none"><li><b>a</b> Click Delete.</li><li><b>b</b> At the delete prompt, click Yes.</li></ul>

**5** Close and exit the application.

## Defining Character Encoding

Use this procedure to define all the file character encoding to be used when setting up the communication channels.

- 1** On Partner Admin Tool menu, click Manage > Advanced.
- 2** Click on Character Encodings tab.
- 3** Click New.
- 4** On Create new encoding object form, type the name of the character, Java encoding, and XML encoding.
  - The default encoding for M3 Java is UCS-2 (UTF-16 with no BOM).
  - The default encoding for M3 RPG is Latin1.
- 5** Click OK to save the new character encoding.

## Managing Character Encoding

Use this procedure to modify details or remove character encoding from MEC database.

- 1** On Partner Admin Tool menu, click Manage > Advanced.
- 2** Click on Character Encodings tab.

3 Select an encoding object from the list.

4 Perform any of the following:

Task	Steps
To edit	<ol style="list-style-type: none"> <li>a Click Edit.</li> <li>b On Edit existing character encoding window, modify the Alias, Java encoding, and XML encoding.</li> <li>c Click OK.</li> </ol>
To delete	<ol style="list-style-type: none"> <li>a Click Delete.</li> <li>b At the delete prompt, click Yes.</li> </ol>

5 Close and exit the application.

## Defining Populating Targets

Use this procedure to define all the targets to populate from the manifest file in run time.

1 On Partner Admin Tool menu, click Manage > Advanced.

2 Click on Populating Targets tab.

3 Click New.

4 On Create new system target, consider the following fields information for the target to populate.

<b>Name</b>	a unique name
<b>XPath</b>	define the XPath that you want to retrieve values from
<b>Description</b>	type a brief description

5 Click OK to save the values.

# Namespaces

To support namespaces in MEC, the following topics are updated for your reference. This section also provides a check list for the consultants to verify the need for manual intervention during an upgrade of their current MEC installations.

- ["Namespace Overview" on page 168](#)
- ["Namespaces in XML Detection" on page 169](#)
- ["Namespaces in XML Transform Process" on page 172](#)
- ["Common Upgrade Scenarios Affecting Namespaces" on page 173](#)
- ["Schema Location" on page 175](#)

## Namespace Overview

MEC depends on identifying XML elements for various business processes. These unique XML elements are the bases of retrieving values in an input XML document and setting values in an output XML document.

In earlier MEC releases namespaces were not used to identify XML elements. In such cases, the XML elements or attributes are used with the same local name, but from different contexts. This results to MEC treating them as the same data.

MEC now uses XML namespaces to qualify XML elements or attributes in an XML where name conflicts can occur.

For example, a `<title>` element can have different meanings and can be applied to different purposes, depending on its context of use.

With namespaces, MEC can get the correct `<title>` element used in an XML. For example: if element `<address>` is used multiple times in a combination of vocabularies in an XML can give MEC the correct address element.

### Additional notes on namespaces

- Use of namespaces in XML documents is not required.
- A namespace defined by a prefix and a URI.
- A default namespace can be defined and it applies to all elements without prefix.

### Quick verification checklist

If problems occur, use the following quick verification checklist:

- Does the input XML use namespaces?



- Do the detection settings targeted to the input XML use namespaces?
- Does the mapping use namespaces (to verify, ask the mapping developer)?
- Do the namespace URIs match among the above three items?

## Namespaces in XML Detection

Detection is the process of matching or identifying input documents with the proper Partner Agreements. MEC does this in three ways: XML, Flat, and Channel Detection. We will focus on XML detection for our discussion on namespace changes.

In earlier MEC versions, detection was done without using namespaces, even if your XML document has, or is using namespace. Detection was handled by a namespace ignorant XML Path. The following figure shows the Detection tab in MEC v9.0.1.0.

### Without using namespace:

The `<Envelope>` on the second line is a simple example with no namespace.

```
<?xml version="1.0"?>
<Envelope>
  <Header>
    <delivery>
      <from>
        <address> this is detected!</address>
      </from>
    </delivery>
  </Header>
</Envelope>
```

### Using a namespace:

The namespace "env" on the third line is defined by URI `http://www.intentia.com/MBM Envelope`

```
<?xml version="1.0"?>
<Envelope xmlns="http://www.intentia.com/MBM"
  xmlns:env="http://www.intentia.com/MBM_Envelope"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xsi:schemaLocation="http://www.intentia.com/MBM MBM_Test_20_in.xsd">
  <Header>
    <env:delivery>
      <env:from>
        <env:address> this is detected!</env:address>
      </env:from>
    </env:delivery>
  </Header>
</Envelope>
```

### Using another namespace:

The namespace "env" on the third line is defined by URI

`http://www.intentia.com/MBM Another Envelope`

```
<?xml version="1.0"?>
<Envelope xmlns="http://www.intentia.com/MBM"
  xmlns:env="http://www.intentia.com/MBM_Another_Envelope"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xsi:schemaLocation="http://www.intentia.com/MBM MBM_Test_20_in.xsd">
  <Header>
    <env:delivery>
      <env:from>
        <env:address>this is detected!</address>
      </env:from>
    </env:delivery>
  </Header>
</Envelope>
```

## New fields in the Detections tab

Three new fields in the Detections tab were introduced to support the use of namespaces.

Field	Description
Actual Xpath	This field is automatically generated. It displays the internal representation of MEC for the XPath
Default Namespace URI	This field represents the default namespace. A default namespace is applied to all elements with out prefix.
Namespace Prefix and Namespace URI	Other namespaces are defined here. You must type this information for every target XML detection item.

## Example

Using the namespace, look at how the following new setting will affect our previous scenarios. Here are the XML target settings we will use:

- Name=**env:from**
- Default Namespace URI=**http://www.intentia.com/MBM**
- Other Namespaces
  - Namespace Prefix = **env**
  - Namespace URI = **http://www.intentia.com/MBM\_Envelope**

### A. Without namespace:

On the third line, the **<Envelope>** is a simple example with no namespace.

```
<?xml version="1.0"?>
<Envelope>
  <Header>
    <delivery>
      <from>
        <address>this is NOT detected!</address>
      </from>
    </delivery>
  </Header>
</Envelope>
```

```
</from>
. . .
```

## B. Using namespace:

On the third line, namespace env is defined by the URI:

**http:// www.intentia.com/MBM Envelope**

```
<?xml version="1.0"?>
<Envelope xmlns="http://www.intentia.com/MBM"
xmlns:env="http://www.intentia.com/MBM_Envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xsi:schemaLocation="http://www.intentia.com/MBM MBM_Test_20_in.xsd">
  <Header>
    <env:delivery>
      <env:from>
        <env:address>this is STILL detected!</address>
      </env:from>
    </env:delivery>
  </Header>
</Envelope>
```

**Note:** The prefix env does not have to match the prefix used in the detections tab. It could be a different element and can still be identified.

## C. Using another namespace:

On the third line, namespace env is defined by the URI :

**http:// www.intentia.com/MBM Another Envelope**

The elements are not in a different namespace and will not match.

```
<?xml version="1.0"?>
<Envelope xmlns="http://www.intentia.com/MBM"
xmlns:env="http://www.intentia.com/MBM_Another_Envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xsi:schemaLocation="http://www.intentia.com/MBM MBM_Test_20_in.xsd">
  <Header>
    <env:delivery>
      <env:from>
        <env:address>this is NOT detected!</address>
      </env:from>
    </env:delivery>
  </Header>
</Envelope>
```

## D. Using a default namespace:

This is another trivial scenarios that will match our defined settings.

On the second line, elements Envelope and Header still match the URI from the settings in Detection. Elements delivery, from, and address (although now in the default namespace) still match the URI from the settings in Detection.

```
<?xml version="1.0"?>
<Envelope xmlns="http://www.intentia.com/MBM"
  xmlns:env="http://www.intentia.com/MBM"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xsi:schemaLocation="http://www.intentia.com/MBM MBM_Test_20_in.xsd">
  <abc:Header>
    <delivery>
      <from>
        <address>this is STILL detected!</address>
      </from>
    </delivery>
  </abc:Header>
  . . .
</Envelope>
```

## E. Using a different prefix:

On the third line, the namespace abc is defined by URI:

**http:// www.intentia.com/MBM Envelope**

which is similar to the URI in Detection.

```
<?xml version="1.0"?>
<abc:Envelope xmlns="http://www.intentia.com/MBM"
  xmlns:abc="http://www.intentia.com/MBM"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xsi:schemaLocation="http://www.intentia.com/MBM MBM_Test_20_in.xsd">
  <Header>
    <abc:delivery>
      <abc:from>
        <abc:address>this is STILL detected!</address>
      </abc:from>
    </abc:delivery>
  </Header>
  . . .
</abc:Envelope>
```

## Namespaces in XML Transform Process

XML transform makes it possible to run a mapping and transform the body part of the XML message according to the XML mapping. When you add this process step select which mapping you want to run. And then select the two new available options in writing namespaces.

Namespaces also affect the XML Transform process in MEC. MEC uses the same logic for identifying elements when retrieving values from your input document. The namespaces used for retrieving values are taken from your input schema document in your XML mapping.

To support XML Namespaces, the XML Transform setup now has the following new selections.

### Ignore namespace of input file

This option allows you to ignore namespaces when retrieving values from your input file. When selected, the XML transform will extract the input values in the same way as it does with the mappings in earlier version.

## Add Schema Location

This option allows you to set the schemaLocation attribute in your output XML document. This is important if you have schema documents that you want to validate the output XML with.

## Empty Namespace

This options allows you to write empty namespaces if your input schema has no namespace declaration.

For example: `xmlns=""`

In XML Transform process settings, a new checkbox is added **"Do not write empty namespace"**. This feature is cleared by default.

## Namespace in output file

In output instance documents all used namespaces will be declared in the document's root tag.

This option allows you to not write NS of output file.

In XML Transform process settings, a new checkbox is added **"Do not write empty namespace for output file"**. This feature is cleared by default.

## Common Upgrade Scenarios Affecting Namespaces

Before going through the common upgrade scenarios, here are some notes on upgrading to the current MEC version.

When upgrading, MEC converts the MBM target detections with the default and other namespaces. This is the only target definition (groups and values) that MEC can infer the proper namespaces with.

Custom XML documents with custom namespaces, however, can not be converted because there is no way for MEC to determine what the correct namespace is. The namespace, if used by your XML document, has to be manually entered in the system through the Partner Administrator.

The following scenarios are classified based on message types being sent to MEC:

### If you are working with MBM messages:

MEC will process the following:

- Automatically enter the namespaces used by MBMs in the Detections tab.
- Create duplicate MBM targets for the old namespace URI of MBMs.

You can verify the MBM namespace you are using with the following:

- Oldnamespace: `http://www.intentia.com/Schemas/MeC`
- Namespace now: `http://www.intentia.com/MBM`

**Note:** MEC could have duplicated all the agreements with each having an MBM and an MBM Old detection. However, it could only cause unnecessary clutter to the Partner Administrator.

**If you are working with Mvx messages:**

Mvx messages do not use namespaces. No changes are needed in the settings for detection and XML transformation.

**If you are working with EDI messages:**

You can compare the namespace used by the input XML, from Amtrix or other EDI converter being used, and the settings in the Detections tab. The defined namespace URIs should match.

**If you are working with a custom XML document:**

You can do any of the following:

- Check if it uses namespaces.
- Check if the mapping, for XML Transform, uses namespaces. You can get this information from the mapping developer.

**If it uses namespaces:**

- Update the Detections tab target settings and enter the namespace prefix, or default, and URIs.

**Note:** The namespace URIs used in the detection and mapping should match. They are essentially the same document.

**If the input document has no namespace but the mapping has**

You can do any of the following:

- Check the Ignore namespace flag from the Edit XML Transform process panel.
- Update the input document and add the namespace.
- Update the mapping and remove namespace of input schema.

**If the input document has a namespace but the mapping has none**

- Check the Ignore namespace flag from the Edit XML Transform process panel.

## Schema Location

### Namespaces

In output instance documents all used namespaces will be declared in the document's root tag.

BOD instance example:

```
<SyncProductionOrder xmlns="http://schema.infor.com/InforOAGIS/2" releaseID="9.2" versionID="2.7.0">
```

If the output instance document contains mapped elements that are declared in custom namespaces these namespaces will also be declared.

BOD instance example:

```
<SyncProductionOrder xmlns="http://schema.infor.com/InforOAGIS/2"
xmlns:cust="http://schema.infor.com/InforOAGIS/Custom" releaseID="9.2" versionID="2.7.0">
```

For more information about custom namespaces, see *M3 Enterprise Collaborator ION Mapper User Guide*.

### Schema Locations

**For an XML processor to be able to validate the document**, schema locations must be given for all namespaces. These locations are given in the attribute "**xsi:schemaLocation**" as multiple value pairs, separated by a space. The first value is the namespace to use. The second value is the location of the XML schema to use for that namespace.

A schema (xsd file) can be located by a local file path, or an internet URL.

#### BOD instance example:

```
<SyncProductionOrder xmlns="http://schema.infor.com/InforOAGIS/2"
xmlns:cust="http://schema.infor.com/InforOAGIS/Custom"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schema.infor.com/InforOAGIS/2 SyncProductionOrder.xsd"
releaseID="9.2" versionID="2.7.0">
```

#### Example with a custom namespace using two custom schemas:

```
<SyncProductionOrder xmlns="http://schema.infor.com/InforOAGIS/2"
xmlns:cust="http://schema.infor.com/InforOAGIS/Custom"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schema.infor.com/InforOAGIS/2 SyncProductionOrder.xsd
http://schema.infor.com/InforOAGIS/Custom CustomHeader.xsd
http://schema.infor.com/InforOAGIS/Custom CustomDetail.xsd"
releaseID="9.2" versionID="2.7.0">
```

**For the XML processor that receives the XML document to be able to validate the XML document**, all xsd files must reside in the XML processor's current directory, otherwise it will not be able to locate the xsd files. If you want to provide a URL and/or a local directory path for the xsd files, you have to give a specific schema location value. If the mapping uses custom namespaces, you also have to type the locations for all custom schemas for the used custom namespaces.

If you type the schema location in the field "Schema Location", you can use any file paths or internet URLs for the schema files.

### Example schema location string and the resulting root tag:

`http://schema.infor.com/2.7.0/InforOAGIS/BODs/Developer/  
SyncProductionOrder.xsd`

```
<SyncProductionOrder xmlns="http://schema.infor.com/InforOAGIS/2"
xmlns:cust="http://schema.infor.com/InforOAGIS/Custom"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schema.infor.com/InforOAGIS/2
http://schema.infor.com/2.7.0/InforOAGIS/BODs/Developer/SyncProductionOrder.xsd"
releaseID="9.2" versionID="2.7.0">
```

### The same example with a custom namespace using two custom schemas:

`http://schema.infor.com/2.7.0/InforOAGIS/BODs/Developer/  
SyncProductionOrder.xsd`

`http://schema.infor.com/Custom Custom\UserAreaExtensions\Customer\  
CustomHeader.xsd`

`http://schema.infor.com/Custom Custom\UserAreaExtensions\Customer\  
CustomDetail.xsd`

```
<SyncProductionOrder xmlns="http://schema.infor.com/InforOAGIS/2"
xmlns:cust="http://schema.infor.com/InforOAGIS/Custom"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schema.infor.com/InforOAGIS/2
http://schema.infor.com/2.7.0/InforOAGIS/BODs/Developer/SyncProductionOrder.xsd
http://schema.infor.com/Custom Custom\UserAreaExtensions\Customer\CustomHeader.xsd
http://schema.infor.com/Custom Custom\UserAreaExtensions\Customer\CustomDetail.xsd"
releaseID="9.2" versionID="2.7.0">
```

#### Note:

- For custom schemas you also have to type the namespace URI.
- The maximum length of the field "Schema Location" is 256 characters.

## Tokens

There are some tokens that you can use to make it easier to type the schema location.

#### Note:

- The schema location string must be given in the format:  
`"defaultNamespaceSchema customNamespaceURI customNamespaceSchema  
customNamespaceURI customNamespaceSchema" etc.`
- The default namespace must not be given, it will be prefixed automatically.



## Custom Prefix

Instead of giving the custom namespace URI you can use the custom namespace prefix used in the mapping, for example "**cust:**", including the ending colon. This schema location string will give the same result as in the tokens above (tokens are marked in bold characters):

`http://schema.infor.com/2.7.0/InforOAGIS/BODs/Developer/SyncProductionOrder.xsd`

**cust:** Custom\UserAreaExtensions\Customer\CustomHeader.xsd

**cust:** Custom\UserAreaExtensions\Customer\CustomDetail.xsd

**Note:** If a custom namespace that is not used in the mapping is given, it is automatically removed, including the corresponding custom schema. This is valid in both instances, when the custom namespace prefix is used and when the custom namespace URI is used, in the schema location string.

## Schema Name

You can use the token "<**s**

If you have several custom schemas for the same custom namespace you only have to give the custom namespace-schema (token) pair once; all custom schemas used for the custom namespace will be listed automatically. This is because there is no way to pin-point a specific custom schema if there are several custom schemas for a custom namespace. This schema location string will also give the same result as the complete schema location string in the first example (tokens are marked in bold characters):

`http://schema.infor.com/2.7.0/InforOAGIS/BODs/Developer/<s> cust:`

`Custom\UserAreaExtensions\Customer\<s>`

You cannot change the default schema name when using this token. Although, you can add a prefix, typically a URL or a file path, and/or a suffix to the schema name. The file extension "xsd" is handled automatically.

### Example:

Tokens are marked in bold characters.

`http://schema.infor.com/2.7.0/InforOAGIS/BODs/Developer/<s> cust:`

`Custom\UserAreaExtensions\Customer\Prod_<s>_2`

```
<SyncProductionOrder xmlns="http://schema.infor.com/InforOAGIS/2"
xmlns:cust="http://schema.infor.com/InforOAGIS/Custom"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schema.infor.com/InforOAGIS/2
http://schema.infor.com/2.7.0/InforOAGIS/BODs/Developer/SyncProductionOrder.xsd
http://schema.infor.com/Custom Custom\UserAreaExtensions\Customer\Prod_CustomHeader_2.xsd
http://schema.infor.com/Custom Custom\UserAreaExtensions\Customer\Prod_CustomDetail_2.xsd"
releaseID="9.2" versionID="2.7.0">
```

- ["Writing a Custom Receive Channel" on page 178](#)
- ["Writing a Custom Send Channel" on page 183](#)

## Writing a Custom Receive Channel

This section provides information on how to set up incoming MEC Communication Channel. This also provides additional information that can help you configure a channel to meet the requirements of your organization.

- ["Custom Receive Communication Channel Overview " on page 178](#)
- ["Creating an Incoming Communication Channel" on page 179](#)
- ["Server-Side Implementation for Incoming Communication" on page 180](#)
- ["Adding Receive Protocol" on page 182](#)
- ["Configuring New Incoming Communication Channel" on page 183](#)

## Custom Receive Communication Channel Overview

The incoming communication channel enables MEC to receive messages. These messages can either be in XML or Flat File format.

When the channel accepts an incoming connection, the message is received based on the following protocol:

- 1 Reading the first four bytes.
- 2 Converting the first four bytes to an integer. This determines how many bytes the message contains and determines how many additional bytes are sent.
- 3 Reading the message to determine the number of bytes received.

To develop an incoming communication channel, you must first develop a sample channel. This sample channel is socket-based and listens for incoming connections on a certain port.

## Communication Plug-in

MEC Communication Plug-in allows your Business Engine (BE) to send and receive messages outside your organization. These messages can either be in XML or Flat file format. You can customize the MEC Communication Plug-in to send and receive messages based on the requirements of your organization.

Use the following file and source codes when setting up the MEC communication plug-in:

- Necessary jar files :  
**<MEC\_ROOT>\ec.jar and <MEC\_ROOT>\classes**
- Public MEC Classes documented with Javadoc :  
**<MEC\_ROOT>\Documentation**

## Creating an Incoming Communication Channel

MEC Communication Channel can be configured to receive incoming messages. You can do this by creating a channel class and implement different methods in the class.

- 1 Write an incoming communication channel class. The class must extend another class called **com.intentia.ec.shared.IncommingChannel**.
- 2 Implement the following abstract methods in the super class:
  - **initiateChannel**
  - **getDelayTime**
  - **cleanUpChannel**
- 3 Implement the **runChannel** method by creating the **selector** and the **ServerSocketChannel** objects.
  - If the select call to the Selector returns an integer strict greater than zero we will iterate over the **selectionKey** objects available to the **selector**.
  - If a key is in the acceptable state we accept the incoming connection and register the connection as readable to the **selector**. Otherwise, check if the key is in a readable state.
  - If the key is in readable state, then create a worker and submit that worker to the thread framework.
- 4 If the channel deals with sockets, then handle the incoming sockets in separate threads. The channel framework supports this through the **com.intentia.ec.shared.IThreadWorker** interface.

- 5 Instead of having the `runChannel` method receive the message, you can put that logic into the `runWork` method of the `IThreadWorker`. The `runWork` method will be called by a separate thread and the `runChannel` method will accept the incoming connection and create an `IThreadWorker` object.

## Server-Side Implementation for Incoming Communication

The process of creating an incoming communication channel involves the writing of a channel class, implementing its methods, and configuring it using the Partner Administration Tool.

This section provides you detailed information about the channel class and its methods.

### IncomingChannel Class

You must create an incoming channel class to enable MEC Communication Plug-In to receive incoming messages.

This class must extend the `IncommingChannel` class.

For example:

```
public class SampleIncommingChannel extends IncommingChannel { // Class
fields goes here }
```

**Need More Details? Check out the following concepts:**

- For a complete code listing of `SampleIncommingChannel`, see [Sample Code - SampleIncommingChannel.package](#).

### initiateChannel Method

The `initiateChannel` method contains the logic for configuring the channel. This abstract method must be implemented in the super class, using the `initiateChannel(ChannelProperty[] props)` method. If the initialization is successful, the return value is true. Otherwise, the return value is false.

`ChannelProperty` contains elements that hold the configuration data. The `IncomingChannel` class must be configured with the following properties:

- Port number (Port) – the number of the port on which the channel will listen for incoming connections.
- Maximum sub-threads (MaxSubThreads) – the maximum number of sub-threads that are allowed to run.
- Read timeout (ReadTimeOut)

For every property, the associated `ChannelProperty` object must be retrieved. From this object we then retrieve the value which can be the default value if a value is not specified. If the default value is not specified and the property is mandatory, then we use the hardcoded value for the property.

The sample code below shows how the property for the port number is retrieved:

```
String propKey = "Port";
// Retrieve the correct ChannelProperty from the array
ChannelProperty prop = getProperty(props, propKey);
String value = getValue(prop);
// If the property did not exist we return false.
if (prop == null) return false;
if (value == null) {
// If value is null but mandatory we issue a warning and
// use the default value defined globally.
if (prop.isMandatory()) {
cat.warn("Using internal default value: " + myPort);
}
}
else {
try {
int tmp = Integer.parseInt(value);
if (tmp <= 0) {
throw new IllegalArgumentException("Negative or
zero value.");}
myPort = tmp;
}
catch (Exception e) {
cat.warn(propKey + " property for SampleIncommingChannel
is invalid. Using internal default value: " + myPort);
}
}
```

## runChannel Method

The runChannel method contains the logic for receiving the message. This method contains logic for the following:

- Creating the ServerSocket object if not already created
- Accepting incoming connections
- Starting a worker that will receive the message

## getDelayTime Method

The time in milliseconds between each successive call to runChannel is given by the getDelayTime method. If this method returns 0, then the runChannel will be called over and over again with no delay.

The implementation of this method looks like this:

```
public long getDelayTime() {
return 0;
}
```

## ThreadWorker Class and runWork method

MEC has a framework for managing threads in an easy way. For example, there is a worker class that implements the `IThreadWorker`. The worker implements the actual receiving of the message.

The `IThreadWorker.runWork` method, which is called by the thread framework, implements logic for the following:

- Creating a manifest

- Reading the size of the message
- Reading the message
- Persisting the message
- Queuing the manifest

Retrieve a `ByteBuffer` object then store the first four bytes from the incoming socket to the said object. A manifest, which acts as a delivery note for an incoming message, is then created. Once the first four bytes are retrieved, convert them into an integer that defines the length of the message.

The `PersistMessage` method is then called with the socket and message length as two arguments. To allow further processing, this message is stored in a file that is accessible to the MEC Server.

## stopPlugIn Method

The `Select` call in the selector is blocking the thread for the communication channel, thus, an error might occur when stopping the channel. To unblock the selector, override the `stopPlugIn` method.

The implementation of this method looks like this:

```
public void stopPlugIn() {
    super.stopPlugIn();
    connectSelector.wakeup();
}
```

The `pausePlugIn` method needs to be overridden in the same way. But in this case it calls the `pausePlugIn` method.

## cleanUpChannel method

After the last call to the `runChannel` method, the `cleanUpChannel` method is called. As a result, the channel is given the chance to execute some logic before terminating. The socket-related objects eventually close.

## Adding Receive Protocol

Use this procedure to add a new receive protocol.

- 1 On Partner Admin Tool menu, click Manage > Advanced.
- 2 Click on Receive Protocols tab, and then click New.
- 3 Type a unique name and the Java class to use.

For example: `com.intentia.ec.communication.DiskIn`

- 4 Click OK to save the new protocol.

Each protocol has its own set of defined properties, for example: a property name, value, and description. An asterisk (\*) indicates whether or not the property is mandatory. Properties set as mandatory must either have an associated default value or you must add it.

For more information on receive channel protocol properties, see the following:

["Polling Type Channel Properties"](#) on page 26

["Server Type Channel Properties"](#) on page 35

## Configuring New Incoming Communication Channel

In order to use the newly created channel, you must configure it in the Partner Administration Tool.

- 1 On Partner Admin Tool menu, click Manage > Advanced.
- 2 Click on Receive Protocols tab and select an object from the list.
- 3 Click Edit.
- 4 On Edit existing receive protocol object, modify the name and Java class.
- 5 Right-click on the table of properties and select Insert Property.
- 6 Type the new property, default value, and description.
- 7 Select Mandatory if required.
- 8 Click OK.

## Writing a Custom Send Channel

This section provides information on how to set up outgoing MEC Communication Channel and additional information that can help you configure the channel and meet the requirements of your organization.

- ["Custom Send Communication Channel Overview"](#) on page 183
- ["Creating an Outgoing Communication Channel"](#) on page 184
- ["Server-Side Implementation for Outgoing Communication"](#) on page 184
- ["Configuring New Outgoing Communication Channel"](#) on page 186
- ["Configuring MEC with HTTPS Communication and Similar Protocols"](#) on page 187

## Custom Send Communication Channel Overview

The outgoing communication channel enables MEC to send messages. These messages can be in either XML or Flat File format.

To develop an outgoing communication channel, you must first develop a sample channel. This sample channel is socket-based and listens for outgoing connections on a certain port. Also, you must develop a channel that implements functionality for sending a message using the HTTPS protocol. This allows the HTTP server within the MEC Server to receive messages.

Using HTTPS means you have to manage digital certificates. This document does not contain any recommendations on how to manage certificates.

By following the examples given in this section, you will be able to create an outgoing communication channel for HTTPS. These sample codes use certificates in a safe and simple way. Since the source codes are only sample codes, you might have to complement them with additional code.

## Creating an Outgoing Communication Channel

MEC Communication Channel can be configured to send outgoing messages. You can do this by creating a channel class and implement different methods in the class.

- 1 Write an **HTTPSOut** class to enable MEC Communication Plug-In to send outgoing messages.

The outgoing communication class must implement the **com.intentia.ec.shared.IMessageSender** interface.

- 2 Create the properties dialog UI used by the **HTTPSOut** class.

- 3 Implement the following methods:

- **IMessageSender.send**
- **IRouting.createContent**
- **IRouting.setProperties**
- **IRouting.getProperties**

- 4 Set up the Partner Administration Tool.

## Server-Side Implementation for Outgoing Communication

The process of creating an outgoing communication channel involves the writing of a channel class, implementing its methods, and configuring it using the Partner Administration Tool.

This section provides you detailed information about the channel class and its methods.

### HTTPSOut Class

An **HTTPSOut** class must be written to enable MEC Communication Plug-In to send outgoing messages. The outgoing communication class must implement the **com.intentia.ec.shared.IMessageSender** interface.



This interface contains the following method:

```
public void send(InputStream messageStream, Manifest manifest, Properties props)
```

The method declaration has the following arguments:

- **messageStream** – stream from where the message to be send can be read from
- **manifest** – the delivery note containing metadata about the message
- **props** – a container holding the properties for the send module

For example, a sample class named **HTTPSOut** is created. The following code shows the **HTTPSOut** class implement the **IMessageSender** class.

```
public class HTTPSOut implements IMessageSender {
    // Class fields goes here
}
```

For a complete code listing of **HTTPSOut**, see [Sample Outgoing Channel](#).

## IMessageSender.send Method

To implement the send method, make sure the path and password are dynamically configured through the system properties, and the property values are extracted from the props argument. The properties also show the host, port, and the path that is used to create an URL. From the URL, get an **HttpsURLConnection** (UC), which should be configured to send data but not to receive data.

Since the MEC HTTP server uses basic authentication, use Base64-encoding to encode the user and password. To set the message as an HTTP parameter, use the **ClientHttpRequest** helper class. Finally, post the created request. If everything is properly set up and if you choose this channel, a message is sent using HTTPS.

## Partner Administration Tool

Before adding the new channel to the system, write a Java class that can present a dialog in the Partner Administration Tool. The Partner Administration Tool allows you to set the properties for the channel. When you select a send communication protocol in the Partner Administration Tool, different dialogs appear depending on the chosen protocol.

## Properties Dialog UI

To be able to set the corresponding properties for the outgoing communication channel that you are about to create, you must create a class displaying an equivalent dialog for the properties used in the **HTTPSOut** class. You will not create the dialog itself, just the buttons, fields, and other for the widgets inside the dialog.

A widget is a part of the Java SWT framework. The widget class is the abstract super class of all user interface objects. Widgets are created, disposed, and issue notifications to listeners when events occur which affect them.

The class containing the UI code must implement the interface `com.intentia.ec.shared.IRoutingUI`. It must contain the following methods:

- **`createContent`** – creates the content for this dialog
- **`setProperties`** – sets the properties to be displayed
- **`getProperties`** – returns the properties For this example, the sample class is named `HTTPSOutPanel`.

For a complete code listing of `HTTPSOutPanel`, see [Sample Outgoing Channel](#).

### **IRouting.createContent Method**

The **`createContent`** method contains code for creating all the widgets needed. The argument is a composite widget with a `FlowLayout` added to it.

### **IRouting.setProperties Method**

The **`setProperties`** method contains code for extracting the values of the given properties object and writing those values to the correct text widget when you open the dialog in edit mode. The following code shows how the value of the property `HTTPSOut.HOST` is written to the host text widget:

```
txtHost.setText(props.getProperty(HTTPSOut.HOST, ""));
```

### **IRouting.getProperties Method**

This method reads the values from the widgets and puts them into a Properties object and returns that object.

## **Configuring New Outgoing Communication Channel**

In order to implement the `HTTPSOut` channel class and the GUI class named `HTTPSOutPanel`, you must add them to the system using the Partner Administration Tool.

- 1 On Partner Admin Tool menu, click **Manage > Advanced**.
- 2 Click on **Send Protocols** tab, and then click **New**.
- 3 On **Create new send protocol object** window, type the protocol name, **Send Class** and **UI Class**.  
You can now create the following channel classes: `HTTPSOut` channel class and the GUI class named `HTTPSOutPanel`.
- 4 Click **OK**, **Close**.
- 5 Go to the **Manage > Communication** tab.
- 6 Click on **Send** tab and click **New** to create a new **Send** object.

- 7 On Create new send object window, define the Channel configuration, Basic configuration, and Unique File name fields.
- 8 Click OK to save the new configuration.

## Configuring MEC with HTTPS Communication and Similar Protocols

In order for MEC Server to support HTTPS, you must create a certificate for the HTTP server and put it in a keystore. You must also add information to the file: `HttpServer.xml`. For a complete code listing of `HttpServer.xml`, see [Sample Outgoing Channel](#).

In addition to configuring a MEC HTTP server to receive messages, configure the MEC Server that sends the messages.

During the SSL handshake, the server certificate is validated. To be able to validate the certificate, import the certificate into a trusted certificate store.

- 1 Access the Windows command prompt and navigate to the MEC Server folder.

- 2 Type the following:

```
keytool -keystore httpKeystore -alias jetty -genkey -keyalg RSA
```

- 3 At the prompts, ensure that you use the same password for the created keystore.

You now have a certified keystore named `httpKeystore`.

- 4 In order for MEC Server to be able to manage HTTPS, add the following lines to the file: `HttpServer.xml`

```
<Call name="addListener">
  <Arg>
    <New class="org.mortbay.http.SslListener">
      <Set name="Port">443</Set>
      <Set name="Keystore"><SystemProperty name=
        "jetty.home" default="."/>/httpKeystore</Set>
      <Set name="Password"#jetty4U</Set>
      <Set name="KeyPassword"#jetty4U</Set>
    </New>
  </Arg>
</Call>
```

- 5 Export the server certificate from the HTTP server keystore to a file by typing the following command:

```
keytool -export -keystore httpKeystore -alias jetty -file server.cer
```

**Note:** When using HTTPS, the port number is 443.

- 6 Copy the certificate to the folder where you want to create your trusted store. To import the certificate into a non-existing store, type the following command:

```
keytool -keystore trustedstore -import -alias jetty -file server.cer -
trustcacerts
```

---

# Sample Incoming Channel



This appendix contains a sample complete code listing for incoming channel.

## Sample Code - SampleIncommingChannel.package

```
import org.apache.log4j.Category;
import java.nio.channels.ServerSocketChannel;
import java.nio.channels.Selector;
import java.nio.channels.SelectionKey;
import java.nio.channels.SocketChannel;
import java.nio.ByteBuffer;
import java.io.IOException;
import java.net.InetSocketAddress;
import java.util.Set;
import java.util.Iterator;
import com.intentia.ec.shared.IncommingChannel;
import com.intentia.ec.shared.Manifest;
import com.intentia.ec.shared.IThreadWorker;
import com.intentia.ec.shared.dbitems.ChannelProperty;
import com.intentia.ec.server.ThreadPool;

public class SampleIncommingChannel extends
IncommingChannel {
    // A log4J category, used for logging.
    public static Category cat = Category.getInstance
(SampleIncommingChannel.class.getName());
    // The port number. Set it to be 112233 as default.
    private int myPort = 112233;
    // The read timeout. Set it to be 5000 msec as default.
    private int readTimeout = 50000;
    // We are dealing with incoming sockets, hence we
    need a ServerSocketChannel.
    private ServerSocketChannel serverChannel;
    // Since we are using java.nio we needs a Selector.
    private Selector connectSelector;

    public SampleIncommingChannel() {
        super(cat);
    }

    public boolean initiateChannel(ChannelProperty[] props) {
        // Start by extracting the property for the port value.
        String propKey = "Port";
        // Retrieve the correct ChannelProperty from the array
        ChannelProperty prop = getProperty(props, propKey);
        String value = getValue(prop);
        // If the property did not exist we return false.
        if (prop == null) return false;
        if (value == null) {
            // If value is null but mandatory we issue a warning and use the
            // implemented default value.
            if (prop.isMandatory()) {
```

```

        cat.warn("Using internal default value: " + myPort);
    }
}
else {
    try {
        int tmp = Integer.parseInt(value);
        if (tmp <= 0) {
            throw new IllegalArgumentException("Negative or zero value.");
        }
        myPort = tmp;
    }
    catch (Exception e) {
        cat.warn(propKey + " property for SampleIncommingChannel is
            invalid. Using internal default value: " + myPort);
    }
}
// Do the same for the MaxSubThreads-property.
propKey = "MaxSubThreads";
prop = getProperty(props, propKey);
value = getValue(prop);
if (prop == null) return false;
if (value == null) {
    if (prop.isMandatory()) {
        cat.warn("Using internal default value: " + maxSubThreads);
    }
}
else {
    try {
        int tmp = Integer.parseInt(value);
        if (tmp <= 0) {
            throw new IllegalArgumentException("Negative or zero value.");
        }
        maxSubThreads = tmp;
    }
    catch (Exception e) {
        cat.warn(propKey + " property for SampleIncommingChannel is
            invalid. Using internal default value: " + maxSubThreads);
    }
}

// Finally the ReadTimeOut-property.
propKey = "ReadTimeOut";
prop = getProperty(props, propKey);
value = getValue(prop);
if (prop == null) return false;
if (value == null) {
    if (prop.isMandatory()) {
        cat.warn("Using internal default value: " + readTimeout);
    }
}
else {
    try {
        int tmp = Integer.parseInt(value);
        if (tmp <= 0) {
            throw new IllegalArgumentException("Negative or zero value.");
        }
        readTimeout = tmp;
    }
    catch (Exception e) {
        cat.warn(propKey + " property for SampleIncommingChannel is invalid.
            Using internal default value: " + readTimeout);
    }
}
// Since we are going to use sub threads we register this channel to the
// MEC ThreadPool.
ThreadPool.getInstance().register(getClass().getName(), maxSubThreads);

// Everything went fine, therefore return true.
return true;
}

public long getDelayTime() {
    return 0;
}

```

```

public void runChannel() {
    // Create the ServerSocketChannel object and register in
    // a java.nio fashion.
    if (serverChannel == null) {
        try {
            connectSelector = Selector.open();
            serverChannel = ServerSocketChannel.open();
            serverChannel.configureBlocking(false);
            InetSocketAddress address = new InetSocketAddress(myPort);
            serverChannel.socket().bind(address);
            serverChannel.register(connectSelector, SelectionKey.OP_ACCEPT);
            logDebug(getPlugInName() + " server started");
        }
        catch (IOException e) {
            logError("Error when inititating " + getPlugInName() + ".
                Shutting down plug-in.", e);
            super.stopPlugIn();
            return;
        }
    }
    try {
        // Note, the connectSelector.select() will block until a
        // connection becomes available. That's why
        // getDelayTime() returns zero.
        if (connectSelector.select() > 0) {
            Set readyKeys = connectSelector.selectedKeys();
            for (Iterator i = readyKeys.iterator(); i.hasNext(); ) {
                SelectionKey key = (SelectionKey) i.next();
                i.remove();
                if (!key.isValid()) continue;
                if (key.isAcceptable()) {
                    SocketChannel channel = null;
                    try {
                        channel = ((ServerSocketChannel) key.channel()).accept();
                        channel.configureBlocking(false);
                        channel.register(connectSelector, SelectionKey.OP_READ);
                    }
                    catch (IOException e) {
                        logError("Error while accepting", e);
                        try {
                            channel.socket().shutdownOutput();
                            channel.close();
                        }
                        catch (Exception ex) {
                        }
                    }
                }
                else if (key.isReadable()) {
                    // The incoming connection is ready to be read. Start a
                    // worker that will read the message.
                    SampleSubThreadWorker worker = new SampleSubThreadWorker();
                    worker.setKey(key);
                    submitWork(worker);
                }
                else {
                    logError("Undefined key interest.");
                }
            }
        }
        catch (IOException e) {
            logError("Error when receiving message.", e);
        }
    }
}

public void cleanUpChannel() {
    try {
        serverChannel.close();
    }
    catch (Exception e) {
    }
    try {
        connectSelector.close();
    }
}

```

```

    }
    catch (Exception e) {
    }
}

private class SampleSubThreadWorker implements IThreadWorker {

    SelectionKey key;
    String type;
    private Manifest manifest;

    public Manifest getManifest() {
        return manifest;
    }

    public void setManifest(Manifest manifest) {
        this.manifest = manifest;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public void setKey(SelectionKey key) {
        this.key = key;
        key.interestOps(key.interestOps() & ~SelectionKey.OP_READ);
    }

    public void runWork() {
        ByteBuffer buffer = ByteBuffer.allocate(4);
        Manifest manifest = Manifest.getManifest();
        setManifest(manifest);
        SocketChannel channel = null;
        try {
            channel = (SocketChannel) key.channel();
            channel.socket().setSoTimeout(readTimeout);
            channel.read(buffer);
            buffer.flip();
            int length = buffer.asIntBuffer().get();
            persistMessage(channel, manifest, length, readTimeout);
            queueManifest(manifest);
        }
        catch (IOException e) {
            logError("Error while reading", e);
        }
        finally {
            try {
                channel.socket().shutdownOutput();
                channel.close();
            }
            catch (Exception ex) {
            }
            key.selector().wakeup();
            key = null;
        }
    }
}
}

```

# Sample Outgoing Channel



This appendix provides sample codes for different types of outgoing channel.

- ["Sample Code for HTTP Server Configuration File" on page 192](#)
- ["Sample Code for HTTPSOut Class" on page 193](#)
- ["Sample Code for HTTPSOutPanel" on page 196](#)

## Sample Code for HTTP Server Configuration File

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Configure PUBLIC "-//Mort Bay Consulting//
DTD Configure 1.2//EN" "http://jetty.mortbay.org/configure_1_2.dtd">
<Configure class="org.mortbay.jetty.Server">
  <Call name="instance" class="org.mortbay.util.Log">
    <Call name="disableLog"/>
    <Call name="add">
      <Arg>
        <New class="org.mortbay.util.log4j.Log4jSink">
          <Call name="start"/>
        </New>
      </Arg>
    </Call>
  </Call>
  <Call name="addListener">
    <Arg>
      <New class="org.mortbay.http.SslListenr">
        <Set name="Port"><SystemProperty name="
          jetty.port" default="80"/></Set>
        <Set name="MinThreads">5</Set>
        <Set name="MaxThreads">100</Set>
        <Set name="MaxIdleTimeMs">3000</Set>
        <Set name="LowResourcePersistTimeMs">5000</Set>
        <Set name="PoolName">WebThread</Set>
      </New>
    </Arg>
  </Call>
  <Call name="addListener">
    <Arg>
      <New class="org.mortbay.http.SslListenr">
        <Set name="Port">443</Set>
        <Set name="Keystore"><SystemProperty name="
          jetty.home" default="."/>/keystore</Set>
        <Set name="Password">jetty4U</Set>
        <Set name="KeyPassword">jetty4U</Set>
      </New>
    </Arg>
  </Call>
  <Call name="addWebApplication">
```



```

    <Arg>/<Arg>
    <Arg><SystemProperty name="jetty.home" default="."/>/web</Arg>
  </Call>
</Configure>

```

## Sample Code for HTTPSOut Class

```

package sample;

import java.net.*;
import java.io.*;
import java.util.Random;
import java.util.Properties;

import org.apache.log4j.*;
import com.intentia.ec.server.DocServer;
import com.intentia.ec.utility.Base64;
import com.intentia.ec.shared.IMessageSender;
import com.intentia.ec.shared.MessageSenderException;
import com.intentia.ec.shared.Manifest;

import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.SSLSession;
import javax.net.ssl.HttpsURLConnection;

public class HTTPSOut implements IMessageSender {

    // Property constants
    /* Property for the host of the http server. */
    public static final String HOST = "Host";

    /* Property for the path of the POST-request. */
    public static final String PATH = "Path";

    /* Property for the user. */
    public static final String USER = "User";

    /* Property for the password. */
    public static final String PASSWORD = "Password";

    /* Property for the port of the http server. */
    public static final String PORT = "Port";

    /* Property for the trusted certificate store path. */
    public static final String TRUST_STORE = "Trust store";

    /* Property for the trusted certificate store password. */
    public static final String TRUST_STORE_PWD = "Trust store pwd";

    static Category cat = Category.getInstance
        (HTTPSOut.class.getName());

    // Create a boundary for the multipart.
    private static Random random = new Random();
    String boundary = "-----" +
        Long.toString(random.nextLong(), 36) +
        Long.toString(random.nextLong(), 36) +
        Long.toString(random.nextLong(), 36);

    /**
     * Sends a message according to the properties in sendGroup.
     * @param inStream The input stream to read the message from.
     * @param manifest The delivery note
     * @param props Container holding the send properties.
     * @throws MessageSenderException
     */
}

```

```

public void send(InputStream inStream, Manifest manifest,
    Properties props) throws MessageSenderException {

    // Extract the needed properties.
    int port = 0;
    try {
        port = Integer.parseInt(props.getProperty(PORT));
    }
    catch (Exception e) {
        throw new MessageSenderException("Invalid port number: "
            + props.getProperty(PORT));
    }
    String host = props.getProperty(HOST);
    String path = props.getProperty(PATH);
    String user = props.getProperty(USER);
    String pwd = props.getProperty(PASSWORD);
    String trustedStore = props.getProperty(TRUST_STORE);
    String trustedStorePwd = props.getProperty(TRUST_STORE_PWD);

    // Set the system properties for the certificate stores.
    System.setProperty("javax.net.ssl.trustStore", trustedStore);
    System.setProperty("javax.net.ssl.trustStorePassword",
        trustedStorePwd);

    path = path == null ? "/" : path;
    if (!path.startsWith("/")) path = "/" + path;
    user = user == null || user.length() == 0 ? null : user;
    pwd = pwd == null || pwd.length() == 0 ? null : pwd;

    URL url = null;
    ClientHttpRequest client = null;
    HttpsURLConnection urlConn = null;

    //Build the URL
    String strURL = "https://" + host + ":" + port + path;

    try {
        // Create the url object.
        url = new URL(strURL);
        // Get the URLConnection from the URL. Since the protocol
        // is https we will be returned a HttpsURLConnection.
        urlConn = (HttpsURLConnection)url.openConnection();
        urlConn.setDoInput(true);
        urlConn.setDoOutput(false);
        urlConn.setUseCaches(false);

        // During the SSL handshake the host specified host name
        // will be verified against the certificate. This
        // implementation accept whatever specified
        // in the certificate.
        HostnameVerifier hv = new HostnameVerifier() {
            public boolean verify(String s, SSLSession sslSession) {
                return true;
            }
        };
        urlConn.setHostnameVerifier(hv);
        // Create the helper class for posting a message as multi-part.
        client = new ClientHttpRequest(urlConn);

        // If the user is specified, we encode the user and
        // password as Base64.
        if (user != null && user.length() > 0 && pwd !=
            null && pwd.length() > 0)
            {String to64 = user + ":" + ((pwd == null) ? "" : pwd);
            urlConn.setRequestProperty("Authorization: Basic",
                Base64.encode(to64, "ISO-8859-1"));}
        urlConn.setRequestProperty("User-Agent",
            "Movex Enterprise Collaborator v" + DocServer.VERSION);
        client.setParameter("name", "output.xml", inStream);
        // Posts the http message.
        client.post();
    }
    catch (MalformedURLException e) {
        String msg = "The message could not be written to
            http address: " + strURL; msg += ".\nThe address does not

```

```

        seems to be in a correct http-format.";
        throw new MessageSenderException(msg, e);
    }
    catch (UnknownHostException e) {
        String msg = "The message could not be written to
            http address: " + strURL; msg += ".\nCould not
            find the host.";
        throw new MessageSenderException(msg, e);
    }
    catch (SocketException e) {
        String msg = "The message could not be written to
            http address: " + strURL; msg += ".\nFailed to connect
            to host. Please verify that the host
            is up and running.";
        throw new MessageSenderException(msg, e);
    }
    catch (IOException e) {
        String msg = "The message could not be written to
            http address: " + strURL; msg += ".\nError while reading
            or writing message.";
        throw new MessageSenderException(msg, e);
    }
    finally {
        // Make sure everything gets closed.
        urlConn.disconnect();
        try {
            urlConn.getOutputStream().close();
        }
        catch (Exception e) {}
        try {
            urlConn.getInputStream().close();
        }
        catch (Exception e) {}
    }
}

/**
 * Helper class for creating a multi-part http message.
 */
protected class ClientHttpRequest {
    URLConnection connection;
    OutputStream os = null;

    private ClientHttpRequest(URLConnection connection) {
        this.connection = connection;
        connection.setDoOutput(true);
        connection.setRequestProperty("Content-Type",
            "multipart/form-data; boundary=" + boundary);
    }

    private void connect() throws IOException {
        if (os == null) os = connection.getOutputStream();
    }

    private void write(char c) throws IOException {
        connect();
        os.write(c);
    }

    private void write(String s) throws IOException {
        connect();
        os.write(s.getBytes());
    }

    private void newline() throws IOException {
        connect();
        write("\r\n");
    }

    private void writeln(String s) throws IOException {
        connect();
        write(s);
        newline();
    }
}

```

```
private void boundary() throws IOException {
    write("--");
    write(boundary);
}

private void writeName(String name) throws IOException {
    newline();
    write("Content-Disposition: form-data; name=\"");
    write(name);
    write('\"');
}

private void pipe(InputStream in) throws IOException {
    connect();
    byte[] buf = new byte[5000];
    int nread;
    synchronized (in) {
        while ((nread = in.read(buf, 0, buf.length)) >= 0) {
            os.write(buf, 0, nread);
        }
    }
    os.flush();
    buf = null;
}

public void setParameter(String name, String filename,
    InputStream is)
    throws IOException {
    boundary();
    writeName(name);
    write("; filename=\"");
    write(filename);
    write('\"');
    newline();
    write("Content-Type: ");
    String type = connection.guessContentTypeFromName(filename);
    if (type == null) type = "application/octet-stream";
    writeln(type);
    newline();
    pipe(is);
    newline();
}

public void post() throws IOException {
    boundary();
    writeln("--");
}
}
```

## Sample Code for HTTPSOutPanel

```
package sample;

import org.eclipse.swt.events.SelectionListener;
import org.eclipse.swt.events.SelectionEvent;
import org.eclipse.swt.widgets.*;
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.swt.layout.GridData;
import org.eclipse.swt.SWT;
import java.util.Properties;
import com.intentia.ec.shared.IRoutingUI;
import com.intentia.ec.shared.Manifest;

import java.io.File;
import java.io.InputStream;
```

```

import java.io.FileInputStream;

public class HTTPSOutPanel implements IRoutingUI,
SelectionListener {

    private Text txtHost;
    private Text txtPort;
    private Text txtPath;
    private Text txtTrustStore;
    private Text txtTrustStorePwd;
    private Button btnSpecUser;
    private Text txtUser;
    private Text txtPwd;
    private Button testButton;
    private Composite composite;

    public HTTPSOutPanel() {}

    public void createContent(final Composite parent) {

        composite = new Composite(parent, SWT.NONE);
        GridLayout gl = new GridLayout(1, false);
        gl.verticalSpacing = 10;
        gl.marginHeight = 0;
        gl.marginWidth = 0;
        composite.setLayout(gl);

        GridData data = new GridData(GridData.FILL_HORIZONTAL);
        GridLayout grpBasicLayout = new GridLayout(2, false);
        grpBasicLayout.verticalSpacing = 10;
        Group basicGrp = new Group(composite, SWT.NONE);
        basicGrp.setText("Basic configuration");
        basicGrp.setLayoutData(data);
        basicGrp.setLayout(grpBasicLayout);

        (new Label(basicGrp, SWT.NONE)).setText("Host:");

        data = new GridData(GridData.FILL_HORIZONTAL);
        txtHost = new Text(basicGrp, SWT.BORDER);
        txtHost.setLayoutData(data);

        (new Label(basicGrp, SWT.NONE)).setText("Port:");

        data = new GridData();
        data.horizontalAlignment = GridData.FILL;
        data.grabExcessHorizontalSpace = true;
        txtPort = new Text(basicGrp, SWT.BORDER);
        txtPort.setLayoutData(data);

        (new Label(basicGrp, SWT.NONE)).setText("URL path:");
        data = new GridData();
        data.horizontalAlignment = GridData.FILL;
        data.grabExcessHorizontalSpace = true;
        txtPath = new Text(basicGrp, SWT.BORDER);
        txtPath.setLayoutData(data);

        data = new GridData(GridData.FILL_HORIZONTAL);
        GridLayout grpSSLLayout = new GridLayout(2, false);
        grpSSLLayout.verticalSpacing = 10;
        Group sslGrp = new Group(composite, SWT.NONE);
        sslGrp.setText("Trusted certificate store");
        sslGrp.setLayoutData(data);
        sslGrp.setLayout(grpSSLLayout);

        (new Label(sslGrp, SWT.NONE)).setText("Path:");
        data = new GridData(GridData.FILL_HORIZONTAL);
        txtTrustStore = new Text(sslGrp, SWT.BORDER);
        txtTrustStore.setLayoutData(data);

        (new Label(sslGrp, SWT.NONE)).setText("Password:");
        data = new GridData(GridData.FILL_HORIZONTAL);
        txtTrustStorePwd = new Text(sslGrp,
            SWT.BORDER | SWT.PASSWORD);
        txtTrustStorePwd.setLayoutData(data);
    }
}

```

```

data = new GridData(GridData.FILL_HORIZONTAL);
GridLayout grpUserLayout = new GridLayout(2, false);
grpUserLayout.verticalSpacing = 10;
Group userGrp = new Group(composite, SWT.NONE);
userGrp.setText("User configuration");
userGrp.setLayoutData(data);
userGrp.setLayout(grpUserLayout);

(new Label(userGrp, SWT.NONE)).setText("");
data = new GridData();
btnSpecUser= new Button(userGrp, SWT.CHECK);
btnSpecUser.setText("Specify user");
btnSpecUser.addSelectionListener(this);
btnSpecUser.setLayoutData(data);

(new Label(userGrp, SWT.NONE)).setText("User:");
data = new GridData();

data = new GridData(GridData.FILL_HORIZONTAL);
txtUser = new Text(userGrp, SWT.BORDER);
txtUser.setLayoutData(data);
txtUser.setEnabled(false);

(new Label(userGrp, SWT.NONE)).setText("Password:");
data = new GridData(GridData.FILL_HORIZONTAL);
txtPwd = new Text(userGrp, SWT.BORDER | SWT.PASSWORD);
txtPwd.setLayoutData(data);
txtPwd.setEnabled(false);

data = new GridData(GridData.FILL_HORIZONTAL);
data.horizontalAlignment = GridData.END;
data.verticalAlignment = GridData.BEGINNING;
testButton = new Button(composite, SWT.PUSH);
testButton.setText("Send test message");
testButton.addSelectionListener(this);
testButton.setLayoutData(data);
}

public void setProperties(Properties props) {
    if (props == null) return;
    txtHost.setText(props.getProperty(HTTPSOut.HOST, ""));
    txtPort.setText(props.getProperty(HTTPSOut.PORT, ""));
    txtPath.setText(props.getProperty(HTTPSOut.PATH, ""));
    txtUser.setText(props.getProperty(HTTPSOut.USER, ""));
    txtPwd.setText(props.getProperty(HTTPSOut.PASSWORD, ""));
    txtTrustStore.setText(props.getProperty(
        HTTPSOut.TRUST_STORE, ""));
    txtTrustStorePwd.setText(props.getProperty(
        HTTPSOut.TRUST_STORE_PWD, ""));
    btnSpecUser.setSelection(txtUser.getText().length() > 0);
    txtUser.setEnabled(txtUser.getText().length() > 0);
    txtPwd.setEnabled(txtUser.getText().length() > 0);
}

private boolean checkFields() {
    if (txtHost.getText().length() == 0) {
        openError(composite.getShell(), "Invalid value",
            "Host cannot be empty."); return false;
    }
    if (btnSpecUser.getSelection() &&
        txtUser.getText().length() == 0) {
        openError(composite.getShell(), "Invalid value",
            "User cannot be empty."); return false;
    }
    try {
        if (Integer.parseInt(txtPort.getText()) <= 0) {
            throw new NumberFormatException();
        }
    }
    catch (NumberFormatException e) {
        openError(composite.getShell(), "Invalid value",
            "Illegal port number: " + txtPort.getText());
    }
}

```

```

    return false;
}
if (txtTrustStore.getText().length() == 0) {
    openError(composite.getShell(), "Invalid value",
        "Trusted store path cannot be empty.");
    return false;
}
return true;
}

public Properties getProperties() {
    Properties props = new Properties();
    if (!checkFields()) return null;
    props.setProperty(HTTPSOut.HOST, txtHost.getText().trim());
    props.setProperty(HTTPSOut.PORT, txtPort.getText().trim());
    props.setProperty(HTTPSOut.PATH, txtPath.getText().trim());
    if (btnSpecUser.getSelection()) {
        props.setProperty(HTTPSOut.USER,
            txtUser.getText().trim());
        props.setProperty(HTTPSOut.PASSWORD,
            txtPwd.getText().trim());
    }
    props.setProperty(HTTPSOut.TRUST_STORE,
        txtTrustStore.getText().trim());
    props.setProperty(HTTPSOut.TRUST_STORE_PWD,
        txtTrustStorePwd.getText().trim());
    return props;
}

public void widgetSelected(SelectionEvent event) {
    if (event.widget == btnSpecUser) {
        if (!btnSpecUser.getSelection()) {
            txtUser.setData(txtUser.getText());
            txtPwd.setData(txtPwd.getText());
            txtUser.setText("");
            txtPwd.setText("");
        }
        else {
            txtUser.setText(txtUser.getData() != null ?
                (String)txtUser.getData() : "");
            txtPwd.setText(txtPwd.getData() != null ?
                (String)txtPwd.getData() : "");
        }
        txtUser.setEnabled(btnSpecUser.getSelection());
        txtPwd.setEnabled(btnSpecUser.getSelection());
    }
    else if (event.widget == testButton) {
        if (!checkFields()) return;
        Properties props = new Properties();
        props.put(HTTPSOut.HOST, txtHost.getText().trim());
        props.put(HTTPSOut.PORT, txtPort.getText().trim());
        props.put(HTTPSOut.PATH, txtPath.getText().trim());
        props.put(HTTPSOut.USER, txtUser.getText().trim());
        props.put(HTTPSOut.PASSWORD, txtPwd.getText().trim());
        props.put(HTTPSOut.TRUST_STORE, txtTrustStore.getText().trim());
        props.put(HTTPSOut.TRUST_STORE_PWD,
            txtTrustStorePwd.getText().trim());

        FileDialog dialog = new FileDialog(composite.getShell(),
            SWT.OPEN); dialog.open();
        if (dialog.getFilterPath() != null &&
            dialog.getFilterPath().length() > 0
            && dialog.getFileName() != null &&
            dialog.getFileName().length() > 0) {
            String fileName = dialog.getFilterPath() +
                File.separator + dialog.getFileName();
            File file = new File(fileName);
            if (!file.isFile()) {
                openError(composite.getShell(), "Invalid file",
                    "Chosen file is not a file:" + file.getName());
                return;
            }
        }
        else if (file.length() <= 0) {

```

```
        openError(composite.getShell(), "Invalid file",
            "Chosen file is empty:" + file.getName());
        return;
    }
    composite.getShell().setCursor
        (Display.getCurrent().getSystemCursor(SWT.CURSOR_WAIT));
    InputStream ins = null;
    try {
        ins = new FileInputStream(file);
        HTTPSOut out = new HTTPSOut();
        Manifest manifest = Manifest.getManifest();
        out.send(ins, manifest, props);
        openInformation(composite.getShell(), "Success",
            "Successfully sent file: " + file.getName());
        composite.getShell().setCursor(null);
    }
    catch (Throwable e) {
        composite.getShell().setCursor(null);
        openError(composite.getShell(), "Error sending",
            "Sending test message failed. " + e.getMessage());
    }
    finally {
        try {
            ins.close();
        }
        catch (Exception e) {}
    }
}
return;
}
}

public void widgetDefaultSelected(SelectionEvent e) {
    widgetSelected(e);
}

private static void openError(Shell shell,
    String title, String msg) {
    int style = SWT.ICON_ERROR | SWT.OK;
    MessageBox box = new MessageBox(shell, style);
    box.setText(title == null ? "Error " : title);
    box.setMessage(msg);
    box.open();
}

private static void openInformation(Shell shell,
    String title, String msg) {
    int style = SWT.ICON_INFORMATION | SWT.OK;
    MessageBox box = new MessageBox(shell, style);
    box.setText(title == null ? "" : title);
    box.setMessage(msg);
    box.open();
}
}
```



---

# Create ConfirmBOD process specifics



This appendix contains process specifics on how to make a ConfirmBOD that is sent from MEC to ION

- ["ConfirmBOD" on page 201](#)

## ConfirmBOD

```
/ConfirmBOD@xmlns = "http://schema.infor.com/InforOAGIS/2" (hard coded)
/ConfirmBOD@xmlns:xsi = " http://www.w3.org/2001/XMLSchema-instance " (hard coded)
/ConfirmBOD@schemaLocation = "http://schema.infor.com/2.6.2/InforOAGIS/BODs/Developer/ConfirmBOD.xsd"
(hard coded)
/ConfirmBOD@versionID = "2.6.2" (hard coded)
/ConfirmBOD@releaseID = "9.2" (hard coded)
/ConfirmBOD@systemEnvironmentCode = 1) from original BOD (/pre:*/@systemEnvironmentCode) 2)
"Production" (hard coded)
/ConfirmBOD@languageCode = "en-US" (hard coded)

/ConfirmBOD/ApplicationArea/Sender/LogicalID = 1) mapping manifest item "map:ionFromLogicalId" 2)
agreement control property "ionFromLogicalId" 3) exception "No FromLogicalId found "
/ConfirmBOD/ApplicationArea/Sender/LogicalID@schemeVersionID = "1.0" (hard coded)

/ConfirmBOD/ApplicationArea/Sender/ComponentID = 1) agreement control property "ionComponentId"
2) "M3BE" (hard coded)
/ConfirmBOD/ApplicationArea/Sender/ComponentID@schemeVersionID = 1) agreement control property
"ionComponentVersion" 2) mapping manifest item "map:m3beVersion"

/ConfirmBOD/ApplicationArea/CreationDateTime = current date and time, normalized

/ConfirmBOD/ApplicationArea/BODID = mapping UUID

/ConfirmBOD/DataArea/Confirm/TenantID = 1) mapping manifest item "map:ionTenantId" 2) mapping
manifest item "map:m3beTenantId" 3) original BOD TenantID (/pre:*/pre:DataArea/pre:*/pre:TenantID)
4) exception "No TenantID element found."

/ConfirmBOD/DataArea/Confirm/AccountingEntityID = 1) mapping manifest item "map:ionAccountingEntityId"
2) mapping manifest item "map:m3beAccountingEntityId"

/ConfirmBOD/DataArea/Confirm/LocationID = 1) mapping manifest item "map:ionLocationId" 2) mapping
manifest item "map:m3beLocationId"

/ConfirmBOD/DataArea/Confirm/OriginalApplicationArea = original BOD ApplicationArea
(/pre:*/pre:ApplicationArea)

/ConfirmBOD/DataArea/Confirm/ResponseCriteria/ResponseExpression@actionCode = "Rejected"

/ConfirmBOD/DataArea/BOD/BODFailureMessage/ErrorProcessMessage/Description = formatted error message
created from MEC error mail

/ConfirmBOD/DataArea/BOD/OriginalBOD/MessageHeader loop:
/ConfirmBOD/DataArea/BOD/OriginalBOD/MessageHeader/MessageHeaderProperty@listID = "MessageHeader"
(hard coded)
/ConfirmBOD/DataArea/BOD/OriginalBOD/MessageHeader/MessageHeaderProperty/NameValue = IONdbIn manifest
```

## Create ConfirmBOD process specifics

---

```
items: com:ionFromLogicalId + com:ionToLogicalId + com:ionBODType + com:ionMessageId +  
com:ionTenantId + com:ionReferenceId + com:ionTargetLogicalId + com:ionEncoding  
/ConfirmBOD/DataArea/BOD/OriginalBOD/MessageHeader/MessageHeaderProperty/NameValue@name =  
"FromLogicalId" + "ToLogicalId" + "BODType" + "MessageId" + "TenantId" + "ReferenceId" +  
"TargetLogicalId" + "Encoding"
```

```
/ConfirmBOD/DataArea/BOD/OriginalBOD/MessageContent = Base64 encoded original BOD
```