

EVENT MANAGEMENT FRAMEWORK

User Manual

Version: 7.2.1

September 2017



4325 Alexander Drive, Suite 100 • Alpharetta GA 30022-3740 • www.aptean.com • info@aptean.com

Copyright © 2017 Aptean. All Rights Reserved. These materials are provided by Aptean for informational purposes only, without representation or warranty of any kind, and Aptean shall not be liable for errors or omissions with respect to the materials. The only warranties for Aptean products and services are those set forth in the express warranty statements accompanying such products and services, if any, and nothing herein shall be construed as constituting an additional warranty. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express written permission of Aptean. The information contained herein may be changed without prior notice. Some products marketed by Aptean contain proprietary software components of other software vendors. Aptean and other Aptean products and services referenced herein as well as their respective logos are registered trademarks or trademarks of Aptean or its affiliated companies.

Contents

Introducing Event Management Framework (EMF) - Product Overview	v
Preparing to use EMF	v
Acknowledgements	v
Getting Started	1-1
Overview	1-2
<i>The EMF Interface</i>	1-2
<i>Setting Up Your System</i>	1-2
<i>Starting the EMF Server</i>	1-3
<i>Creating EMF Processes</i>	1-4
<i>Setting a Default Language for EMF Output</i>	1-4
<i>Licensing and Modularity in EMF</i>	1-5
<i>Viewing License Information for a Repository</i>	1-6
Setting Up EMF	2-1
Overview	2-2
<i>Adding and Configuring Repositories</i>	2-2
<i>Logging Into the EMF System</i>	2-19
<i>Using the EMF Tree View</i>	2-20
<i>Maintaining the EMF Server</i>	2-26
<i>EMF Security</i>	2-92
<i>Managing Operators</i>	2-95
<i>Logging in EMF</i>	2-107
Building your EMF Processes	3-1
Overview	3-2
<i>Creating and Managing EMF Processes</i>	3-2

Example of Link Conditions	3-12
<i>Example</i>	3-12
<i>The EMF Process Builder Modules</i>	3-50
Source Properties	3-314
Define Parent Iterator Element	3-316
<i>Example results for XML to Data Section with and without a Parent Iterator defined</i>	3-316
Add element	3-319
Add attribute	3-320
Add JSON value	3-321
Add field	3-323
Apply sample	3-324
Editing / Deleting items	3-327
Destination Properties	3-328
Define Parent Iterator Element	3-329
Key field	3-330
<i>Key Field Example for XML to Data section mapping</i>	3-330
Group By field	3-333
Add element	3-334
Add attribute	3-337
Add JSON value	3-338
Add field	3-340
Apply sample	3-343
Editing / Deleting items	3-346
<i>Accessing an Element's Position in XML</i>	3-349
Dealing with Undefined Elements in XML	3-351
<i>Example for a command being returned to EMF</i>	3-453

EMF Concepts and Components	4-1
Overview	4-2
<i>Fundamental EMF Technical Concepts</i>	4-2
<i>Considerations for Sizing and Optimizing an EMF Deployment</i>	4-11
<i>Recipient Administration</i>	4-25
<i>Overview of SAP System Data Sources</i>	4-40
<i>Configuring Data Sources</i>	4-55
<i>Database Trigger Wizard</i>	4-151
<i>White Lists</i>	4-160
<i>Event Definitions</i>	4-162
<i>Managing Authentication Protocols</i>	4-164
<i>Using Multiple Languages in EMF</i>	4-173
<i>About EMF Data Store - Sharing Data between Processes</i>	4-174
<i>Using Dynamic Functions in EMF Processes</i>	4-196
<i>Java API Guide</i>	4-282
<i>SOAP</i>	4-329
<i>Dead Letter Queue Browser</i>	4-339
<i>Exclusive Queue - Single Thread Operation/Message Order Dependency</i>	4-341
<i>Production Mode</i>	4-342
<i>Event Plans</i>	4-345
Advanced	5-1
Multi Residency in EMF	5-2
<i>Creating a New EMF Instance</i>	5-2
<i>Configuring the new EMF Instance</i>	5-3
<i>Upgrading EMF in Multi Residency</i>	5-8
Troubleshooting	6-1
Overview	6-2

<i>Checking your EMF system configuration</i>	<i>6-2</i>
<i>Known issues, problems and error messages</i>	<i>6-2</i>
<i>Specific Issues</i>	<i>6-2</i>
<i>General Problems</i>	<i>6-5</i>
<i>Error Message</i>	<i>6-6</i>

Introducing Event Management Framework (EMF) - Product Overview

EMF is a versatile tool that allows you to monitor your business data on an ongoing basis, and notify selected recipients of significant events as and when they occur. EMF works with any existing corporate data sources that are accessible via JDBC and SAP, and EMF processes are delivered using standard systems and devices such as email, fax, SMS, and mobile phones. Recipients need no additional software or hardware.

They can also respond to incoming EMF processes, and interact with the EMF system using executable EMF processes that can automatically run third party applications.

Preparing to use EMF

In order to use EMF, you must [first set up the repository database](#). You must then install a valid [license](#) (a temporary license is provided with EMF, but this is only suitable for trial purposes owing to certain limitations - see [Licensing and Modularity in EMF](#) for further details).

When your system has been properly configured and is working correctly, you can begin to create the EMF processes. This is done by dragging icons representing the various [process builder modules](#) onto the [EMF Process Design Graph](#) and creating links between them, like a flow chart. Each icon on the Design Graph has a number of settings available, and you can change these by double-clicking the icons to open their properties pages. Comprehensive online Help is available for all of these, or the Help buttons on the dialogs.

For further information see the [Setting up Your System](#) and [Creating EMF Processes](#) topics in the [Getting Started](#) section, or else go straight to the main [Configuring the EMF System](#) and [Creating and Managing EMF Processes](#) sections of the online Help.

Acknowledgements

[Copyright © Aptean 2016](#). All rights reserved

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>)

[How to Use EMF Help](#)

[Getting Started](#)



1

Getting Started

Overview

Before you can create and run EMF Processes, you must ensure that your EMF system is correctly set up and configured. This can be a complex procedure and is often best undertaken by system administrators or representatives of Aptean. Further details are available in [Setting up Your System](#).

When your system is up and running correctly, you can start [creating your EMF Processes](#) using the [EMF Process Builder view](#), and then test, verify, and debug the processes before running them.

The EMF Interface

There are two aspects to the EMF interface:

- The **EMF Tree view** is first displayed when you log in to EMF. This view consists of a number of icons displayed down the left side of the view, similar to Windows Explorer. Click on these icons to expand the relevant details view on the right side of the view.

Note: If you have not successfully logged into EMF, the tree is not displayed.

- The **EMF Process Builder view** is accessed by double-clicking the icon for an existing EMF Process in the tree view, or by right-clicking on an EMF Process folder and clicking **New Process**. You can have as many EMF Process builder views open as you wish.

[Introducing EMF](#)

[Configuring the EMF System](#)

[Creating and Managing EMF Processes](#)

Setting Up Your System

There are four main areas of your system that need to be configured before you can create and run EMF Processes:

- **Configuring the EMF System**, including the [machines](#) that the EMF server is running on, your [queue provider](#), your [logical queues](#) and [language](#) options. All these tasks are carried out using the views that you can access from the icons in the EMF tree view.
- **Creating and setting up your EMF databases**, including your repository database and any client databases that you want to access. See Adding and Configuring Databases.
- **Configuring and starting the EMF server** - see [Starting the EMF Server](#).
- Adding and managing system recipients - see [Recipient Management](#).
- Assigning access rights and security settings - see [Operator Management](#).

These can be complex tasks and are often best undertaken by system administrators or representatives of Apteian.

Further details and full instructions are available in the [Configuring the EMF System](#) Help topics.

[How to Check EMF Configuration Details](#)

[Introducing EMF](#)

[Creating EMF Processes](#)

Starting the EMF Server

The EMF Server is the process engine that runs the system, and **it must be running in order for EMF Processes to be executed**.

Starting the EMF Server on a Windows Platform

The EMF server runs as a Windows service.

The installation automatically configures the service. If the service does not exist, or you need to re-register it, then see [Adding EMF as a Windows Service](#).

If you configured the service to start automatically, then it will start whenever you restart the server machine. If you want to start it manually, select **Start > Settings > Control Panel > Administrative Tools > Services**. In the **Services** screen, right-click **EMF Service** and select **Start**.

To start the EMF server manually from the EMF Administrator:

1. Start your queuing software, unless you configured your queue provider to start automatically or you are using the internal queue provider (see [Managing the machines that EMF runs on](#)).
2. In the EMF Administrator, select the menu item **Server > Start Local Server**. This will start the EMF Server service.

To start the EMF server manually from the Windows Start menu:

1. Start your queuing software, unless you configured your queue provider to start automatically or you are using the internal queue provider (see [Managing the machines that EMF runs on](#)).
2. Start the EMF server by selecting **EMF 7 > Server > Start EMF Server**. This will start the EMF Server service. If the EMF server has started correctly, you should see EMFService.exe listed in the Task Manager.
3. To view the console output, run the EMF Administrator and select **Server > Connect to local server**. The server console window opens and a "Connected" message will be displayed if the server is running successfully.

To start the EMF server as a Windows service

Important: If you are running the repository database server and/or queuing software

other than the internal ActiveMQ queue provider on the same machine as the EMF server, **and** you have configured the EMF service to start automatically, you must configure [dependencies](#) to ensure that the EMF service starts only after the database server and/or queuing software have started

[Stopping the EMF Server](#)

Creating EMF Processes

The stages involved in creating EMF Processes are as follows:

1. Create a new EMF Process. The **EMF Process Builder view** opens automatically with the **EMF Process Design Graph** in the center.
2. Drag one or more **Initiator**, **Processing** and **Output** modules onto the Design Graph and create links between them.
3. Customize the various options for the modules by double-clicking on them to open their properties pages.
4. Customize the links between the modules by double-clicking on them to open the **Link Properties** and **Link Conditions** screens, and modify the behavior of the EMF Process using **Processing modules**.
5. **Validate**, **debug**, **save** and **activate** your EMF Process. You can then fire your EMF Processes manually, or automatically depending on the criteria set in the various initiator modules.

For full details and instructions, follow the relevant links below:

[Creating and Managing EMF Processes](#)

[The EMF Process Builder View](#)

[The EMF Process Builder Modules](#)

[Validating](#), [Debugging](#) and [Activating EMF Processes](#)

[Introducing EMF](#)

[Configuring the EMF System](#)

Setting a Default Language for EMF Output

You can define the default language to use when sending out EMF Processes by selecting the required language on the **Default Language** tab of the EMF **Settings** screen, which is one of the General Tasks in the EMF tree view.

Whichever language is enabled (by clicking the box next to it) will be used for all new EMF Processes unless a specific alternative is chosen.

[Defining Languages for EMF to use](#)

EMF Process General Tasks

Configuring the EMF System

Licensing and Modularity in EMF

Important: what you can do within the EMF system depends on the current license and your personal security privileges. If you cannot see or use any aspects of EMF you should consult your system administrator and check these before contacting technical support.

EMF can be installed and used in a variety of different ways that reflect the requirements of the end users. This is achieved by offering a variety of licensing options so that customers can select the features and functionality that they require and control access to, and use of, the software.

Each licensing option offers a different set of modules and/or functionality, and for this reason it may be that your installation does not include all of the possible options. Therefore you may find that certain features that are described in the Help are not available to you, although each installation is self-complete and will not lack any features that are required in order to function in the intended manner.

Licensing is generally the responsibility of your vendor. However, certain OEMs may also be able to create their own licenses depending on their commercial agreement. Licenses can be granted and distributed by a variety of means (e.g. a license generation API, permitting a variety of licensing applications, and/or a Web-based application).

The licensing and modularity of EMF has the following potential implications:

- Your installation of EMF may be time-limited (e.g. for evaluation purposes). Your installation will be fully functional until the trial period expires, after which you must upgrade to a full license in order to continue using it.
- Your installation of EMF may be restricted to a maximum number of users per repository.
- Certain features and/or modules may be unavailable.
- You may not be able to copy EMF between folders or users.

You can upgrade your license at any time in order to enable and extend missing and/or limited functionality. For further information, contact your supplier.

Additional notes on licenses:

- Licenses expire at 23:59, server's local time, on the date of expiry.
- You cannot install a license for a repository if you already have a license installed with the same license number. You must first [remove the previous license](#) before attempting to install the new license.

Installing a license

Note: You must create your repository database before you can install a license. For more information about creating databases, refer to the Quick Start Guide or online Help.

EMF requires a valid license before you can log in to a repository and begin using the software. Whenever you create a new repository you automatically receive a temporary license for it as a short term safeguard in case you did not request a permanent license before installation.

Note: You will not be able to access the repository after the temporary license expires. In order to continue using your EMF repositories, you must install a valid permanent license. If you did not receive a license with EMF, please consult your supplier.

For more information about installing the license file, see *Event Management Framework 7 Installation and Configuration Guide*.

[Introducing EMF](#)

Viewing License Information for a Repository

You can use the **License Manager** tab to select the active license for the current repository and to delete expired and/or redundant licenses.

To view the License Manager section:

1. Expand the **Repository** and then expand **System** in the EMF tree view.
2. Click **License Manager** and on the left pane, the License Manager section is displayed.
3. Select the required **License number** from the list of the license files that you have installed. A summary of the license properties is displayed (including the license type - e.g. temporary, with details of the expiry date, or permanent). Each license file can contain one or more licenses, and details of the licenses in the selected file are displayed in the lower half of the screen.

The other columns in the **License Manager** pane give the following information:

- **Expiry:** The date when the license ceases to be valid (NB: the license will expire at 23:59, local time of the server, on the date stated).
- **Max Operators:** The maximum number of people who can be authorised to log in to the repository.
- **Max Users:** The maximum number of recipients who can be set up to receive output messages.
- **Max Processes:** The maximum number of processes that can be created within the system.
- **Max Machines:** The maximum number of machines that can be set up within the system.
- **Max JDBC Datasources:** The maximum number of JDBC Datasources that can be created in the repository.
- **Max SAP Datasources:** The maximum number of SAP Datasources that can be created in the repository.

To remove a license: Right-click the required license and click **Delete**.

[Repository Properties JDBC Tab](#)

[Licensing and Modularity in EMF](#)



2

Setting Up EMF

Overview

These Help topics will assist you in setting up EMF so that it functions optimally for your purposes. There are seven main sections:

- [Adding and Configuring Repositories](#)
- [Logging into the EMF System](#)
- [Using the EMF Tree view](#)
- [Maintaining the EMF Server](#)
- [Security](#)
- [Operator Management](#)
- [Logging](#)

Adding and Configuring Repositories

A repository database stores all the information about the EMF processes and server configuration. To build processes and successfully run the EMF server, a repository must first be created.

Before you can create any EMF Processes you must create your repository database. The database is created using the [EMF Repository Wizard](#) utility, which is accessed by using the menu path **File > New Repository**.

EMF uses two types of database:

- The Repository database contains information on the EMF system (e.g. details of your email server) and any EMF Processes that you create. You select the repository database that you want to use each time you log in to EMF (see [the Login screen](#)).

You can have as many repository databases as you are licensed to have, and can use each one for a different purpose (e.g. one for training purposes, one for testing, and one for active customer and company EMF Process processing). However the EMF server on any one machine, can only use a single repository database at any one time.

- The **Client databases** include all the databases against which EMF Processes are run. EMF can connect to JDBC or SAP data sources (see [JDBC Connections](#) and [SAP Connections](#)) and use the information that is contained within them to create Recipient sections and Data sections.
 - **Recipient** sections contain user information that has been brought into the EMF System from a client database by a [Recipient module](#).
 - **Data** sections contain non user-related information that has been brought into the EMF System from a client database by a Data module, for example, the [SAP](#) or [SQL](#) modules.

Updating EMF databases

When upgrading to a newer version of EMF, the repository database has to be upgraded. If you apply service packs or upgrades to your installed version of EMF, you may occasionally need to upgrade the repository database as well. You will be prompted to do this before you are allowed to log in to that repository.

Repository Compatibility Settings

Compatibility settings allows you to select settings that will maintain compatibility with older versions of EMF. In newer versions some functionality has been improved that would cause old processes to function differently - and possibly fail. On upgrading old repositories the settings are set automatically to maintain compatibility. New repositories created will by default, use the new improved functionality. Right click the **Repository** node and select the **Compatibility settings** option from the pop-up menu.

[Creating New Repositories](#)

[Connecting to An Existing Repository](#)

[Upgrading Repositories](#)

[Repository Compatibility Settings](#)

Creating Databases With the EMF Repository Wizard

You can create and configure new repository databases for the EMF system using the **EMF Repository Wizard**. Click **File > New Repository** to access the wizard.

In the EMF Repository wizard you can do the following:

- Choose **Express**, **Typical** and **Advanced** mode, depending on the level of customization you require.
- Create EMF repository database.
- Generate the JDBC settings for your databases.
- Install a license for your new repository.
- Save the database scripts and EMF server configuration file for later deployment.

The EMF Repository wizard opens with the **Welcome** screen.

To continue and create your new database, click the **Next** button to proceed to the **Configuration type** screen.

[Selecting a configuration type](#)

[Express](#), [Typical](#) and [Advanced](#) configuration types

Selecting a Configuration Type

The **Configuration type** screen is used to select the type of database configuration you want to perform. Use the information in this topic to help you decide which configuration type best suits your needs. If you are a new user, it is recommended that you choose either **Express** or **Typical**.

The **Express** configuration type is designed for users who have installed the entire EMF system - Administrator and Server - on a single machine, and who want as much of the configuration settings as possible to be automated. All you need to specify is your database server type and location, as well as the database user. The EMF Repository Wizard will do everything else for you.

Select **Typical** if you want to perform any of the following configuration tasks:

- Specify a different machine as the EMF Server (not your local machine).
- Import data into the repository.
- Install a license for your repository.
- Provide your own name for the repository and configuration databases (**Express** automatically names these for you).
- Specify the details of the user for connecting to the EMF databases.
- View the database creation scripts.
- Save the system setup file so you can run it later.

Note: You can import data and install licenses in the EMF Administrator after creating the EMF databases.

You can also select **Typical** if you do **not** want your queue provider to start automatically when the EMF Server starts.

Note: You can [configure the automatic startup of the queue provider](#) later in the EMF Administrator. If this is the only task you want to perform in addition to the **Express** tasks, you may prefer to choose the **Express** configuration type and configure the queue provider startup in the EMF Administrator after running the configuration.

Select **Advanced** if you want to perform any of the following configuration tasks (you will also be able to perform all the tasks listed above for the **Typical** configuration):

- Configure the database connection string.
- Specify a location other than the local EMF server directory for the <Repository Name>.properties file.
- Provide different names for the repository and configuration databases (**Typical** uses the same name for both).
- Drop existing users, tablespaces or databases if they already exist.

Also, select **Advanced** if you do **not** want Unicode database(s)

[Using the EMF configuration wizard](#)

Express Configuration Type

What you need to know before you continue

- Your database server type, Oracle or SQL Server.
- SQL Server machine name or Oracle connection details.
- The username and password of the user who will create the EMF databases. For Oracle, this user must have DBA rights; for SQL Server, the user must have database creation and Security Administrator rights. The user must already exist.

Note: Some of these screens are database-specific, so you will not see those that do not apply to your database type. For allowable characters in database names and usernames, see [Allowable characters](#).

Database server screen

Select either **SQL Server** or **Oracle** as your database server type.

SQL Server connection screen (SQL Server only)

1. Enter the machine name on which the SQL Server is located in the **SQL Server machine name** textbox. If the database server is on your local machine, you can enter "localhost" or "127.0.0.1" instead of the machine name.
2. Specify the instance name of the SQL server, or leave it blank, if it has none.
3. To use Windows **NT integrated security**, select the checkbox (the **Username** and **Password** text boxes will be disabled).

Note: While you can use Windows authentication to create the repository databases, the product requires Mixed Mode or SQL Server authentication to be enabled on the database server to function.

If you do not want to use Windows authentication, enter the user details as follows:

- In the **Username** textbox, type the name of an existing user with database creation and Security Administrator rights, for example "sa".
- In the **Password** textbox, type the password of the user.

If the database server is running and accessible from your machine, click the **Test connection** button to test the connection.

Oracle connection screen (Oracle only)

The following methods can be used to define the connection to the Oracle database:

- Thin Driver
- OCI

1. To connect to the Oracle database using the **Thin Driver** type, select the **Thin**

Driver option and enter the following details:

- Specify the machine on which the Oracle Server is located in the **Machine name** text box. If the database server is on your local machine, you can enter "localhost" or "127.0.0.1" instead of the machine name.
 - In the **Connect using** group, select:
 - a. **SID** - to use the SID name of the Oracle database instance. Enter the SID name in the following field.
 - b. **Service Name** - to use the service name of the Oracle database instance. Enter the service name in the following field. For example, to connect an Oracle 12 database, in the service name field, type the name of the pluggable database (PDB).
 - In the **Port number** field, type the connection port number, for example "1521".
2. To connect to the Oracle database using the **OCI** type, select the **OCI** option and enter the following details:
 - Type the TNS name of the Oracle database instance in the **Oracle TNS name** field.
 3. In the **Oracle : User Details** dialog box, specify the details as follows:
 - In the **User name** field, type the name of an existing user with DBA rights, for example "SYSTEM".
 - In the **Password** field, type the password of the user. The password cannot be blank.
 - If the database server is running and accessible from your machine, click **Test connection** to test the connection.
 4. Click **Next** to continue.

[Using the EMF configuration wizard](#)

Performing a Typical Configuration

What you need to know before you continue:

- Name or IP address of the machine on which the EMF server is installed.
- Your database server type, SQL Server or Oracle.
- SQL Server machine name or Oracle connection details.
- The username and password of the user who will create the EMF databases. For Oracle, the user must have DBA rights; for SQL Server, the user must have database creation and Security Administrator rights. The user must already exist.
- (For SQL Server only) The details of the user that will connect to the EMF databases (database owner). (For Oracle only) The details of the EMF user and tablespace.
- The name you want to call the repository and configuration databases.
- The details of the license if you want to install one during setup.

The following screens are described in this topic:

- [EMF server machine](#)
- [Database server](#)
- [SQL Server connection](#)
- [Oracle connection](#)
- [SQL Server EMF database configuration](#)
- [Oracle EMF tablespace configuration](#)
- [Install license](#)
- [Update Server](#)

Note: Some of these screens are database-specific, so you will not see those that do not apply to your database type. For allowable characters in database names and usernames, see [Allowable characters](#).

EMF server machine

Type the name or IP address in the **EMF server machine** text box. If the EMF Server is on your local machine, you can enter "localhost" or "127.0.0.1" instead of the machine name.

Database server screen

Select either **SQL Server** or **Oracle** as your database server type.

SQL Server connection screen (SQL Server only)

1. Specify the machine on which the SQL Server is located in the **SQL Server machine name** text box. If the database server is on your local machine, you can enter "localhost" or "127.0.0.1" instead of the machine name.
2. To use Windows **NT integrated security**, select the checkbox (the **Username** and **Password** text boxes will be disabled).

Note: While you can use Windows authentication to create the repository databases, the product requires Mixed Mode or SQL Server authentication to be enabled on the database server to function.

If you do not want to use Windows authentication, enter the user details as follows:

3. Specify the instance name of the SQL server, or leave it blank, if it has none.
4. To use Windows **NT integrated security**, select the checkbox (the **Username** and **Password** text boxes will be disabled).

Note: While you can use Windows authentication to create the repository databases, the product requires Mixed Mode or SQL Server authentication to be enabled on the database server to function.

If you do not want to use Windows authentication, enter the user details as follows:

- In the **Username** textbox, type the name of an existing user with database creation and Security Administrator rights, for example "sa".
 - In the **Password** text box, type the password of the user.
5. If the database server is running and accessible from your machine, click the **Test connection** button to test the connection.
 6. Specify the instance name of the SQL server, or leave it blank, if it has none.
 7. To use Windows **NT integrated security**, select the checkbox (the **Username** and **Password** text boxes will be disabled).

Note: While you can use Windows authentication to create the repository databases, the product requires Mixed Mode or SQL Server authentication to be enabled on the database server to function.

If you do not want to use Windows authentication, enter the user details as follows:

- In the **Username** text box, type the name of an existing user with database creation and Security Administrator rights, for example "sa".
 - In the **Password** text box, type the password of the user.
8. If the database server is running and accessible from your machine, click the **Test connection** button to test the connection.
 - In the **Username** text box, type the name of an existing user with database creation and Security Administrator rights, for example "sa".
 - In the **Password** text box, type the password of the user.
 9. If the database server is running and accessible from your machine, click the **Test connection** button to test the connection.

Oracle connection screen (Oracle only)

The following methods can be used to define the connection to the Oracle database:

- Thin Driver
 - OCI
1. To connect to the Oracle database using the **Thin Driver** type, select the **Thin Driver** option and enter the following details:
 - Specify the machine on which the Oracle Server is located in the **Machine name** text box. If the database server is on your local machine, you can enter "localhost" or "127.0.0.1" instead of the machine name.
 - In the **Connect using** group, select:
 - a. **SID** - to use the SID name of the Oracle database instance. Enter the SID name in the following field.
 - b. **Service Name** - to use the service name of the Oracle database instance. Enter the service name in the following field. For example, to connect an Oracle 12 database, in the service name field, type the name of the pluggable database (PDB).

- In the **Port number** field, type the connection port number, for example "1521".
2. To connect to the Oracle database using the **OCI** type, select the **OCI** option and enter the following details:
 - Type the TNS name of the Oracle database instance in the **Oracle TNS name** field.
 3. In the **Oracle : User Details** dialog box, specify the details as follows:
 - In the **User name** field, type the name of an existing user with DBA rights, for example "SYSTEM".
 - In the **Password** field, type the password of the user. The password cannot be blank.
 - If the database server is running and accessible from your machine, click **Test connection** to test the connection.
 4. Click **Next** to continue.
 5. In the **Oracle: New Repository details** dialog box, specify the details as follows:
 - In the **Username** field, type the name of the user whose default tablespace will be the EMF tablespace. This name is also used for the tablespace. The user cannot already exist.
 - In the **Password** field, type the password of the user. The password cannot be blank.
 - Type the name of the EMF repository in the **Repository name** field.

SQL Server EMF database configuration screen (SQL Server only)

- **If the person who will connect to the EMF databases (the database owner) is an existing user**, enter their **Username** and **Password**.
- **If the user does not already exist**, enter a new **Username** and **Password**. A new user will be created later on, when you complete the Wizard and run the configuration.
- **If the user already exists** on the database server, and if it is running and accessible from your machine, click the **Test connection** button to test the connection. An error message, for example "Login failed", will be displayed if you entered an incorrect password for the user. You will also get an error if the user does not already exist or there is a problem connecting to the database server.
- Type the name of the EMF repository in the **Repository Database name** text box. This name is also used to name the repository and configuration databases, so it must be a valid SQL Server database name. See [allowable characters](#) for more information.

Oracle EMF tablespace configuration screen (Oracle only)

- In the **Username** text box, type the name of the user whose default tablespace will be the EMF tablespace. This name is also used for the tablespace. The user **cannot**

already exist.

- In the **Password** text box, type the password of the user. The password cannot be blank.
- Type the name of the EMF repository in the **Repository name** text box.

Install license (optional)

A temporary license is automatically installed whenever you create a new repository. However, if you have a permanent license that you want to install now, select the **Install a license** checkbox and enter the path and filename of the license file (.txt) in the **License filename** textbox.

Note: You can also install a permanent license later in the EMF Administrator.

Update Server

1. If you want to update the EMF server to use the newly created repository, then select the **Update server to use the newly created Repository** check box.
2. Click Browse and select the path to the server's EMF.properties file.
3. Now click the **Next** button to proceed to the Summary screen.

[Using the EMF configuration wizard](#)

Advanced Configuration Type

What you need to know before you continue

- Name or IP address of the machine on which the EMF server is installed.
- The location to which you want to save the EMF server database connection properties.
- Your database server type, SQL Server or Oracle.
- Connection details for database server, including server or TNS name, user and password.
- (For SQL Server only) The details of the user that will connect to the EMF databases (database owner). (For Oracle only) The details of the EMF user and tablespace.
- The name you want to call the repository and configuration databases.
- The details of the license if you want to install one during setup.
- The details of the XML file if you want to import EMF data into the repository.

The following screens are described in this topic:

- [EMF server machine](#)
- [Server database connection properties filename](#)
- [Database server](#)
- [Database connection](#)

- [SQL Server database owner](#)
- [Oracle EMF tablespace user](#)
- [SQL Server EMF databases](#)
- [Oracle EMF tablespace configuration](#)
- [Install license](#)
- [Update Server](#)

Note: Some of these screens are database-specific, so you will not see those that do not apply to your database type. For allowable characters in database names and usernames, see [Allowable characters](#).

EMF server machine screen

Browse for the machine on which the EMF server is installed, or type the name or IP address in the **EMF server machine** text box. If the EMF Server is on your local machine, you can enter "localhost" or "127.0.0.1" instead of the machine name.

Database server screen

Select either **SQL Server** or **Oracle** as your database server type.

Database connection screen

The screen depends on whether you have SQL server or Oracle. Both options are described below.

SQL Server connection screen (SQL Server only)

1. Specify the machine on which the SQL Server is located in the **SQL Server machine name** text box. If the database server is on your local machine, you can enter "localhost" or "127.0.0.1" instead of the machine name.
2. To use Windows **NT integrated security**, select the checkbox (the **Username** and **Password** text boxes will be disabled).

Note: While you can use Windows authentication to create the repository databases, the product requires Mixed Mode or SQL Server authentication to be enabled on the database server to function.

If you do not want to use Windows authentication, enter the user details as follows:

3. Specify the instance name of the SQL server, or leave it blank, if it has none.
4. To use Windows **NT integrated security**, select the checkbox (the **Username** and **Password** text boxes will be disabled).

Note: While you can use Windows authentication to create the repository databases, the product requires Mixed Mode or SQL Server authentication to be enabled on the database server to function.

If you do not want to use Windows authentication, enter the user details as follows:

- In the **Username** textbox, type the name of an existing user with database creation and Security Administrator rights, for example "sa".
 - In the **Password** text box, type the password of the user.
5. If the database server is running and accessible from your machine, click the **Test connection** button to test the connection.
 6. Specify the instance name of the SQL server, or leave it blank, if it has none.
 7. To use Windows **NT integrated security**, select the checkbox (the **Username** and **Password** text boxes will be disabled).
-

Note: While you can use Windows authentication to create the repository databases, the product requires Mixed Mode or SQL Server authentication to be enabled on the database server to function.

If you do not want to use Windows authentication, enter the user details as follows:

- In the **Username** text box, type the name of an existing user with database creation and Security Administrator rights, for example "sa".
 - In the **Password** text box, type the password of the user.
8. If the database server is running and accessible from your machine, click the **Test connection** button to test the connection.
 - In the **Username** text box, type the name of an existing user with database creation and Security Administrator rights, for example "sa".
 - In the **Password** text box, type the password of the user.
 9. If the database server is running and accessible from your machine, click the **Test connection** button to test the connection.

Oracle connection screen (Oracle only)

The following methods can be used to define the connection to the Oracle database:

- Thin Driver
 - OCI
 - User-defined (this method gives full control over the JDBC properties that will be used to connect)
1. To connect to the Oracle database using the **Thin Driver** type, select the **Thin Driver** option and enter the following details:
 - Specify the machine on which the Oracle Server is located in the **Machine name** text box. If the database server is on your local machine, you can enter "localhost" or "127.0.0.1" instead of the machine name.
 - In the **Connect using** group, select:
 - a. **SID** - to use the SID name of the Oracle database instance. Enter the SID name in the following field.

- b. **Service Name** - to use the service name of the Oracle database instance. Enter the service name in the following field. For example, to connect an Oracle 12 database, in the service name field, type the name of the pluggable database (PDB).
 - In the **Port number** field, type the connection port number, for example "1521".
2. To connect to the Oracle database using the **OCI** type, select the **OCI** option and enter the following details:
 - Type the TNS name of the Oracle database instance in the **Oracle TNS name** field.
3. To connect to the Oracle database using **User-defined** type, select the **User-defined** option, enter the following details:
 - In the **Driver name** field, select one driver from the drop down list or type your own driver name.
 - In the **URL** field, type the database connection URL string.
 - In the **Properties** table, specify the details as follows:
 - a. In the **serverName** field, same as the **Machine name** described above
 - b. In the **databaseName** field, name of the particular database on the server, also known as the "SID" in Oracle terminology
 - c. In the **serviceName** field, type the service name of the Oracle database instance
 - d. In the **driverType** field, type the driver type of the connection, for example "thin", "oci8", etc.
 - e. In the **portNumber** field, type the connection port number, for example "1521".
 - f. In the **xa_oracleFailoverCheck** field, type true or false
 - g. You could also add your own properties or remove the existing fields by the **Add\Remove** buttons
4. In the **Oracle : User Details** dialog box, specify the details as follows:
 - In the **User name** field, type the name of an existing user with DBA rights, for example "SYSTEM".
 - In the **Password** field, type the password of the user. The password cannot be blank.
 - If the database server is running and accessible from your machine, click **Test connection** to test the connection.
5. Click **Next** to continue.
6. In the **Oracle: New Repository details** dialog box, specify the details as follows:
 - In the **Username** field, type the name of the user whose default tablespace will be the EMF tablespace. This name is also used for the tablespace. The user cannot already exist.
 - In the **Password** field, type the password of the user. The password cannot be blank.
 - Type the name of the EMF repository in the **Repository name** field.

SQL Server database owner (SQL Server only)

- In the **Username** textbox, type the name of the user who will connect to the EMF databases (the database owner). If this user does not already exist, a new user will be created when you run the configuration later.

Important: you cannot use the SQL Server "sa" user, or any other SQL Server reserved user. If this is a new user, it is recommended that you enter a password for security reasons. A blank password may cause login problems if you are using SQL Server 7.

- In the **Password** textbox, type the password of the user.
- If the database server is running and accessible from your machine and the user already exists on the database server, click the Test connection button to test the connection. An error message, for example "Login failed", will be displayed if you entered an incorrect password for the user. You will also get an error if the user does not already exist or there is a problem connecting to the database server.

Oracle EMF tablespace user (Oracle only)

- In the **Username** textbox, type the name of the user whose default tablespace will be the EMF tablespace. If you want to drop the user if they already exist, select the checkbox below. If the user already exists and you have not selected this checkbox, then the configuration will fail when it runs.
- In the **Password** textbox, type the password of the user. The password cannot be blank.

SQL Server EMF databases (SQL Server only)

Type the names of the EMF configuration and repository databases in the text boxes provided. You can choose to drop the databases if they already exist by selecting the **Drop the database if it already exists** checkbox. If you have not selected this checkbox and the databases **do** exist, then the configuration will fail when it runs.

Oracle EMF tablespace configuration screen (Oracle only)

- Type the names of the EMF tablespace in the textbox provided. You can choose to drop the tablespace if it already exists by selecting the **Drop the tablespace if it already exists** checkbox. If you have not selected this checkbox and the tablespace does exist, then the configuration will fail when it runs.
- The default tablespace creation parameters are: **CREATE TABLESPACE** **%%ORATABLESPACENAME%%** **DATAFILE '%%ORATABLESPACENAME%%'** **SIZE 50M REUSE AUTOEXTEND ON NEXT 500K MAXSIZE UNLIMITED**. If you want to modify any of these parameters, or add additional parameters, you can do so in the **Override tablespace creation** textbox.

If you modify any of the parameters or use additional parameters, you must type *all* the parameters you want to use for tablespace creation. For example, if you only want to change the database size to 100MB but keep the rest of the parameters as they are in the Repository Wizard defaults, you must type **CREATE TABLESPACE %ORATABLSPACE% DATAFILE '%ORATABLSPACE%' SIZE 100M REUSE AUTOEXTEND ON NEXT 500K MAXSIZE UNLIMITED**. If you do not specify a path, the default path will be used.

You can use the following replacement tokens in your tablespace parameters:

%ORATABLSPACE%, %MACHINE% and %DBOWNER%.

Tablespace creation parameter examples

```
CREATE TABLESPACE MYTABLESPACE DATAFILE 'C:\MyData\mydata1.dbf' SIZE 100M REUSE
AUTOEXTEND ON NEXT 500K MAXSIZE UNLIMITED
```

```
CREATE TABLESPACE MYTABLESPACE DATAFILE 'C:\mydata1' SIZE 50M, DATAFILE 'D:\mydata1'
SIZE 50M REUSE
```

Install license

A temporary license is automatically installed whenever you create a new repository. However, if you have a permanent license that you want to install now, select the **Install a license** checkbox and enter the path and filename of the license file (.txt) in the **License filename** textbox.

Note: You can also install a permanent license later in the EMF Administrator.

Update Server

1. If you want to update the EMF server to use the newly created repository, then select the **Update server to use the newly created Repository** check box.
2. Click **Browse** and select the path to the server's EMF.properties file.
3. Click the **Next** button to proceed to the **Summary** screen.

[Using the EMF configuration wizard](#)

Allowable Characters in Database and Usernames

Certain fields in the EMF Repository Wizard have restrictions on the type of characters you can enter. This topic details these restrictions for both SQL Server and Oracle.

SQL Server database and user names

The first character must be one of the following:

- A letter as defined in the Unicode Standard 2.0. The Unicode definition of letters includes Latin characters from a through z and from A through Z, in addition to letter characters from other languages.
- The underscore (_).

Note: Because identifiers starting with the following characters have special meaning in SQL Server, it is recommended that you do not use them: "at" sign (@), double "at" sign (@@), number sign (#) and double number sign (##)

Subsequent characters can be:

- Letters as defined in the Unicode Standard 2.0.
- Decimal numbers from either Basic Latin or other national scripts.
- The dollar sign (\$), underscore.
- The "at" sign, number sign (only if the first character is not an "at" sign or number sign respectively).

Other restrictions:

- The database name or username (in both uppercase and lowercase) must not be a Transact-SQL reserved word.
- You cannot use embedded spaces or special characters.
- Database names and username must be from 1 and 128 characters in length.

Oracle tablespace and user names

- Tablespace names and usernames must not be longer than 30 characters
- The first character must be a letter.
- Subsequent characters can be any combination of letters, numerals, dollar signs (\$), pound signs (#), and underscores (_).

Note: If you enclose an Oracle name in double quotation marks ("), it can contain any combination of legal characters, including spaces but **not** quotation marks. Names in quotation marks are case-sensitive.

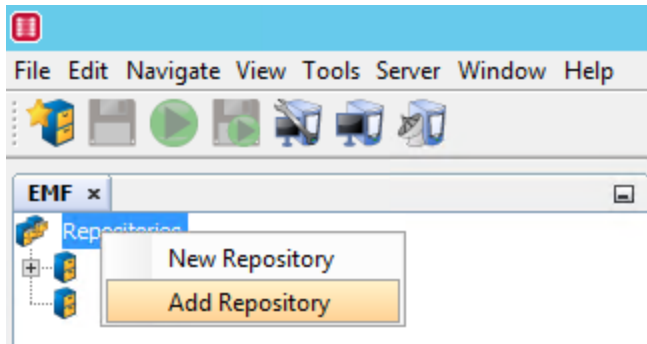
- Tablespace names and usernames are converted to upper case, and are not case-sensitive except when enclosed by quotation marks.

[Using the EMF configuration wizard](#)

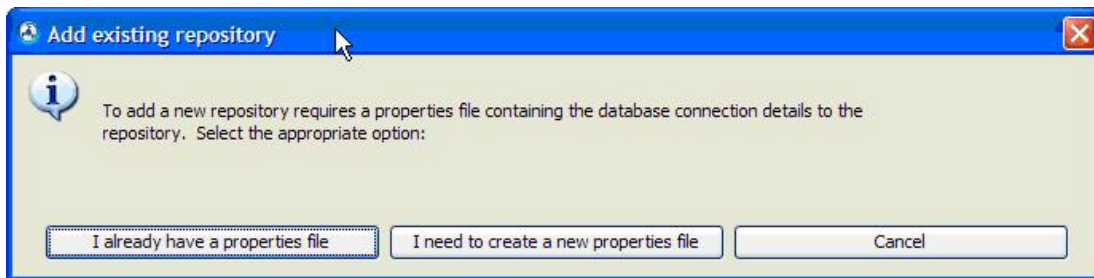
Connecting to An Existing Repository

This topic describes the procedure to connect to an existing repository, for example, if you have installed EMF on a new machine and need to connect to the existing repository.

On the EMF tree, right-click the **Repositories** node at the top and select **Add Repository** from the menu.



A dialog box appears, asking whether you already have a properties file containing connection details to the existing repository.



- If you created the repository database on another machine and can connect to the repository on that machine, copy the properties file from that machine to the one you are using.
- Select **I already have a properties file** and then browse to the file and select it to connect to the repository.
- If you no longer have the properties file, you will require to re-enter the database connection information. Select **I need to create a new properties file**. A dialog box will be displayed, asking you to specify the database type you are connecting to (Oracle or Microsoft SQL Server). Select the appropriate option. A dialog box will be displayed, asking you to set the connection details as follows:
 - For Oracle:
 - serverName – The IP address or name of the computer on which SQL Server is running.
 - databaseName – The EMF repository database name.
 - user – The user who has rights to access the EMF repository database.
 - password – The password for the user.
 - For Microsoft SQL Server:
 - serverName – The IP address or name of the computer on which SQL Server is running.
 - instance – The SQL Server instance name (check with your database administrator if you are not sure).
 - databaseName – The EMF repository database name.

- user – The user who has rights to access the EMF repository database.
- password – The password for the user.

For more information on sample settings, see [JDBC Driver Configuration Details](#).

Finally, click **OK**. You will be prompted to enter the name of the properties file to save the new settings.

Upgrading EMF Repository Databases

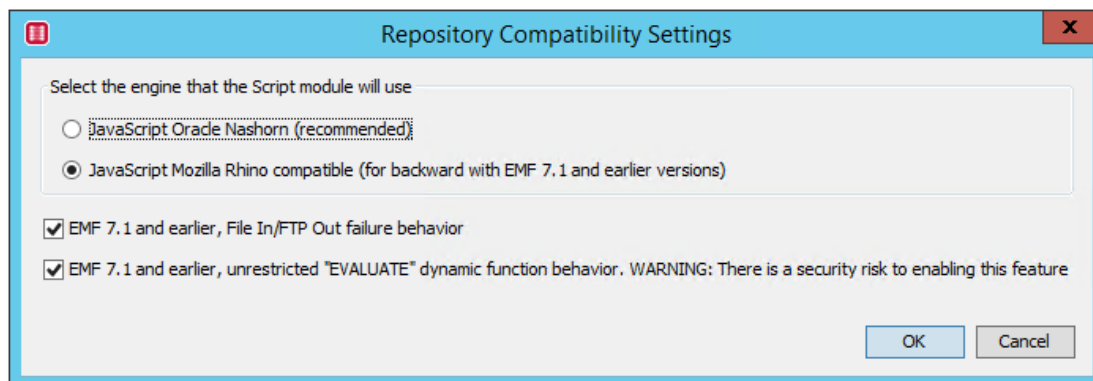
If you upgrade to a newer version of EMF, or apply a service pack, you may find that some of your databases are outdated. You will be prompted to upgrade your database before you can then log in to the repository.

Important: Before upgrading your databases you should create a backup copy in case of any problems. If necessary, ask your database administrator to do this.

[Creating a new Database](#)

Repository Compatibility Settings

Compatibility settings allows you to select settings that will maintain compatibility with older versions of EMF. In newer versions some functionality has been improved that would cause old processes to function differently - and possibly fail. On upgrading old repositories the settings are set automatically to maintain compatibility. New repositories created will by default, use the new improved functionality. Right click the **Repository** node and select the **Compatibility settings** option from the pop-up menu.



Repository Compatibility Settings defines the following options:

- The **JavaScript Script engine** in EMF is used in the following areas of EMF:
 - Javascript module.
 - The EVALUATE dynamic function.
 - The data transformation module transformer blocks.

- The data section toolbox module filter option.
- The SQL Insert module to evaluate the data value.

The newer **JavaScript Oracle Nashorn** engine implements the ECMAScript 5.1 specification. There are some language changes over the **JavaScript Mozilla Rhino** engine so some scripts may no longer run without modification. The following link provides some details of moving scripts from the Rhino engine to Nashorn <https://wiki.openjdk.java.net/display/Nashorn/Rhino+Migration+Guide>. Old scripts may also run under Nashorn but the output may be slightly different. For example the following dynamic function \$EVALUATE('2 + 2')\$ would output "4.0" with the Rhino engine, but "4" under Nashorn.

- If **EMF 7.1 and earlier File In/FTP Out failure behavior** is selected, then when a file cannot be accessed/created in the File In/FTP out module then a warning is logged. With the new behavior, the module will error. The process will then fail, or an error link can be added to take alternative action.
- If **EMF 7.1 and earlier, unrestricted "EVALUATE" dynamic function behavior** is selected, all **EVALUATE** dynamic functions are processed and resolved. This however opens up possible security risks. Therefore it is recommended to de-select this option so only top level **EVALUATE** dynamic functions are resolved, and **EVALUATE** dynamic functions embedded in other message/data sections will be not be processed (a warning will be logged instead).

Security considerations: Similar to SQL Injection attacks, it is possible for a malicious attacker to insert an EVALUATE statement into an incoming message which would then cause the JavaScript to run on the EMF server when the data was processed. For example, a POP3 or IMAP listener receiving an incoming email – when the **EMAIL_IN** message section was processed in EMF, if it contained an **EVALUATE** dynamic function this would also be processed and the unknown script command executed. It would be easy for an attacker to send an email to a mailbox that EMF was monitoring containing malicious code. By de-selecting the unrestricted **EVALUATE** behavior this risk is eliminated.

Logging Into the EMF System

When you start the EMF UI, you must log in before you can access the system.

The **Login Screen** contains three fields (**Operator name**, **Password** and **Directory**) that must be completed before the EMF system will allow access.

- **Operator Name:** Set up by the Administrator in the System operators [Administration](#) screen. It is this operator ID which is used to gain access to the repository and all EMF Processes. Security options, as set by the Administrator, define which parts of the EMF system the operator has access to. Only operators with EMF System Entry Security Rights enabled can log in to the EMF System.

Each **Operator name** that you can log in with has [security rights](#) assigned to it. When the correct password and repository are entered, login will proceed and you can use the areas of the EMF system that you have been given access to. EMF system access for operators is controlled using the [Roles](#) functionality.

Note: When EMF is first installed, there are four default operators (SuperAdmin, Admin, Server, and Script). Server and Script are system users, and are not designed to be logged in with. The default password for SuperAdmin and Admin is a blank password. On initial login, you will be forced to change this and set a password that matches the default password policy.

- **Password:** Set up by the Administrator in the System Operators [Administration](#) screen, this is the password that allows access to the repository.
-

Important: Passwords are case sensitive.

- **Directory:** This is the name of the directory where the Operator details that you are logging in with are stored. If you are logging in with internal operators that have been created within EMF and do not exist outside of EMF, then select **local**. If you have configured external operators in your Roles, then select the JNDI service name from the directory list that contains the connection details to the LDAP store in which your user details are stored. Your system administrator should be able to provide you with this information. Note that only those JNDI services that have the check box **Enable operators to login against this service** are available in the drop-down list.

[EMF Security](#)

Using the EMF Tree View

When you have successfully logged in to EMF, a vertical row of icons is shown in the left pane of EMF Administrator. This is known as the **EMF Tree View**. These icons provide access to all the configuration and setup options that affect the working of the EMF administrator. They are described here in the order in which they appear.

This section of the Help lists all the icons that appear in the EMF tree view and explains their function.

Important: You should have some understanding of the EMF System before changing any of these settings.

To access the settings and options in the Tree view:

Click the relevant icon. Icons with a plus sign (+) next to them can be expanded to reveal further sub-options.

In many cases, you can create a new item of the appropriate type by right-clicking in the pane on the right of the screen (or on the icon itself) and selecting **New (item)**.

- **EMF Process folders** are where you store your EMF Processes. There is one system folder that cannot be removed or renamed, as well as any folders that you create.
 - The **System** folder contains EMF Processes that are used by the EMF system itself. For example, **SMSSystemProcess** is used to locate EMF Processes that should be fired by an SMS message that has been received by the EMF System.
 - **Any other folders that you create** can be used to store and organize your EMF Processes in whichever way is most meaningful and useful. Note that you must have Folder creation security rights in order to create new folders.

For further information, see [Folder organization](#).

- The **System** node contains the various system configuration components that define how the system works and looks. See [Setting System Options](#).
- The **Recipient Administration** node contains the details of the recipients of the EMF system. It allows you to view and manage details of the recipients of the EMF system. See [Recipient Administration](#).
- The **Data Sources** node contains the Data Sources that are used to create the connections required to run your EMF against client databases. See [Creating Connections to Data Sources](#).
- The **Services** node is used to define the different output methods and services that can be used to send EMF Processes. See [Services](#).
- The **SAP GRC** node contains the details of SAP Domains. It allows you to create new domains and select the SAP connection that is used to retrieve Authorization/Field objects from the selected SAP connection. See [Manage Domains](#).
- The **Triggers** node contains the database triggers that can be used to fire an EMF process. Both the SQL Server/Oracle triggers use the EMF API to communicate with the EMF Server. See [Database Trigger Wizard](#).
- The **White Lists** node contains the white list definition that defines the list and the fields that are used to identify white listed items. A white list is a list of items that have been marked as being expected or allowable. See [White List Definition](#).

[Configuring the EMF System](#)

The System Node

The items in the **System** section of the EMF tree view are concerned with configuring system settings and options for your installation of EMF. You can view and modify details of the various items by double-clicking their icon to open the relevant Properties screen. In many cases you can also create a new item of the appropriate type by right-clicking in the right-hand pane and selecting **New (item)**.

The **System** section contains the following:

- [Queue Providers](#) is used to identify and connect to the queueing technology that you are using (see **Queues**, below). It tells the EMF server where to locate the queueing system configuration and how to connect to it.
- [Languages](#) is used to define the languages that can be used when creating Output Formatting modules.
- [Machines](#) is used to define the physical computers that EMF is running on. Each machine specified here has two further icons beneath it; [Servers](#) (used to configure the server instances running on each machine) and Activity (used to view status information on the activities being undertaken by these server instances).
- License Manager is used to install a license, or change the existing license. For more information, see [Viewing License Information for a Repository](#).

- [Queues](#) is used to create and define your system processing (logical) queues, which are used to ensure guaranteed delivery of EMF Processes.
- The items under the **Net Security** icon are used to install and manage information about Key stores and other authentication protocols. See [Managing Authentication Protocols](#) and [Managing Key Stores](#).
- [Operators](#) is used to create operators in the EMF system or edit or remove the details of an existing operator. A new installation of EMF contains four operators, including a default operator which is used to log in to the system.
- [Roles](#) is used to limit an operator's access to the EMF system. You can add a new role, or edit or remove the details of an existing role. Also see [Naming and Describing a Role](#).

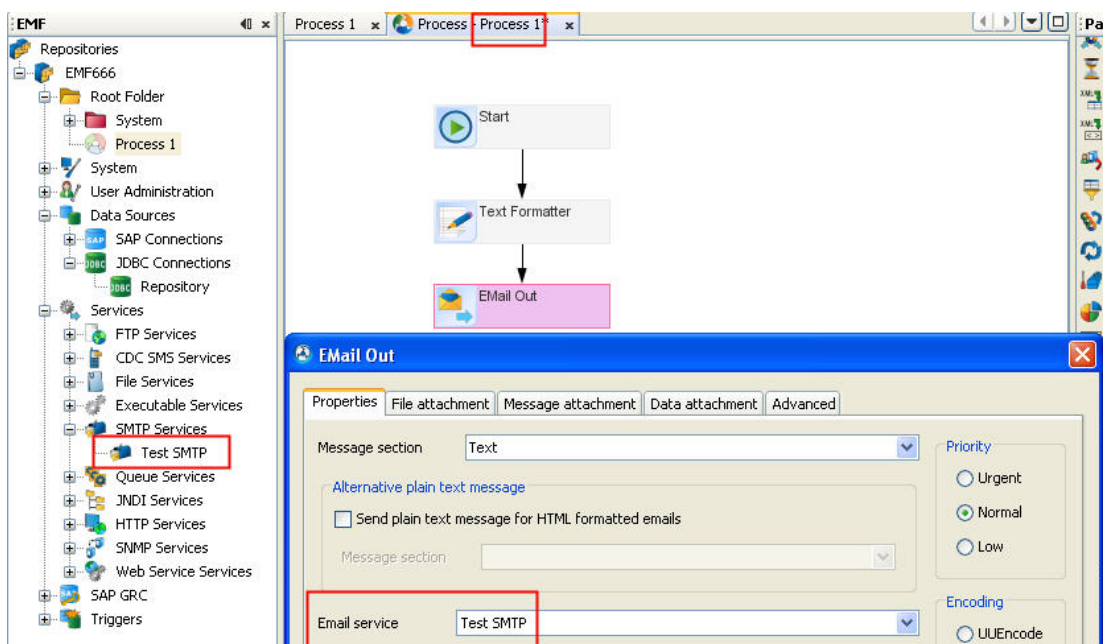
[The Icons in the EMF Tree View](#)

[Configuring the EMF System](#)

Find Usages

The **Find Usages** option available for most items in the EMF tree allows you to view all instances of where the item is being used. Right-click the item of interest and click **Find Usages** to display a list of processes, users, and services that use a particular item.

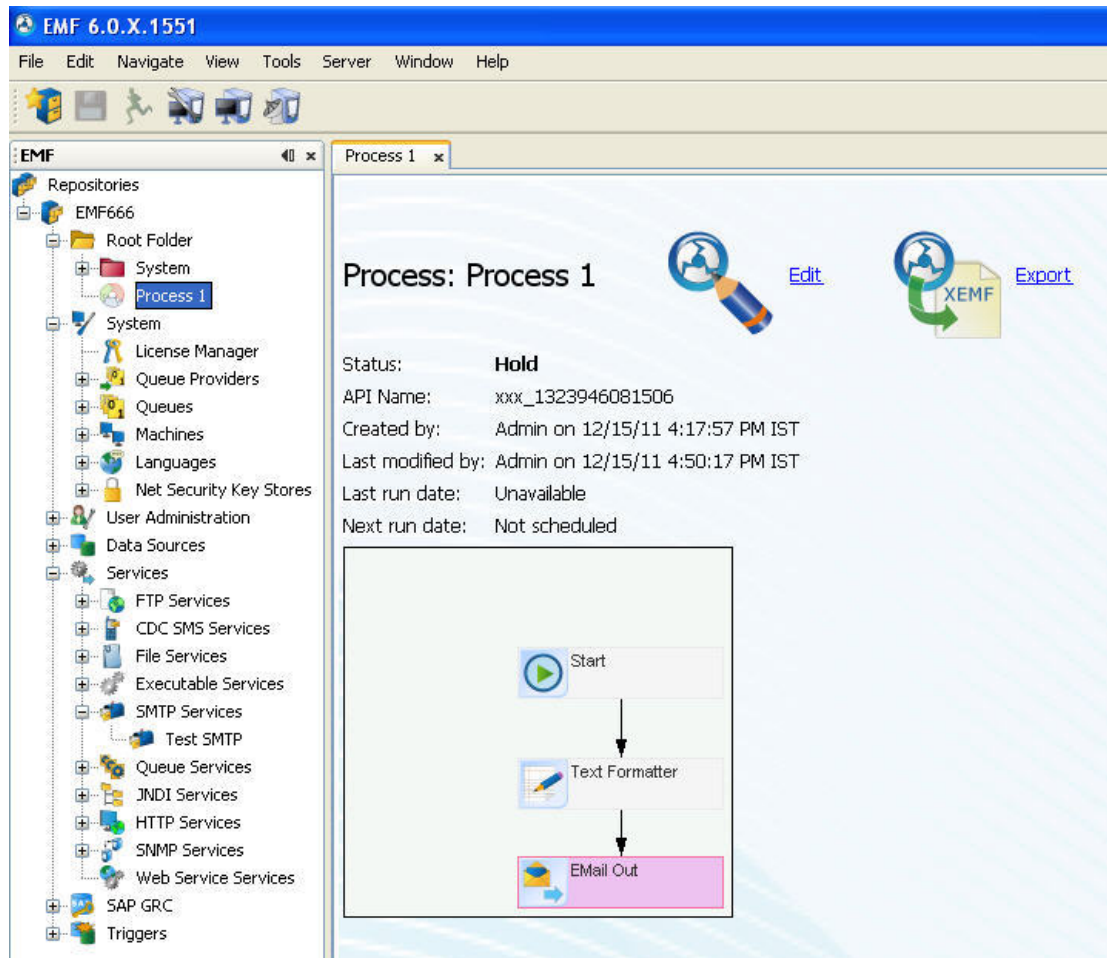
For example, consider a process 'Process 1' with an Email Out module which uses an existing SMTP service.



1. Right-click on the SMTP service 'Test SMTP' and click **Find Usages**. The process 'Process 1' is displayed in the Find Usages window. If the Find Usages window is not visible, select **Usages** from the Window menu to open it.
2. Click the plus sign (+) next to the process name. It displays the module in which the SMTP service is used.

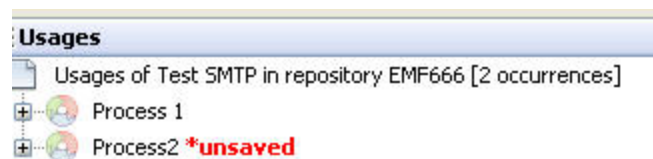
3. Right-click the process name to select additional options:

- **Select in Explorer** - The process name is highlighted in the tree view.



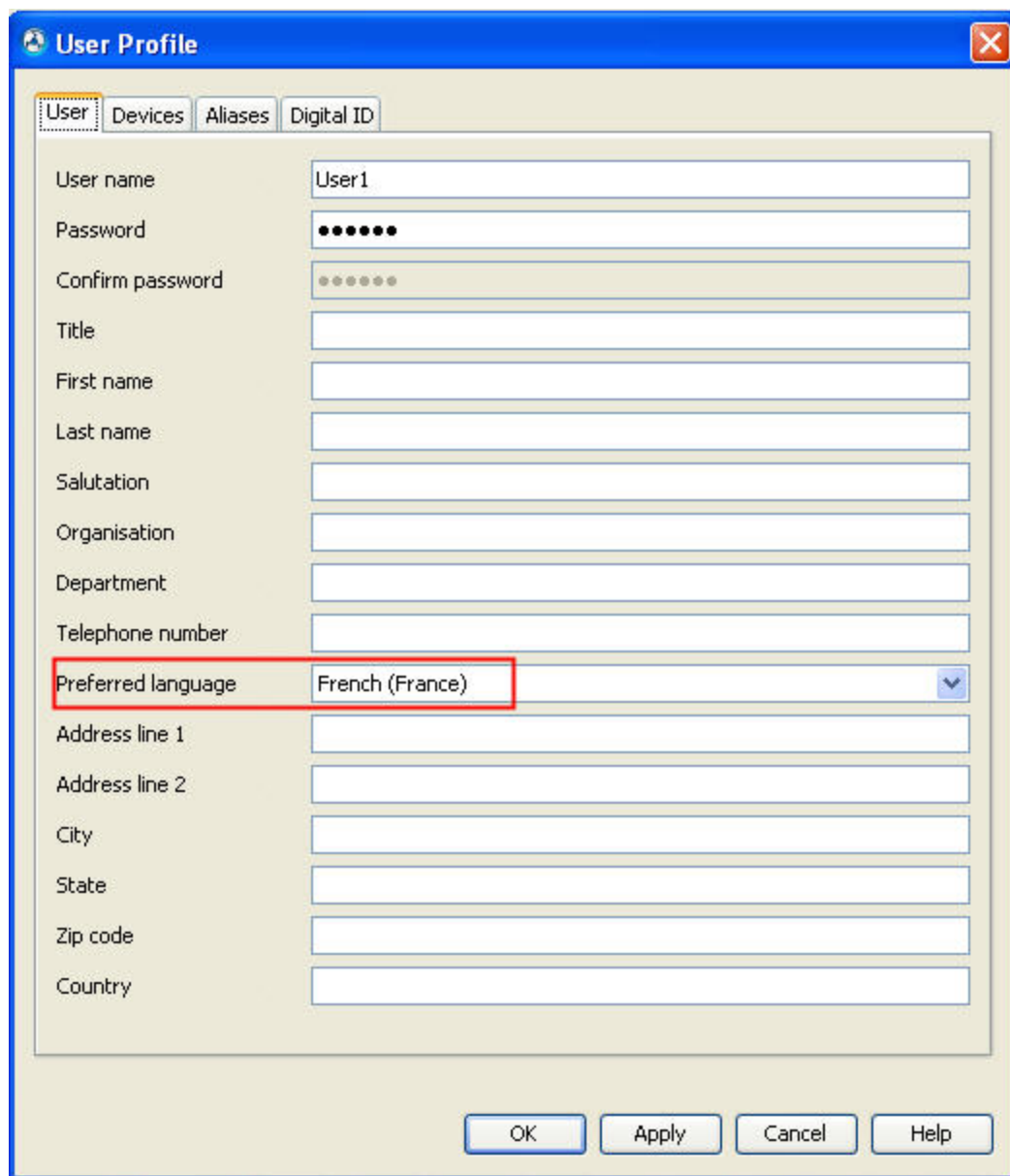
- **Edit** - Opens the process for editing in the EMF Process Builder.

If an item is being used in a process that is currently being edited, then it will be marked as 'unsaved'.



This helps when you want to delete an item, because you cannot delete it when it is still being used.

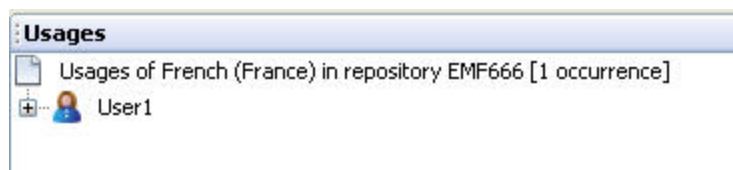
The **Find Usages** option is not limited to displaying usages of items in processes. For example, click **Find Usages** on French language to see the list of users having French as their preferred language.



The 'User Profile' dialog box is shown with the 'User' tab selected. It contains various input fields for user information. The 'Preferred language' field is highlighted with a red rectangle and shows 'French (France)' selected from a dropdown menu. The fields are as follows:

Field	Value
User name	User1
Password	••••••
Confirm password	••••••
Title	
First name	
Last name	
Salutation	
Organisation	
Department	
Telephone number	
Preferred language	French (France)
Address line 1	
Address line 2	
City	
State	
Zip code	
Country	

Buttons at the bottom: OK, Apply, Cancel, Help.



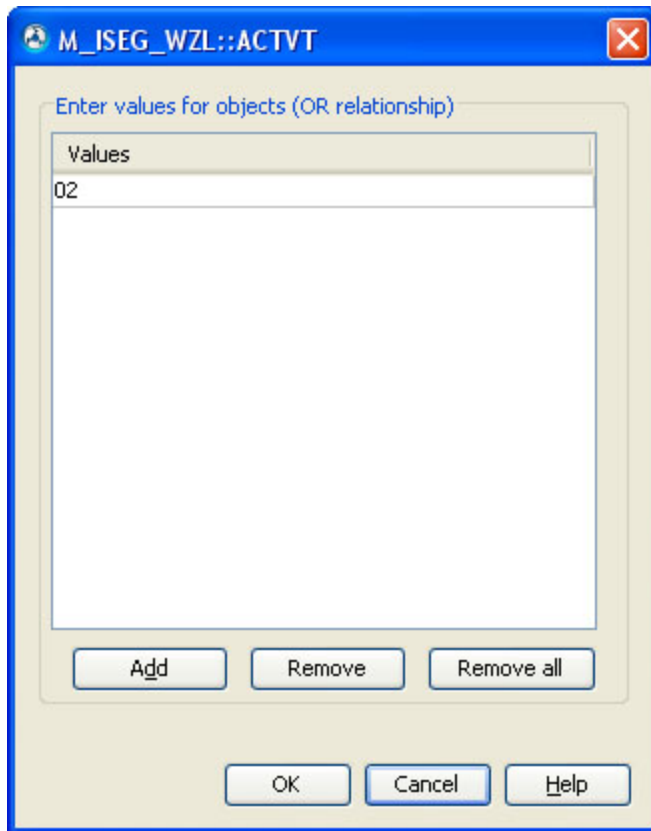
Manage Domains

In the tree view, click (+) next to the **SAP GRC** icon. Right-click **SAP Domains** and select **New**.

- Authorizations defined will override the values for the same Object/Field combination in the main Properties tab of the SAP Authorization module. Authorizations defined, that do not have the object/field defined in the main properties tab of the SAP Authorization module are ignored when the SAP Authorization module is run. Authorizations defined here will follow the same AND/OR relationship behavior as in the SAP Authorization module.

- Page 2-25

5. Click **Remove** to delete the selected row/domain detail.
6. To enter data in the Values column, double-click the cell. This brings up the following dialog box.



7. Click **Add** to add empty rows to the grid.
8. Enter the the values and click **OK** to register the new values on the object details grid.
9. Click **Cancel** to abort any changes.

Maintaining the EMF Server

The following topics contain information intended to assist you in setting up and maintaining the EMF system.

Note: Most of the tasks that are described here are performed using the utilities and applications that you can access from the EMF **Tree view**. The icons in the tree view are described [here](#) in the order in which they appear.

The following areas are covered in these topics:

- [Managing the physical machines that EMF runs on.](#)
- [Configuring your queue provider](#) and setting up the EMF queues.
- [Configure EMF Server Settings](#)
- [Managing the EMF Server](#), including and running it as a Windows service.

If you experience problems with your installation of EMF, please [consult this list](#) of places where EMF configuration details can be checked in case of errors.

Managing the Machines that EMF Runs On

The EMF system (administrator, server, services and queues) can all run on the same machine or, to increase efficiency and handle large volumes of traffic, you can use multiple machines. You can view all the machines that EMF [server instances](#) are running on by clicking the **Machines** icon in the **System** section of the EMF tree view. A list of the machines is displayed in the right-hand pane. If you want to run server instances on a machine that is not listed here you must add it to the list.

To view details of a machine:

- Right-click on the machine name and select **Edit**.

To view the servers running on a machine:

- Double-click the machine name.

To add or modify details of a machine:

1. Right-click in the list view and select **New machine**, or double-click an existing machine, to display the **Machine** screen.
2. Enter the machine's network identification details (either the IP address or an actual name assigned to the machine) in the **Name** field.
3. Enter any additional information about the computer, e.g. the name of the person who normally uses it, in the **Description** field.
4. **If you would like your Queue provider to start automatically when this server starts**, select the **Auto-start Queue Provider** option and then:
 - Enter a **Maximum period for startup** (this is the maximum amount of time, in milliseconds, to wait for the Queue Provider to start up before reporting a failure or error). Alternatively, select the **Wait forever** option to force the server to wait indefinitely.
 - If you wish, select an alternative **Startup Script** to control how the Queue provider starts. In most cases you should not need to change this setting from the default script that is generated automatically when you create (or upgrade) a database.

Note: Whenever you use the [EMF Repository Wizard](#) to create a new database, the queue provider is set to start automatically, by default with the EMF server. If you wish to disable auto-start you must choose "typical" or "advanced" setup and deselect the **Start queue provider automatically when EMF server starts** option (this is not available with the express setup).

Important: in order to use the **Auto-start Queue Provider** option, your queueing software must either be on the same physical machine as the EMF server, or else installed on a mapped network drive (the scripts run in DOS mode, which cannot use UNC paths).

5. If you would like the Queue provider to shut down automatically when the server stops, select the **Auto-stop Queue Provider** check box and enter the location of the **Stop Script**. In most cases you should not need to change this setting from the default script that is generated automatically when you create (or upgrade) a database.
6. To enable the embedded queue provider, select the **Use internal queue provider** check box. A third-party external queue provider is not required in this case. The internal queue provider configuration (location of the saved messages, max queue sizes, and so on) can be changed by editing the `public\emf\6\activemq.xml` file. By default, the public folder is present in the c-drive. To change the storage space/location edit the following parameters in the file:
 - `systemUsage`: disk space parameters
 - `dataDirectory`="`${emf.queue.datadir}`queue (2 locations in file)"

Warning: If a large amount of queue data is expected, make sure the disk has enough available free space for message storage.

Please refer to <http://activemq.apache.org> for more information on configuring this file. Contact EMF Support to check if your configuration changes are compatible with EMF.

Note: The active queue provider under the System -> Queue Providers node must be changed to connect to the internal queue provider. The provider URL is `failover://(tcp://localhost:61616)?timeout=3000&maxReconnectDelay=10000&jms.prefetchPolicy.queuePrefetch=0` and the initial context factory as `org.apache.activemq.jndi.ActiveMQInitialContextFactory`.

In a multiple machine load-balanced environment, the URL should be adjusted to include all machines that are running the internal queue provider, such as `failover://(tcp://server1:61616,tcp://server2:61616)?timeout=3000&maxReconnectDelay=10000&jms.prefetchPolicy.queuePrefetch=0&randomize=false`

[Explanation of Server Instances](#)

[Configuring System Options](#)

[Configuring the EMF System](#)

Configuring Server Instances

Most of the EMF [Initiator modules](#) require you to create a corresponding **server instance** associated with a server running on a particular machine (although you can have multiple server instances running under a single service).

To view and create server instances:

1. Expand the **Machines** icon from the **System** section of the EMF tree view.
2. Right-click the required machine to display all the server instances on the selected machine.

3. To create a new server instance,
 - Right-click the **Servers** icon, select **New**, and then choose the name of the server you want to create.

The following are available (click a link to learn more about it):

- [POP3 Email Listener](#)
 - [Queue Listener](#)
 - [SNMP Listener](#)
 - [HTTP Listener](#)
 - [IMAP Email Listener](#)
- Right-click the **Servers** icon, select **New > System**, and then choose the name of the server you want to create.

The following are available (click a link to learn more about it):

- [Server Manager](#)
- [Scheduled Alerts Retriever](#)
- [Escalated Alerts Retriever](#)
- [Debug Log Manager](#)
- [Statistics Log Manager](#)
- [Sent Message Log Manager](#)
- [Audit Log Manager](#)
- [Synchronized Alerts Retriever](#)
- [Event Plan Clean-up Manager](#)
- [Running Event Plan Manager](#)
- [Missed Events Manager](#)

[Managing the Machines that EMF Runs On](#)

[Scaling EMF](#)

[Configuring System Options](#)

Audit Log Manager

You can use the **Audit Log Manager** server to define how often the EMF system [Debug log](#) should be checked to see if it has exceeded the size or age specified in the Audit logging [Size Management Screen](#). If it is older than the maximum specified age or larger than the maximum specified size, the Audit Log Manager will delete the oldest entries to make room for new ones.

The default setting is to check the audit log once per minute, but you can change this if you wish.

To create or modify an Audit Log Manager:

1. Select the machine on which the Audit Log Manager server is running from the

System > Machines section of the EMF tree view.

2. Right-click the **Servers** icon and then:
 - To create a new server instance, right-click in the list view, click **New > System**, and then select **Audit Log Manager** to display the **Audit Log Manager** screen.

Note: Typically, a machine should only ever have one **Audit Log Manager** instance.

 - To modify an existing server instance, double-click the icon for the appropriate server instance.
3. Enter the name of the Audit Log Manager to be created in the **Name** text box.
4. Enter the required frequency in the **Check Frequency** field as a number of seconds. This is the only option that requires setting (the location of the log file is obtained from the Audit logging component when it is polled by the server).
5. Select the [Advanced](#) tab to specify advanced options.

[Adding Server Instances](#)

[Explanation of Audit Logging](#)

[Configuring the EMF System](#)

Sent Message Log Manager

You can use the **Sent Message Log Manager** server to define how often the EMF system The Sent Message log should be checked to see if it has exceeded the size or age specified in the Sent Message logging [Size Management Screen](#). If it is older than the maximum specified age or larger than the maximum specified size, the Sent Message Log Manager will delete the oldest entries to make room for new ones.

The default setting is to check the Sent Message log once per minute, but you can change this if you wish.

To create or modify a Sent Message Log Manager:

1. Select the machine on which the Sent Message Log Manager server is running from the **System > Machines** section of the EMF tree view.
2. Right-click the **Servers** icon and then:
 - To create a new server instance, right-click in the list view, click **New > System**, and then select **Sent Message Log Manager** to display the **Sent Message Log Manager** screen.

Note: Typically, a machine should only ever have one **Sent Message Log Manager** instance.

 - To modify an existing server instance, double-click the icon for the appropriate server instance.
3. Enter the name of the Sent Message Manager to be created in the **Name** text box.

4. Enter the required frequency in the **Check Frequency** field as a number of seconds. This is the only option that requires setting (the location of the log file is obtained from the Billing logging component when it is polled by the server).
5. Select the [Advanced](#) tab to specify advanced options.

[Adding Server Instances](#)

[Configuring the EMF System](#)

Debug Log Manager

You can use the **Debug Log Manager** server to define how often the EMF system [Debug log](#) should be checked to see if it has exceeded the size or age specified in the Debug logging [Size Management Screen](#). If it is older than the maximum specified age or larger than the maximum specified size, the Debug Log Manager will delete the oldest entries to make room for new ones.

The default setting is to check the debug log once per minute, but you can change this if you wish.

To create or modify a Debug Log Manager:

1. Select the machine on which the debug log manager server is running from the **System > Machines** section of the EMF tree view.
 2. Right-click the **Servers** icon and then:
 - To create a new server instance, right-click in the list view, click **New > System**, and then select **Debug log manager** to display the **Debug Log Manager** screen.
-
- Note:** Typically, a machine should only ever have one **Debug Log Manager** instance.
-
- To modify an existing server instance, double-click the icon for the appropriate server instance.
 3. Enter the name of the Debug Log Manager to be created in the **Name** text box.
 4. Enter the required frequency in the **Check Frequency** field as a number of seconds. This is the only option that requires setting (the location of the log file is obtained from the Debug logging component when it is polled by the server).
 5. Select the [Advanced](#) tab to specify advanced options.

[Adding Server Instances](#)

[Explanation of Debug Logging](#)

[Configuring the EMF System](#)

POP3 Email Listener

The **POP3 Listener** server instances poll the POP3 Email Servers for email messages, ensure that these are handled properly, and initiate the appropriate EMF Processes as necessary.

Important: The POP3 Listener supports both plain text and HTML email messages, as well as email attachments.

You can define an email account, choose how often an email server should be polled, and set other parameters using the email listener server **Properties** screen. When mail is received by the specified account, the [POP3 Router](#) is sent the message details to decide which EMF Processes are to be fired.

Creating a EMF-specific email account

It is **strongly recommended** that you establish a separate email account specifically for use with the EMF POP3 Listener, for example EMF Processes@organization.com.

Suggestions for keeping a record of the email messages retrieved:

- **(Recommended)** Set up two email accounts and configure the first email account to auto forward messages to the second email account. The POP3 listener retrieves (and deletes) the messages from the second email account, leaving a record of all the messages in the first email account. All read messages are retained on the server.
- Use EMF to write the information to another location, for example send an email message or write the message to a file.

To create or modify a POP3 listener:

1. Select the machine on which the server should run from the **System > Machines** section of the EMF tree view.
2. Right-click the **Servers** icon, and then:
 - To create a new server instance, right-click in the list view and click **New** and then click **POP3 Listener**.
 - To modify an existing server instance, double-click the icon for the appropriate server instance.
3. Enter the name of the POP3 Listener to be created in the **Name** text box.
4. Select a System EMF Process that contains a POP3 router from the **Process** drop-down list. This is the EMF Process that will be used to check whether any EMF Processes should be fired when an email is received.
5. Specify the name or address of the POP3 Email server that should be polled in the **POP3 server** text box .
6. Enter the **Port number** that you want the POP3 listener to use. The port number that is reserved for the POP3 protocol is **110**, but you may want to specify a different port number.
7. Select the **Authentication** mode to use when connecting to the emails server. If secure authentication is selected, then the POP3 listener will try to use the following methods to authenticate the user: CRAM-MD5, PLAIN, DIGEST-MD5.

8. Enter the **User Name** and **Password** for the email account to listen to.
9. Select the [Email Retrieval](#) tab to specify where messages are retrieved from and the type of messages to retrieve.
10. Select the [Message Properties](#) tab to define settings for handling HTML email messages and attachments.
11. Select the [Advanced](#) tab to specify a **Polling Interval** to define how often the listener should check for email-initiated process and to set other advanced options.

[Adding Server Instances](#)

[IMAP Email Listener](#)

[Configuring System Options](#)

IMAP Email Listener

Internet Message Access Protocol (IMAP) is a method of accessing electronic mail or bulletin board messages that are kept on a (possibly shared) mail server

The EMF **IMAP Listener** polls the IMAP Servers for any incoming email messages, initiating the appropriate EMF Processes as necessary.

Important: The IMAP Listener supports both plain text and HTML email messages, as well as email attachments.

You can define an email account, choose how often the servers should be polled, and set other parameters using the IMAP email listener server **Properties** screen. When mail is received by the specified account, the [Email Router](#) is sent the message to decide which EMF Processes are to be fired.

Unlike the POP3 Listener, it is possible to configure the IMAP Listener to read only unread messages and not to delete messages as they are read from the email account.

To create or modify an IMAP listener:

1. Select the machine on which the server should run from the **System > Machines** section of the EMF tree view.
2. Right-click the **Servers** icon, and then:
 - To create a new server instance, right-click in the list view and click **New** and then click **IMAP Listener**.
 - To modify an existing server instance, double-click the icon for the appropriate server instance.
3. Enter the name of the IMAP Listener to be created in the **Name** text box.
4. Select a System EMF Process that contains an email router from the **Process** drop-down list. This is the EMF Process that will be used to check whether any EMF Processes should be fired when an email is received by the account being defined in this IMAP Listener.
5. Enter the machine name/IP address of the **IMAP Server** that should be polled.

6. Enter the **Port number** that you want the IMAP listener to use. The port number that is reserved for the IMAP protocol is **143**, but your IMAP server may be running on a different port number.
7. Select the **Authentication** mode to use when connecting to the emails server. If secure authentication is selected then the IMAP listener will try to use the following methods to authenticate the user: CRAM-MD5, PLAIN, DIGEST-MD5.
8. Enter the **User Name** and **Password** for the email account to listen to.
9. Select the [Email Retrieval tab](#) to select where messages are retrieved from and the type of messages to retrieve.
10. Select the [Message Properties](#) tab to define settings for handling HTML email messages and attachments.
11. Select the [Advanced](#) tab to specify a **Polling Interval** to define how often the listener should poll the IMAP Servers for any incoming email messages and to set other advanced options.

[Adding Server Instances](#)

[Configuring System Options](#)

Escalated Alerts Retriever

The **Escalated Process Retriever** server instances monitor any EMF Processes that have escalation parameters in order to check whether they have been responded to. If the required responses are not received, the EMF Process is escalated.

Note: By default, one **Escalated Alerts Retriever** already exists in a new installation.

To create or modify an Escalated Alerts Retriever:

1. Select the machine on which the server should run from the **System > Machines** section of the EMF tree view.
 2. Right-click the **Servers** icon, and then:
 - To create a new server instance, right-click in the list view and click **New > System > Escalated Alerts Retriever**.
-
- Note:** Typically, a machine should only ever have one **Escalated Alerts Retriever** instance.
-
- To modify an existing server instance, double-click the icon for the appropriate server instance.
 3. Enter the name of the Escalated Alerts Retriever to be created in the **Name** text box.
 4. Select the [Advanced](#) tab to specify a **Polling Interval** to define how often the EAR should check for responses and to set other advanced options.

[Adding Server Instances](#)

[Configuring System Options](#)

HTTP Listener Server

The **HTTP Listener** server listen for incoming requests from clients and make sure that any responses are sent back to the correct client. If a client request is successful the connection is stored in the 'client connection store' (accessible by any EMF runtime components), which then generates a unique key associated with the stored connection. This key, together with the ID of the HTTP Listener and the response queue, are then stored in a message section that the listener creates in the EMF Process, in the format:

```
ListenerID=<HTTPListenerID>;ClientConnectionKey=<ClientConnectionID>;Response
Queue=<QueueName>.
```

Note: If you want to be able to send responses back to the client from an EMF Process, ensure that you include and configure an [HTTP module](#) with **Client response** enabled. This HTTP module can be in the same EMF Process as the HTTP in module (see [example](#)), or it can be in a slave EMF Process cascaded to by the EMF Process that contains the HTTP in module (see [example](#)).

To create or modify an HTTP listener server:

1. Check that a suitable **HTTP Response Queue** has been configured. By default, the HTTP response queue is **HTTPQ**. However, you can use any queue you like as long as it is not used by any other parts of the EMF system. **The response queue must be on the same machine as the HTTP listener.**
2. Select the machine on which the server should run from the **System > Machines** section of the EMF tree view.
3. Right-click the **Servers** icon, and then:
 - To create a new server instance, right-click in the list view and click **New > HTTP Listener**.
 - To modify an existing server instance, double-click the icon for the appropriate server instance.
4. Enter the name of the HTTP Listener to be created in the **Name** text box.
5. Select the system **EMF Process** that should be run when a request from an HTTP client is received, from the drop-down list of all those on your system.

Note: Normally, you should select the preconfigured **HTTP System EMF Process**.

6. Select the **Response queue** that should be polled for generated client responses from the drop-down list of all the configured queues.

Note: by default, the HTTP response queue is **HTTPQ**. However, you can use any queue you like as long as it is not used by any other parts of the EMF system. **The response queue must be on the same machine as the HTTP listener.**

7. Enter the **Port Number** that the HTTP Listener should monitor for incoming requests.
8. Specify whether a secure transaction is required and, if so, which type, by selecting the required **Socket type** from the dropdown list. The available options are:
 - **Unsecured** - data is not protected.
 - **SSL (Secure Sockets Layer) 2.0**

- **SSL (Secure Sockets Layer) 3.0**
- **TLS (Transport Layer Security) 1.0**

Note: If you select **SSL 2.0**, **SSL 3.0** or **TLS 1.0** as the Socket type, you should further define your security settings on the [SSL settings](#) tab (this is not available if you select **Unsecure**).

- Specify how long an incoming connection should be kept open while waiting for data from the client, by entering a number of seconds in the **Request timeout** field.
 - Specify the maximum number of simultaneous connections that the HTTP listener should process by entering a number in the **Max. connections** field.
-

Note: the maximum number of threads that will be created for processing incoming HTTP connections is defined by the MAX_HTTP_LISTENER_THREADS setting in **EMF.properties** - the default setting is 20. When the maximum number of threads in use is reached, any additional connections must wait for a thread to become available. Increasing the maximum number of threads will increase the memory and processor requirements of the server, the optimum value is dependant upon each individual installation.

- If you want to keep alive the HTTP connection that the Listener service is using once it is established, then click the **Enable** check box and specify the time (in seconds) the connection should be kept open for waiting for other incoming requests. If no client request has been made within the specified timeout period, the connection is closed. This can improve performance when multiple incoming requests are expected from the same source.
- Select the [Request handling](#) tab to specify whether or not to respond immediately to requests that use a particular method and, if so, what that response should be.
- Select the [Advanced](#) tab to specify a **Polling Interval** to define how often internally the HTTP Listener checks to see if a response is ready to send back to the client. The [Request data](#) section allows you to specify whether incoming requests should be parsed, stored in their entirety, or both.

[How HTTP Works in EMF](#)

[Adding Server Instances](#)

[Configuring System Options](#)

Queue Listener

The **Queue listener** polls the external queue for any incoming queue messages that should initiate EMF Processes.

Note: the communication between EMF and the queueing system is handled by the Java Message Service (JMS). For further information on JMS see [Messaging and the Java Message Service](#) or consult the JMS home page at <http://java.sun.com/products/jms/>.

To create or modify a queue listener:

1. If you have not already done so, configure an [external queue provider](#) and an [external queue](#) using the **Services > Queue services** section of the EMF tree view.
2. Select the machine on which the server should run from the **System > Machines** section of the EMF tree view.
3. Right-click the **Servers** icon, and then:
 - To create a new server instance, right-click in the list view and click **New > Queue Listener**.
 - To modify an existing server instance, double-click the icon for the appropriate server instance.

Note: Unlike most other server instances, there is no default Queue listener server instance.

4. Enter the name of the Queue Listener to be created in the **Name** text box.
5. Select a [System EMF Process](#) that contains a Queue router from the **Process** drop-down list (the queue router is used to ensure that the correct EMF Process(s) are fired when an incoming queue message is detected).
6. Select the **Queue provider name** and **Queue name** that you want the Queue listener to poll.
7. If you want to define a rule for specifying which messages should be selected from the queue, type your query in the **JMS selector string** field (this is optional). Only those messages whose headers match the selector string will be delivered.

Example: "JMSType = 'car' AND color = 'blue' AND weight > 2500" selects messages with a message type of car and color of blue and weight greater than 2500 lbs.

The message selector syntax is based on a subset of SQL92, a SQL standard produced by the American National Standards Institute (ANSI) and the International Standards Organization (ISO). For more information on message selector syntax see [Messaging and the Java Message Service](#) or consult the JMS home page at <http://java.sun.com/products/jms/>.

8. Select the [Advanced](#) tab to specify a **Polling Interval** to poll the external queue for any incoming queue messages and to set other advanced options.

[Server Instance Screen](#)

[Configuring System Options](#)

Scheduled Alerts Retriever

The Scheduled Alerts Retriever server is used to specify how often the system should check the repository for EMF Processes that are scheduled to run. This class of servers ensure that scheduled EMF Processes are run on time.

Note: By default, one **Scheduled Alerts Retriever** already exists in a new installation.

To create or modify a Scheduled Alerts Retriever:

1. Select the machine on which the server should run from the **System > Machines** section of the EMF tree view.

2. Right-click the **Servers** icon, and then:
 - To create a new server instance, right-click in the list view and click **New > System > Scheduled Alerts Retriever**.

Note: Typically, a machine should only ever have one **Scheduled Alerts Retriever** instance.

 - To modify an existing server instance, double-click the icon for the appropriate server instance.
 3. The **Scheduled Alerts Retriever** screen is displayed. If you are creating a new server instance, enter the name of the Scheduled Alerts Retriever to be created in the **Name** text box.
 4. To determine if a process started late, the Start time margin has to be set in the Scheduled Alerts Retriever. For example, if a process is set run at 10:15 a.m., but actually runs at 11:30 a.m., and the Start time margin is set to 1 hour, it means that the process started late. Adding 1 hour to 10:15 a.m. gives the process a start time margin of up to 11:15 a.m.
-
- Note:** The start time margin is only used for those processes that log the process start/complete events. (See [Log process start/complete to audit log](#))
-
- **Start time margin period:** This is the number of time units that the process can delay in starting.
 - **Start time margin interval:** Defines the units - i.e. seconds, minutes, hours, days, weeks or months. For example, if the process can start up to 30 minutes late, enter **30** as the run time period and **Minutes** as the run time interval.
 5. Select the [Advanced](#) tab to specify a **Polling Interval** to define how often the system should check the repository for EMF Processes that are scheduled to run and to set other advanced options.

[The Server Instance Screen](#)

[Configuring System Options](#)

Server Manager

The **Server manager** servers read from a queue and control which worker threads are created or destroyed, and which EMF Processes they process.

Each server manager is assigned to read a particular queue, and you must have at least one server manager assigned to each physical queue in the queueing software. However, server managers can control multiple worker objects concurrently. Each Server manager must be uniquely identified.

Each time the server manager pulls an item from a queue it reads the worker type from the queue item, then creates a worker thread of that type and passes the EMF Process information to the worker thread. When the worker thread has completed, the server manager terminates it.

Note: A server manager contains no innate assumptions about its context in the EMF Server. It can read any valid queue, and can manage any type of valid worker object. It plays no part in the transformation of an EMF Process.

To create or modify a server manager server:

1. Select the machine on which the server should run from the **System > Machines** section of the EMF tree view.
2. Right-click the **Servers** icon, and then:
 - To create a new server instance, right-click in the list view and click **New > System > Server Manager**.
 - To modify an existing server instance, double-click the icon for the appropriate server instance.
3. The **Server Manager** screen is displayed. If you are creating a new server instance, enter the name of the Server Manager to be created in the **Name** text box.
4. Select the **Queue** that EMF Process objects are to be read from.
5. Select the required **Work in progress queue**, where the EMF Process objects are to be passed after processing.

Note: EMF Process objects cannot be returned to the queue from which they were taken - e.g. EMF Processes taken from the Primary Input Queue cannot be returned to the PIQ. After processing, they are placed on a different queue (e.g. the Work In Progress Queue).

6. Define how many threads should be used to complete tasks using the selected queue by entering the required number in the **Number of workers** field. The more threads you specify, the more memory is used.
7. Select the [Advanced](#) tab to specify advanced options.

[Configuring System Options](#)

[The Server Instance Screen](#)

SNMP Listener

Before you can run an EMF Process containing an [SNMP in](#) module you must configure an SNMP listener server.

To create or modify an SNMP Listener Server:

1. Select the machine on which the server should run from the **System > Machines** section of the EMF tree view.
2. Right-click the **Servers** icon, and then:
 - To create a new server instance, right-click in the list view and click **New > SNMP Listener**.
 - To modify an existing server instance, double-click the icon for the appropriate server instance.

3. Enter the name of the SNMP Listener to be created in the **Name** text box.
4. Select a system **EMF Process** that contains an SNMP router module (this is used to decide which EMF Processes should be fired when the server instance receives an incoming SNMP message).
5. **Select which events are receivable** by selecting the check boxes for the traps that you want the SNMP listener server instance to receive. Clear the check boxes for events you do not want the SNMP server instance to receive. If you do not select any SNMPv3 events, the **Security** tab will not be available.
6. Enter the **Port** number you want the SNMP listener server to use (the default is 162).
7. If you are using SNMPv1 or SNMPv2c security, enter the **Community** string. If you are using SNMPv3 security, leave this field empty and select the [Security](#) tab.
8. Select the [Advanced](#) tab to specify advanced options.

[SNMP Overview](#)

[Configuring the EMF System](#)

Statistics Log Manager

You can use the **Statistics Log Manager** server to define how often the EMF system [Statistics log](#) should be checked to see if it has exceeded the size or age specified in the Statistics logging [Size Management Screen](#). If it is older than the maximum specified age, or larger than the maximum specified size, the Statistics Log Manager will delete the oldest entries in order to make room for new ones.

The default setting is to check the statistics log once per minute, but you can change this if you wish.

To create or modify the Statistics Log Manager:

1. Select the machine on which the Statistics log manager server is running from the **System > Machines** section of the EMF tree view.
 2. Right-click the **Servers** and then
 - To create a new server instance, right-click in the list view, click **New > System**, and then select **Statistics Log Manager** to display the **Statistics Log Manager** screen.
-
- Note:** Typically, a machine should only ever have one **Statistics Log Manager** instance.
-
- To modify an existing server instance, double-click the icon for the appropriate server instance.
 3. Enter the name of the Statistics Log Manager to be created in the **Name** text box.
 4. Enter the required frequency in the **Check Frequency** field as a number of seconds. This is the only option that requires setting (the location of the log file is obtained from the Statistics logging component when it is polled by the server).
 5. Select the [Advanced](#) tab to specify advanced options.

[Adding Server Instances](#)

[Explanation of Statistics Logging](#)

[Configuring the EMF System](#)

Synchronized Alerts Retriever

The Synchronized Alerts Retriever is used to detect timeouts on the synchronization and cascade modules in an EMF Process. Timeouts are caused when an EMF Process branch is expected to return within a specified period, but for some reason doesn't (e.g. because an error occurred while processing the branch). The action that happens on the timeout depends on the option selected on the module.

The checking process is called "polling", and it can make quite heavy demands on your system's processors and databases. Normally the synchronizer polls every three seconds (that is, 3000 milliseconds) but you can increase this if you feel it is making excessive demands on your resources or lower it if you have a powerful system and/or small databases.

Note: By default, one **Synchronized Alerts Retriever** already exists in a new installation.

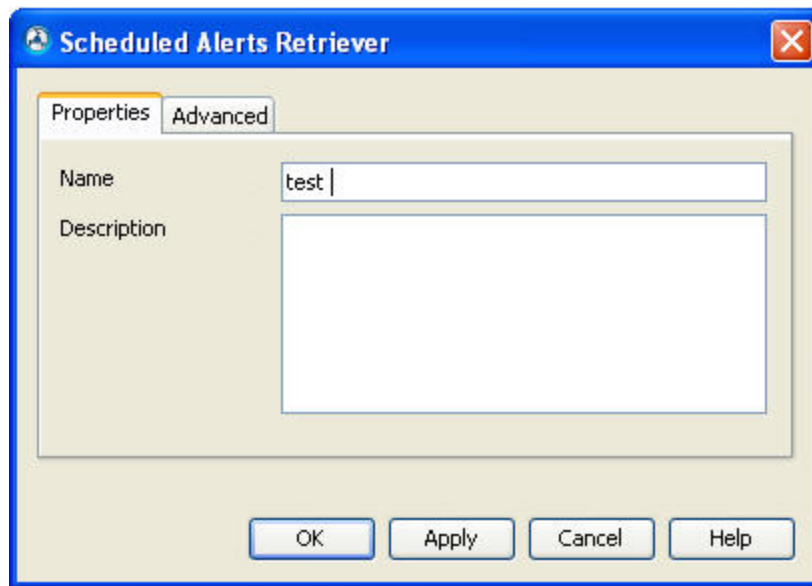
To create or modify a Synchronized Alerts Retriever:

1. Select the machine on which the server should run from the **System > Machines** section of the EMF tree view.
2. Double-click the **Servers** icon, and then:
 - To create a new server instance, right-click in the list view and click **New > System> Synchronized Alerts Retriever**.

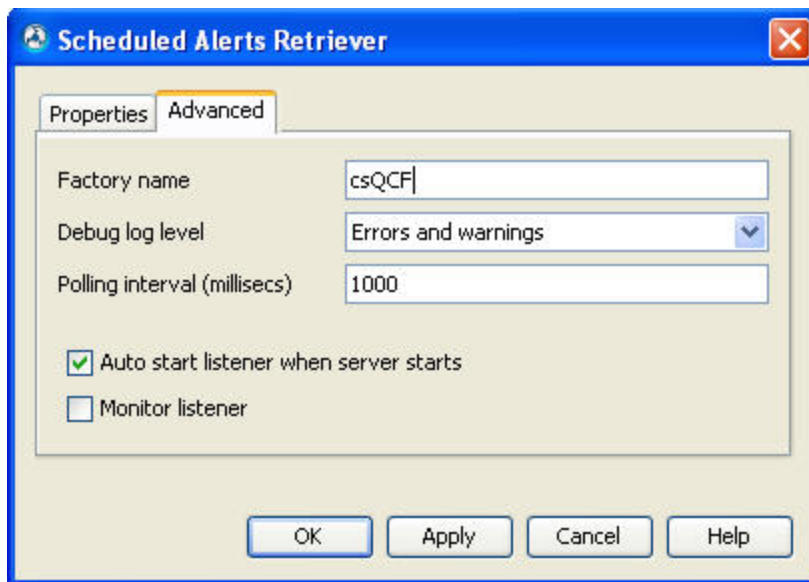
Note: Typically, a machine should only ever have one **Synchronized Alerts Retriever** instance.

- To modify an existing server instance, double-click the icon for the appropriate

server instance.



3. Enter a name and description.
4. Select the [Advanced](#) tab to specify a **Polling Interval** to define how often the Synchronized EMF Processes retriever should check for available information and to set other advanced options.



[Synchronization Module](#)

[Cascade Module](#)

Event/Event Plan Clean-up Manager

The **Event/Event Plan Clean-up Manager** is used to clean the events table in the repository. It has the ability to clean completed event plans and stored published events from the repository, through the configuration of the [Event Plan Monitor](#) and the [Published Event Store](#).

To create or modify an Event/Event Clean-up Manager

1. In the **System > Machines** section of the EMF tree view, select the machine on which the **Event/Event Clean-up Manager** server instance is running.
2. Right-click the **Servers** icon:
 - To create a new server instance,
 - Right-click in the list view and select **New > System > Event/Event Plan Clean-up Manager** to display the **Event/Event Plan Clean-up Manager** screen.
 - To modify an existing server instance,
 - Double-click the icon for the appropriate server instance.
3. In the **Name** field, enter the name of the **Event/Event Plan Clean-up Manager** to be created .
4. In the **Check Frequency** field, enter the value of the required frequency in seconds.

Note: This is the only field that requires setting. It is the interval, in seconds, at which the **Event/Event Plan Clean-up Manager** runs.

5. Select the [Advanced](#) tab to specify advanced options.

[Event Plan Monitor](#)

[Publish Event Store](#)

[The Server Instance Screen](#)

[Configuring System Options](#)

Running Event Plan Manager

The **Running Event Plan Manager** performs the following tasks:

- The manager notifies the [Expect Event](#) modules for any events that are overdue so that the overdue branches on the process will run.
- If **Number of events to wait for** *maximum* option is set to unlimited in the [Expect Event](#) module, the manager will resume those modules that have received enough events to meet the *minimum* specified requirement.

To create or modify a Running Event Plan Manager

1. In the **System > Machines** section of the EMF tree view, select the machine on which the **Running Event Plan Manager** server instance is running.

2. Right-click the **Servers** icon:
 - To create a new server instance,
 - Right-click in the list view and select **New > System > Running Event Plan Manager** to display the **Running Event Plan Manager** screen.
-
- Note:** Typically, a machine should only ever have one **Running Event Plan Manager** instance.
-
- To modify an existing server instance,
 - Double-click the icon for the appropriate server instance.
 3. In the **Name** field, enter the name of the **Running Event Plan Manager** to be created.
 4. In the **Check Frequency** field, enter the value of the required frequency in seconds.
-
- Note:** This is the only field that requires setting. It is the interval, in seconds, at which the **Running Event Plan Manager** runs.
-
5. Select the **Advanced** tab to specify advanced options.

[Configuring Server Instances](#)

[Configuring System Options](#)

Missed Events Manager

The **Missed Events Manager** processes the published events that have not been used by any [Event Plan](#). It will try to find any active [Event Plan Initiator](#) module or [Expect Event](#) module waiting for the unprocessed event in the event store.

Following are example scenarios where an event can be missed and must be picked up by the **Missed Events Manager**:

- A process is on hold when the initiating event is published. The initiating event will be in the event store, as **unprocessed**. If the process is then made **active**, the **Missed Events Manager** will initiate the process using the event from the event store.
- There is a small window of opportunity where an [Expect Event](#) module is processed at exactly the same time as the event it will wait for is published. In this instance, the published event does not see the expect event waiting for the event, and the expect event does not see the event in the published store as it is yet to be written. Therefore, the event will be missed. The **Missed Events Manager** will pick this up the next time it is run and ensure the event is correlated correctly to the [Expect Event](#) module.

The unprocessed events have an internal retry counter in order to avoid the **Missed Events Manager** processing the same events repetitively without success (as this is an intensive process). Once the retry count has reached its limit (default is three), then no further attempt is made to check that event. The retry counter may be manually reset through the [Published Event Store](#).

To create or modify the Missed Events Manager:

1. In the **System > Machines** section of the EMF tree view, select the machine on which the **Missed Events Manager** server instance is running.
2. Right-click the **Servers** icon:

- To create a new server instance,
 - Right-click in the list view and select **New > System > Missed Events Manager** to display the **Missed Events Manager** screen.

Note: Typically, a machine should only ever have one **Missed Events Manager** instance.

- To modify an existing server instance,
 - Double-click the icon for the appropriate server instance.
3. In the **Name** field, type the name of the **Missed Events Manager** to be created .
 4. In the **Check Frequency** field, type the value of the required frequency in seconds.

Note: This is the only field that requires setting. It is the interval, in seconds, at which the **Missed Events Manager** runs.

5. Select the **Advanced** tab to specify advanced options.

[Configuring Server Instances](#)

[Configuring System Options](#)

IMAP Email Retrieval Settings

You can use the **Email Retrieval** properties page of the IMAP Listener to specify advanced options on items such as the type of messages (read or unread) that will cause an EMF Process to be fired.

- **MailBox** - This defines the name of the mailbox to read messages from. A Mailbox is like a folder, and on some email servers can be hierarchical e.g. **Issues\Support**. Most email servers use the name **INBOX** as the mailbox new email first arrives in.
- **Messages to read** - This defines which messages should be read. This can either be **All** messages in the mailbox or **Unread Only**, which only processes previously unread messages in the mailbox. This setting should be used in conjunction with the **Action to perform on read messages** setting.

Note: Messages read by an external email client looking at the same mailbox/account as EMF will also mark the message as being read, and therefore may prevent EMF from processing the message if **Unread Only** is selected.

- **Action to perform on read messages** - This defines the action to perform on the message after it has been processed. **Delete** will remove the message from the email server. **Mark as read** will leave the message on the email server.

Note: **Messages to read** and **Action to perform on read messages** are typically configured in one of two combinations, **All** and **Delete** or, if the message history is required to be maintained on the email server **Unread Only** and **Mark as read**.

- **Advanced properties** - These allow advanced settings to be defined. The full list of settings is defined in the [Sun JavaMail](#) documentation. Generally a user will not need to change these. An example of setting a property would be to change the default timeout for the IMAP connection. To do this, add a property of **mail.imap.connectiontimeout** and set the value to be the required timeout period in milliseconds.

Note: To connect to a secure IMAP server (with SSL connection), add the following properties in the Advanced properties table:

Name	Value
mail.imap.socketFactory.class	javax.net.ssl.SSLSocketFactory
mail.imap.socketFactory.fallback	false

[IMAP Email Listener](#)

[Configuring Server Instances](#)

[Configuring the EMF System](#)

POP3 Email Retrieval Settings

You can use the **Email Retrieval** tab of the **POP3 Listener** dialog box to specify advanced options on items such as the type of messages (read or unread) that will cause an EMF Process to be fired.

- **Messages to read** - This defines that **All** messages in the mailbox should be read. This setting should be used in conjunction with the **Action to perform on read messages** setting.
- **Action to perform on read messages** - This defines the action to perform on the message after it has been processed. **Delete** will remove the message from the email server. **Leave** will leave the message on the email server.
- **Advanced properties** - These allow advanced settings to be defined. The full list of settings is defined in the [Sun JavaMail](#) documentation. Generally a user will not need to change these. An example of setting a property would be to change the default timeout for the POP3 connection. To do this, add a property of **mail.pop3.connectiontimeout** and set the value to be the required timeout period in milliseconds.

Note: To connect to a secure POP3 server (with SSL connection), add the following properties in the **Advanced properties** table:

Name	Value
mail.pop3.socketFactory.class	javax.net.ssl.SSLSocketFactory
mail.pop3.socketFactory.fallback	false

[POP3 Email Listener](#)

[Configuring Server Instances](#)

[Configuring the EMF System](#)

POP3/IMAP Email Listener Message Settings

You can use the **Message Properties** tab of the **POP3 Listener** or **IMAP Listener** dialog box to specify message settings such as the preferred message format to extract from an incoming message and whether or not to extract embedded content:

- If an incoming message contains both HTML and plain text versions of the message, you can specify which of these to extract (and insert in the POP3IN Message section) by selecting **Text** or **HTML**.

If you select HTML, you can choose not to extract embedded content (e.g. images) by deselecting **include embedded content**. This option is enabled by default for HTML messages, and is not available for plain-text messages as these cannot contain embedded content.

- If you do not want to extract the alternative message format type when an email does not contain your preferred format, select **Only use preferred message content**. This option is *not* enabled by default.
- If you want to extract plain text attachments from emails, select **Save attachments as data section**. This option is *not* enabled by default.

If you select this option, the attachments are saved into a system data section called POP3IN_ATTACHMENTS. This section has the following format:

Field name	Description
FileName	Name of the attachment
ContentType	"Text" or "Binary", depending on the content-type header of the attachment
Content	The contents of the attachment as a string. In the case of binary attachments this is Base64 encoded.

For more information on manipulating data section attachments, see the following topic:

[Attaching Files From a Data Section](#)

Scripting Examples (pop3attachments.js and base64decoding.js)

[POP3 Email Listener](#)

[IMAP Email Listener](#)

[Configuring Server Instances](#)

[Configuring the EMF System](#)

Properties of POP3IN_RD

When an Email in module has triggered the EMF Process, this data section can be processed by the EMF Process. It contains data from the [POP3 Listener Listener Server Instance](#). Various information can be collected using different properties, which are described below.

Property	Description
MsgSectName	Returns the value POP3IN as this is the name assigned to the message section of the email.
RecipientEmail	Returns the email address used by the POP3 Listener Server Instance.
RecipientName	Returns the user name used by the POP3 Listener Server Instance
ToCcBcc	Returns an integer value representing

POP3IN_RD should be entered as the 'DataSectionName' value and the properties should be entered as the 'FieldName' parameter values for whichever dynamic function is being used.

Properties of POP3IN_SD

When an Email In module has triggered the EMF Process, this data section can be processed by the EMF Process. It contains data from the email received by the [POP3 Listener Listener Server Instance](#) email account. Various information can be collected using different properties, which are described below.

Property	Description
MsgSectName	Returns the value POP3IN as this is the name assigned to the message section of the email.
From	Returns the user name (if available) of the user that sent the email.
Subject	Returns the content of the subject field of the email
EmailAddress	Returns the email address of the user who sent the email.
ReplyToName	Returns the user name to be used if a reply is sent.
ReplyToEmail	Returns the email address to be used if a reply is sent.

POP3IN_SD should be entered as the 'DataSectionName' value and the properties should be entered as the 'FieldName' parameter values for whichever dynamic function is being used.

HTTP Advanced Data tab

You can use the **Advanced data** tab of the HTTP Listener Properties pages to specify how incoming requests should be parsed. The resulting constituent parts of the request are then stored in the EMF Process object. Incoming requests can be stored as they are in a single message section, or they can be parsed and stored as separate data/message sections, or both.

Note: The message sections that are created are automatically given names by EMF, and you cannot change these.

1. If you want to store the entire contents of the message in a single message section that will be automatically created (and named **HTTPIN_STREAM**), select **Store whole request stream**.
2. Alternatively, if you want to parse the message and store the different constituent parts in separate sections, select the required parts in the **Parse request** box:
 - Select **Message body** to store the main body of the message (if any) in a message section that will be automatically created and named **HTTPIN_MSG**
 - Select **Headers** to store the headers that are sent with the request in a data section that will be automatically created and named **HTTPIN_HEADERS**
 - Select **Parameters** to store any parameters that are sent with the request in a data section that will be automatically created and named **HTTPIN_PARAMS**

[HTTP Listener Server Properties](#)

[The SSL Settings tab](#)

[The Request Handling tab](#)

[Adding Server Instances](#)

Security Tab (SNMP Listener Server Instance)

If you wish to use SNMPv3 security features for SNMPv3 Traps and Informs you must enter the relevant details for using SNMPv3 security in the **Security** tab of the SNMP Listener server instance Properties pages.

Note: If you are using SNMPv1 or SNMPv2 security, you do not need to use this tab and should enter the relevant string on the [Properties](#) page.

The screenshot shows the 'SNMP listener' window with the 'Security' tab selected. The 'Security level' dropdown is set to 'None'. The 'Authentication protocol' dropdown is also set to 'None'. The 'User name' field is empty. The 'Authentication password' and 'Private password' fields are masked with '*****'. The 'Context name' and 'EngineID' fields are empty. The 'Close' and 'Help' buttons are at the bottom right.

Fill in the details as follows:

- **Security level:** Select the SNMPv3 security level from this drop-down list. The available options are **NoAuthNoPriv** (no authentication, no privacy), **AuthNoPriv** (authentication, no privacy) and **AuthPriv** (both authentication and privacy). AuthPriv offers the highest level of security.
- **Authentication protocol:** If you selected AuthNoPriv or AuthPriv as the security level, select **SHA-1** or **MD5**.
- **User name:** Type the user in whose name the requests or traps are being sent. The user must exist on both the source and destination SNMP network elements. In addition, the user's security settings on both source and destination elements must match.
- **Authentication password:** If you selected an authentication protocol, type the authentication password for the user here.
- **Private password:** If you selected AuthPriv as the security level, type the encryption password here.
- **Context name:** Type the SNMP context name here. This is an octet string that together with the EngineID uniquely identifies a particular SNMP context, or set of management information. Management information can exist in more than one context, and an SNMP entity can access multiple contexts.
- **EngineID:** Enter the ID of the authoritative SNMP engine for Trap and Inform requests here. This is usually the SNMP agent. If this is left blank, the EMF server will retrieve the EngineID automatically or generate its own.

[SNMP Listener Server](#)

[Configuring the EMF System](#)

System EMF Processes

System EMF Processes (stored in the **System** EMF Processes folder in the EMF Processes tree view) are used internally by the EMF system for error handling, the routing of incoming messages (e.g. via SNMP, POP3, HTTP, and Queue), and similar functions. A default set of system EMF Processes is created automatically when you install EMF.

Important: EMF relies on the system EMF Processes in order to function correctly, and you should not attempt to modify or change the status of the system EMF Processes.

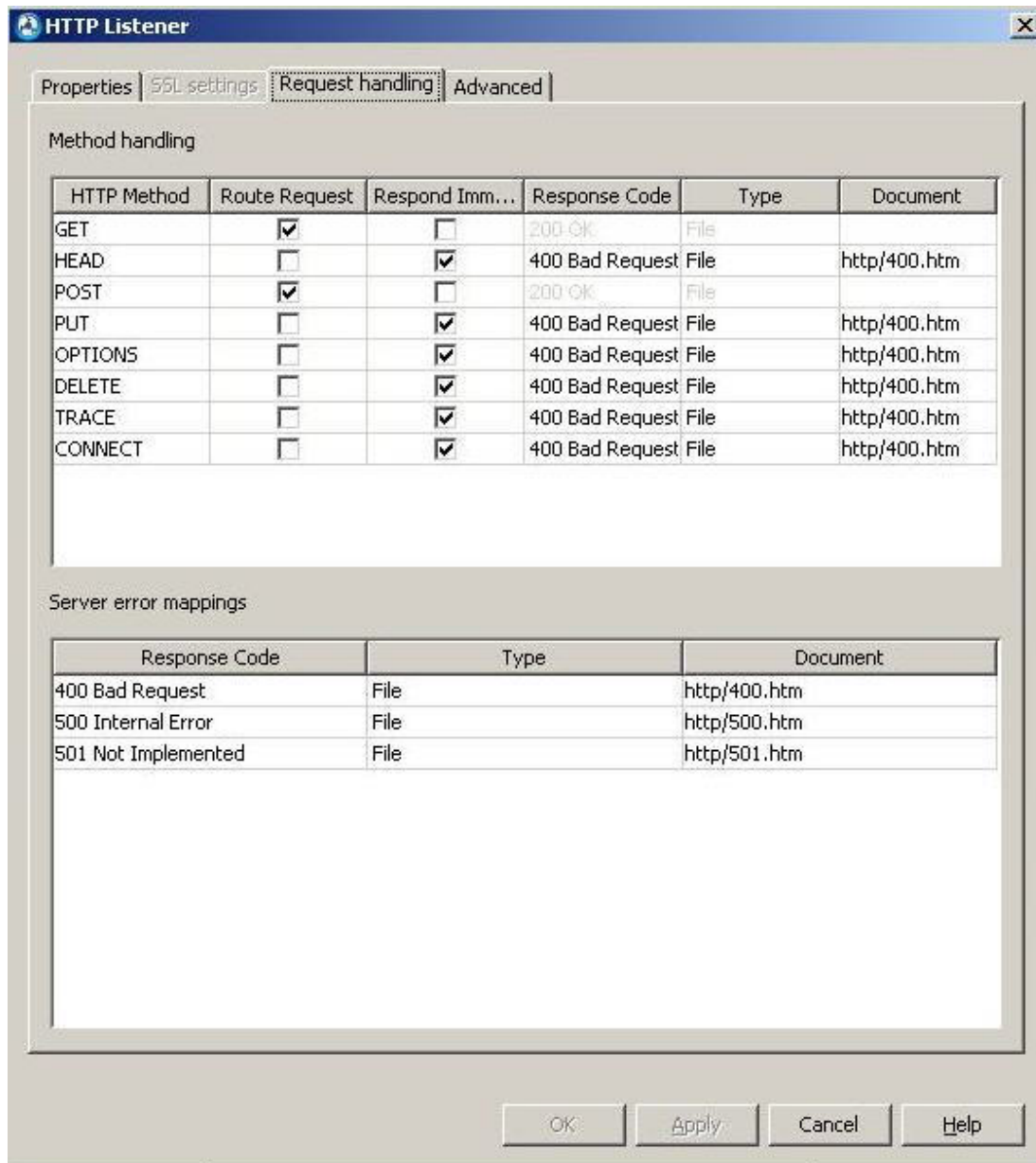
Note: you should avoid storing your own EMF Processes in the System folder, unless you need to create your own system EMF Processes.

[Fundamental EMF Concepts](#)

The HTTP Request Handling tab

You can use the **Request handling** tab of the [HTTP Listener](#) dialog box to determine how to handle HTTP requests that use each HTTP method. This includes determining whether to:

- Route a request through the appropriate EMF process, and
- Respond immediately to a request and configure the corresponding response.



The **Request handling** tab enables you to perform the following:

- Handle HTTP requests
- Configure server error mappings

Handling HTTP Requests

The HTTP listener supports eight HTTP methods (verbs) - **GET, HEAD, POST, PUT, OPTIONS, DELETE, TRACE** and **CONNECT**. The Method handling area enables you to choose to respond differently to each method.

The Method handling area contains the following fields:

- **HTTP Method** - Indicates the method.
- **Route Request** - Select the check box corresponding to an HTTP method to indicate that this request should be routed to the EMF process selected in the **Properties** tab.
- **Respond Immediately** - Select the check box to determine whether the EMF server should send a response immediately to an HTTP request. If you select this along with the corresponding **Route Request** check box, then the HTTP module should not be used to send a client response from within the process that the request is routed to (as this would result in two responses being sent for one request, which is invalid).

For example, if you have created a system to perform batch processing of data being sent through an HTTP Post method request continuously throughout the day, you can configure the HTTP Listener to accept the data and send back an HTTP 202 (Accepted) code. The code would contain informational content and stores the payload into a message or data section to be processed later.

- **Response Code** - Select an appropriate response code from the drop-down list if the corresponding **Respond Immediately** check box is selected.
- **Type** - Select the type of response to which to map the method. The response can be mapped to a file or an URI (Uniform Resource Indicator).
- **Document** - Type the file name and location or the URI containing the response to be sent.

Configuring Server Error Mappings

Server error mappings allow you to configure responses to be sent when the EMF server encounters errors.

To modify a default Server Error mapping:

The Server error mappings area displays the default HTTP response code mappings to files or URIs as follows:

Response Code	Type	Document
400 Bad Request	File	http/400.htm
500 Internal Error	File	http/500.htm
501 Not Implemented	File	http/501.htm

You can only modify the Type and Document field settings for default mappings.

To modify a default server error mapping, do the following:

1. In the Type field corresponding to a response code, click on the type and select File or URI from the drop-down list.
2. Enter a valid URI or a file location and name in the Document field, based on the response code type.

Examples:

- **File** - http/400.htm
- **URI** - http://www.mycompany.com/ErrorPage.htm

3. Click **Apply** or **OK** to save the settings.

The response code will be mapped to the specified file or URI.

[HTTP Listener](#)

[The SSL Settings tab](#)

[The Request Data tab](#)

[Adding Server Instances](#)

[Configuring the EMF System](#)

HTTP SSL Settings tab

You can use the **SSL Settings** tab of the [HTTP Listener Server Properties](#) pages to configure the Java classes that should be used when communicating over Secure Sockets Layers, and specify the EMF server certificate to use in the handshake.

Note: the SSL settings tab is not available if you selected **Unsecure** as the **Socket Type** on the **Properties** page.

To specify SSL settings:

1. Enter the name of the **Provider class** that implements the SSL provider interface.
2. Enter the name of the **HTTPS class** that implements an http or https URL stream handler interface.
3. If you would like to validate any certificates sent by clients, select the **Validate client certificates** option. You must then select the appropriate **Listener certificate** to use, from the list of those that you have defined in the [Key store](#). The selected certificate will be used when handshaking with a client over a secure socket.

If you do not enable the **Validate client certificates** option, any certificates are ignored.

[HTTP Listener Server](#)

[The Request Handling tab](#)

[The Request Data tab](#)

[Adding Server Instances](#)

[Configuring the EMF System](#)

Running EMF on Multiple Machines

This topic outlines the steps you need to follow to run the EMF Server on multiple machines. You may want to do this for scaling or load balancing reasons, or to introduce more redundancy in your EMF system.

To configure multiple machines:

1. After installing the EMF Administrator and EMF Server on the required machines, use the [EMF Processes Repository Wizard](#) in the EMF Administrator to create your configuration and repository databases.
2. From the machine where you ran the Configuration Wizard, check that the **<EMF Installation Directory>\Server\<Repository Name>.properties** file specifies the correct details for the database server machine. For example, if it uses "localhost", you will need to change this to the IP address or machine name.
3. Now copy this file to the **Server** directory on all your EMF Server machines.
4. If you are not using the default repository, amend **EMF.properties** on all the EMF Server machines to add the name of the repository in the following format:
RepositoryName=<repository name>.
5. In the EMF Administrator, if you are using the internal queue provider, change the **Provider URL (System > Queue providers node)** so it refers to an IP address or machine name, rather than "localhost".
6. In the EMF Administrator, add the new machines by right-clicking the **System > Machines** node and selecting **New machine**. Then enter the details as described in [Managing the Machines That EMF Runs On](#).

Note: Only select **Auto-start** and **Auto-stop queue provider** if the queuing system is on the same machine as the EMF Server. If you are using a queuing system other than the internal queue provider, you will need to write your own startup scripts before you enable the **Auto-start** and **Auto-stop** options.

7. After adding the EMF Server machines, you must configure at least a PIQ server manager and WIP server manager for each new machine. To do this, right-click the machine name in **System > Machines** and select **New server**. Then enter the details as described in [Server Manager](#).

To start multiple machines:

1. Start your queue provider, or start a EMF Server that has been configured to autostart the queue provider.
2. Start the rest of your EMF Servers.

[Explanation of Server Instances](#)

[Configuring System Options](#)

[Configuring the EMF System](#)

Scaling EMF

In a development system it is usually adequate to install EMF on a single machine, or across two machines (EMF Administrator and EMF Server). However, in a production system where you may be running thousands of EMF Processes, or particularly complex EMF Processes, you may find it necessary to configure a distributed system to improve performance. This topic provides general guidelines for configuring a distributed EMF system.

EMF scales in the following ways:

- Multithreading
- Concurrent and asynchronous execution
- Queueing

Important: Because EMF makes extensive use of queueing, for the system as a whole to scale, maintain high availability and load balance work, your chosen queueing technology must also address these issues. You should consult your queueing system's documentation to see how the technology can be scaled.

Note: If you have scaled your queueing technology by configuring additional instances of the queue server, you can specify a different queue factory name in the [Server manager screen](#). This allows EMF to obtain the connection details to a different instance of the queue server. To do this, the additional queue server instances and queue connection factories must be configured in your queueing system. The queue servers must be within the same cluster, and must be referencing the same queues.

Before you start: Investigate the current bottlenecks in your system - is it the queueing system, the EMF Server, processing power, the database server, or your output delivery methods (for example, email server)? You should try to avoid bottlenecks outside of EMF.

If you have determined that the bottleneck is EMF, use the following techniques to help you scale your system:

- Increase throughput by increasing the number of worker threads that a server manager can allocate. [Server managers](#) take work from their queue whenever they have a thread available. The more threads you have, generally the better the throughput on the queue.
- Install the EMF Server on [multiple machines](#) and configure server managers to monitor existing queues - this provides true concurrency, load balancing and failover. For example, for each machine you must configure at least a PIQ manager and WIPQ manager. This means that not only do you have the thread pools of multiple server managers at your disposal, but if one machine goes down, work on the queues can still be processed by server managers on other machines. Load balancing is achieved because each server manager takes work from the queue it is monitoring whenever it has a thread available.
- Create EMF Processes that maximize use of the EMF queues by:
 - targeting specific queues in your [output services](#) (by default, output services use the existing csPIQ and csWIPQ). For example, if you have a large number of email EMF Processes, you may want to use a dedicated SMTP service email queue, or even set up a number of SMTP services, each targeting a different dedicated queue.
 - using the **Split recipients** option in your [Recipients module](#). This creates and queues a new EMF Process object for each group of recipients. In other words, for each group of recipients, a new worker thread will be allocated by the server manager.
 - using branches in your EMF Processes. Each time an EMF Process is branched, an EMF Process object is created for each branch and placed on the queue indicated

by the first module in the branch (usually csWIPQ). When on this queue it is picked up by a new worker thread.

- using the [Cascade module](#) to increase concurrency. For example, if you have an EMF Process with a data section containing a large number of SQL INSERT statements, instead of making one worker thread perform all the INSERTS, you could drop in a Cascade module that cascades to a slave EMF Process per row. For each row in the data section, the slave EMF Process object will be placed on the primary input queue, thus making full use of the worker thread pool.

Note: in this example, EMF will be performing concurrent INSERTs in the database server. If there are many rows, this could mean that your database server becomes a bottleneck. You should consult your Database Administrator before attempting multiple concurrent database requests.

[Configuring the EMF System](#)

Advanced Tab

You can use the **Advanced** tab of the **<server instance> Listener** dialog box to specify advanced options.

Depending on the server instance that you are configuring, you can set some or all of the following options:

1. If you want to use a connection factory, enter the name of the connection factory in the **Factory name** field.

Note: EMF uses the factory name to look up the connection factory details in the queueing system's bindings registry. It then uses the connection factory to create a connection to the queue server. So, specifying a different factory name in the **Server** screen allows EMF to connect to a different queue server. This is useful if you are scaling your queueing technology and have configured additional instances of your queue server. Note that all the queueing servers must reference the same queues and must be in the same cluster.

2. Select a [Debug log level](#) to specify the amount of detail that you want this server instance to save to a log.
3. Specify polling periods in the **Polling interval (millisecs)** field (e.g. entering 1000 would check once every second).
4. Select the **Auto start listener when server starts** check box if you want the server instance to start automatically when the EMF Server starts on the specified machine.
5. Select the **Monitor listener** check box if you want the monitor to detect when the listener has failed and should it fail, to attempt to automatically restart it.

Following are the additional settings for server instances:

- **HTTP Listener** - You can use the **Advanced** tab of the **HTTP Listener** dialog box to specify how incoming requests should be parsed. For details, see [HTTP Advanced Data](#)

[tab.](#)

- **Queue Listener** - Queue messages can be in the Java Message Service (JMS) Text or Map Message format. The message, header, and data sections that the Queue Router passes to the EMF Process are displayed in the **Message section**, **Header section**, and **Data section** fields of the **Advanced** tab of the **Queue Listener** dialog box. For details, see [Queue Router Module](#).
- **SNMP Router** - The data sections that the SNMP Router passes to an EMF Process are displayed in the **Data section** and **OID section** fields of the **Advanced** tab of the **SNMP Listener** dialog box. For details, see [SNMP In Data Section Fields](#).

[Configuring Server Instances](#)

SNMP In Data Section Fields

The SNMP Router passes two data sections to an EMF Process: SNMP_DATA and SNMP_OIDS. The following tables give the name and description of the fields for these data sections.

SNMP_DATA

Field Name	Description
Type	The type of request. Allowable values are Trap and Inform.
Version	The version of SNMP that is being used (1, 2c or 3)
SourceAddress	The IP address of the message source
Community	The community string (SNMPv1 and SNMPv2c only)
Enterprise	The enterprise object identifier (in dot notation)
TrapType	This is the trap type for a SNMPv1 trap. Options are Cold start, Warm start, Link down, Link up, Authentication failure, egp Neighbor loss and Enterprise specific. For more information on these trap types, see Generic Trap Types .
SpecificCode	If the TrapType is Enterprise specific, then this is a trap code defined within the enterprise.
TrapOID	The object identifier that identifies the trap in SNMPv2c or SNMPv3.
Uptime	How long the sender has been "alive".

SNMP_OIDS

Field Name (for each object)	Description
ObjectID	The object identifier (written in dot notation)
Type	The object type. For more information on object types, see Object Data Types .
Value	The object value

Note: There may be multiple objects in the SNMP_OIDS data section.

[SNMP Router Icon](#)

[Back to Start of Initiator Modules](#)

Configuring the EMF Queues

EMF requires a queuing system to manage the connections between the different parts in the system, and to ensure guaranteed delivery of EMF Processes. EMF has a build in embedded queue provider which is recommended for general use.

Important: These internal system queues are not the same as the queues used by the [Queue Listener](#) and [Queue Output](#) modules in the EMF Process Builder (although you can use the same provider, if you wish). To configure an external queue provider and external queue for use with these modules, see [Queuing Services](#).

Note: The communication between EMF and the queuing system is handled by the Java Message Service (JMS). For further information on JMS see [Messaging and the Java Message Service](#) or consult the JMS home page at <http://java.sun.com/products/jms/>.

Selecting a queue provider

The choice of queuing technology that you use is dependent on your business requirements (EMF supports the internal queue provider it comes bundled with). In order to work with EMF, the queuing system must be JMS Compliant, must use point-to-point communications, and must run under Sun Java Virtual Machine (JVM).

Configuring your queuing system

Your queuing system should be configured prior to the installation of EMF.

For specific configuration instructions for other queuing systems, please refer to the documentation that is provided with your system.

Note: If you wish, you can have your queue provider start up automatically with the server. See [Managing the Machines that EMF Runs On](#).

Changing your queue provider

If you need to change details of your queue provider, or wish to use a different provider, you can do so using the [Queue Provider screen](#) which is available by selecting **Queue providers** from the **System** section of the EMF tree view.

[Creating Queues](#)

[List of EMF Queues](#)

[Configuring the EMF System](#)

Configuring Your Queue Provider

You can use the **Queue provider screen** to identify the queuing technology that you are using, to tell the EMF server where to locate the queuing system configuration, and how to connect to it.

To configure your queue provider:

1. Select **Queue providers** from the **System** section of the EMF tree view.
2. Double-click the icon for the queue provider.

Note: You can only define a single queue provider. If you change your queue provider, you must modify the details in this screen to reflect the new details.

3. Enter the relevant details as follows.
 - **Initial Context Factory:** the Java Initial Factory Content for JNDI lookup. For the internal queue provider (ActiveMQ) it should be **org.apache.activemq.jndi.ActiveMQInitialContextFactory**.
 - **Provider URL:** the URL for lookup Queue Provider details. For the internal queue provider (ActiveMQ) it should be **failover://(tcp://localhost:61616)?timeout = 10000&maxReconnectDelay= 20000&jms.prefetchPolicy.queuePrefetch = 0**

Important: The above example assumes that you are running the queues on the same physical machine as the server. If you are running the queues and the server on different machines, you must change the **Provider URL** to reference the IP address of the queue provider machine (the default setting, **localhost**, will only look on the local machine).

 - **Factory Name:** The name of the Queue factory. For the internal queue provider (ActiveMQ), it should be **csQCF**.
 - **User Name:** the username to use when logging in to the queue provider.
 - **Password:** the password to be used when logging in to the queue provider.
 - **Dead Letter Queue (DLQ):** To change the default time that a failed message will stay on the dead letter queue before being deleted, set a time in seconds. Setting a time of zero will mean messages never expire, and will stay on the queue until removed by some other means. Leaving messages on the DLQ for longer than you require can cause the queues to fill and stop the message broker from releasing large blocks of message storage. If you are not worried about seeing messages on the DLQ, then set the time to 1 second. Individual processes can override this setting and have a specific time set for them (see [Changing General Settings for an EMF Process](#)).

Important: If you modify any of the settings for an existing queue, you must shut down and restart the EMF Server in order for your changes to take effect.

If you are not using the internal queue provider (ActiveMQ), you should refer to your message queue documentation for more information.

Note: The communication between EMF and the queuing system is handled by the Java Message Service (JMS). For more information on JMS, see [Messaging and the Java Message Service](#) or consult the JMS home page at <http://java.sun.com/products/jms/>.

[Creating Queues](#)

[Configuring System Options](#)

Creating Queues

The number of queues that you configure for your system will depend largely on your business needs. Performance is directly proportional to the number of physical queues that are in the system, therefore, as the volume of EMF Processes increases, you must also increase the number of queues.

It is recommended that you configure a minimum of four physical queues: a Primary Input Queue, a Work in Progress Queue, a System Queue, and an additional queue where you can map your other logical queues for the remaining services. To scale up the EMF system, additional physical queues will be required and you may need to move towards a one-to-one mapping of your logical queues to physical queues.

To view and create your logical queues:

Click the **Queues** icon in the **System** folder of the EMF tree view.

The existing queues are displayed in the right-hand pane - see this [list of EMF queues](#).

Note: For assistance calculating your queue requirements, please contact EMF.Support@aptean.com

To add a new logical queue:

1. Right-click in the right-hand pane of the **Queues** view and select **New queue** to display the **Queue** screen.
2. Enter an appropriate **Name** for the queue.
3. Enter the **Lookup Name** in the appropriate field.
4. If you wish, enter a **Description** to be reminded of its purpose.
5. Select **Exclusive Queue** to run the queue in [Exclusive](#) mode.

[Configuring the EMF Queues](#)

[List of EMF Queues](#)

[Configuring System Options](#)

[Exclusive Queue - Single Thread Operation/Message Order Dependency](#)

List of EMF Queues

To view your logical queues: Click the **Queues** icon in the **System** folder of the EMF tree view. The existing queues are displayed in the right-hand pane.

The following logical queues are mandatory and must be present:

- **Primary input queue (PIQ).** This is the queue on which an EMF Process is first placed when it is initiated. EMF Processes which have not yet been released are removed from this queue by its server manager and put into processing. The only information contained in an EMF Process on the PIQ is the start module, which is responsible for populating the EMF Process object with the list of modules to run, links between the modules and any data object which has been passed to the EMF Process from initiating objects.
- **Work in progress queue (WIP).** EMF Processes that are waiting to be processed are placed on this queue, while waiting for a **Server Manager** instance to allocate a thread for them to run on. Items on the WIP are run on any **Server Manager** instance as all modules are constructed to have a common interface.
- **Dead-letter queue (DLQ).** This is used for EMF Processes that have failed during processing. All items which fail during processing are moved to this queue after they fail or after any specified retries fail.
- **Debug queue.** This is used in debug mode. EMF Processes on this queue are executed in single step mode from the administrator.
- **HTTP Response queue (HTTPQ).** This contains serialized Response objects, created by the [HTTP module](#) (acting in response mode), which contain the content to send back to the client together with the key for the connection of the requesting client.

In addition to the above queues, you may wish to use the following queues:

- **Email queue (EMAIL).** This is a dedicated queue for email output that you can use if required.
- **Command queue, Response queue and System queue.** These queues are not currently used by the EMF system and you can use them for own purposes if required.

Note: all output services can be configured to use their own queue (by default they use the **WIP** queue). Specifying a queue other than WIP for an output service may improve performance if there is a bottleneck in the EMF output.

To create a new queue

Right-click in the list of queues and select **New queue** to display the [Queue screen](#).

Important: any queue you create in the EMF Administrator must also be configured in your queueing system. In addition, you should also set up at least one [Server Manager](#) to monitor the new queue. The exception to this rule is the HTTP Response Queue which is used exclusively by HTTP listeners.

Note: the communication between EMF and the queueing system is handled by the Java Message Service (JMS). For further information on JMS see [Messaging and the Java Message Service](#) or consult the JMS home page at <http://java.sun.com/products/jms/>.

[Creating Queues](#)

[Configuring the EMF Queues](#)

[Configuring the EMF System](#)

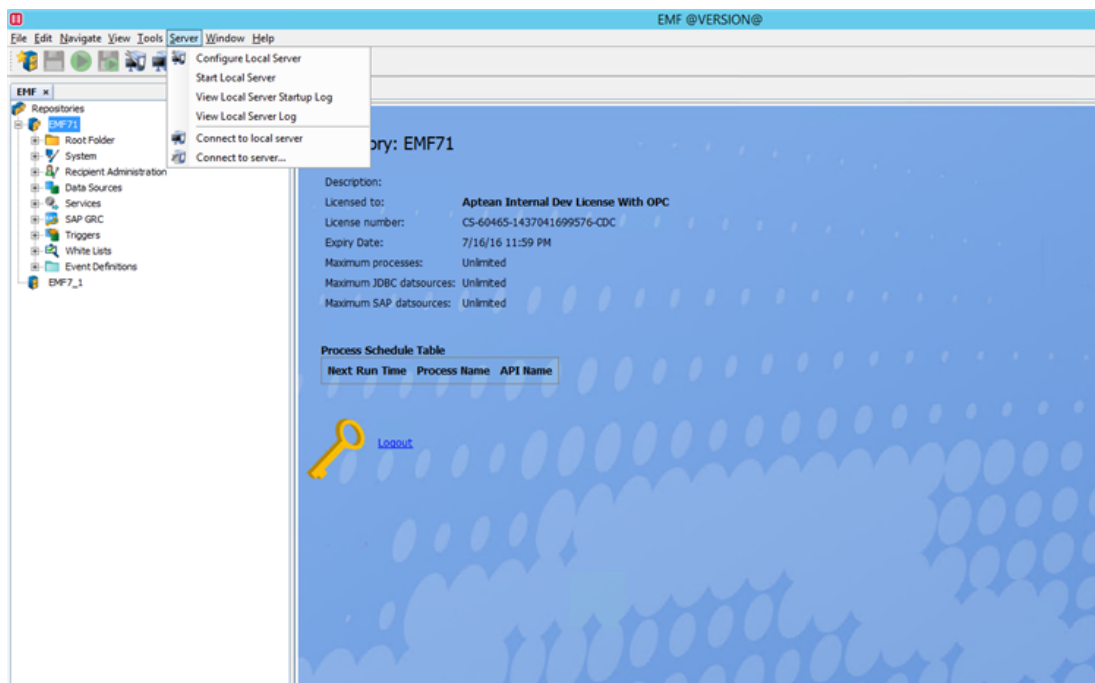
Configuring the EMF Server

The following topics contain information intended to assist you in performing the initial start up of the EMF server:

- [Configure the settings that the EMF Server uses to start up](#)
- [Configure the database](#)
- [Define Server logging settings](#)
- [Define the RMI port connections](#)
- [Define Java JAR libraries](#)
- [Manually Change Startup Settings.](#)

Configuring the Local Server

To configure the settings that the EMF Server uses to start up, on the **Server** menu, click **Configure Local Server**.

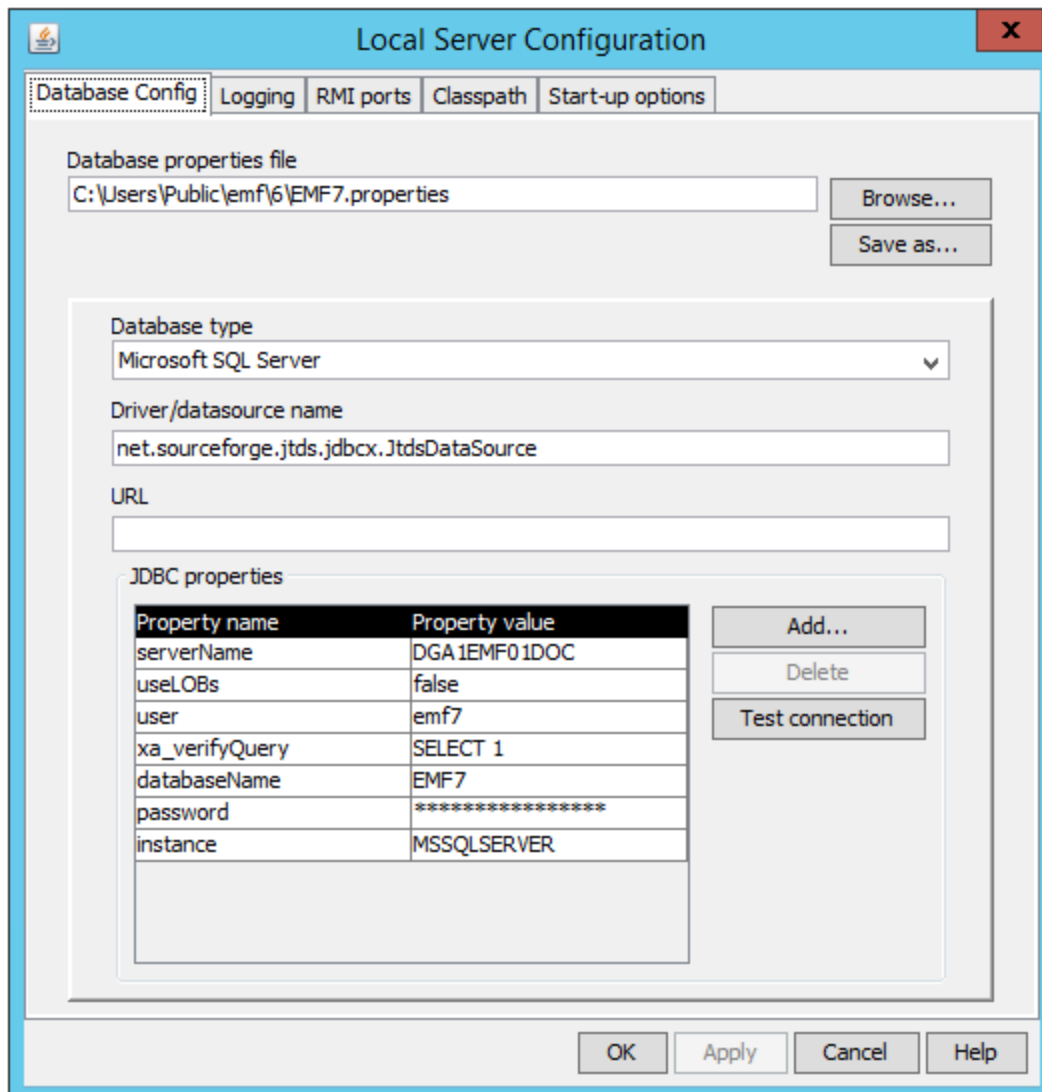


This option defines the following:

- [Database connection to the repository](#)
- [Server logging settings](#)
- [RMI port connections](#) needed for the EMF administrator to connect to the EMF server
- The [Java JAR libraries](#) to use to initialize the server
- [Start-up options](#)

Note: The configuration details for the EMF server are stored in a file named `EMF.properties` in the `<All Users>/Application Data>/emf/6/server` directory for Windows. If preferred, you can edit this file manually using a text editor. For more information, see [Manually Changing Startup Settings](#).

The server configuration should be correctly configured automatically when you create a new repository.



The settings can be adjusted using the following tabs:

- [Database config](#)
- [Logging](#)
- [RMI ports](#)
- [Classpath](#)
- [Start-up options](#)

Configuring The Database

The screenshot shows the 'Local Server Configuration' dialog box with the 'Database Config' tab selected. The 'Database properties file' is set to 'C:\Users\Public\emf\6\EMF7.properties'. The 'Database type' is 'Microsoft SQL Server'. The 'Driver/datasource name' is 'net.sourceforge.jtds.jdbcx.JtdsDataSource'. The 'URL' field is empty. The 'JDBC properties' table lists several properties: serverName (DGA1EMF01DOC), useLOBs (false), user (emf7), xa_verifyQuery (SELECT 1), dbName (EMF7), password (*****), and instance (MSSQLSERVER). Buttons for 'Add...', 'Delete', and 'Test connection' are visible next to the table. At the bottom are 'OK', 'Apply', 'Cancel', and 'Help' buttons.

Property name	Property value
serverName	DGA1EMF01DOC
useLOBs	false
user	emf7
xa_verifyQuery	SELECT 1
dbName	EMF7
password	*****
instance	MSSQLSERVER

1. Enter the required information in the **Database Config** tab to define the connection details to the repository database as follows:
 - Enter the name and path of the properties file containing the database connection details in the **Database properties file** box. The connection details will be displayed in the panel below.

You can click **Browse** to search the file system for the required file, and use **Save as** to save the configuration to a new file name.

Note: **Browse** will be enabled only if you are running the configuration program on the same server machine. If you are connecting to a remote server, it will be disabled.

2. Select the database type of the configuration database from the **Database type** drop-down list (EMF supports Oracle and Microsoft SQL Server). See [here](#) for example settings.
3. Enter the name of the JDBC driver or data source to be used to access the configuration database in the **Driver/datasource name** box.
4. Enter the URL to be passed to the JDBC driver to identify the required database, in the **URL** box (not required for a data source). For further information, refer to the documentation supplied with the driver.
5. Optionally, enter additional properties in the **JDBC properties** table; for example, user, password, and the name of the configuration database (as displayed in the Connection String Builder).
 - To add a property, click **Add** to display the **Add Property** dialog box.
 - To delete a property, select the property and then press **Delete**.
 - To modify a property, double-click the property name or value and then modify it.

The following additional (optional) properties are applicable:

- **xa_minPoolSize:** The number of connections that will be initially created in the pool, and held in the pool at all times. The default value is -1, which indicates an unlimited number.
- **xa_maxPoolSize:** The maximum number of connections that the pool will give out. The default value is -1, which indicates an unlimited number. If this value is set to 0, attempts to gain further connections after the minimum number of connections have been retrieved will fail. If this value is set to any number greater than 1, when the maximum number of connections has been retrieved, the pool will wait for a connection to become available. If no connection becomes available within the time specified by xa_maxWait (see below) the request will fail.
- **xa_maxIdleTime:** The maximum amount of time (in seconds) that a connection can remain in the pool unused. Any connections that are still in the pool and unused after this time will be released. The default value is 600 (10 minutes).
- **xa_propertyCycle:** Determines how frequently (in seconds) to check for connections in the pool that have been idle for the time specified by xa_maxIdleTime and should be removed. The default value is 600 (10 minutes).
- **xa_oracleFailoverCheck:** Allows you to specify whether a check should be performed on every Oracle connection to verify whether the connection is still valid. Enabling this option allows the EMF server to recover in an event in which the network or database fails, but will have a detrimental effect on performance. The default setting for this property is **True**.
- **xa_disableInternalPooling:** Prevents the server from using its own internal pooling. This allows you to use a third-party data source that has its own pooling (e.g.

oracle.jdbc.pool.OracleConnectionCacheImpl). The default setting for this property is False.

- **xa_maxWait:** Determines the amount of time (in seconds) to wait for a connection to become available (see xa_maxPoolSize). A setting of 0 or less indicates an indefinite wait, as long as xa_maxPoolSize is set to greater than 0. The default value is 0.
- **xa_ignorePooledDatasourceInterface** (only applicable if you are using the Microsoft JDBC driver for SQL Server 2000): This property should be set to True in order to force the EMF connection pool to use the appropriate method of implementing the Microsoft driver.

Click **Test connection** to attempt a connection to the database using the specified configuration details, and display the **Test database** connection screen. This screen reports the success (or failure) of the various stages of connecting to the database, and displays a warning if the database was not recognized as a configuration database.

Defining Logging Features

Define logging features of the server using the **Logging** tab as follows:

- **Backup log file** - Any log messages that cannot be logged to the repository database will be logged to this file. The system may not be able to log to the database file because a connection has not yet been established, or because of a connection failure. You can click **Browse** to display a file chooser dialog box allowing you to search for the required file (**Browse** is only available if the configuration program is run as a standalone application on the server machine, or if the user interface was selected to start when the server starts).
- **Log file** - Runtime exceptions that occur are logged to this file. This can aid in problem resolution. You can click **Browse** to display a file chooser dialog box allowing you to search for the required file (**Browse** is only available if the configuration program is run as a standalone application on the server machine, or if the user interface was selected to start when the server starts). See the **Log to file** option below for the type of information that is logged.
- **Log to file** - Refers to what type of information is logged to the Log file described above. The permitted values are:
 - **Exception stack traces only** - This logs only the stack traces (prior to EMF 6.1, this was the only option).
 - **All logged messages** - Any other messages that the server generates (DEBUG/WARNINGS/ERRORS) will also be logged to this file. This option should be used with caution as it will slow performance of the server and can lead to very large log files.
- **Trace level** - This refers to the amount of messages that are displayed to a console or User interface attached to the server. This can aid in identifying problems. Tracing more messages will have a detrimental effect on the performance of the server. The permitted values are:

- **None** - No messages are traced.
- **Debug log level** - The log level of the server (for system level messages) or process (for process execution messages) as defined in the EMF administrator determines the amount of information that is output.
- **All** - All trace messages will be displayed. Note that this setting could result in a very large number of messages being output.

Configuring RMI Ports

The **RMI (Remote Method Invocation) ports** tab allows you to define the rmi port configuration to be used by the server.

If the server is to be restricted on the ports that it will use to communicate with the EMF Administration UI, you must configure these manually. Note that the server may use the same port number on multiple clients.

- **RMI port number** defines the port number that the EMF server will be registered to RMI. This is used by remote clients (Such as the EMF Administration UI console) to contact the EMF server. The default is **50001**.
- **Client Port Connections** allows you to restrict the ports that the server can use to communicate with the EMF Administration UI. Note that the server can use the same port number for multiple clients.
 - **To add a port connection**, click **Add Ports** to specify the port numbers and hosts.
 - **To delete or modify an existing port connection**, right-click on it and select **Edit** or **Delete**.

The other tabs on the EMF Processes Server Configuration Utility are:

- [The Database Config tab](#) (allows you to define the settings to connect to the repository database).
- [The Logging tab](#) (allows you to define the logging features of the server).
- [The Classpath tab](#) (allows you to define additional JAR or ZIP files and directories containing class files for the Server).
- [The Start-up options tab](#) (allows you to specify the time to wait if a database connection could not be made to the database server during start-up).

Configuring the Server Environment

You can use the **Classpath** tab of the [Configure Local Server](#) utility to define additional JAR files, ZIP files, and directories containing class files, that need to be accessed by the Server.

Adding Jar, Zip, and other files

Note: The default jar (and zip) files that are installed in the lib directory are automatically included in the EMF Server's environment, you only need to specify additional files (including any added to the lib directory) on the **Classpath** tab. These will be made available in the order that they appear in the list, from top to bottom.

- The **JAR and class folders on the classpath** pane list the additional jars, zips and directories that will be made available to the server. You can edit these directly - note that you must add a trailing backslash to the end of directory paths, otherwise the server will not find the classes.
- The **Add folders** button displays a dialog to allow you to search for a folder. You can either browse for a folder, or enter a full path directly in the file name section. Remember to add a trailing backslash to the directory name.
- The **Add JARS** button displays a dialog to allow you to search for .jars and .zip files. You can either browse for a file, or enter a full path directly in the file name section (you cannot select multiple files).
- The **Delete** button deletes the selected item from the list.
- The **Move up** and **Move down** buttons move the selected entry up or down one place in the list.

The above entries are stored in the **EMF.properties** file under 'CODEBASE'.

Configuring the server environment manually

If you wish, you can define additional .jar files manually instead of using the Server Configuration utility. If you do this you must also edit the EMF.properties file and add the .jar files to the CODEBASE entry according to the following rules:

- Each entry must be in the form of a URL, e.g. file:path/file_name
- Any backslashes in the path must be escaped (preceded with a backslash character). Therefore server\lib must be written as server\\lib. Alternatively you can use forward slashes, e.g. server/lib.
- Entries must be separated by a space.
- Spaces or blanks in a path name need to be replaced with %20. All \ need to be replaced with \\ . All : need to be replaced with \: . Therefore the Windows path c:\Program Files\companyname\product\Server\lib\xa_admin.jar will have to be written as: file\:c:\\Program%20Files\\companyname\\product\\Server\\lib\\xa_admin.jar
or
file:c:/Program%20Files/companyname/product/Server/lib/xadmin.jar
- The path must be accessible from the directory in which the server is running, and therefore any relative path must be defined from there (e.g. file:.\myjars\newjar.jar to specify a file named newjar.jar in the myjars directory under the directory from which the server is running). Alternatively, you can ensure that all files are referenced using an absolute path.

Example:

```
CODEBASE=file\:./xa_
admin.jarfile\:./lib/activation.jarfile\:./lib/AdventNetSnmplibfile\:./lib/fscontext.
jarfile\:./lib/jakarta-regexp-1.1.jar file\:./lib/jcert.jarfile\:./lib/jdbc2_0-
stdext.jar file\:./lib/jms.jarfile\:./lib/jndi.jar file\:./lib/jnet.jar
file\:./lib/jsse.jar file\:./lib/mail.jar file\:./lib/pop3.jar
file\:./lib/providerutil.jar file\:./lib/sap.jar file\:./lib/smqclient.jar
file\:./lib/soap.jar file\:./lib/ssl0.jar file\:./lib/xalan.jar
file\:./lib/xercesImpl.jar file\:./lib/xml-apis.jar file\:./lib/xa_jdbcutil.jar
```

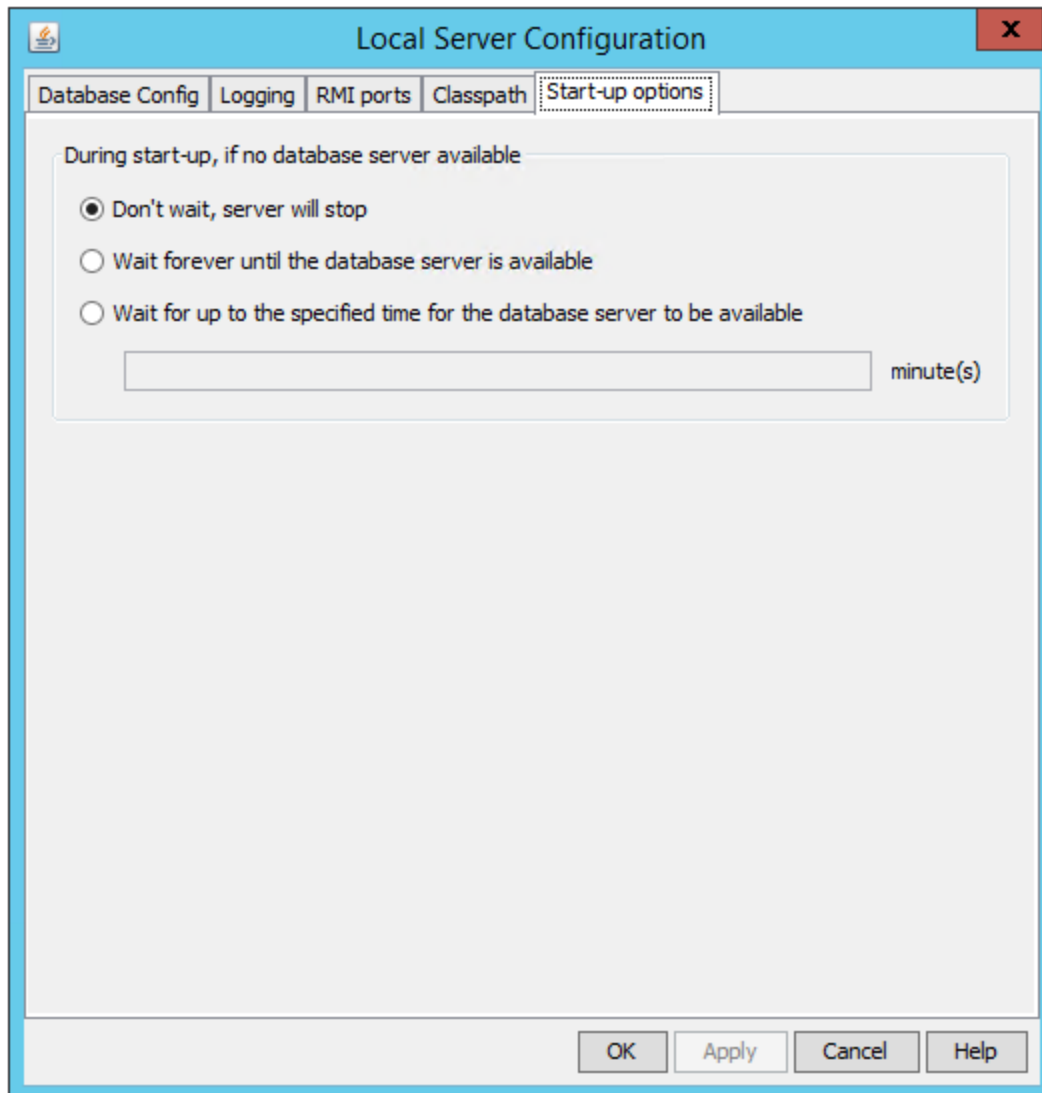
The other tabs on the EMF Processes Server Configuration Utility:

- [The Logging tab](#) (allows you to define the logging features of the server)
- [The RMI Ports tab](#) (allows you to define the rmi port configuration to be used by the server)
- [Start-up options tab](#) (allows you to specify the time to wait if a database connection could not be made to the database server during start-up)

[Configuring the Local Server](#)

Configuring the Start-up options

Start-up options tab allows you to specify the time to wait if a database connection could not be made to the database server during start-up.



Select any of the following options:

- **Don't wait, server will stop:** Select this option if the EMF server should stop if no connection to the database server can be made available during start-up.
- **Wait forever until the database server is available:** Select this option to make the EMF Server wait till a database connection can be successfully made.
- **Wait for up to the specified time for the database server to be available:** Select this option to specify the time in minute(s) for the EMF Server to wait for the database server connection. If no connection can be made within that time then the server will shutdown.

Note: If a connection cannot be made because of invalid user credentials when connecting to the database, the EMF server will always shutdown immediately and ignore any configured wait options.

Manually Changing Startup Settings

You can change the startup properties of the EMF server using the [Configure Local Server](#) utility or by manually editing the file named `EMF.properties` in the `<All Users>/Application Data>/emf/6/server` directory on Windows.

If you choose to edit `EMF.properties`, you can change the following settings:

- **ExceptionLogFile** - The output file to which any run-time exception will be output (may be useful in problem resolution). The default is `exception.log`.
- **TraceLevel** - Refers to the information that is output to the server console (it does not affect the information that is logged to the EMF repository database). The options are 0 (no trace information is produced), 1 (the default - tracing is as defined in the debug log level of the server or executing process), and 2 (all trace information is output).
- **BackupLogFile** - The output file to which logging will occur when a database connection cannot be established. The default is `backuplogfile.csv`.
- **ConfigDBPropertiesFile** - The name of the file containing the connection details to the EMF configuration database. Note that this file is not installed with the EMF server; it is created when a new repository is created.
- **DatabaseType** - The default database type. Type 1 is Oracle, Type 2 is SQL Server.
- **RMIREGISTRY_PORT** - Defines the port number of the RMI registry to which the EMF server is registered. This is the port number that the remote client, such as a console will use to attach to the server. The default for EMF is 50001.
- **CODEBASE** - The 'classpath' the server will use to find the program code it will execute. You can add additional jar files, e.g. database drivers, to this entry so that the server will know where to find them.
- **SM_PORT** - Defines the port number that the Server will use to communicate with the EMF administrator (and other utilities) if the port numbers have been restricted in the EMF Permissions file. If the port numbers have not been restricted, an anonymous (random) port number will be assigned. This entry would not normally be changed, although if the allowed ports are restricted and this port is being used by another product then another port may be assigned.
- **SM_REG_PORT** - Defines an additional port for the server to use to maintain a communication with RMID. The same considerations apply as to `SM_PORT`.
- **SM_RESTART_WIN** - The command that the EMF Administrator will run to restart the EMF server service on the Windows platform.
- **SM_START_WIN** - The command that the EMF Administrator will run to start the EMF server service on the Windows platform.
- **SM_STOP_WIN** - The command that the EMF Administrator will run to stop the EMF server service on the Windows platform.
- **DF_RECURSION_DEPTH** - The maximum number of levels in nested dynamic functions (default is 20).
- **STARTUP_HOST** - The IP address that the EMF Server should use when it registers itself with RMID upon startup.

Note: This entry is optional; the default is `localhost` or `127.0.0.1` which is translated to the machine's primary IP address.

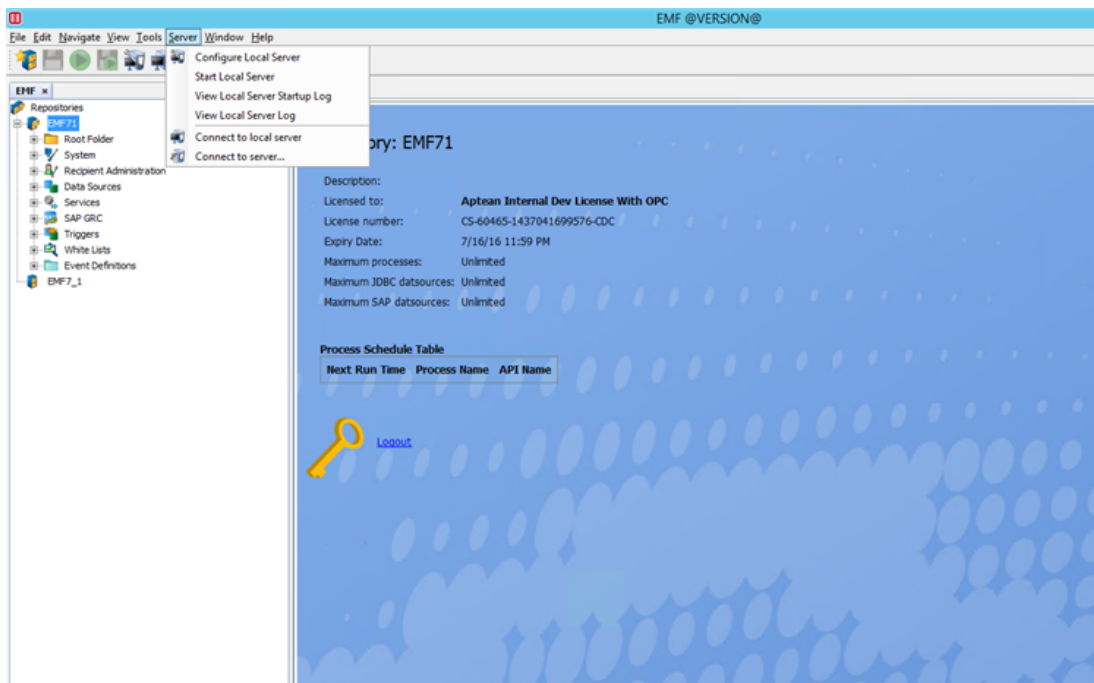
Managing the EMF Server

The following topics contain information intended to assist you in starting and managing the EMF server and running it as a windows service:

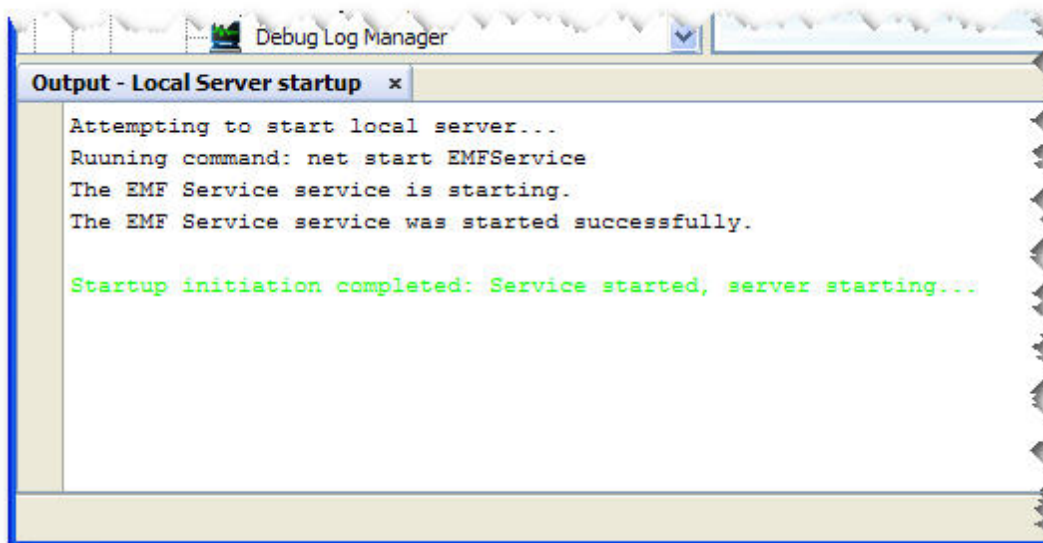
- [Start the EMF server](#)
- View logs ([Local Server Startup Log](#) and [Local Server Log](#))
- Connect to servers ([Local Server](#) and [EMF server](#))
- [Monitor the server activity](#)
- [Stop the EMF Server](#)
- [Run EMF as a Windows service.](#)
- [Start EMF components in the required order](#)
- [Check EMF Configuration details](#)

Starting the Local Server

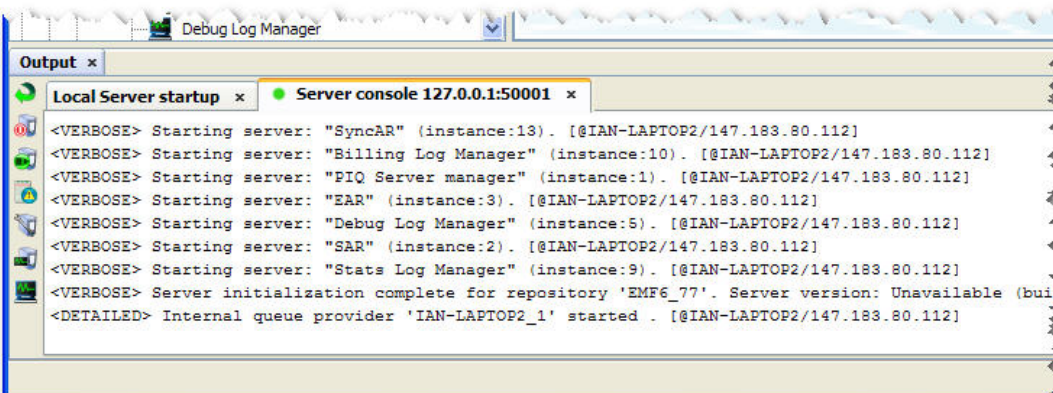
On the **Server** menu, click **Start Local Server** to start the EMF Server service on the same local machine that the EMF Administrator is running on. The service should not already be running.



The server start-up process can be viewed in the **Output** window that will be displayed. (If the window is not displayed, then select **Windows** > **Output** to view it).



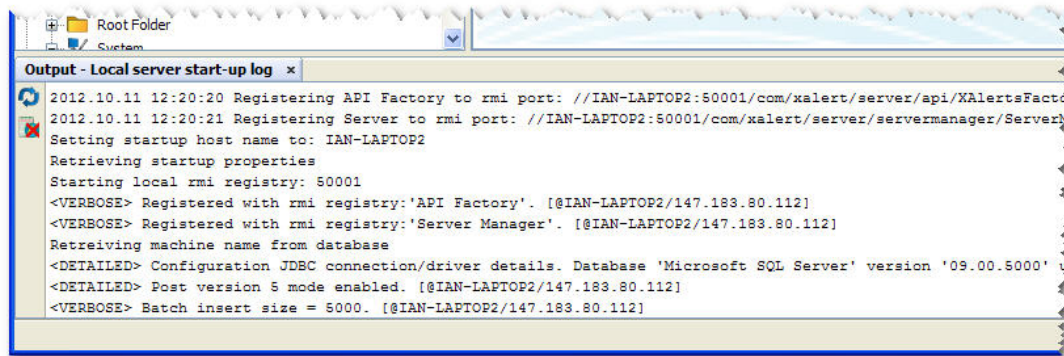
After the first part of the server start-up sequence is successful, the [Server console](#) window will be displayed.



Viewing the Local Server Startup Log

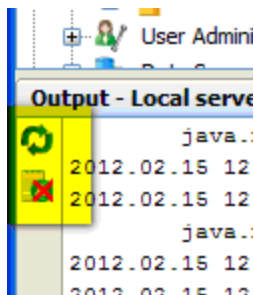
On the **Server** menu, click **View Local Server Startup Log** to display the EMF Server startup log for the local computer.

This startup log constitutes the contents of the file <All users>\Application Data\emf\6\server\sa_startup.txt. This log contains the startup log generated when the EMF Server service was started. It is helpful to view the log to diagnose problems when the server fails to start.



The console window has two additional icons:

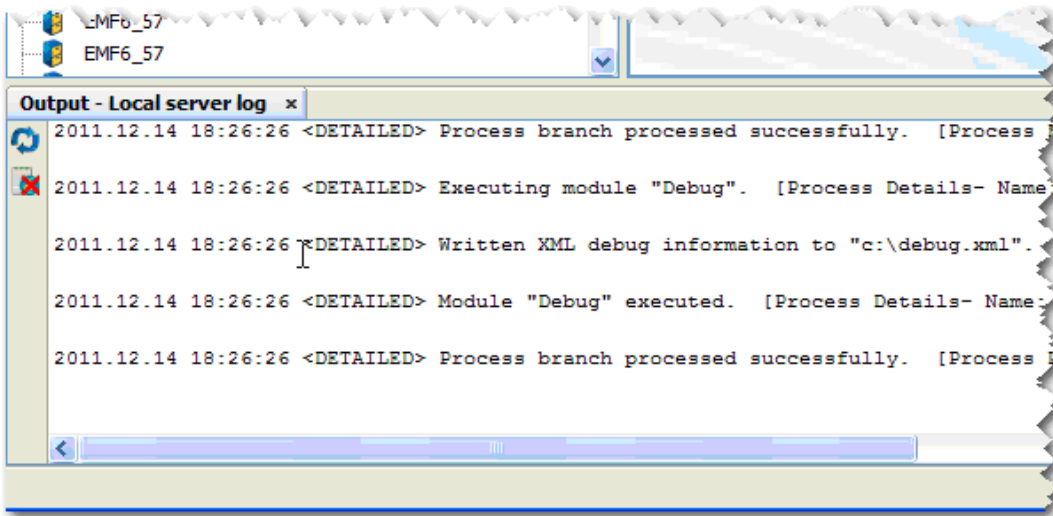
- **Refresh** - Reloads the console window with the updated contents of the startup log file.
- **Clear** - Deletes the contents of the startup log. The file on the disk will be made empty and the console display will be cleared.



Viewing the Local Server Log

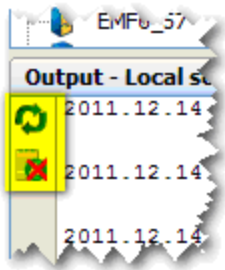
On the **Server** menu, click **View Local Server Log** to display the EMF Server log file for the local computer.

The default location for this file is <All users>\Application Data\emf\6\server\exception.log. Content logged to this file depends on the settings made in the server configuration [Logging tab](#).



The console window has two additional icons:

- **Refresh** - Reloads the console window with the updated contents of the log file.
- **Clear** - Deletes the contents of the log file. So the file on the disk will be empty and the console display will be cleared.



Connecting to the Local Server

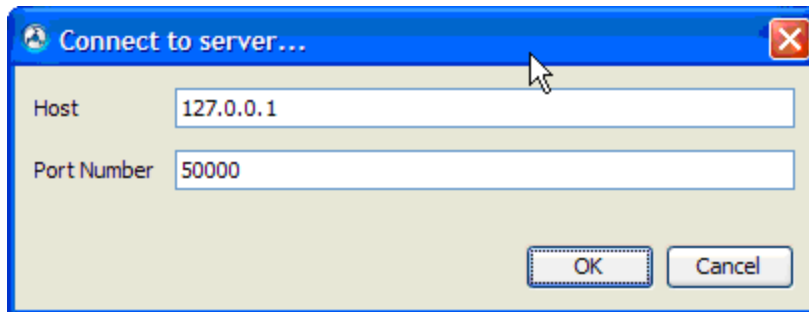
On the **Server** menu, click **Connect to Local Server** to open the server console window, and to connect it to the EMF server running on the local computer.

This allows the server activity of the local server to be monitored. See [Monitoring the Server Activity](#) for more information.

Note: To connect to an EMF server running on a remote machine, use [Connect to server](#).

Connecting to the EMF Server

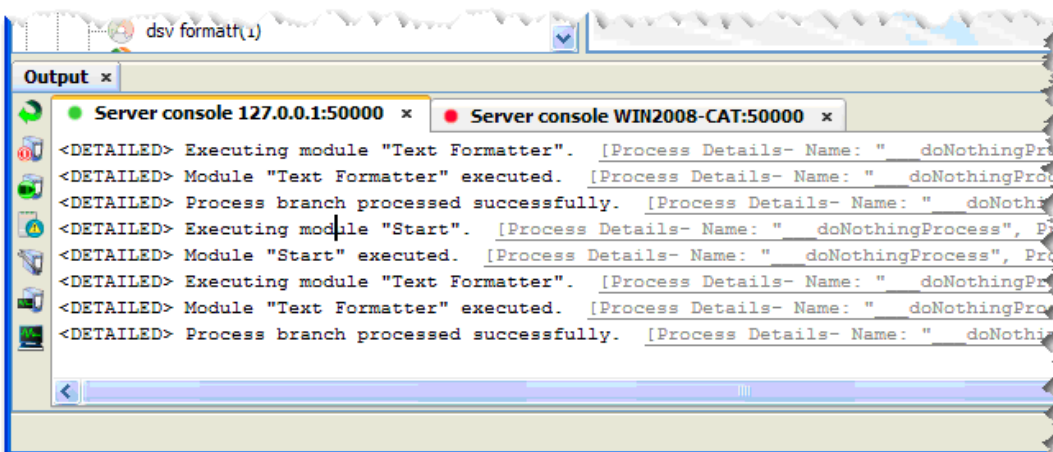
On the **Server** menu, click **Connect to Server** to enable connection to an EMF server running on a remote machine. The host (computer name or IP address) that the remote server is running on should be entered in the dialog box, along with the RMI port that the server is running on (the default is 50001).



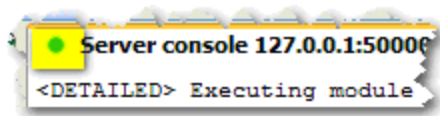
On clicking **OK**, the server console window will be displayed, and will attempt connecting to the EMF server running on the specified machine. This enables the server activity of that server to be monitored. See [Monitoring the Server Activity](#) for more information.

Monitoring the Server Activity

The **Server console** window enables monitoring the server activity. It is possible to have more than one server console window open at the same time to monitor the activity on different servers.

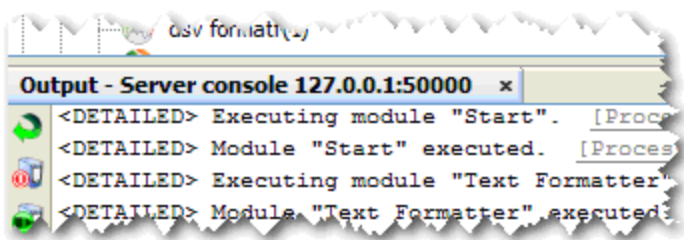


The above figure shows two server console windows displayed. One is connected to the local server 127.0.0.1 and the other to the computer named WIN2008-CAT. If the console window is successfully connected, it will have a green dot next to the server name to indicate this.



A red\grey dot next to the server name, as shown in the connection to the WIN2008-CAT computer, indicates that the server is not currently connected. It however continues to try to connect in the background. So if the server on the remote machine was started, as soon as the server console could connect, it would automatically connect. It is not required to close and reopen the console window to make it connect to the server.

Note: If there is only one output tab open, then the extra tab level is removed automatically to save space. However, this has the drawback of no longer showing the connection status icon, as shown below. To view the connection status, open another output window so that a minimum of two tabs are displayed (such as the IDE Log window: **View > IDE Log**).



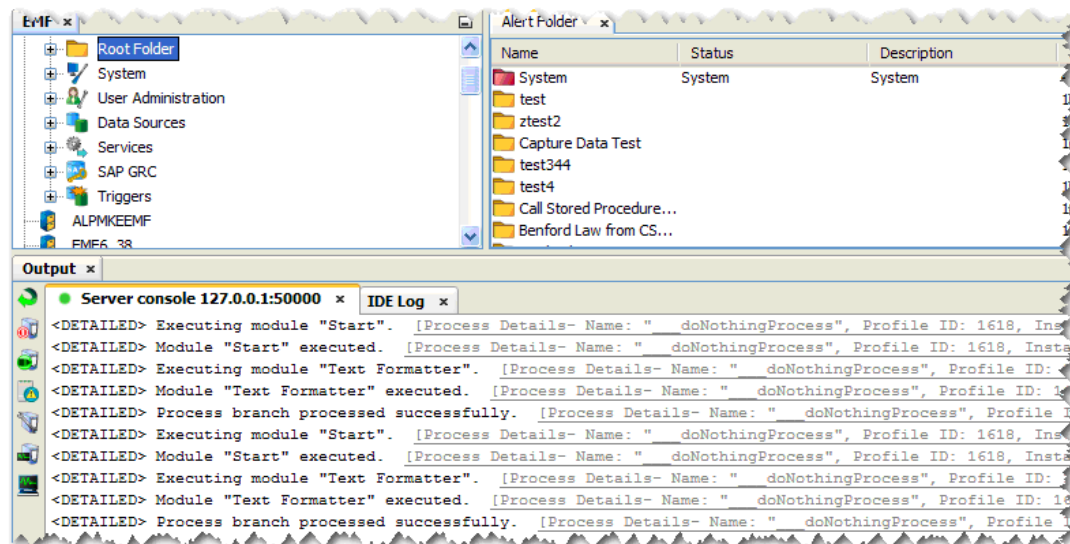
Hyperlinked Messages

The server console output displays all the messages that the server generates, as and when they are generated. It is equivalent to the message display in the separate Server Manager Application prior to EMF 6.1.

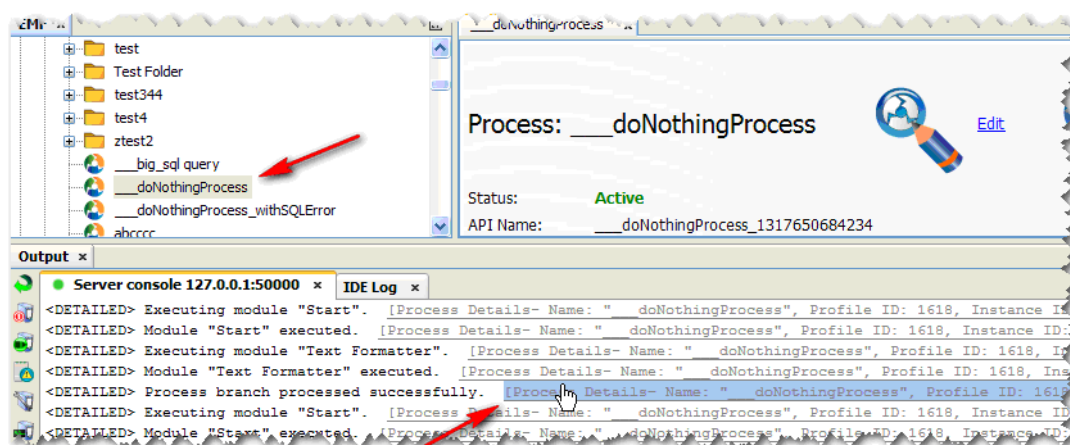
All the standard ERROR/WARNING/DETAILED/VERBOSE messages that are generated by the server are hyperlinked (wherever possible) to allow the entity that caused the message to be quickly navigated to.

Note: Before being able to click on one of the hyperlinks, you need to be logged into the appropriate repository in the EMF Administrator.

For example, if the EMF server generates the following messages, and you click one of the hyperlinks in the console window associated with a message, it will automatically select the related item in the tree.



This feature is demonstrated below:











Auto Scrolling of the Console Window

The **Server console** output window is designed in a way that if you scroll to the bottom of the window, any new output generated in the window will automatically cause the window to scroll up, so that new content remains visible at the bottom of the window (auto-scroll). If the user scrolls the window to any other position than at the bottom, then the content will stop auto-scrolling, allowing the user to easily read the content. Any new content is still added to the bottom, but the window will not automatically scroll to keep the new content in view.

Console Operations

The **Server console** window has a number of icons down towards the left hand side to

perform operations specific to this window. All operations except the first are disabled, unless the window is connected to a running EMF server. The operations are:

-  - Bring/do not bring the **Server console** window to the front on update. This is a toggle state that determines whether the console window should be brought to the front when new output is written to the console window.
 -  - Stop the EMF server - Shuts down the running server.
 - Controlled shutdown - Causes the server to wait until all processes currently being processed are in a safe state to stop (i.e., they are in a persistent state in a queue or database). If, for example, a script module is running a very long operation, this would wait for it to finish.
 - Immediate shutdown - Starts a controlled shutdown, but if the controlled shutdown does not complete within 30 seconds (because a long operation is running or the process is stuck in an endless loop), then the shutdown will force terminating the server.
-  - Restart the EMF server - Restarts the server. While the server is being restarted, the **Server console** window will lose contact with the server. It should automatically reconnect after the server has restarted. If you are trying to restart a remote server and it fails to start again for some reason, then you need to access the remote machine the server is on, to restart it.
-  - View Server log - Displays the server log, similar to [View Local Server Log](#). The difference is that the information comes from the running server, and hence can be displayed even if the server is on a remote computer.
-  - Opens the EMF server configuration dialog box. This displays the server configuration, similar to [Configuring the Local Server](#) dialog box. The difference is that the information comes from the running server, and hence can be displayed and altered even if the server is on a remote computer. You will not be able to do the following when compared to [Connecting to the Local Server](#) configuration:
 - Testing the repository connection details.
 - Browsing to a different database properties file.
-  - Opens the EMF [Server Activity Monitor](#) for the connected server.
-  - Changes the trace level of messages that the server displays in the console. (This does not affect the messages that will be logged to the repository database log).
-  - Clears the console output. It is advisable to clear the console output every now and then to stop it building up, as the build up will increase memory requirements on the EMF Administrator.

Other Console Window Operations

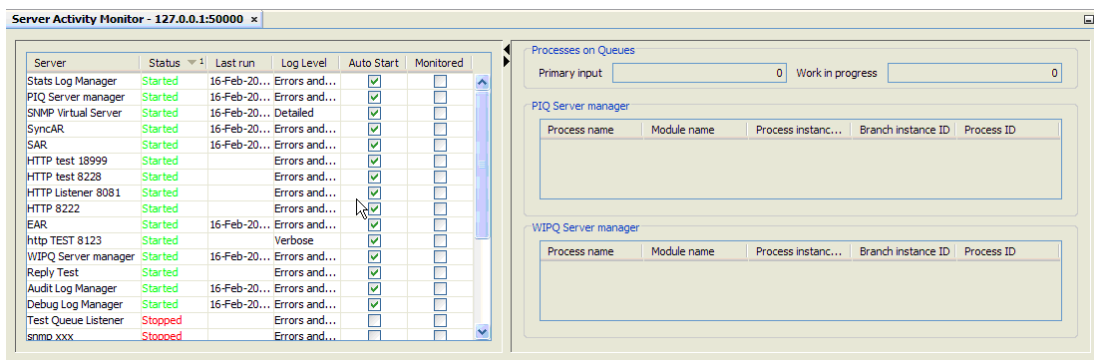
Right-clicking in the console window displays a menu with additional features:

- Find - Allows you to search for a string of text in the output window.
- Filter - Filters out all lines that do not contain the specified text. For example, if you have a process named `ABC` and need to view only those messages related to that process, set a filter `ABC`.

- Wrap Text - Wraps the text output over multiple lines.
- Large Font/Small Font/Choose Font - Allows the font of the text to be altered. Note that the font changes apply to all text in the output window and all output windows.
- Clear - Clears the console output. It is advisable to clear the console output every now and then to stop it building up, as the build up will increase memory requirements on the EMF Administrator.

The Server Activity Monitor

The Server Activity Monitor displays the live status of the connected server. To open the Activity Monitor connected to a server, it is required to first connect to a running server through the console window (see [Connecting to the Local Server](#) and [Connecting to the EMF Server](#)).



The display shows the list of servers/listeners that are running for the connected EMF server (that are defined under the **System > Machines** node in the EMF repository). The table displays the following columns:

- Name - Name of the server/listener.
- Status - Whether the server/listener is started or stopped.
- Last run - The date and time the server/listener last tried to process work. Note that some listeners are constantly listening for incoming data and therefore do not run at regular intervals; hence the last run column will always be blank (such as the HTTP Listener).
- Log level - The amount of messages the server will be logging.
- Auto Start - Whether the server/listener was set to auto-start. If it was set to auto-start, and if the EMF server is running but its status is showing *Stopped*, then it must have failed for some reason. In this case, check the logs.
- Monitored - Whether the server/listener has been set to be monitored internally by the EMF server.

By right-clicking any server/listener and accessing the context menu specific to it, you have the ability to start/stop/restart the server/listener, and also to change the amount of information it is logging.

Server Manager Activity Monitoring

Servers of type **Server Manager**, for example, the system **PIQ Server Manager** and **WIPQ Server Manager**, have additional items enabled on the context menu that allows them to be monitored. Selecting either **Monitor activity in 1** or **Monitor activity in 2** will cause that server manager to have its activity displayed in the right-hand side of the window. Activity monitor 1 is the top activity monitor on the right hand side, and activity monitor 2 is at the bottom. By default, the system **PIQ Server Manager** is assigned to activity monitor 1 and **WIPQ Server Manager** to activity monitor 2; hence in most configurations, it is never required to change the assignments.

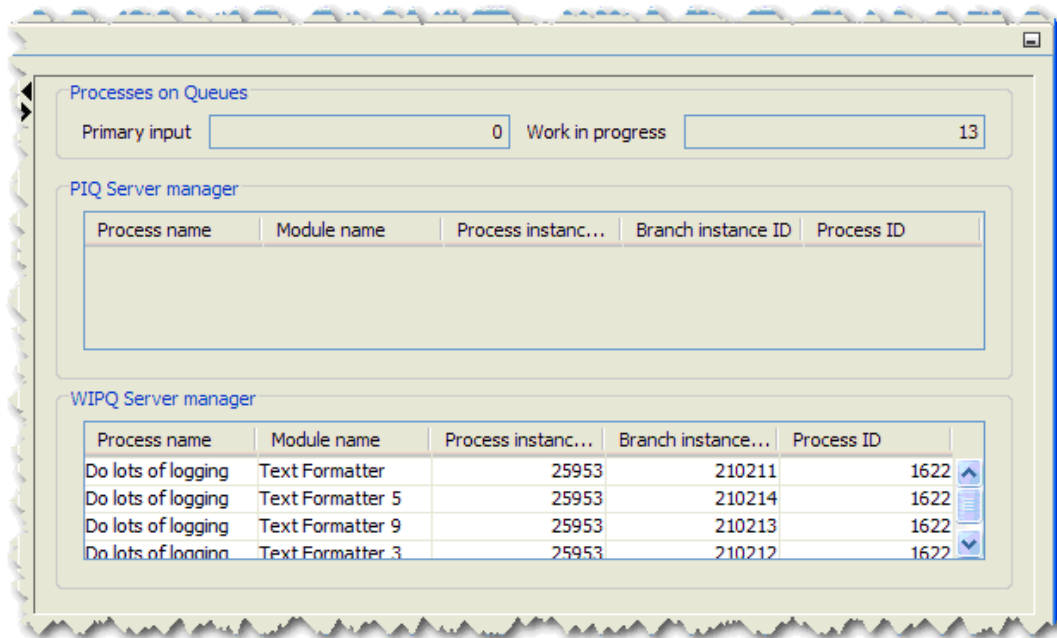
The right-hand side of the window shows the following information:

- Processes on Queues - This displays how many processes are currently queued in the EMF default system queues, Primary input (PIQ), and Work in progress (WIPQ). These are the processes that are waiting to be processed. While a process is being processed, it is not on the queue, and therefore the count will not indicate that.

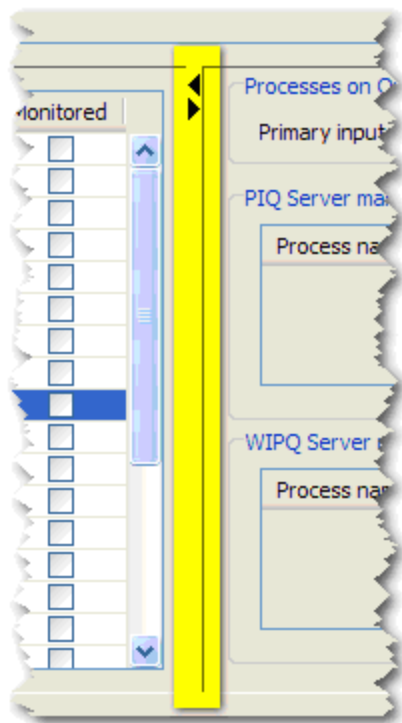
Note: If you are running single processes on the server one at a time, or less than the number of worker threads you have defined using the server manager, then these numbers will constantly read zero.

- Server Manager Activity - This is divided into two sections, which by default will be monitoring the **PIQ Server Manager** and the **WIPQ Server Manager**. When either of these server managers is processing a process from the queue, it will display an entry in the table to indicate the process and module that it is currently processing. Each line in the table represents a different worker thread that is processing a different process (or part of a process).

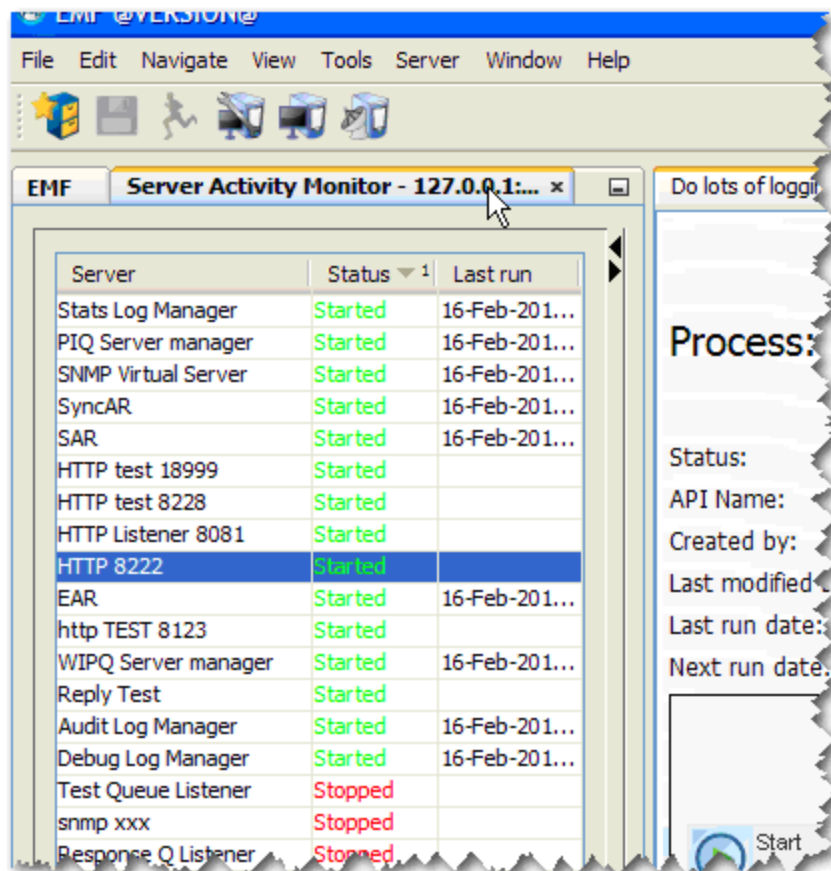
The following figure shows the activity monitor running a process that branches many times. The **Work in progress** queue count indicates that there are messages waiting to be processed, while the **WIPQ Server manager** activity monitor lists all the worker threads that are currently being processed.



The divider bar that separates the two sections of the activity monitor can be dragged to the left or right to determine how much of the two sections should be visible. The two black arrows can also be used to hide or show the entire left or right sections.



For example, it is possible to hide the right side or hide some of the columns (right-click the column heading to access the menu) so that the window can be docked well into the same pane as the EMF tree.

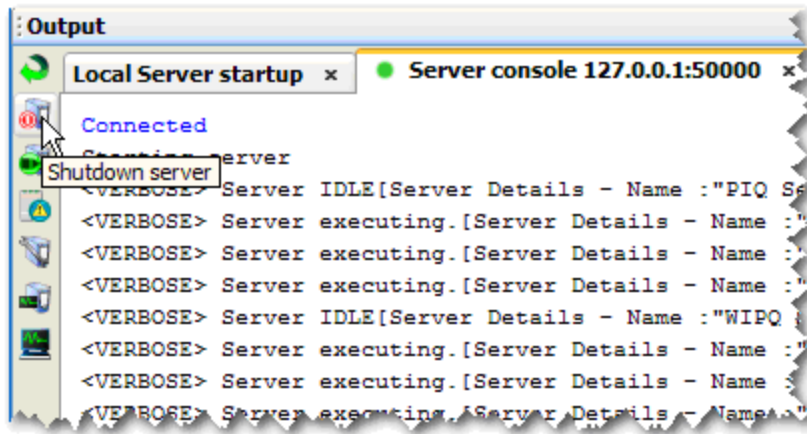


Stopping the EMF Server

To stop the EMF server:

Follow the instructions that are appropriate for the Windows platform on which the EMF server is installed. To stop the EMF Server service, do one of the following:

1. Open the **Control Panel**, go to the **Services** window, right-click the EMF Service, and select **Stop**.
2. Through the EMF Administrator, connect to the server console display and click the Shutdown server button.



3. Click **Start**, point to **EMF 7 > Server > Shutdown EMF Server**.

[Starting the EMF server](#)

[Configuring the EMF system](#)

Adding EMF as a Windows Service

The EMF server will automatically be configured to run as a Windows service after installation. To manually register or unregister the server as a Windows service, perform the following:

To register the EMF server as a Windows service:

1. Click the **Start** button and point to **EMF 7 > Server**, then click **Register as Service** (alternatively you can run **ServiceInstall.cmd**, located in the root folder of your server installation). If the registration has been successful, you should see **EMF Service** listed in the Services window.
2. Configure the EMF server as below.

To unregister EMF as a Windows service:

Click the **Start** button, point to **EMF 7 > Server**, and then click **Unregister as Service** (alternatively you can run **ServiceUninstall.cmd** located in the root folder of your server installation).

To configure the EMF service:

1. Select **Start > Settings > Control Panel > Administrative Tools > Services**.
2. Double-click **EMF Service** in the **Services** screen to display the **Properties** screen.
3. Select **Automatic** (the default setting) or **Manual** from the **Startup type** drop-down list in the **General** page.

Important: If you set the Windows service to start automatically, and are running the queuing software and/or database server on the same machine, you must [configure dependencies](#) for the service. This can be done after you have finished with the Properties screen.

4. Click the **Log On** tab.
5. In the **Log On** page, select the **This account** option. Then enter the details of your personal user account.

Note: If you use the **Local System account**, you will not be able to read/write any files out to remote machines that use network shares.

6. Make any other changes you require in the Properties screen.
7. Click the **OK** button to save your changes and close the screen.

To start the EMF server:

See the following topics:

[Starting the EMF server](#)

[Configuring the EMF System](#)

[Configuring the EMF Service](#)

Configuring the EMF Service

The EMF Service configuration tool is used to change the settings the service will run with. It allows you to enter additional Java command arguments to control the Server environment.

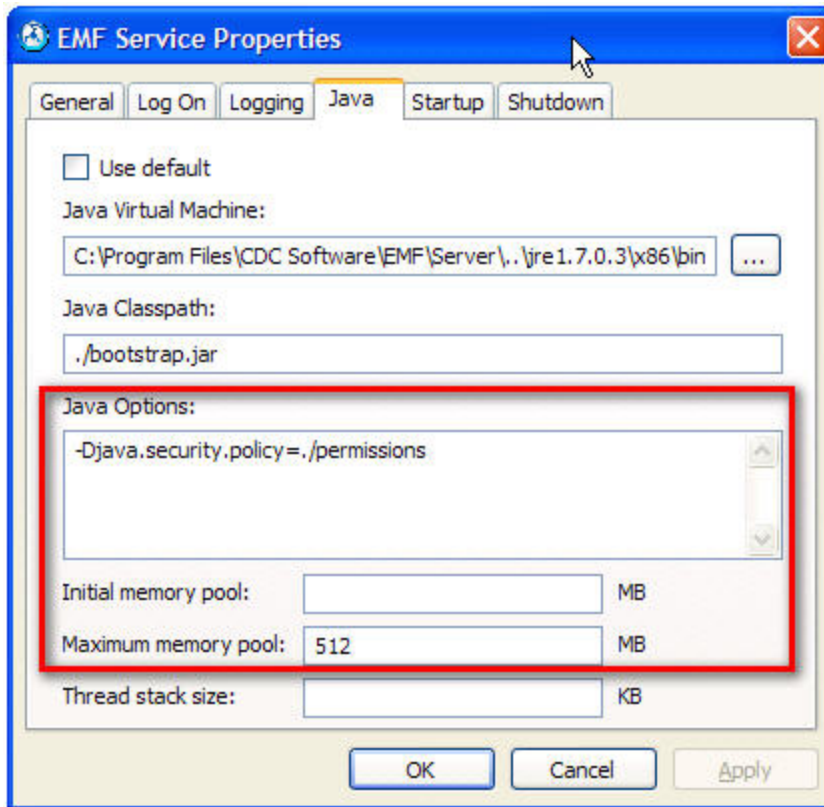
The most common use is to specify the amount of memory that is to be made available to the server.

Warning: Entering invalid settings can stop the server from starting.

To run the EMF Service configuration, select **EMF 7 > Server > Configure EMF Service** from the Windows **Start** menu, or alternatively, run the file `EMF/Server/ServiceConfigure.cmd`. You may need to run this command with Administrator privileges dependant on your rights and the operating system.

Defining memory allocations and other Java command arguments

On launching the Service Configuration tool, switch to the **Java** tab.



The settings that you may need to change are highlighted in the above figure in red. Changing other settings may cause your server not to run, and is not recommended unless advised by qualified support staff.

To change the memory settings, alter the initial memory pool and maximum memory pool.

Note: The initial memory pool size is also the minimum amount of memory, and this is always available - any memory that is required above this amount will be allocated dynamically up to the specified maximum. If you anticipate that you will be running particularly memory-hungry EMF Processes, or if your system is running other applications that might restrict the memory available to the server, you can specify a larger initial size.

Note: You should take care when specifying a maximum size. This is an absolute value and cannot be exceeded, so you should ensure that you allocate enough memory to handle the largest EMF Process that you will process (If an EMF Process reads more data than can be held in the available memory, you will receive an `Out of memory error`.).

On the other hand, if you specify an upper limit that is too high for your system (i.e., close to the amount of physical RAM installed on the machine), you might experience a large amount of disk swapping and this could seriously impact your system performance. You can allocate more memory if you are running with a 64-bit JVM. Allocating more than about 1GB of memory on a 32-bit JVM may fail (even if you have more physical memory), and the server will not start.

Other JVM settings that you may need to set are configured in the **Java Options** field. Enter one line for each setting. Do not put more than one setting on a line. You can see a full list of the Options supported by the Java application launcher at: <http://docs.oracle.com/javase/7/docs/technotes/tools/windows/java.html>

Generally, you would only ever want to modify the "non-standard options" listed here.

Note: After you change any settings, you need to restart the service.

Starting EMF Components in the Required Order

The EMF server requires connections to the queuing software and the repository database server in order to start successfully. It is important that the various aspects of the EMF system start up in a particular order, and in order to ensure that the EMF service does not start until after the database server and/or queuing software have started you may need to configure **dependencies**. You do not have to do this if both the database server and queuing software are installed on different machines to the EMF server.

You will need to configure dependencies if:

- You have installed the repository database server and/or the queuing software on the same system machine as the EMF server, **and**
- You have configured the EMF service to start automatically.

If both the above are true, it is not possible to determine or specify the order in which services will start when the machine is rebooted unless you configure dependencies. Ideally, the components should be installed on different machines.

To configure dependencies on a UNIX machine:

Rename the files in the /etc/rc.d directory to change the sequence number that forms part of the file name.

To configure dependencies on a Windows machine:

Use the tool named **sc.exe** that is supplied with Windows. For operation and syntax, refer to [Using SC.Exe with EMF](#).

For further information on sc.exe, type `sc config /?` at the command prompt, or refer to the following article on the Microsoft Support Web site: [How to create a Windows service by using Sc.exe](#)

[Configuring the EMF System](#)

Using SC.Exe with EMF

If you need to [configure dependencies on a Windows machine](#), you must obtain the tool named **sc.exe**. This is supplied with the operating systems on newer versions of Windows. On older versions of Windows, it is available with the appropriate Windows Resource Kit. The following syntax should be used when using **sc.exe** with EMF (for further information on

sc.exe, type [sc config /?](#) at the command prompt, or refer to [this article](#) on the Microsoft Support Web site).

Determining the Current EMF Service Dependencies

To find out what dependencies (if any) are currently configured for the EMF service, type the following at the command prompt:

```
sc qc "EMFService"
```

The output should look something like this:

```
[SC] GetServiceConfig SUCCESS

SERVICE_NAME: EMFService

        TYPE : 10  WIN32_OWN_PROCESS
        START_TYPE : 2
        AUTO_START_ERROR_CONTROL : 1
        NORMAL_BINARY_PATH_NAME : %%windows_product_install%%\Server\XalertsService.exe
        LOAD_ORDER_GROUP :
        TAG : 0
        DISPLAY_NAME : EMF Service
        DEPENDENCIES :
        SERVICE_START_NAME : LocalSystem
```

Make a note of any dependencies listed next to DEPENDENCIES.

If all the dependencies you want to configure are listed, you do not need to perform any other further steps.

Finding the Service Names of the Dependencies

Before you can add services as dependencies in Windows using sc.exe, you need to know the service name of the service. This may be different to the name displayed for the service in the **Services** screen.

To find out the service names of the services you want to configure as dependencies, do the following:

1. At the command prompt, type `sc query ri=`

The query command with the ri option allows you to step through all the services currently configured for the machine. Services are listed in index order. If you don't specify an index (ri= xxx), the query command starts at an index of 0.

The query command does not list all the services at one time. A partial list is displayed, starting at the index specified by the ri= xxx option. At the end of this list, the query command tells you what index you should use if you want to continue stepping through the services.

2. If necessary, when prompted, type `sc query ri= xxx`, where xxx is the resume index.

Important: Note the space between the equals (=) sign and the index.

3. When you find a service you want to configure as a dependency, make a note of the SERVICE_NAME.
4. Repeat **steps 2-3** until you find all the services you want to configure as dependencies

If you are using an Oracle database server, you should locate the service names of both the Oracle service and the Oracle listener service.

Adding the Service Names as Dependencies

Now that you know the service names of all the dependencies you want to add, you can use the config command to add them to the EMF service. When you add dependencies, all the dependency information for the service is overwritten. Therefore, if any services were already listed as dependencies for the EMF Service, you need to enter them again.

Important: Stop the EMF Service before continuing.

At the command prompt, type the following:

```
sc config "EMFService" depend= "yyy"
```

where yyy is a forward slash separated list of the service names already listed as dependencies for the EMF service **and** the service names you want to add as dependencies.

For example, if you are using SwiftMQ and SQL Server, you would type the following:

```
sc config "EMFService" depend= "SwiftMQ/MSSQLServer"
```

Important: Note the space between the equals (=) sign and the list of dependencies.

If the dependencies were added successfully, you will see the following message:

```
[SC] ChangeServiceConfig SUCCESS
```

Verifying the Dependencies

To verify whether the dependencies were added successfully, do the following:

1. Restart the EMF Service.
2. At the command prompt, type the following:

```
sc qc "EMFService"
```
3. Confirm that the required dependencies are listed next to DEPENDENCIES.
4. If they are, you can restart your server machine to check that the EMF Service starts without problems.

[Starting EMF Components in the Required Order](#)

[Configuring the EMF System](#)

How to Check EMF Configuration Details

This topic describes all the locations where EMF configuration details can be checked in case of errors, or changed if you are altering your system architecture (for example, moving the EMF Administrator to a different machine).

JAR Files

Many of the EMF components require access to one or more JAR files. For a complete list of the JAR files and where they should be placed, refer to [JAR files in EMF](#).

JDBC Drivers

In EMF, the JAR file needs to be added either to the `auto_jar` directory or to the classpath using the classpath editor. Click **Tools > Classpath** to access the classpath editor.

For the EMF Server to find the driver, the correct path and filename must be entered in the Environment tab of the EMF Processes Server Configuration utility. This information can also be entered manually in the **<Repository Name>.properties** file (CODEBASE property).

For more information on configuring JDBC drivers, refer to [JDBC Driver Configuration Details](#).

Repository Database

All locations where repository database name and connection details can be configured:

- **EMF Processes Server Configuration utility (DB Config tab)**. This is saved to **<Repository Name>.properties**.
- **EMF.properties** (`<Repository Name>propertiesFile` - specifies the location of the **<Repository Name>.properties** file)
- **<Repository Name>.properties** (all properties)

Queue Providers

Locations where the queue provider and queue details can be configured:

- **All queuing systems**: Queue provider dialog in the EMF Administrator. Also for external queue provider details, the Services\Queue services node.
- Internal queue provider configuration is stored in **Users/Public/emf/6/activemq.xml**.
- **SwiftMQ: smqr1.properties** on the Queuing System Machine

EMF.properties and <Repository Name>.properties

These files are used by the following components:

- **EMF Server**
- **Java Admin API**

For all of these components, the **EMF.properties** file must correctly reference the location of the **<Repository Name>.properties** file, and the **<Repository Name>.properties** file must have the correct connection details for the configuration database. For further information on the entries in **EMF.properties**, see here.

Required location of the **EMF.properties** and **<Repository Name>.properties** files:

- **EMF Server** - the EMF Server uses **EMF.properties** and **<Repository Name>.properties** files to determine server startup, database connection and environment information. The **EMF.properties** file should be in the directory specified in XA_HOME (by default this is the EMF Server directory), but the **<Repository Name>.properties** file can be anywhere as long as it correctly referenced in **EMF.properties**.
- **Java Admin API** - the Java Admin API needs access to **EMF.properties** and **<Repository Name>.properties** files. **EMF.properties** must be located in the working directory of the Java Virtual Machine (JVM) of the machine using the API. The working directory of the JVM is the directory from where the "java" command is run. **<Repository Name>.properties** can be anywhere as long as it is referenced correctly in the **EMF.properties** file.

[How to Check Your User and Database Names and Details](#)

[Troubleshooting](#)

EMF Security

EMF security controls the ability of a user to access objects in the EMF Administrator through the user interface.

Accessing an object means creating, reading, updating or deleting an object. In the case of folders, it also means subscribing to a folder. Permissions for each of these operations can be separately set to **true** (operation allowed) or **false** (operation not allowed).

Setting User Access Rights

Access rights are assigned to roles. Operators inherit access rights from the roles of which they are a member. All the roles that an operator is a member of are checked for access rights; if any of these roles is permitted to perform an operation, then the operator will also be able to perform that operation.

Security Categories

Objects can be found within the left panel tree view of the EMF Administrator. The secured types are:

- Groups, recipients and aliases (in Recipient administration)
- Roles and Operators (in System)
- Audit, billing, statistics and debug logging (in Logging)
- JDBC and SAP connections (in Data sources)
- Key store (in Net security)
- Languages (in System)
- Machines (in System)
- Queue provider and queues (in System)
- Repositories (in System)
- Server instances (in Machines)
- Services and all services under this node
- EMF Processes

Note: Other types can be considered as being subordinate to these types. Access to the parent type therefore controls access to the child types as well.

Folders, including standard folders, also have access rights; access is set per individual folder.

- Creation rights for folders control whether a user can create sub-folders within the folder.
- New folders inherit all the access rights of the parent folder.
- If a group is given Read rights to a folder, it will also be given Read rights to all ancestor folders.

Individual **modules** in the EMF Process Builder view also have security rights which control whether or not a particular module can be used or modified.

Additional secured objects not visible in the tree view are:

- **Licenses:** allows users to view the license information for a particular repository
- **Login:** allows users to log in to the EMF Administrator.

Important: To use certain components, it may be necessary to grant rights to other components first. For more information, see [list of component dependencies](#).

[Login Screen](#)

[Roles](#)

Security Rights for EMF System Operators

The EMF System Administrator can assign different security rights to different operators, which affect how much of the EMF functionality they are able to access and what they can do. Security rights are allocated to groups of operators using the [Roles Security tab](#), and each member of the role has the same rights. Where an operator is a member of more than one role, the security rights are combined so that if any of the roles has a particular right then the operator will also have it.

There are two default operator roles already set up when you install EMF - **Default** and **Security Administrators**. **Security Administrators** have limited access rights that allow them to manage roles, while the **Default** role has all available access rights to EMF.

Important - An operator's security rights can be modified at any time by any EMF system operator that has update access to the Roles functionality.

You cannot change security rights for an operator while they are logged in to EMF. If you change an operator's security settings, the changes do not become active until the operator has logged out and logged in again.

[Login Screen](#)

[Roles](#)

Security Right Dependencies

Important: All components except those listed below have no dependencies.

Component	Component
Repositories	You must have Licenses rights to view information in the License information tab of the Repository screen.
Servers (server instances)	You must have Machines rights.
All formatters except DSV	You must have Languages rights.
XSL Transformer	You must have Languages rights.
LDAP	You must have Recipients rights.
Recipients	You must have Groups rights. If you want to create an alias to assign to the recipient, you will also need Aliases rights.
All modules	You must have EMF Processes rights and rights to the Start module.
Conditional links	You must have EMF Processes rights and rights to the Start module.
All services	You must have Services rights.
Operators	You must have Roles rights.

Special notes

- To begin creating an EMF Process, you need rights to EMF Processes and the Start module (in addition to rights to any modules you intend to use in the EMF Process). You also need rights to the folder in which you want to create the EMF Process.

Important: If you do not have rights to the Start module, you will not be able to create an EMF Process, even if you have rights to a number of other modules.

- You must have Login rights to log in.
- When you enable Services rights, you automatically give the user rights to File Services.

[Login Screen](#)

[Groups](#)

Managing Operators

You can view and manage details of the operators of the EMF system by using the **Operators** section of the EMF tree view. You can do the following:

- [Add new operators, and edit or remove details of existing operators](#)
- [Create roles for operators](#)
- [Assign security rights to roles containing operators](#)

Adding, editing and managing operators

To get operator information into the EMF system:

Use the Operators section of System in the EMF tree view to add each operator's details individually. For an overview of adding and managing EMF system operators, see [Managing System Operators](#).

When you have all the details of your operators in the EMF system, you can create roles and assign them security rights (see [Managing Roles of Operators](#)).

For general information, see [Managing System Operators](#).

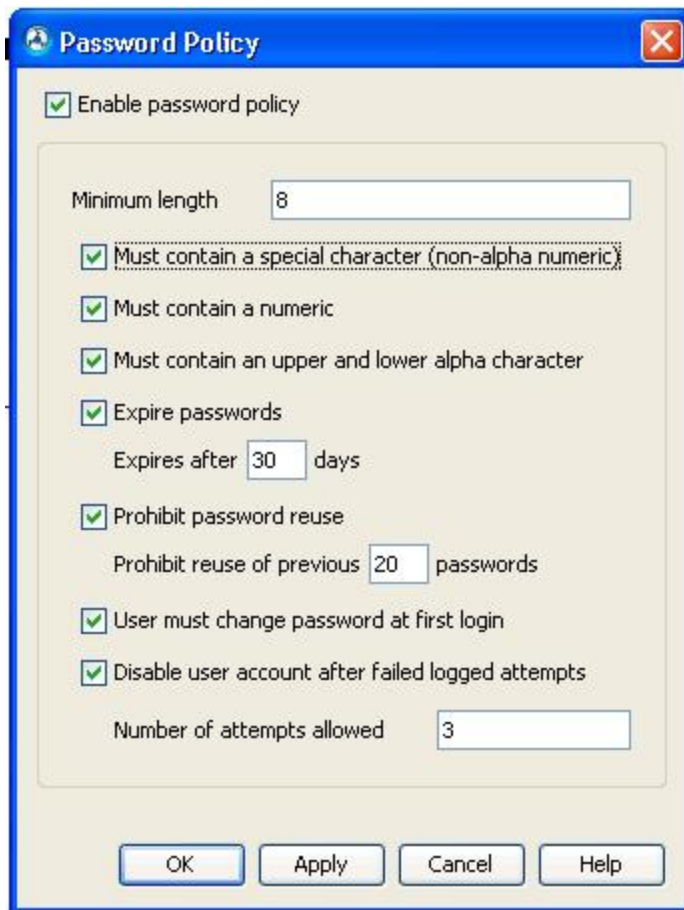
Reset the Super Administrator password

If you have forgotten your password or locked out because of too many failed attempts then an Administrator account can be used to unlock it. If you have been locked out of the Super Administrator account, then you will need a reset key from the EMF Support team. For more information, see [Reset the Super Administrator Password](#).

Enabling EMF Password Policy

A password policy is a set of rules designed to enhance computer security by encouraging operators to deploy strong passwords and use them correctly. The EMF administrator can set a password policy for users logging on to EMF.

1. From the EMF tree view, double click **System**, then right-click **Operators** and select **Password Policy**. The **Password Policy** dialog box will be displayed.
2. Select the **Enable Password Policy** check box.



3. Specify the minimum number of characters for the password in the **Minimum length** box.
4. Set the password complexity by selecting the following options as required:
 - **Must contain a special character** to ensure that the password contains a minimum of one non-alphanumeric special character.
 - **Must contain a numeric** to ensure that the password contains a minimum of one numeric character.
 - **Must contain an upper and lower alpha character** to ensure that the password contains a minimum of one upper and lower alphanumeric character each.
 - **Expire passwords** to enable password expiry and setting a new password periodically.

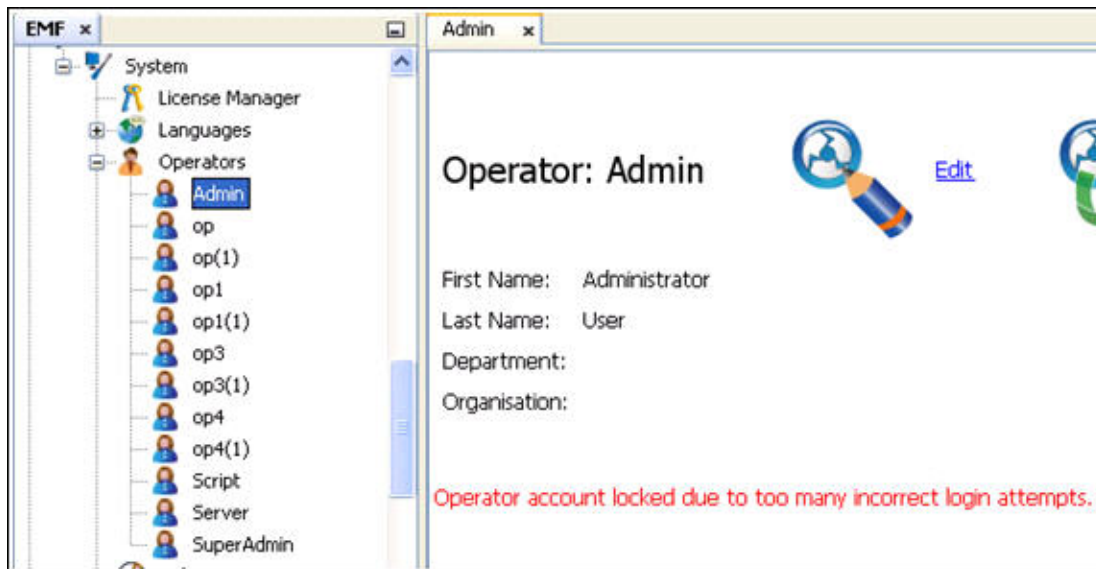
- Specify the time interval for password expiry in the **Expires after <n> days** box.
 - **Prohibit password reuse** to prevent the reuse of previously used passwords.
 - In the **Prohibit reuse of previous <n> passwords** box, specify the number of previously used passwords which should not be repeated.
 - **Operator must change password at first login** to prompt the operator to change the password when they log on to EMF for the first time.
 - **Disable operator account after failed logged attempts** to disable operator login after the number of attempts specified in the **Number of attempts allowed** box.
5. Click **Apply/OK** to save the password policy settings.

Important:

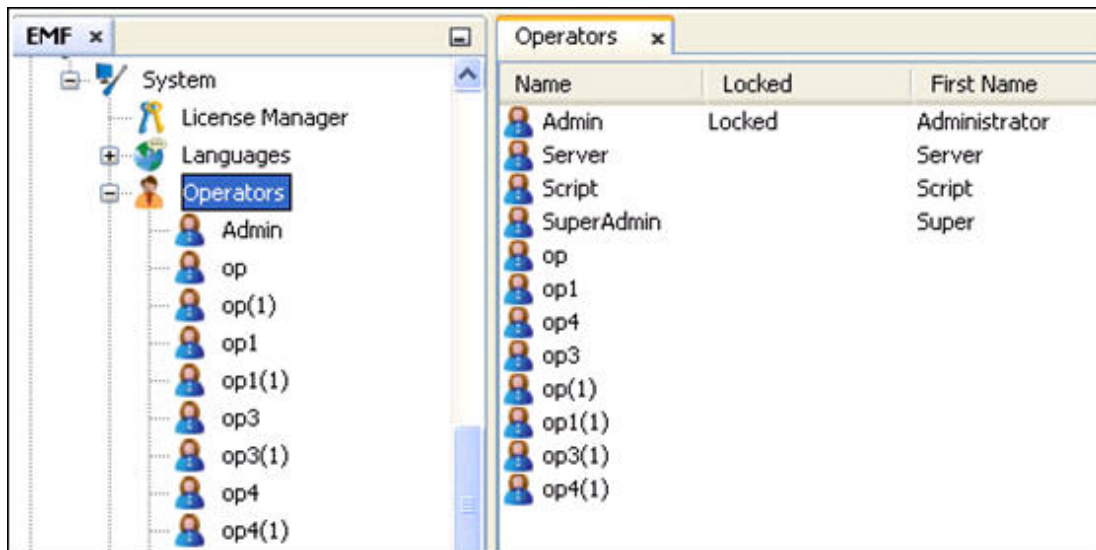
- When an operator logs in, if the password does not meet the current password policy (or if the password has expired), the operator will be forced to change the password before logging in.
- Operators can change their password details (and other settings on their operator profiles) by accessing the **Edit my Profile** menu option from the **Repository** menu. Even if an operator has no access rights to Operators, the operator can access his own profile.



- An operator must be in the Security Administrators role to be able to access the **Password Policy** dialog.
- When an operator's account is locked (due to too many failed login attempts), it can be unlocked by an operator with read/update rights to operator profiles by right-clicking the locked account and selecting **Unlock**.



- All locked operator accounts can be viewed in the detail view of the **Operators** node.



Note: The password policy set using EMF Administrator only applies to operators logging on to the EMF Administrator. It is not applicable to operators logging on to the GRC workspace. Therefore:

1. If an operator account is blocked from too many failed login attempts, then the operator will still be able to log on to the GRC workspace.
2. If the password is set to expire, then the operator will not be forced to change the password when logging on to the GRC workspace (currently, the operator cannot change the password in the GRC Workspace).

Managing System Operators

To add, delete or modify EMF system operators:

1. Double-click the **System** icon in the EMF tree view and then double-click the **Operators** icon to display a list of all the existing operators in EMF system. A new installation of EMF contains three default operators that cannot be removed or renamed (SuperAdmin can be renamed):
 - **SuperAdmin** is the default security administrator operator account that you use to create roles and give others rights to create roles.
 - **Server** is used by EMF and allows the EMF servers (e.g. the Escalated EMF Process Retriever) to log in to the repository.
 - **Script** is also used by EMF; it allows the Script module to log in. The rights assigned to the servers and Script module after successful login define what they can do within EMF.

A new installation will also contain a default operator, which is used to login to the system. This can be removed or renamed.

 - **Admin** is the default system administrator operator account that you use when you first access the EMF system after installation.
2. **To delete an operator**, right-click the operator name and select Delete to remove the selected operators from the repository. All references to those operators are also removed.
3. **To create a new operator**, right-click in the list view and select **New operator**. To modify an existing operator, double-click its name in the **Operator name** column. The Operator screen is displayed.

4. The **Operator** tab of the Operator screen is for personal details on the individual that is to be brought into the EMF system. Enter the relevant information as follows:

- **Operator Name:** The name that the person must use when logging in to EMF. It is used purely for identification purposes and need not be the person's real name. For more information, see [The Login Screen](#).

Note: The only mandatory fields on this screen are: **Operator name**, which is used for identification in EMF, **Password**, which is the password that will be used to access the EMF system, and **Language**, which is the primary language that the operator will be using. All the other fields on this screen are purely for your information.

- **Password:** The password that is associated with the Operator name in the Login screen. The default password for the EMF administrator is blank.
- **Language:** This is the default language that will be used to send EMF processes to the operator.

The following fields are optional and primarily for your information.

- **Title:** Title of the operator. For example, Mr, Ms.
- **First name:** First name of the operator. For example, Matilda.
- **Last name:** Last name of the operator. For example, Smith.
- **Organization:** The organization to which the operator belongs. For example, Smith Software.
- **Department:** The department in which the operator works. For example, Research & Development.
- **Telephone number:** A standard telephone number.
- **Salutation:** A salutation. For example, Hello.
- **Address Line 1:** The house number (or building name) and street name associated with the operator.
- **Address Line 2:** The name of the district.
- **City:** The name of the town or city.
- **State:** The name of the state, county or area.
- **Zip Code:** The zip code or post code.
- **Country:** The country where the operator lives.

Note: The country has no effect on time zone settings.

[Back to Start of Operator Management](#)

Managing Roles of Operators

You can use **roles** to limit an operator's access to the EMF system. You can add a new role, or edit or remove the details of an existing role, by using the **Roles** screen (in System) in the EMF Tree view.

Within EMF, two types of operators can be assigned to a role:

- **Internal operator:** This type of operator exists only within EMF and is created and managed within EMF (under the System->Operators node).
- **External operator:** This is a user that exists with an LDAP store (for example, Microsoft Active Directory), and those details are mapped onto a role within EMF, so that user can log in to EMF using their LDAP credentials (i.e., their LDAP user name and password).

When you first create an operator, the operator is not assigned to any role. It needs to be assigned to a role before it can be used to log in to the system.

You can create as many operator roles as you want, each with different access rights and privileges, and each operator can belong to any number of roles. When an operator is a member of more than one role, if any of these roles has a particular right, then the operator will also have it. Access rights for each EMF area can be granted on a **Create, Read, Update** or **Delete** basis (for more information, see the Roles Screen [Security Tab](#)).

There are two default roles in EMF:

- **Default:** This role - which can be deleted - has full access rights, so new users should generally not be added to this role. The default operators named **Admin**, **Server**, and **Script** belong to this role.
- **Security administrators:** This role has limited access rights and allows you to create operators and roles only. You cannot change the security settings for the Security Administrators role, so if you want to restrict operators from accessing any aspect of EMF, you should put them in a different role, creating one specifically for this purpose if necessary. The default operators named **Admin** and **SuperAdmin** belong to this role.

Overview of EMF Role Security settings

Every EMF operator that exists within the repository is a member of one or more roles, and each role has its own security settings. This allows you to restrict what individual operators can do within the EMF administrator, system and builder.

You can limit or prohibit access to every function and utility in the EMF tree view, and every module in the EMF builder, by changing the security rights that the different roles have to them. Thus, you can allow people to browse the system and see everything but change nothing, to create items but not update them, to modify EMF Processes but not delete them, and so on.

Note: To administer repositories, an operator must have read rights on licenses (at least), as well as the required repository rights, so that the license info can be loaded in the UI.

[Roles Screen](#)

[Back to Start of Operator Management](#)

Naming and Describing a Role

You can use the **Properties** tab of the Roles screen to enter a name and descriptive information for each role. This name is used throughout the EMF system.

To enter a name and description for a role:

1. Double-click the **System** icon in the EMF tree view and then double-click the **Roles** icon to display a list of all the existing roles in the EMF system.
2. Double-click the required role to open the Roles screen. The **Properties** tab is displayed (for information about the other tabs see [Security](#), [Operators](#), [External Operators](#) or [Folders](#)).

3. Enter a **Name** for the role that will be used as a reference to the role throughout the system.
4. You can optionally enter a **Description** that defines the purpose of the role.

[Role Administration](#)

[Back to Start of Operator Management](#)

Adding Operators to a Role

You can use the **Operators** tab of the [Role screen](#) to define which operators should be placed in the role and given the security rights that you define on the [Security](#) and [Folders](#) tabs.

To view the Operators tab:

1. Double-click the **System** icon in the EMF tree view and then double-click the **Roles** icon to display a list of all the existing roles in the EMF system.
2. Double-click the required role to open the **Roles** screen and select the **Operators** tab (for information about the other tabs, see [Properties](#), [Security](#), [External Operators](#), or [Folders](#)).

The **All operators** panel on the left of the **Operators** tab shows all available operators, and the **Selected operators** panel on the right shows the operators that are currently in the role.

- **To add an operator to the role:** Select the required operator and click the > button.
- **To add several operators to the role:** Hold down the SHIFT or CTRL keys while selecting the operators, and then click the > button.
- **To add all operators to the role:** Click the >> button.
- **To remove operators from the role:** Select the required operators in the right-hand pane and use the < or << buttons.

[Roles Screen](#)

[Role Administration](#)

[Back to Start of Operator Management](#)

Adding External Operators to a Role

You can use the **External Operators** tab of the **Roles** screen to assign users and groups that exist in an LDAP store (for example, Microsoft Active Directory) to a role. Users (and users within the groups) will then be given the security rights that you define on the [Security](#) and [Folders](#) tabs.

Note: Currently, External Operators are only tested against Microsoft Active Directory. Group functionality will not work against any other LDAP store.

To view the External Operators tab:

1. Double-click the **System** icon in the EMF tree view and then double-click the **Roles** icon to display a list of all the existing roles in the EMF system.
2. Double-click the required role to open the **Roles** screen and select the **External Operators** tab (for information about the other tabs, see [Properties](#), [Security](#), [Operators](#) or [Folders](#)).
3. Select the [JNDI Service](#) that contains the configuration to retrieve the LDAP users and groups. Only those services that have the check box **Enable operators to login against this service** are available from the drop-down list. Only one service can be used against each role - but duplicate Roles can be created if it is required to configure external operators from more than one LDAP directory.

The first time a JNDI service is all the queries defined in the service are run to read in the available users and groups. This may take some time. This information is cached locally, so subsequently when that service is selected, there is only a small delay to read in the information locally. If the contents of the LDAP store have changed (new users added or removed) or the queries in the service have been modified, click **Refresh Cache** to renew all the information in the cache for that service.

Note: When the user logs in, the EMF UI does not use this cached information and will access the LDAP store to find the up to date information on users/groups.

The **All operators** panel on the left of the **Operators** tab shows all available operators, and the **Selected operators** panel on the right shows the operators that are currently in the role.

- **To add an operator to the role:** Select the required operator and click the > button.
- **To add several operators to the role:** Hold down the SHIFT or CTRL keys while selecting the operators, and then click the > button.
- **To add all operators to the role:** Click the >> button.
- **To remove operators from the role:** Select the required operators in the right-hand pane and use the < or << buttons.

The filter box under the **All operators** panel can be used to filter the results.

- To filter the **All Operators** list, type in the first few characters of the user/group you are looking for.

Assigning Folder Access Rights to Roles

You can use the **Folders** tab of the Roles screen to define what security rights (if any) operators in this role have to the folders that are used to store EMF shortcuts.

To view the Folders tab:

1. Double-click the **System** icon in the EMF tree view and then double-click the **Roles** icon to display a list of all the existing roles in the EMF system.

2. Double-click the required role to open the **Roles** screen and select the **Folders** tab (for information about the other tabs, see [Properties](#), [Security](#), [External Operators](#), or [Operators](#)).

The available folders are listed in the left-hand **Folder** pane, while the permissions for the selected folder are displayed in the right-hand **Permissions** pane. Child folders automatically take on the permissions of the parent folder unless manually changed.

3. Select a folder from the **Folder** pane. The existing permissions for the selected folder are displayed in the **Permissions** pane.
4. Click in the check boxes to enable or disable permissions as follows:
 - **Folder Create** - members of this group can create subfolders
 - **Folder Read** - members of this group can view subfolders
 - **Folder Update** - members of this group can change the content and names of subfolders
 - **Folder Delete** - members of this group can delete subfolders

[Roles Screen](#)

[Role Administration](#)

[Back to Start of Operators Management](#)

Defining Security Rights for a Role

The **Security** settings for the roles that an operator belongs to will affect what they can and cannot do in the EMF tree view. Certain menu commands and options may be unavailable (or visible but grayed out and not editable) if the operator does not have the correct privileges. Note that if you want to allow an operator to modify an output formatting module, you must enable their access rights to the appropriate language component(s) as well.

Note: You cannot change security rights for an operator while they are logged in to EMF. If you change an operator's security settings, the changes do not become active until the operator has logged out and logged in again.

You can use the **Security** tab of the [Role screen](#) to define security access rights for a role. Any operators in the role will have these same access rights (and possibly others if they are also members of other roles with greater privileges).

Note: You cannot change the security settings for the **Security Administrators** role. If you wish to restrict operators from accessing any aspect of EMF, you should put them in a different role, creating one specifically for this purpose if necessary.

To view the Security tab:

1. Double-click the **System** icon in the EMF tree view and then double-click the **Roles** icon to display a list of all the existing groups in the EMF system.

2. Double-click the required role to open the **Roles** screen and select the **Security** tab (for information about the other tabs, see [Properties](#), [Operators](#), [External Operators](#) or [Folders](#)).

The **Security** tab consists of a number of horizontal rows, and each row refers to a different component of the EMF system (named in the first column). These components are divided between the three tabs labelled **System**, **Modules**, and **Services**. The items on the **Modules** and **Services** tabs correspond to the items available in the [EMF Process Design Graph](#) and [Services](#) section of the EMF tree view respectively, the items on the **System** tab correspond to the other items in the EMF tree view (NB: **Operators** are those people who can log in to the EMF administrator).

Note: The items that appear on the **Security** tab depend on your particular installation, license and any customization that may have been done (including OEM-specific components).

You can give role members different access rights to each of these components by clicking the relevant check boxes as follows (you can select multiple rows at the same time and give all the selected rows the related rights at the same time):

- **Create** - allows the role members to create items of the type defined by the Component Name.
- **Read** - allows the role members to view the information contained by items of the type defined by the Component Name. You cannot disable read permission while create, delete or update permissions are enabled.
- **Update** - allows the role members to access the information contained by items of the type defined by the Component Name and change it.
- **Delete** - allows the role members to delete items of the type defined by the Component Name. (NB: you can never delete a **Start** module from an EMF process and the Delete security settings for this module are ignored).

Important: Many components and modules are interdependent and you may not be able to modify certain settings if you do not have the appropriate rights to all other relevant components (for example, to use the Java EMF Runtime_API, you must have Create rights). For a complete list of dependencies, refer to [security right dependencies](#). Also, changes to the read, delete, or create security settings for the Start module have no effect as this is always available.

Note: If you remove read rights for a component, you will automatically remove all other rights. Similarly, if you assign any rights to a module for the first time, read rights will be automatically enabled. If you assign Create rights to a component, Update rights are automatically given. Uncheck the Update rights in order to remove Update access.

Information about your current status is displayed above the **System**, **Modules** and **Services** tabs.

For more information about the other tabs on the System Roles screen, see [Properties](#), [Operators](#) or [Folders](#).

[Roles Screen](#)

[Role Administration](#)

[Back to Start of Operator Management](#)

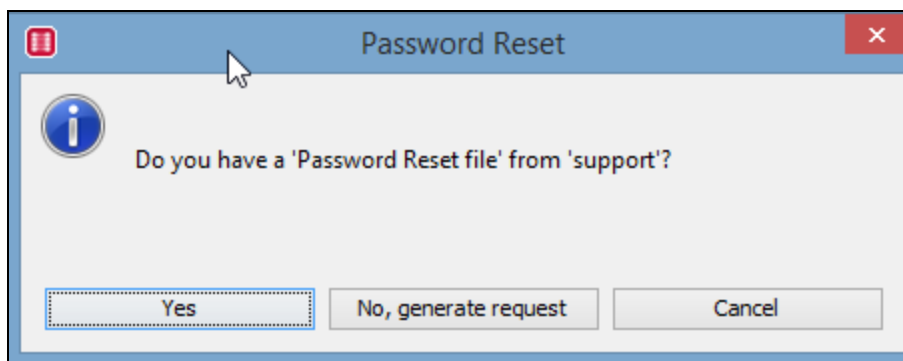
Reset the Super Administrator Password

If you have forgotten your password or locked out because of too many failed attempts then an Administrator account can be used to unlock it. If you have been locked out of the Super Administrator account, then you will need a reset key from the EMF Support team.

To reset a super administrator password

1. On the login screen, right-click the **Repository** and select the **Reset Super Administrator password** menu option.

The **Password Reset** dialog box appears.



2. Select **No, generate request**. You will then be prompted for a location to save the **password reset request** file.
3. Save the **password reset request** file and email it to the "support contact".
4. Once the "Support contact" provides the **password reset** file, right-click the **Repository** and select the **Reset Super Administrator password** menu option.
5. In the **Password Reset** dialog box, select **Yes**.
6. Select the **password reset** file sent by the "support contact".

After successful validation of the file, the system prompts you to reset the **Super Admin** password.

Note: The **password reset** file is valid for a fixed period of time, hence, it is recommended to use it immediately.

7. Reset the **Super Admin** password.

Note: Audit events are logged for each unlock attempt (whether successful or a failure).

Logging in EMF

You can use the **logging** feature to keep a record of what happens within the EMF system. You can then use the logs to review the various activities.

The Logging records different information:

- Records all changes within the administrator. It also records when EMF processes run and finish (if enabled on an individual process). For further information and instructions, see [Audit Logging](#).
- Records the messages that are sent to recipients by the various output modules. This information can then be used for diagnostic, billing, and invoicing purposes. For information on how to enable this, see [Sent Message Logging](#).
- Monitors and logs information on server activity. It records details of problems that may occur within the servers in order to help in their diagnosis and resolution. Logs information on the amount of information that is processed by data and recipient modules when an EMF Process runs and about EMF Process branch splits. For further information and instructions, see [System Logging](#).

To view the logs:

- Click **Window** and then **Log** to view details of each log. For details, see [System Logging](#).
- Click **Window** and then **Audit Log** to view details of each audit log. For details, see [Audit Logging](#).

[Managing the Size of the EMF Log Files](#)

[Managing the Size of the Audit Log Files](#)

[Extracting and Removing Information From Logs](#)

System Logging

System logging keeps a record of server activity within the EMF system. This includes information about when processes run, recording each step. The amount of information logged depends on the setting on an individual process. It includes all sent message log records. This information can be used to help in diagnosing and resolving these problems.

Log View

The contents of the logs are displayed in a number of columns, each with a heading at the top. When you first open a log, the events are shown in chronological order, with the most recent at the top, but you can sort them according to any of the available criteria by clicking the header of the appropriate column.

You can filter the contents of one or more columns by entering text in the appropriate column filter, this allows you to see, for example, only those entries that relate to a specific EMF Process. For further information on filtering, see the EMF Help.

To access the Log view:

Click **Window** and then click **Log**. This displays the **Log** view pane with the following options:

- **Log Level:** The log level of the server (for system level messages) or EMF Process (for EMF Process execution messages) as defined in the EMF administrator determines the amount of information that is output.
- **Instance ID:** the instance identifier of the processed EMF Process. If an EMF Process is fired on ten separate occasions, then there have been ten EMF Process Instances, which each have a unique ID.
- **Date:** The date of the log.
- **Time:** The time of the log.
- **Message:** The message logged.
- **Module:** The module for which the log has been created.
- **Machine:** The machine name on which EMF is installed.
- **Process:** The name of the EMF Process.
- **Recipient:** Information about the message sent to a recipient (sent message logging).
- **Listener:** The name of the listener that is logging the message.

You can sort based on any of the above-mentioned columns by clicking on that column. Right-click on the log and click **Open** and view the [Log entry detail](#) screen.

[Managing the Size of the EMF Log Files](#)

[Logging in EMF](#)

Log Entry Detail Screen

The **Log entry detail** screen displays information about the server activity in the EMF system in a more convenient way. It allows you to view all the information about a single entry in the EMF system log. The information displayed is identical to that in the Log table view.

To view the Log entry detail screen:

Double-click an entry in the logging list view, or right-click it and select **Open**.

You can view other events in the list by using the up and down arrows to scroll through them.

- **Log Level:** The log level of the server (for system level messages) or EMF Process (for EMF Process execution messages) as defined in the EMF administrator determines the amount of information that is output.
- **Instance ID:** the instance identifier of the processed EMF Process. If an EMF Process is fired on ten separate occasions, then there have been ten EMF Process Instances, which each have a unique ID.
- **Date:** The date of the log.
- **Time:** The time of the log.
- **Module:** The module for which the log has been created.
- **Machine:** The machine name on which EMF is installed.
- **Name:** The name of the EMF Process.
- **Recipient:** Information about the message sent to a recipient (sent message logging).

- **Listener:** The name of the listener that is logging the message.
- **Description:** The message logged.

[System logging](#)

[Extracting and Removing Information From Logs](#)

[Managing the size of the EMF log files](#)

[Logging in EMF](#)

Managing the Size of the EMF Log Files

Use the **Log Configuration** page to manage and restrict the size of the EMF log.

To view the Log Configuration page:

Click the **Configure** button to view the **Log Configuration** properties page.

- **To delete log entries according to their age**, select the **Purge entries older than specified time** option. You can then select **Minutes**, **Hours**, or **Days** and specify the maximum age of log entries as a number of these. Note that **you must shut down the EMF server** and restart it for any changes to take effect.
- **To delete entries after the log reaches a set maximum number of entries**, select the **Purge oldest entries when max number of entries exceeded** option. When the number of entries in the database reaches the number specified in the **Max number of entries** field, the oldest entries will be deleted to make room when newer ones arrive. Note that **you must shut down the EMF server** and restart it for any changes to take effect.

Note: You can select both the **Purge entries older than specified time** and **Purge oldest entries when max number of entries exceeded** options. In this case, the log is cleared according to age and total number.

[Extracting and Removing Information From Logs](#)

[System Logging](#)

[Logging in EMF](#)

Audit Logging

Audit logging keeps a record of all changes in the EMF Administrator. It also records when processes run and finish (if enabled on an individual process). This information can be useful to gather specific information about which EMF Processes fire when, in order to assess whether they should continue to be active, and to trace the source of unexpected or malicious functioning.

To disable Audit logging:

Audit logging is enabled by default. If you want to turn off Audit Logging, clear the **Audit**

logging option on the **Audit log** screen.

The EMF system has the potential for misuse, both unintentional and malicious (for example, using Data Sources that can return large volumes of data, large numbers of outputs, etc.). Although auditing cannot prevent such occurrences, it can trace the cause of the problem and, if necessary, make an individual accountable for the changes that lead to the end result. This information can then be used to prevent future occurrences of the same problem.

To view the Audit log:

Click **Window** and then click **Audit Log** to display the Audit log table view, which displays six columns of information about all the events that have taken place since it was last cleared (for information on clearing the log, see the [Audit Log](#) screen):

- **Type** - The type of the entry.
- **Date** - The date and time of the entry.
- **Audit Event** - The description of the event.
- **Operator** - The name of the system operator that performed the action.
- **Session** - The session for the log entry.
- **ID** - The instance identifier of the processed EMF Process. If an EMF Process is fired on ten separate occasions, then there have been ten EMF Process Instances, which each have a unique ID.

You can view details of an individual event in dialog form (see the [Audit entry detail](#) screen) by double-clicking on an entry and using the up and down arrows to scroll through the events.

[Viewing details of an individual entry in the Audit log](#)

[Managing the size of the Audit log](#)

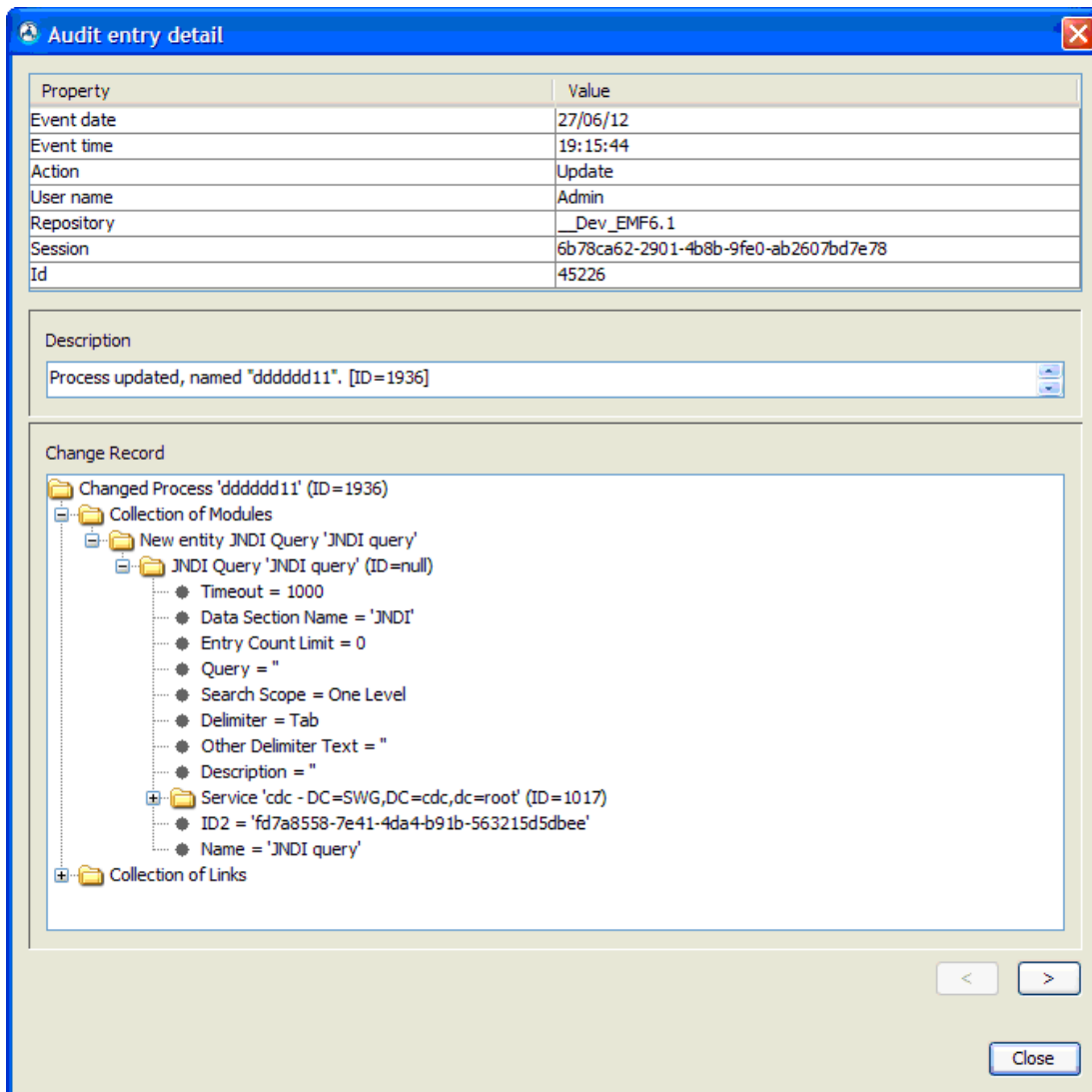
[Logging in EMF](#)

Audit Log Entry Detail Screen

The **Audit entry detail** screen provides a more convenient way to view all the information about a single entry in the Audit log. The information displayed is identical to that in the Audit log table view, with the addition of a **Repository** field.

To view the Audit entry detail screen:

Double-click an entry in the logging list view, or right-click it and select **Open**.



You can view other events in the list by using the up and down arrows to scroll through them.

- **Event Date** - the date of the entry.
- **Event Time** - the time of the entry.
- **Action** - the action performed by the operator within EMF.
- **Operator Name** - the name of the system operator that performed the action.
- **Repository** - the database that the operator was logged into.
- **Session** - The session for the log entry.
- **Id** - the Id assigned to the log entry.
- **Description** - a description of the entry.

[Overview of Audit logging](#)

[Extracting and Removing Information From Logs](#)

[Managing the size of the Audit log](#)

[Logging in EMF](#)

Managing the Size of the Audit Log Files

Use the **Audit Log Configuration** page to manage and restrict the size of the Audit log.

To view the Audit Log Configuration page:

Click the **Configure** button to view the **Audit Log Configuration** properties page.

- **To disable Audit logging:** Audit logging is enabled by default. If you want to turn off Audit Logging, clear the **Enable** option.
- Select **Full logging of all changes** to enable auditing of every change that is made to an item in EMF (For example, a change to a data source, or adding a new module to a process). When full logging is not enabled, an item change event is logged, but the exact changes to the item will not be logged. Full logging records all changes, including changes to the item. To view the changes to an item, double-click and open the appropriate audit record in the audit log. The dialog box will show the changed items in the Change Record section. If an item is created or deleted, the audit record will contain all the values of that item.

Note: Enabling full logging slows down the performance of EMF. Opening and saving items in EMF will take more time. Enabling full logging also increases the size of the EMF repository database considerably, which can be controlled by ensuring that old records are purged regularly using the settings shown below. If you do not require auditing full changes, it is recommended to disable the option.

Audit Log Configuration

Log Auditable Events

☒ Enable

☒ Full logging of all changes

Log Management (performed by the server)

☒ Purge entries older than specified time

Time days

☐ Purge oldest entries when max number of entries exceeded

Max number of entries

OK Apply Cancel Help

- **To delete log entries according to their age**, select the **Purge entries older than specified time** option. You can then select **Minutes**, **Hours** or **Days** and specify the maximum age of log entries as a number of these. Note that **you must shut down the EMF server** and restart it for any changes to take effect.
- **To delete entries after the log reaches a set maximum number of entries**, select the **Purge oldest entries when max number of entries exceeded** option. When the number of entries in the database reaches the number specified in the **Max number of entries** field the oldest entries will be deleted to make room when newer ones arrive. Note that **you must shut down the EMF server** and restart it for any changes to take effect.

Note: You can select both the **Purge entries older than specified time** and **Purge oldest entries when max number of entries exceeded** options. In this case, the log is cleared according to age and total number.

[Overview of Audit logging](#)

[Viewing details of an individual entry in the Audit log](#)

[Extracting and Removing Information From Logs](#)

[Logging in EMF](#)

Extracting and Removing Information From Logs

You can use the **Log Extractor** to export or delete information from any of the logging components within the EMF System. You can define start and end times and dates for the log extraction process, and the relevant entries can either be exported to a text file or deleted entirely.

To extract or delete logged entries:

1. Click **Windows**, and then click **Audit Log** or **Log**.
2. Click **Export**. This displays the **Export to MS Excel format** dialog box.
3. If you want to export the information to a file, enter a location and file name for the file (or use the **Browse** button to select a location).
4. Click **Export** to save their contents of entries within your selected range to the file specified, or **Delete** to remove them from the log. Exporting log entries saves their content to a file and leaves the original entries intact.

Note: Before deleting log entry details, you might like to export them first to ensure the information is saved for future reference.

[Introducing EMF](#)

[Logging in EMF](#)



3

Building your EMF Processes

Overview

The Help topics in this section will assist you in [building and managing your EMF processes](#). They describe how to create, modify, debug, store, and manage your EMF Processes.

These topics also provide instructions on [importing and exporting EMF Processes](#) and give advice on organizing your EMF Processes in [folders](#).

They also describe the different types of [modules](#) you can use when building an EMF Process or template.

Creating and Managing EMF Processes

This section of the Help describes how to create, modify, debug, store and manage your EMF Processes. It includes information on how to use the EMF Process builder view with a brief overview of each of the modules and how to place and link them. It also includes instructions for importing and exporting EMF Processes and using templates, and gives advice on organizing your EMF Processes in folders.

Creating EMF Processes

Note: in order to create any EMF Processes, you need to have [security](#) and [licensing](#) rights for the **Start** module. This is in addition to security and licensing required for any other EMF Process Builder modules you want to use.

You design, validate, save, schedule and run your EMF Processes using the **EMF Process Builder View**, which opens automatically when you create a new EMF Process or double-click an existing one in the EMF Process folders (see below).

The central element of the Builder view is the **EMF Process Design Graph**, which is where you design the actual layout and flow of the EMF Process. You do this by dragging icons representing the various modules from the surrounding toolbars and creating links between them to create a layout in the form of a flow-chart. You can then modify the properties and behavior of the modules and links by right-clicking them to access their **Properties** pages.

For further information see the [Guide to Building EMF Processes](#).

Storing and managing EMF Processes

You store your EMF Processes in folders, accessed by opening **EMF > EMF Process folders** in the EMF Tree view. When you click on an EMF Process folder, the **EMF Processes View** screen opens and displays all the EMF Processes in the selected folder. There are two default system folders, templates and system, and you should create further folders of your own in which to store the EMF Processes that you create.

For further information see [Storing and Managing Your EMF Processes](#).

Debugging EMF Processes

Occasionally you may find that an EMF Process does not run as intended, or that there are problems with the data gathering or output. In such cases you can try to debug the existing EMF Process to determine where the problem lies.

For further information see [Debugging EMF Processes](#).

Importing and exporting EMF Processes

Normally, when you create an EMF Process, you will always run it from the same repository database. However there may be times when you need to copy or move an EMF Process (or other repository information) to a different database, or share an EMF Process with another installation, and you can do this by exporting the EMF Process as an XML file and importing it into a different repository.

For further information see [Importing and Exporting EMF Processes](#).

Building EMF Processes

You build your EMF Processes using the [EMF Process Builder](#) view that is shown whenever you create a new EMF Process or double-click an existing one in your EMF Process folders.

To create a new EMF Process:

1. In the EMF Tree view, right-click the [folder](#) in which you wish to place the EMF Process and select **New EMF Process**. The [Create new EMF Process](#) screen opens, requiring you to assign a name to the EMF Process.
2. Enter the **Name** and, optionally, a **Description** (recommended for all systems where there are a large number of EMF Processes) and click **OK** to open the Builder View.

The central area of the EMF Process Builder view is the [EMF Process Design Graph](#), which is a customizable graphic design screen. To the right side of the central area is the palette containing icons that represent the various [modules](#) which you can use when creating your EMF Processes. You drag these icons onto the Design Graph and then arrange and link them (like a flow chart) so that they are processed in the required order.

Note: The modules and functionality that you will have access to when designing or modifying EMF Processes will depend on your access rights, as defined on the [Security](#) tab of the **System > Recipient Administration > Groups** screen.

To create EMF Processes, you must have security and [licensing](#) rights for the **Start** module, as well as appropriate rights for any other modules you want to use.

All modules have a number of customizable parameters (or **Properties**), and you access these by double-clicking on the module icons after they have been placed in the Design Graph. You can also modify [links between the modules](#) so that they perform various filtering operations - again, double-click the links to access their Properties pages.

When you are happy with the design of your EMF Process, you must [validate](#) it to ensure that it does not contain any errors, and if there are any errors you should fix these - for example, by [debugging](#) it. Finally, you must [set its status](#) to active before you can [run](#) it.

Designing EMF Processes

You design and build your EMF Processes using the [EMF Process Builder](#) view that is shown whenever you create a new EMF Process or double-click an existing one in your EMF Process folders.

To create and design a new EMF Process:

1. In the EMF Tree view, right-click the [folder](#) in which you want to place the EMF Process and select **New EMF Process**. The [Create new EMF Process](#) screen opens, requiring you to assign a name to the EMF Process.
2. Enter the **Name** and, optionally, a **Description** (recommended for all systems where there are a large number of EMF Processes) and click **OK** to open the EMF Process Builder view.
3. Click the palette icons for the [modules](#) that you want to use and drag them onto the Design Graph. The modules that you choose to include will depend on the intended function and recipients who will receive the EMF Process, but every EMF Process must have at least one each of the following:
 - An **Initiator** Module that tells the EMF Process when to run.
 - A **Start** module that forms the link between the initiator and processing modules.
 - A **Processing** Module that tells the EMF Process what to do.
 - An **Output** Module that defines what to do with the resulting information.

Note: In most cases, you can have as many of each type as you want, but you can (and must) only have one Start module.

4. Double-click the module icons to open their **Properties** pages and modify the settings as required (you can also open the Properties page by right-clicking the module icon and selecting **Properties**).

Note: The **Start** module, and certain other modules, have no settings that you can change and hence no Properties pages.

5. After all the modules have been added, select the **Save** icon from the tool palette to save the EMF Process. The EMF Process Designer will [validate](#) the EMF Process and display any validation messages, including error and informational messages in the message area.
6. When you are ready to run the EMF Process, right-click in an empty area of the design graph and click Properties to view the EMF Process [Properties](#) pages, and then set its **Status** to **Active**.

7. You can add comments to the process. Right-click in the **Process Designer** and click **Add comment**. This displays a comment pop-up in the designer, where you can add comments to the process.

EMF Process status

When you first create an EMF Process, its **Status** is incomplete, but before you can run it, you must set its status to Active. The available options are:

- **Active** - The EMF Process is activated and ready to run or running. You should not select this state unless the EMF Process can be considered finished and working.
- **Hold** - The EMF Process is ready to run but is currently suspended. Select this state if you are making changes and do not want the EMF Process to be fired while the changes are in progress.
- **Incomplete** - The EMF Process is not yet ready to run. This is the default state assigned to an EMF Process when it is first created.

[Guide to Building EMF Processes](#)

[The EMF Process Builder View](#)

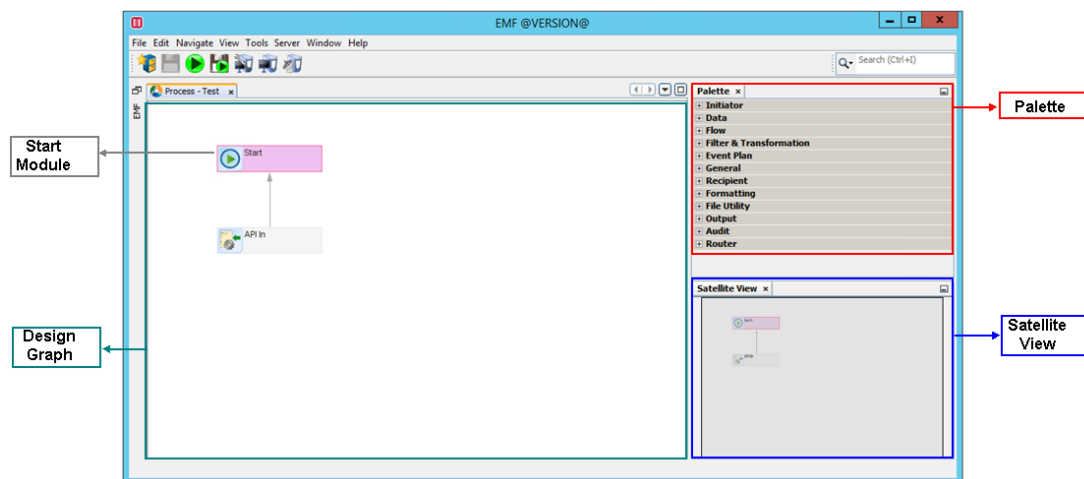
[The EMF Process Design Graph](#)

[Saving and Running EMF Processes](#)

[Managing EMF Processes](#)

Using The EMF Process Builder View

You design and build your EMF Processes in the **EMF Process Builder View**, which opens automatically whenever you create a new EMF Process or double-click an existing one in any of the EMF Process folders.



The central area of the EMF Process Builder view is the EMF Process Design Graph, which is a customizable graphic design screen. To the right side of the central area is the palette containing icons that represent the various modules, which you can use when creating your EMF Processes. You drag these icons onto the Design Graph and then arrange and link them (like a flow chart) so that they are processed in the required order. Below the palette is the satellite view which gives you a thumbnail view of the process you are currently designing.

The EMF Process Design Graph contains, at a bare minimum, a [Start Module](#). The [Start Module](#) forms the link between the initiator and processing modules. It must be present in all EMF processes, and as a result, is automatically added on the design graph whenever a new EMF Process is created. The [Start Module](#) is also where the [Process Interface](#) is defined. Opening up the [Start Module](#) properties displays the process interface setting.

As you drag other module icons onto the Design Graph, [links](#) are automatically formed between the new modules and their adjacent modules, thus showing the processing flow of the EMF Process. You can delete and move these links, and modify their behavior and properties.

The **Message area** at the bottom displays information about what has been loaded when the Design Graph opened as well as any messages about the currently open EMF Process such as validation status informational messages, debugging information and any errors that occur.

Note: Hold down the Ctrl key and move the mouse wheel to zoom in/out the Process builder view.

[The EMF Process Design Graph](#)

[The EMF Process Builder Modules](#)

[Linking EMF Process Builder Modules](#)

[How to Create and Design an EMF Process](#)

Adding Modules to the Design Graph

You create and modify EMF processes by dragging icons representing the various modules from the [EMF Process Builder](#) toolbars onto the **EMF Process Design Graph**, the central element of the builder where EMF processes are laid out and designed visually.

Note: There may be more module icons than can fit on the available toolbar space. If some of the expected modules do not seem to be present, they may be hidden - try maximizing the EMF process builder so that it occupies all of your display. If you still cannot see them all you should detach the dockable toolbar and drag it somewhere else, then resize it appropriately.

There are four types of module that you can use in an EMF process (for further details see [The EMF Process Builder Modules](#)):

- The **Start** module acts as a bridge between the initiator modules and the processing and output modules.

- **Initiator Modules** are the modules that specify when and why an EMF process should run. You can place as many Initiator modules in an EMF process as you wish, as long as they all point to the Start module.
- **Processing modules** include all modules which function to retrieve, update and process data, maintain flow control, define recipients, and format messages.
- **Output modules** send data to devices or files.

To add a module to an EMF Process:

Drag the icon for the appropriate module from the surrounding toolbars to the appropriate place in the EMF process sequence. When you release the icon, the Builder automatically creates a link to it from the nearest preceding module (for more information on adding and removing links, see [Linking EMF Process Builder Modules](#)).

Each module has a name and, optionally, description. The **Name** is displayed under its icon while the **Description** (if available) is displayed above it.

To name or describe a module:

Right-click on the module icon and click **Name** or **Description**. You can then enter the required name or description in the **Text** field.

To modify the behavior and properties of a module:

Double-click on the relevant icon to bring up its **Properties** screen. Each of these is different, and you can find more information about specific module properties in the Help section for those modules.

To move or delete modules:

- **To move a module:** Click on the module icon to select it, then hold the mouse button down and drag it to a new position.
- **To delete a module:** Select the icon and press **Del** or right click and select delete from the drop down menu. All links to the selected icon are also deleted.

Important: To delete a module from an EMF process that has been saved to the database, you must have delete [security rights](#) for that module (if the EMF process has not yet been saved, you do not need these rights).

To select multiple modules:

Select the **Multiple Modules** tool icon from the tool palette at the top of the design graph. You can then select more than one icon either by holding down the **Ctrl** key while selecting the modules, or by holding the mouse button down and drawing a box around the required icons. All items within that box will be selected, including any items that are only partially within the box.

[How to Create and Design an EMF Process](#)

[Managing EMF Processes](#)

Linking EMF Process Builder Modules

The EMF Process **Design Graph** is the area where EMF Processes are laid out and designed visually. When you drag a new module onto the graph, an appropriate link is automatically created to it from the nearest previous module. You can delete, modify and create new links as you wish.

You can apply **conditions** to links (e.g. if and when not to follow that link) using [Conditional Links](#) and prevent certain information from being sent to subsequent modules by [removing information from EMF Process threads](#). You can also use [Waypoints](#) to control the links routing allowing better manageability and presentation.

Creating links

As mentioned above, each module that you add to your graph is automatically linked to the nearest preceding module. This link is represented by a line with an arrow at one end, thus showing the processing flow of the EMF Process.

To manually create a link:

1. With default configuration settings, right-click (and hold the button down) on the source module.
2. Drag the mouse over the destination module and release the button to create the link.

Note: The mouse button action can be configured to the left button instead in the **Tools->Options->Process Builder Behaviour** tab.

To delete a link:

Select the link, and press **Del** (or right-click and select **Delete**).

To modify the behavior of a link:

Click on a link to select it, then right-click and click **Edit properties**. For more information, see [Conditional Links](#).

[Copying Settings Between Links](#)

[EMF Process Builder Modules](#)

[How to Design an EMF Process](#)

[The EMF Process Builder View](#)





[Using the EMF Process Design Graph](#)

[Managing EMF Processes](#)

Applying Conditions to Links

A link will default to always executing in the server. However, it is possible to set conditions on a link that will control whether it executes or not: a link can be set to run when an expression evaluates to true. Links can also be set to run as a "default" type - when all links using conditions have not run, then links with the default condition set are run.

To set a link condition:

1. Double-click on the link that you want to modify, to display the **Link Conditions screen**.
2. Select from the available **Link Options**:
 - **Run always** - The link will always execute. This is the default type used in EMF, and is shown on the builder graph by a solid line 
 - **Run on condition** - If this option is selected, the link will only run if the expression evaluates to true. These links are shown as dotted lines 
 - **Run when all other links using conditions have not run** - If this option is selected, the link will run only if all other links with conditions set have not run. These links are shown with alternate dots and dashes 
 - This option can be used to act as a default or exception handler. This path will only be taken if all other links using Run on condition evaluate to false.
 - **Error Link** - This link is run when a module fails to process and the server normally reports an error for the same. It allows the process to continue processing down the branch and corrective action to be taken. For example, e-mail the concerned individual that a database connection is down. 
3. Select the queue (defined in the System Queues) from the **Queue to run on** drop-down menu.

The link will always run on the default queue, unless specified. Once a process is placed on another queue, all further processing of that branch will stay on the new queue until the process branches again (it will then be placed in the work in progress queue defined by the server manager that is running the branch) or a different queue is specified on another link. By specifying different queues, the process can be better load balanced and the flow controlled. If a different queue is specified on a link between the initiator module and the start module, then this will control the queue that the process is initially run on.

Note: The queue must have a [server manager](#) associated with it, else the process placed on the queue will not be run.

4. Enter the first expression in the **Evaluate this expression** box. This will be evaluated against the expression in the **Against this expression** box (see below). To insert a [dynamic function](#), right-click in the **Evaluate this expression** area and select the required function.
5. Select the **Trim Spaces** check box if you want to remove character spaces leading or trailing a value in tabulated data.

6. Select the required operator from the **Using this operator** list. The options available will depend on whether String or Numeric is chosen in the **Perform Evaluation as** field (see above).

If a **Numeric** comparison is selected, the following comparison operators are available:

- **Equal to (=)** - Evaluates whether the first expression is equal to the second expression.
- **Less than (<)** - Evaluates whether the first expression is less than the second expression.
- **Greater than (>)** - Evaluates whether the first expression is greater than the second expression.
- **Less than or equal to (<=)** - Evaluates whether the first expression is less than or equal to the second expression.
- **Greater than or equal to (>=)** - Evaluates whether the first expression is greater than or equal to the second expression.
- **Not equal to (<>)** - Evaluates whether the first expression is not equal to the second expression.

If a **String** comparison is selected, the following operators are available:

- **Contains** - Evaluates whether the first expression contains the second expression.
 - **Does not contain** - Evaluates whether the first expression does not contain the second expression.
7. Enter the second expression in the **Against this expression** box. This will be evaluated against the expression in the **Evaluate this expression** box (see above) using the selected operator. To insert a [dynamic function](#), right-click in the **Against this expression** area and select the required function. Select the **Trim Spaces** option if you want to remove character spaces leading or trailing a value in tabulated data.
8. If you need to apply the same settings to other links, see [Copying Settings Between Links](#).

[The Remove tab](#)

[Example of Link Conditions](#)

[Removing information from EMF Process threads](#)

[Go to Start of Process Builder Modules](#)

Removing Information From EMF Process Threads

Normally, each link in an EMF Process passes on all the information that it receives. However it may be that certain information, once processed by one or more modules, is not required further along in the thread. In this case you may be able to improve the speed and efficiency of the EMF Process by removing it.

To remove redundant information from an EMF Process thread:

1. Right-click the link that immediately follows the last module where the redundant information was needed and click **Conditions**.
2. Select the **Remove** tab to display the **Remove** screen.
3. The **Data** tab displays a list of all the data sections that would normally be transferred along the selected link. To remove a data section from the thread, and prevent further processing, select it in the **Available data sections** list and click the **>** button. Alternatively click the **>>** button to remove all data sections from the thread. Note that some information may be generated programmatically and so will not appear in the list of sections; in this case you can enter its name in the **Other Data Section** box and then add it.
4. You can click the **<** and **<<** buttons to replace sections that you may have inadvertently removed by mistake.
5. Select the **Messages** tab and repeat the above step for any unnecessary message sections, if required.
6. The **Recipients** and **Errors** tabs allow you to remove all existing recipients or error messages from the thread. Note that subsequent modules may add more recipients, or generate errors.
7. If you need to apply the same settings to other links, see [Copying Settings Between Links](#).

[Link Conditions](#)

Waypoints

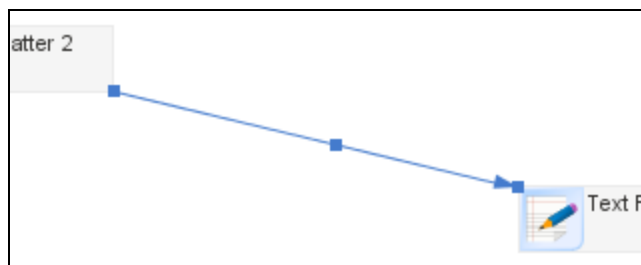
Waypoints provide more control of the links routing allowing better manageability and presentation. When two or more links are added between the same modules (for example, a loop module to another module and back again), waypoints will automatically be added to the two links (if none already exist). The waypoints will be offset from each other so that they do not overlap, allowing the two links to be visible and selectable at the same time.

Note: You can configure the size of the waypoint in the [Options](#) dialog box.

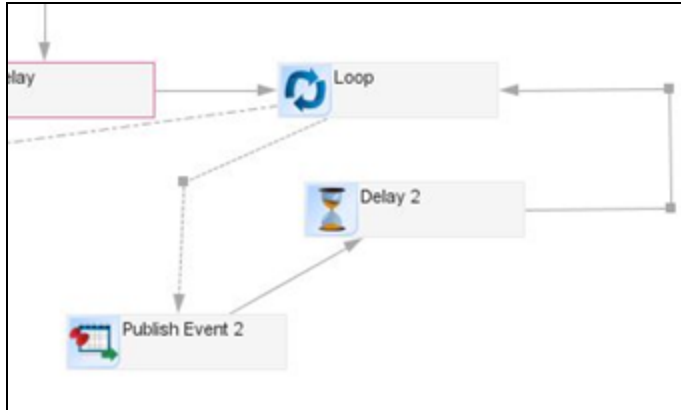
To add a waypoint

- Right-click the link at the position where you want to add the waypoint and select **Add Waypoint** option.

This adds a waypoint at the closet point in the link that you clicked, as shown.



Multiple waypoints can also be added in a single link, as shown.



To remove a waypoint

Note: Right-click on the empty space in the process builder to ensure that the link is not selected.

- Right-click on the waypoints to be removed (only the waypoints should be highlighted, not the link) and select **Delete Waypoint(s)** option.

To remove all the waypoints on a link

- Right-click the link and select **Delete all waypoints** option.

To move a waypoint

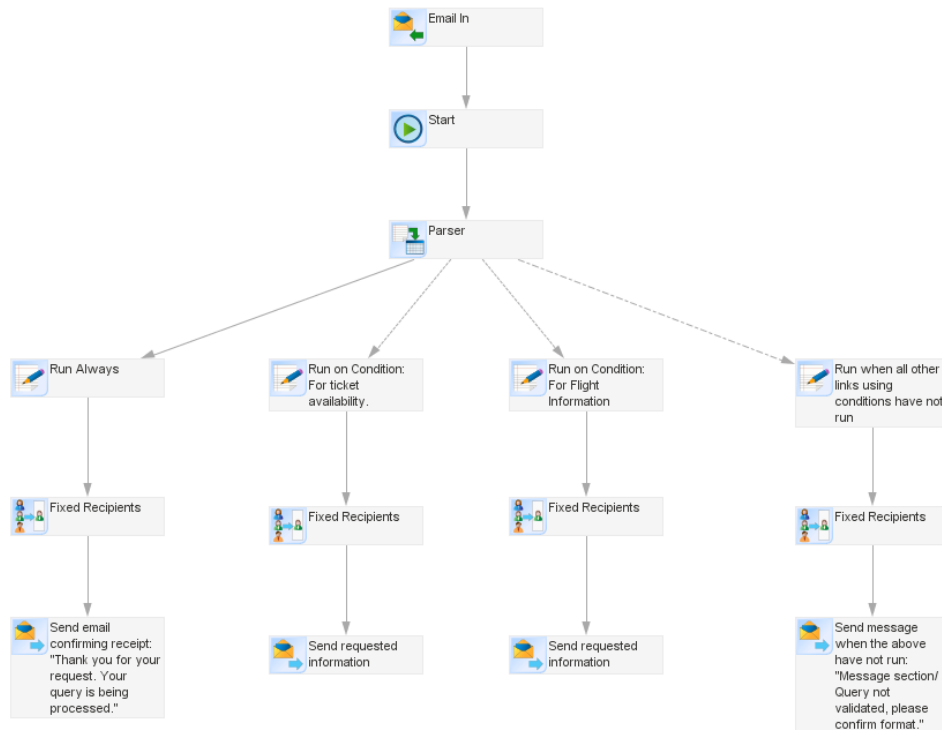
- Select the waypoint to be moved and drag to the desired position.

Example of Link Conditions

This is an example illustrating the use of Link Conditions in the EMF Process Builder.

Example

You may wish for different EMF Process branches to run depending on the content of a received Email message.



Suppose you have set up a service to provide users with interactive flight details and ticket availability for various events (among them sporting events) via Email.

You may then use one EMF Process branch for each of the following:

- All Email senders requiring a confirmation of receipt of query.
 - Link set as **Run always**
- All Email senders checking flight details, for example from Washington to Atlanta.
 - Link set as **Run on condition**
- All Email senders checking ticket availability, for example for the Harlem Globetrotters.
 - Link set as **Run on condition**
- All Email senders requiring feedback on an invalid query.
 - Link set as **Run when all other links using conditions have not run**

The additional first and last branches will cater for all cases and exceptional cases.

[Link Conditions](#)

[Go to Start of Process Builder Modules](#)

Copying Settings Between Links

If you wish, you can copy the settings of a link (appearance, font, name, tooltip, [conditions](#), and "[remove](#)" settings) and paste them onto one or more other links. This is particularly useful if you have set up complex conditions and need to use the same settings elsewhere in your EMF Process.

To copy settings between links:

1. Select the link that contains the settings that you want to copy by left-clicking on it.
2. Right-click the link and select **Copy link properties**.
3. Left-click to select the link that you want to paste the settings onto.
4. Right-click the destination link and select **Paste link properties**.
5. If required, repeat steps **3** and **4** to paste the settings onto other links.

[Linking EMF Process Builder Modules](#)

[Applying Conditions to Links](#)

[Removing Information From EMF Process Threads](#)

Options

From the Tools menu, select **Options**. There are three options in this menu:

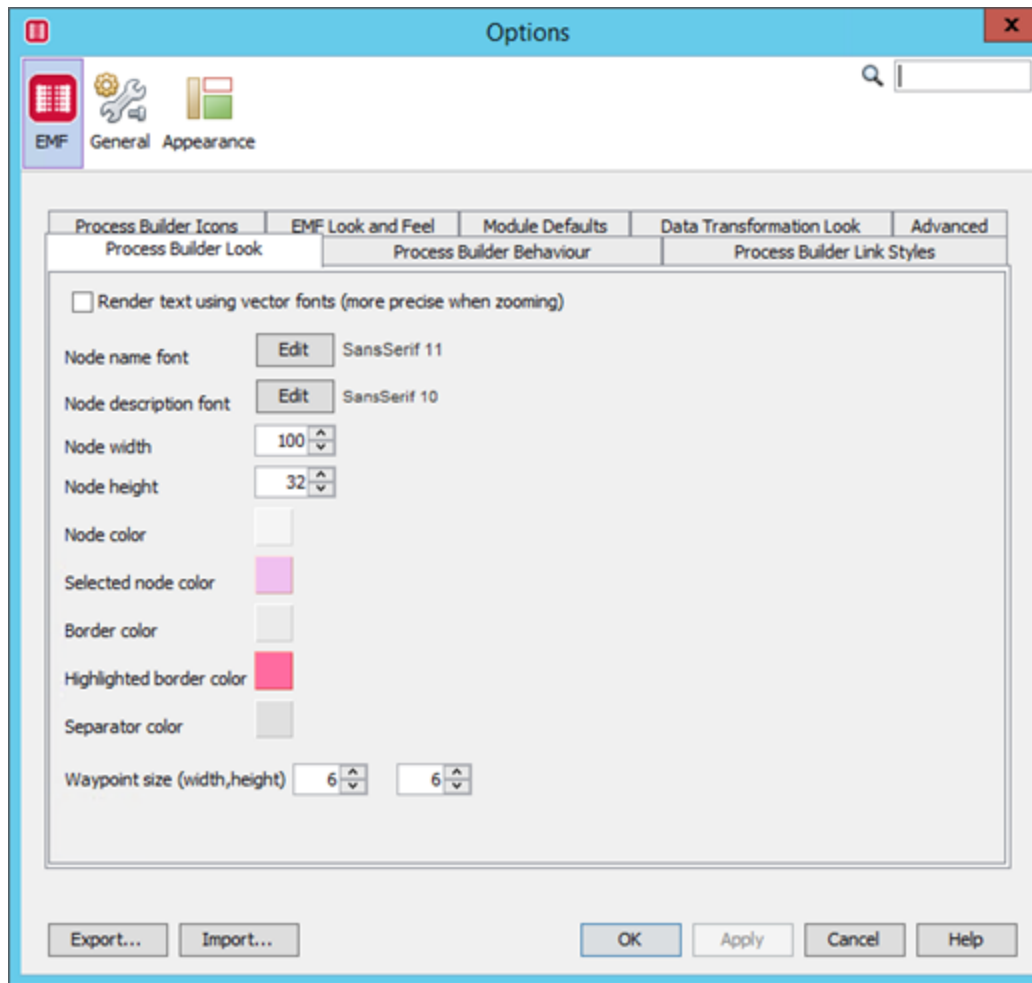
- [EMF](#)
- [General](#)
- [Appearance](#)

EMF

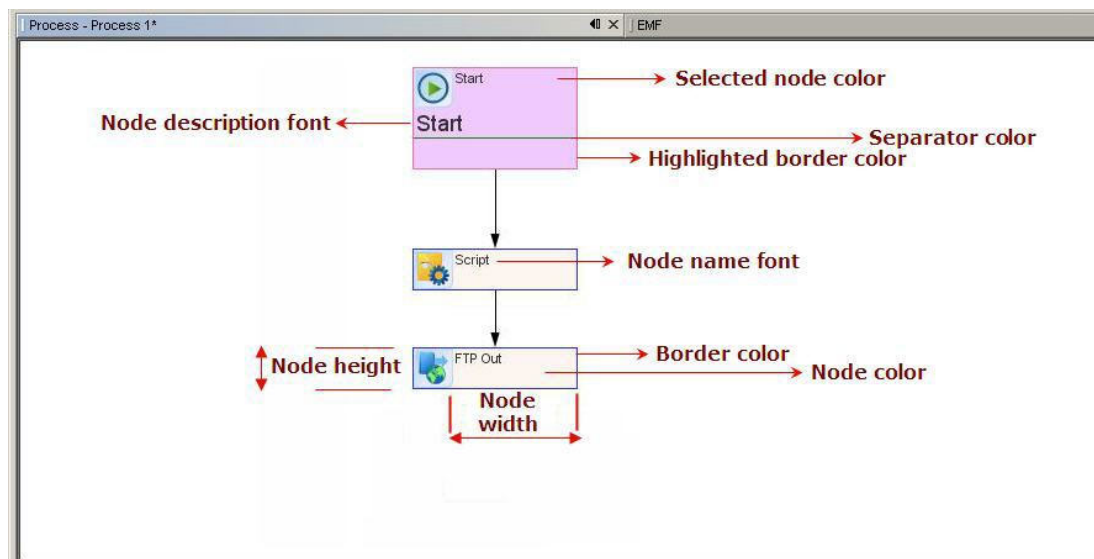
Use the EMF option (menu) to configure the way EMF nodes, icons, and processes look.

Process Builder Look

The **Process Builder Look** tab allows you to configure fonts, size, and colors of the nodes in the process builder.

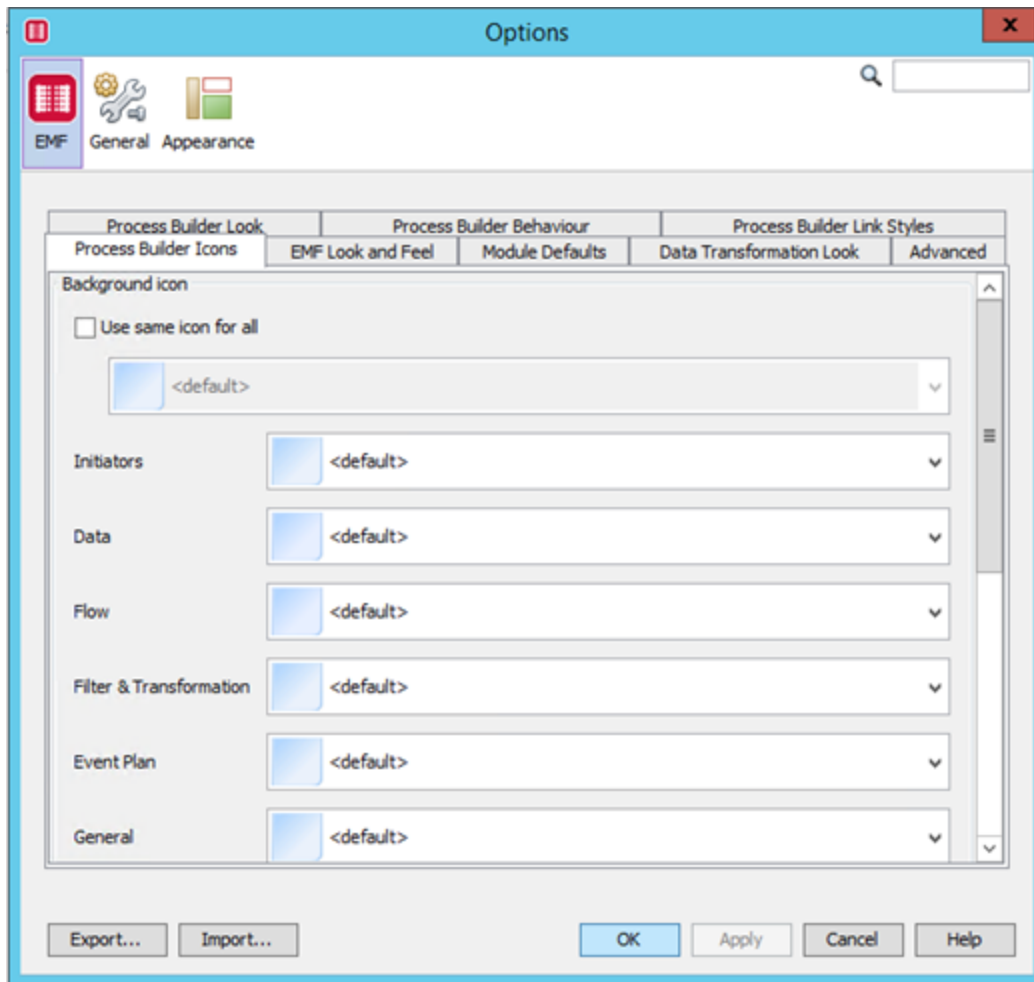


The options are indicated in the following sample process:

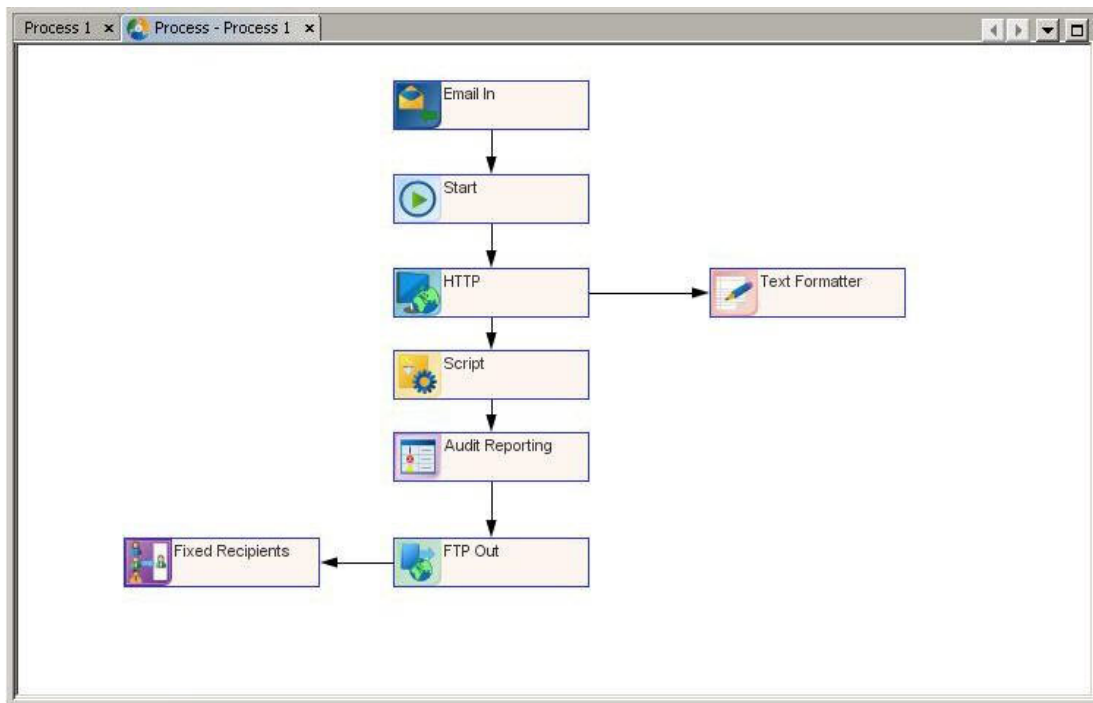


Process Builder Icons

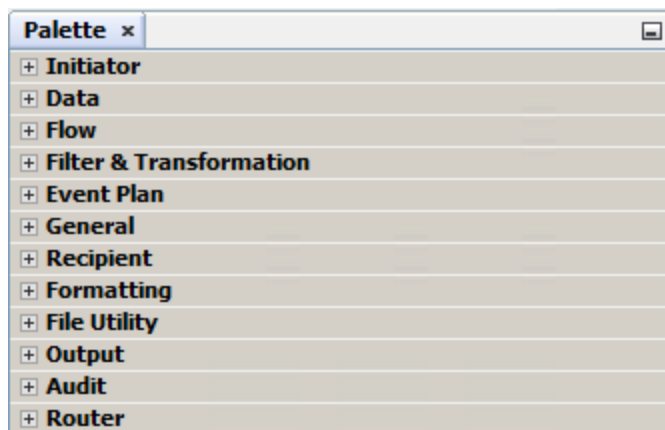
This tab allows you to choose icon backgrounds. Select the **Use same icon for all** check box to use the same background icon for all components.



A sample process with different background icons is shown below:

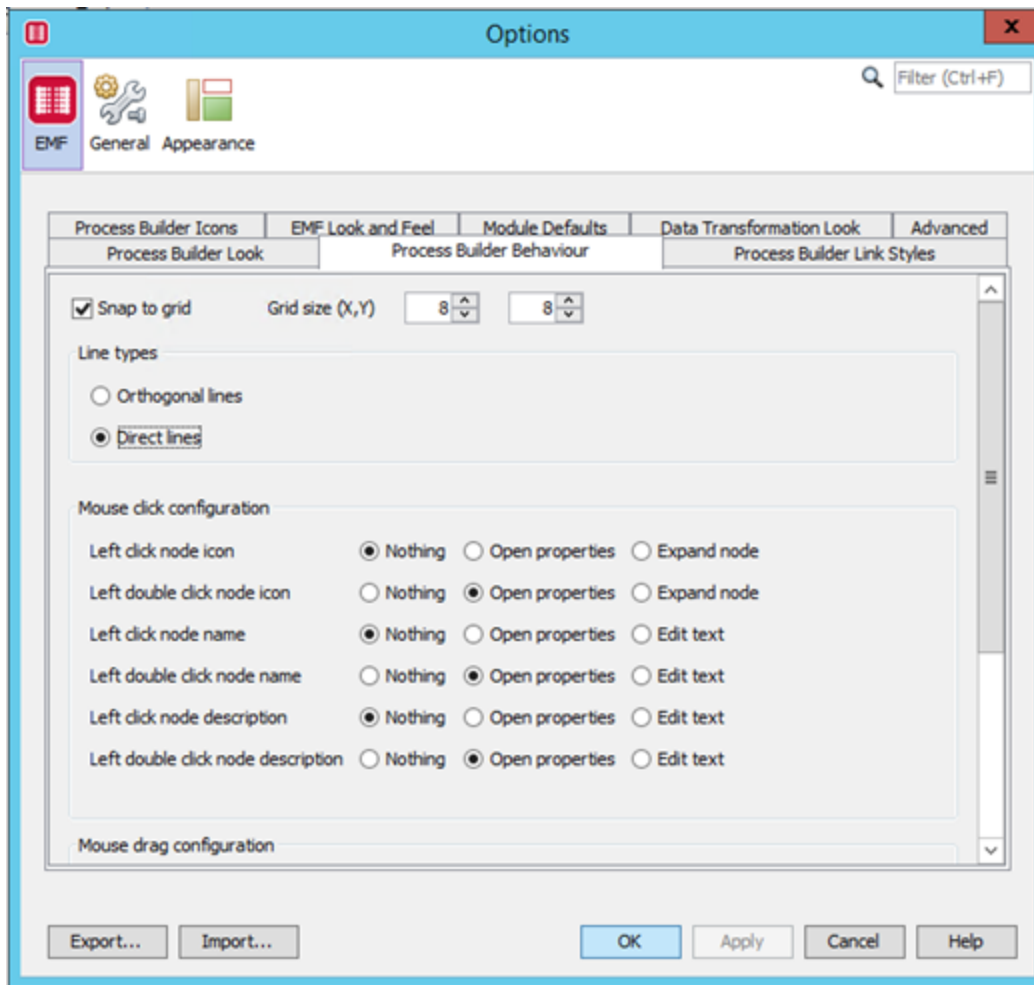


The components are divided into groups in the Palette as shown below:

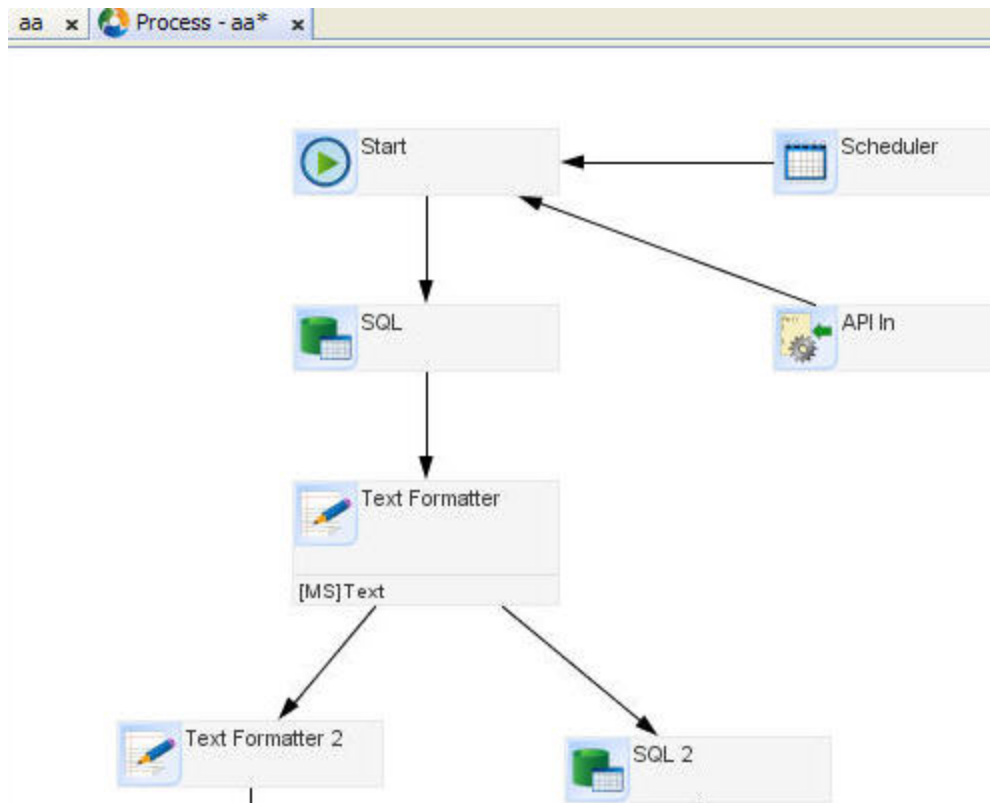


Process Builder Behaviour tab.

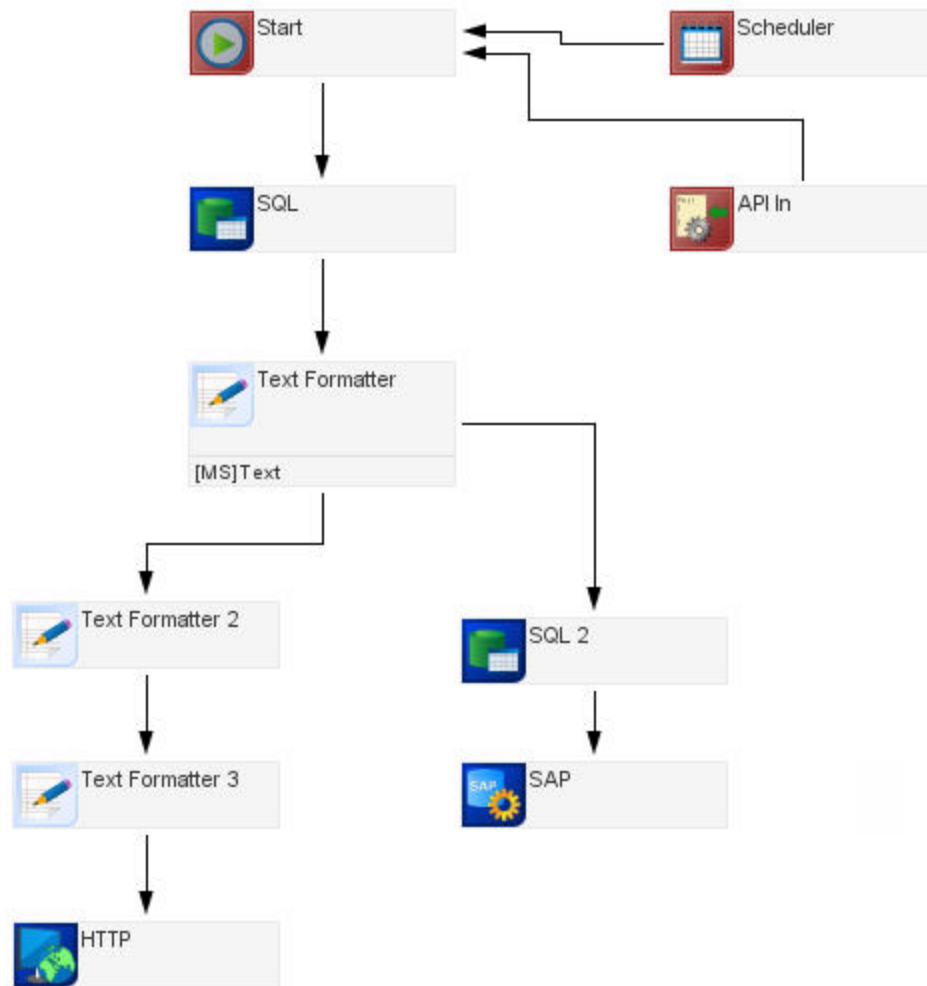
This tab has options to increase or decrease grid size, choose line types and select the snapshot background color.



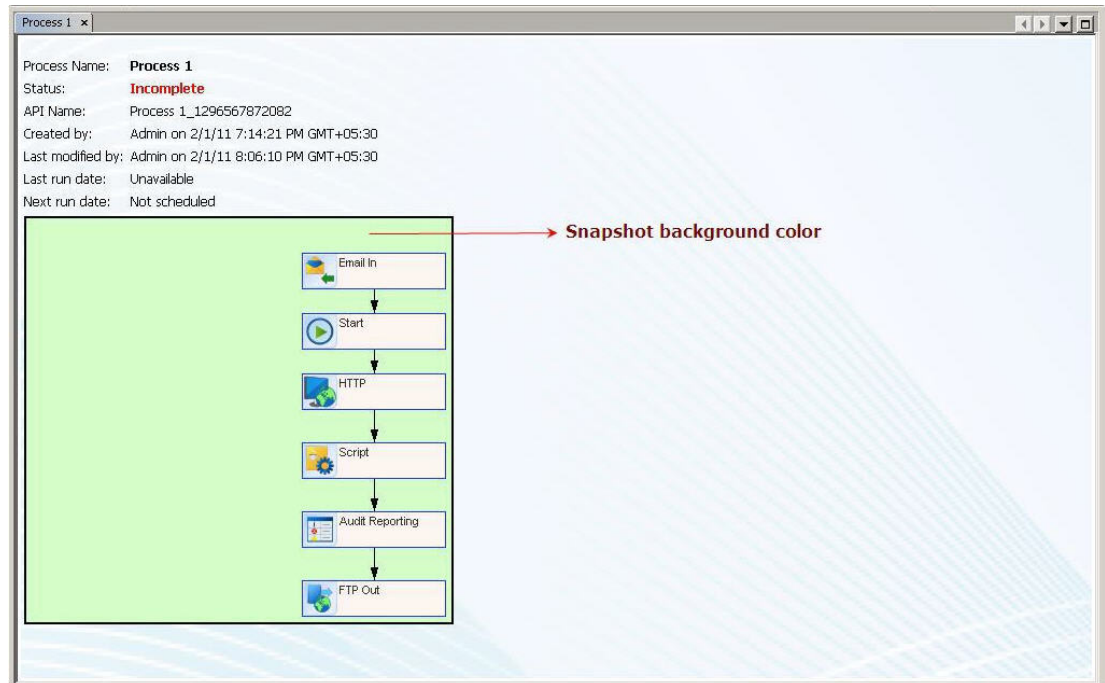
1. Select **Direct Lines** for the arrows to be shown in straight lines, as shown below:



2. If you select Orthogonal lines, the process looks like this:



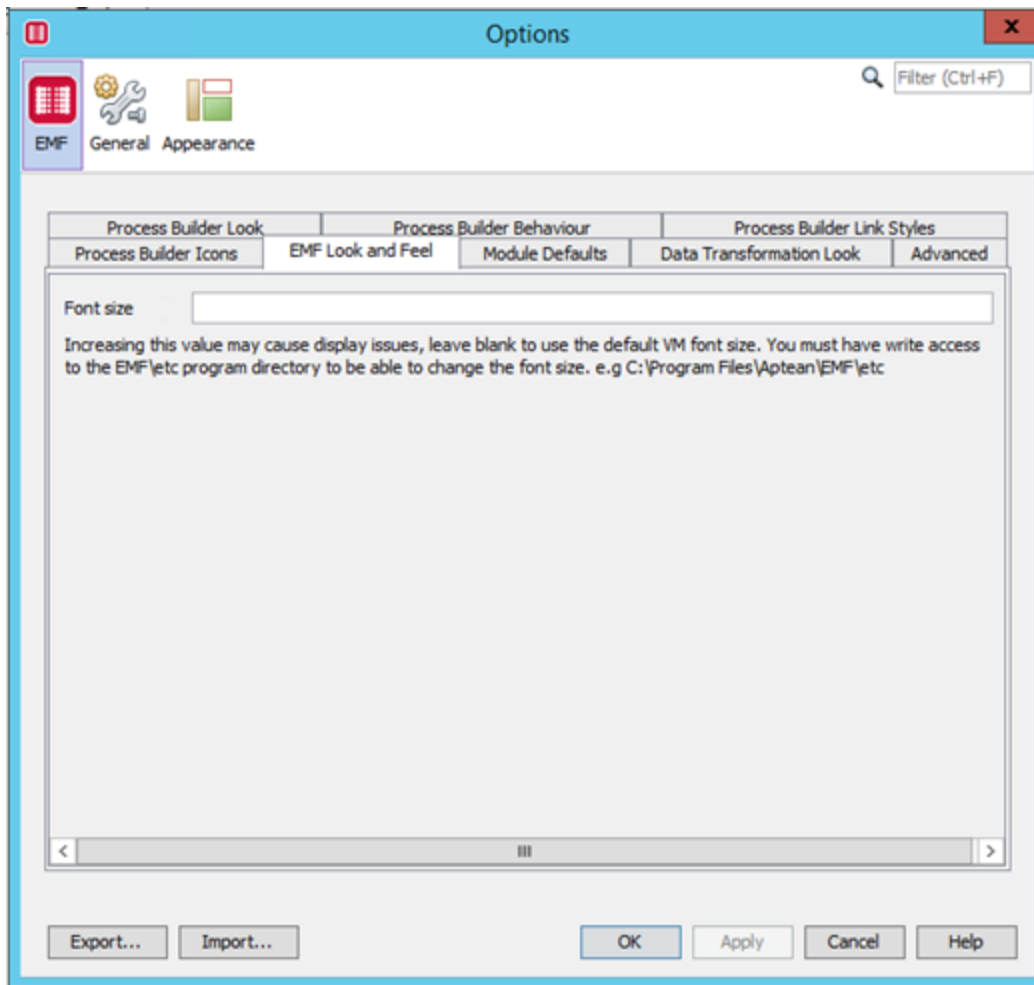
3. Click on the up and down arrows to increase or decrease the grid size.
4. Every time you save a process, a snap shot is created and this can viewed in the detail tab. The background color of this snapshot can be changed and can be set by using the option available on this tab.



5. To configure various mouse actions on the nodes, select the appropriate option from the **Mouse click configuration** and **Mouse drag configuration** sections.

EMF Look and Feel

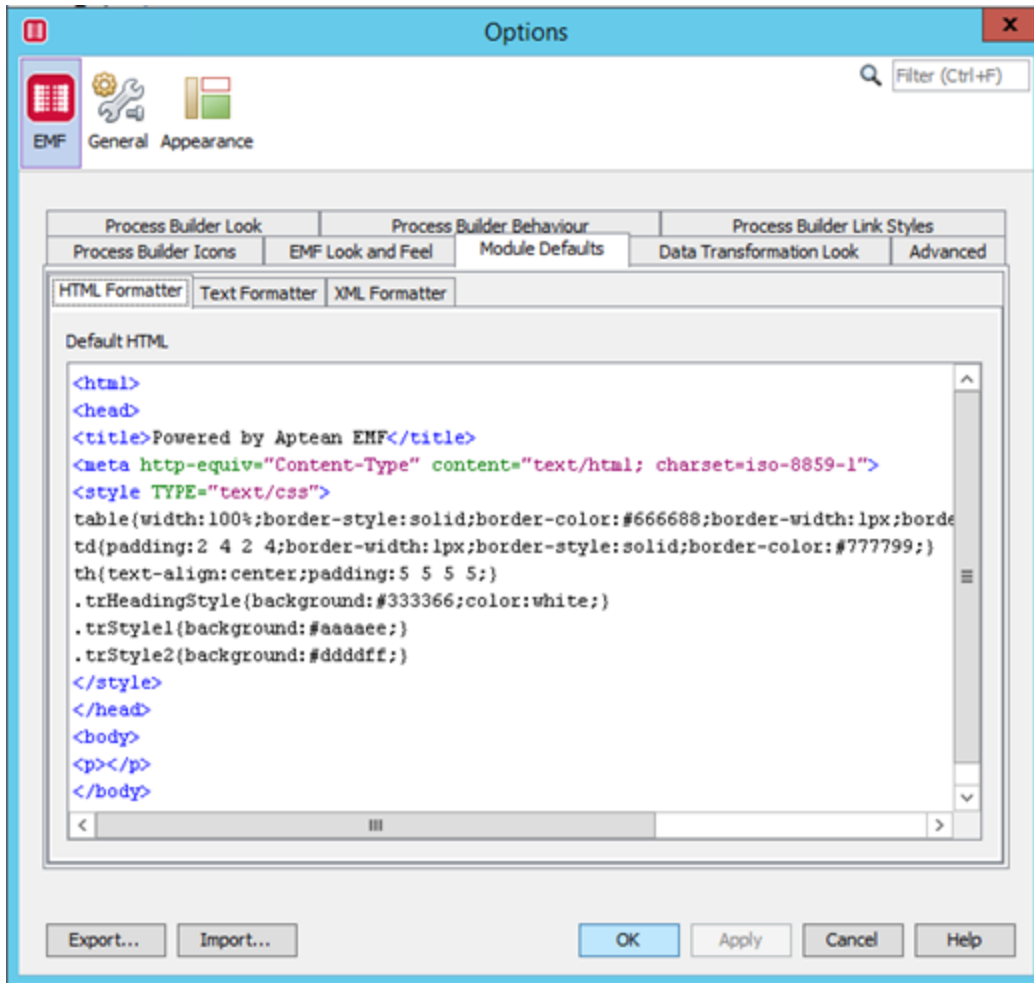
This tab allows you to select the font size.



To increase the font size, enter a value in the Font size text box. However, it is recommended that you use the default values.

Module Defaults

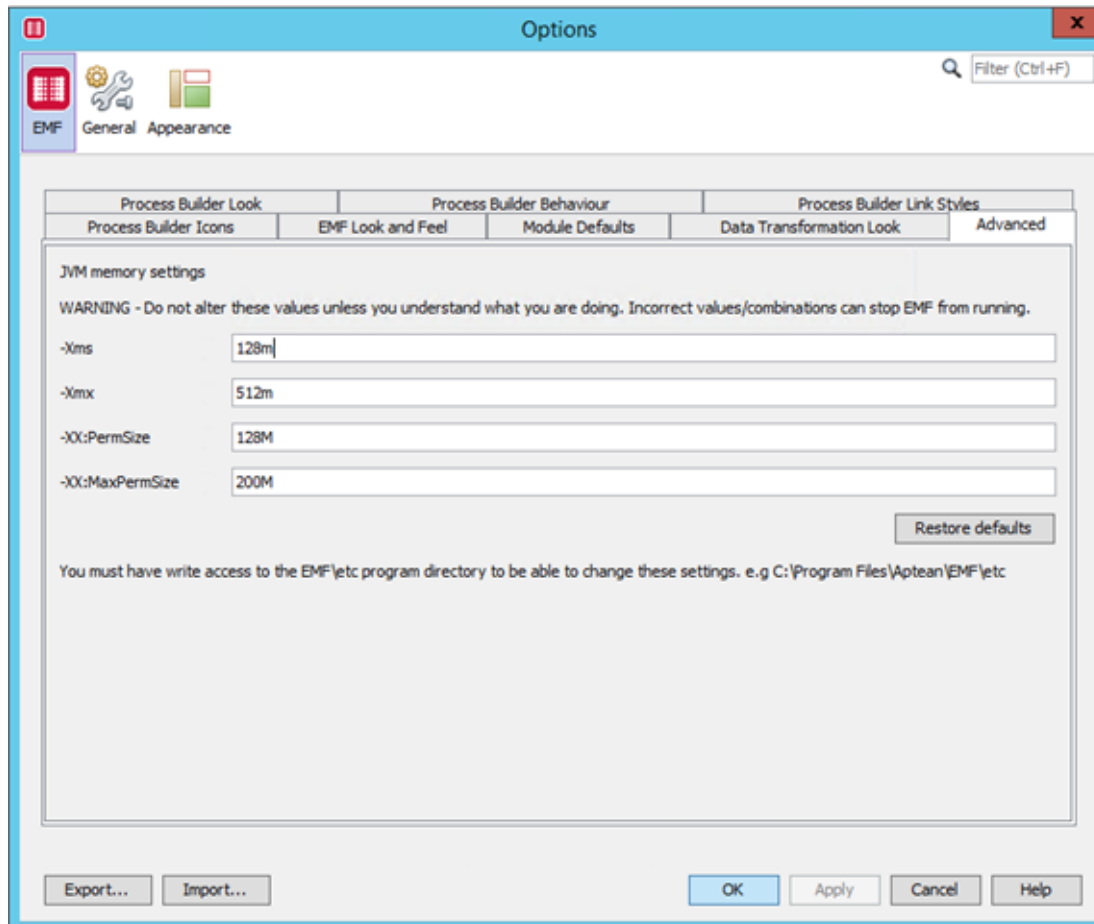
This tab allows you to specify the default content for some Formatting modules.



1. Select the HTML Formatter tab to specify default content for the HTML Formatter module. Enter HTML code in the Default HTML section.
2. Select the Text Formatter tab to specify default content for the Text Formatter module. Enter text in the Default Text section.
3. Select the XML Formatter tab to specify default content for the XML Formatter module. Enter XML code in the Default XML section.
4. Click **OK** to save and close the window.

Advanced

This tab allows you to change the JVM memory settings. You can achieve better performance by configuring these parameters.

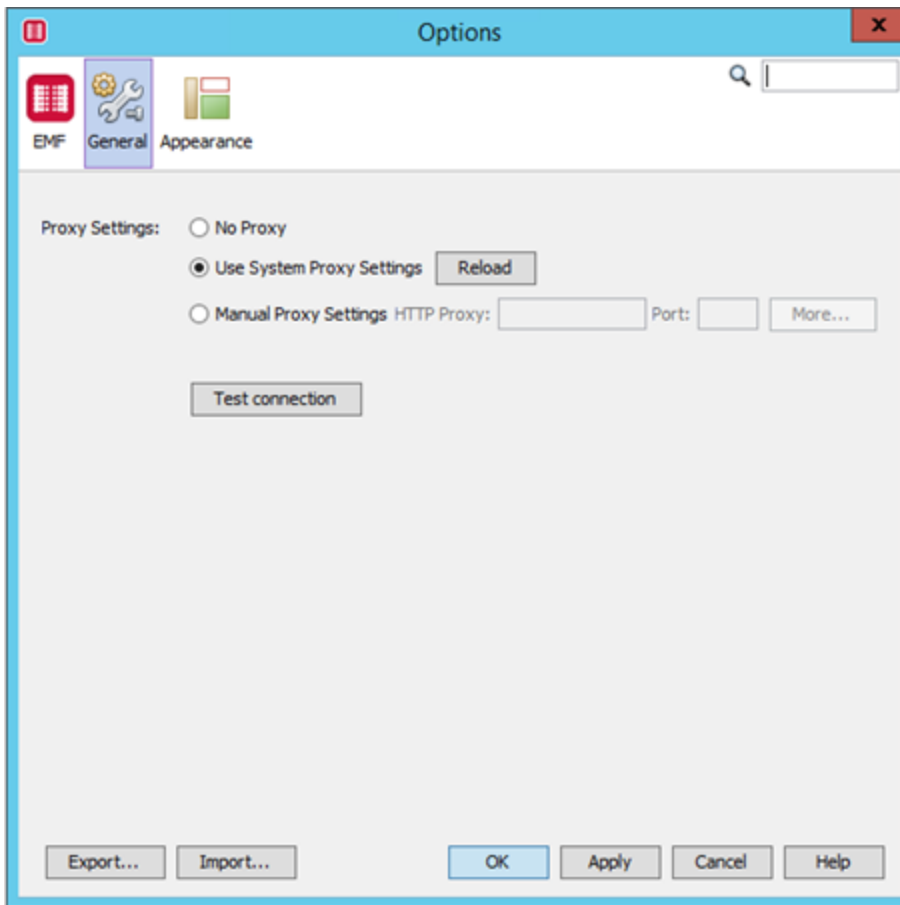


Important: Do not change these values unless you are sure. Incorrect values can stop EMF.

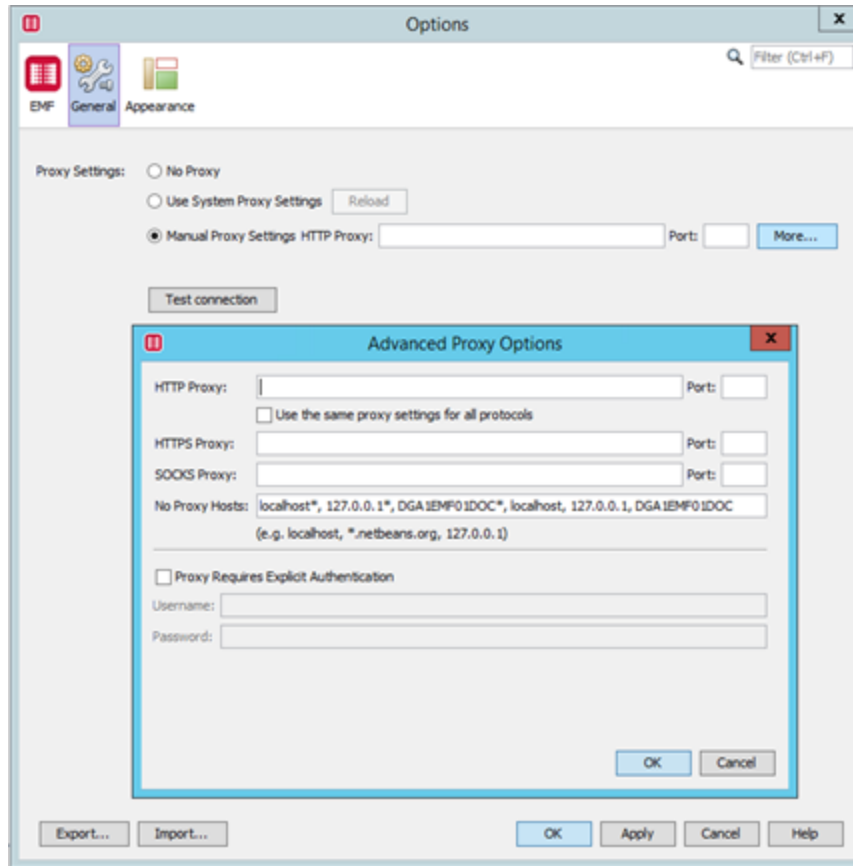
Parameter	Meaning
-Xms	Initial java heap size
-Xmx	Maximum java heap size
-XX:PermSize	Permanent generation size
-XX:MaxPermSize	Maximum permanent generation

General

Use the **General** option (menu) to configure your default browser.



1. Select the check box of your choice to configure your proxy settings. You have 3 options - not using a proxy, using the system proxy settings or manual proxy settings. Click the **Test Connection** button to test the connection
2. If you select manual proxy settings, enter the HTTP Proxy manually. Click **More** for further options.



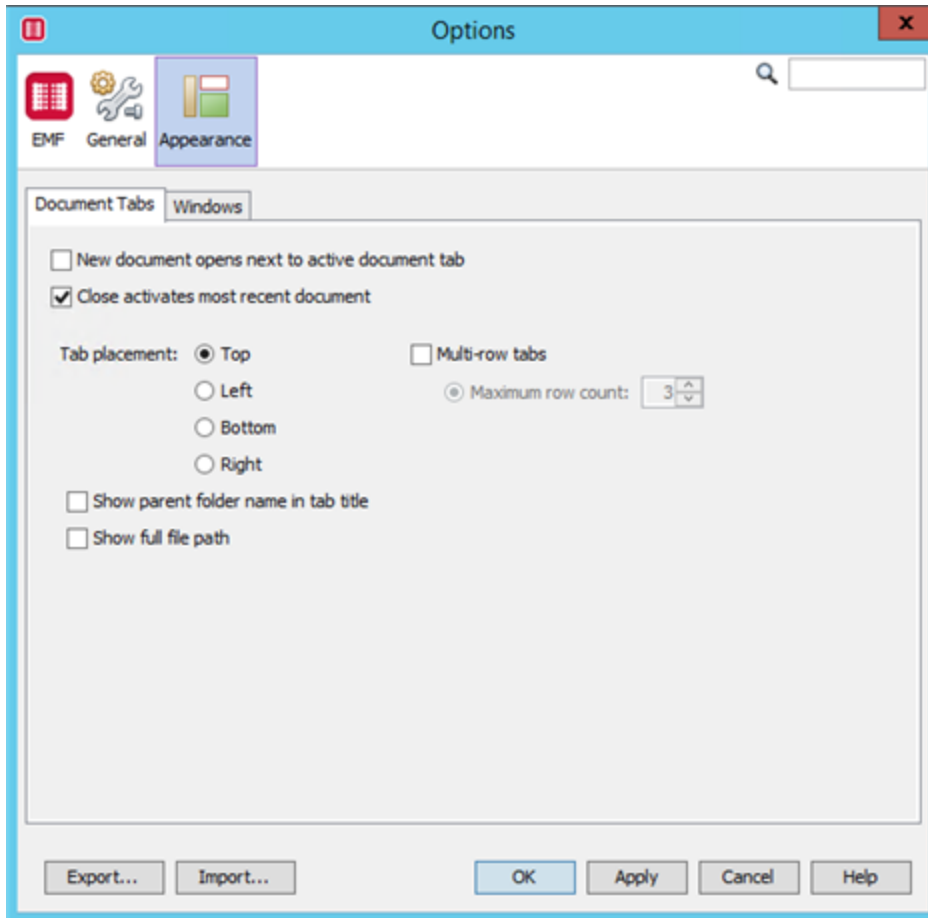
3. Select the **Proxy Requires Authentication** check box to enter credentials for authentication.
4. Click **OK** to save your changes.

Appearance

Use the **Appearance** option (menu) to configure options regarding Appearance and Files.

Document Tabs

The **Document Tabs** tab allows you to modify the position of tabs in the document (EMF process) area.

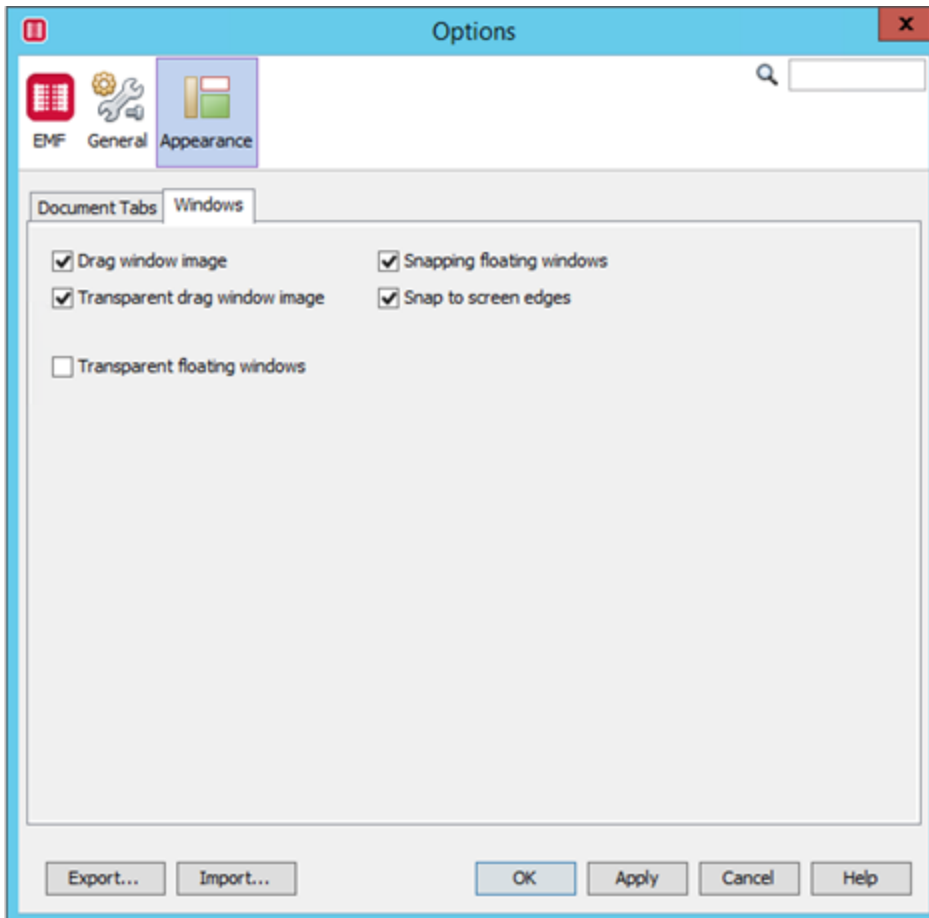


1. The **Document Tabs** tab allows you to modify the following options:
 - **New document opens next to active document tab:** If selected, a new document (EMF process) opens in the same docking spot next to the active document (EMF process) tab.
 - **Close activates most recent document:** If selected, the most recent document (EMF process) becomes active after an active document (EMF process) is closed.
 - **Tabs placement:** Select where to place the editor tabs. The possible tab locations are:
 - Top
 - Bottom
 - Left
 - Right
 - **Multi-row tabs:** If selected, the editor tabs display in multiple rows. Select the **Maximum row** count.
 - **Show parent folder name in tab title:** If selected, the parent folder name is displayed in the tab title.

- **Show full file path:** If selected, full process path is displayed above / below the tab title.
- 2. Click **OK** to save your changes.
- 3. You can also Export/Import settings that you saved. Click **Export** to save the settings in a ZIP file on your local machine. Select **Import** to upload settings to the application.

Windows

The **Windows** tab allows you to configure how the windows will look and behave.



1. The **Windows** tab enables you to specify the following settings:
 - **Drag window image:** If selected, the EMF UI displays a small copy of the window content when dragging a window.
 - **Transparent drag window image:** If selected, the EMF UI displays a small transparent copy of the window content when dragging a window.
 - **Snapping floating windows:** If selected, undocked windows snap to edges of other undocked windows when being dragged.
 - **Snap to screen edges:** If selected, the EMF UI snaps an undocked window to the edges of the screen.
 - **Transparent floating windows:** If selected, the EMF UI makes floating windows semi-transparent a few seconds after they go out of focus.

2. Click **OK** to save your changes.
3. You can also Export/Import settings that you saved. Click **Export** to save the settings in a ZIP file on your local machine. Select **Import** to upload settings to the application.

EMF Process Validation

When you have finished building your EMF Process, you must **validate** it before you can save it or set its status to Active. EMF Process validation checks for problems with the construction of the EMF Process that could prevent it from running properly; any error or warning messages that result from validation are displayed in the Information/Messages pane of the EMF Process Builder Design Graph.

Validation is performed at two levels: module-by-module and overall EMF Process level.

- **Validation of individual modules** within an EMF Process is performed internally within the module's corresponding business object component.
- **Context-sensitive validation of EMF Processes**, (i.e. validating each module with respect to the other modules in the EMF Process) is carried out at the EMF Process level. There are two types of EMF Process level validation that can be carried out:

- **Generic Context-Sensitive Validation of Modules** can be carried out centrally based on a fixed set of rules applied to all modules according to a classification of module types (Categories). Context rules are defined and apply to module categories. Each module in an EMF Process that belongs to the category can be checked against the rule. Additional sub-categories can be specified in the rule to indicate how the sub-category modules can be placed in an EMF Process with respect to the main category. For example, one rule is:

Initiator modules (main category) can only connect directly to Start modules (sub-category). In the case of this rule, the EMF Process could not be saved if this rule is violated.

Within the EMF Process Builder tool, generic context-sensitive validation occurs:

- At EMF Process save time
- When links are added to modules.
- When the Validation button on the tool bar is pressed.
- **Specific context-sensitive validation of modules** is validation of a highly module-dependent nature. For example, If an Email output module is in the EMF Process, then there must be a module which assigns Email recipients preceding it.

Module dependent context-sensitive EMF Process validation is more complex validation that cannot be dealt with in the simple rules discussed above or where simple rules are applied conditionally based on specific values of properties of the module itself or other modules in the EMF Process.

Within the EMF Process Builder tool, this specific context-sensitive validation occurs at EMF Process save time or when the validate button is pressed.

[Guide to Building EMF Processes](#)

[Saving and Running EMF Processes](#)[Managing EMF Processes](#)

Changing the Settings for an EMF Process

Using the **EMF Process Properties** screen, you can change the name, description, status, debug level, settings and other properties for an EMF Process. Examples of other properties that you can change are priority, log levels and how many times to retry a failed EMF Process.

To open the EMF Process Properties screen:

Right-click in an empty area of the EMF Process Design Graph and click **Properties**. EMF Process Properties screen has the following fields:

- **Name** - The name that you assign to the EMF Process. The EMF Process will be referred to by this name throughout the System.
- **API Name** - The name assigned to the EMF Process for use with the API that you are using (if any). The default value is the same as the **Name**.
- **Description** - An optional description of the EMF Process. This field can be used to help establish to different users what the EMF Process is for from the EMF Process folder containing the EMF Process.
- **Status**: The status of the EMF Process:
 - **Active** - The EMF Process is activated and ready to run or running. You should not select this state unless the EMF Process can be considered finished and working.
 - **Hold** - The EMF Process is ready to run but is currently suspended. This state could be selected if you were making changes and did not wish for the EMF Process to be fired while the changes were in progress.
 - **Incomplete** - The EMF Process is not yet ready to run. This is the default state assigned to an EMF Process when it is first created.
- **Number of retries**: When an EMF Process instance is being processed, it may encounter problems (for example, due to the high workload of the EMF system or a congested network). The number that you enter in this field defines how many time the system should try processing the EMF Process instance before placing it on the Dead-letter queue.
- **Debug Log Level**: The level of information that will be added to the Debug Log about any problems encountered. See [Debug Logging](#).
- **Log user messages**: When this checkbox is enabled, the process will log user messages, as defined in the [Log Message module](#), in addition to the **Debug log level**. For example, if the **Debug log Level** is set to **Errors and warnings** and the **Log user messages** option is selected, and the **Log Message module** is set to log message as User Message, then the console window will display both the errors and warnings and any user messages defined in the Log Message module.
- **Priority**: Defines the general location of the EMF Process in the system Queue. For example: if the priority is set to **low** the EMF Process will be placed at the bottom of

the queue. If the priority is set to **high** the EMF Process will be placed near the top of the Queue, but underneath any other high priority EMF Processes (in order to ensure that previously fired high priority EMF Processes are processed first).

- **Persist EMF Process When Queued:** When enabled, every time the EMF Process branches (i.e. splits), the EMF Process will be queued.
- **Statistics Logging checkbox:** When this checkbox is enabled, the information processed by an instance of the EMF Process will be stored in the Statistics Log. See [Statistics Logging](#).
- **Dead Letter Queue (DLQ):** To change how long a failed message will stay on the dead letter queue before being deleted, clear the "Use default expiry time" and set a time in seconds. Setting a time of zero, will mean messages never expire, and will stay on the queue until removed by some other means. Leaving messages on the DLQ for longer than you require can cause the queues to fill and stop the message broker from releasing large blocks of message storage. If you know a process will fail, and are not worried about seeing it on the DLQ, then set the time to 1 second. The default time for messages to remain on the DLQ is set in the [queue provider dialog](#).
- **Log process start/complete to audit log:** When this checkbox is enabled, the process start and complete events are written to the Audit Log. See [Process Status Monitor view](#). If process performance is a consideration, then you may wish to turn off this feature as there is a small overhead to log the start and complete events for each process.

Note: Log process start/complete must be enabled to be able to cancel a running process from the Process Status Monitor view.

- **Expected run time period:** This is the number of time units that the process normally takes to run.
- **Expected run time interval:** Defines the units – i.e. seconds, minutes, hours, days, weeks, or months. For example, if the process takes half an hour to run, enter 30 as the run time period and Minutes as the run time interval.

After you make the required changes, close the **Properties** screen and save the EMF Process.

[The Error Handling tab](#)

[How to Design an EMF Process](#)

[Guide to Building EMF Processes](#)

[Managing EMF Processes](#)

[Sent Message Logging](#)

Saving and Running EMF Processes

You can save the EMF Process to the repository database at any time by clicking the Save icon on the toolbar.

The EMF Process must be [validated](#) before it can be saved; if there are errors you will be informed and they will need to be fixed before you can save the EMF Process (see [Debugging EMF Processes](#)).

Important: When you have saved an EMF Process, you will need to have Delete [security rights](#) if you want to remove any modules from the EMF Process.

Once the process is saved, click the **Run Process** icon on the toolbar to run it. To run, the EMF Process status must be **Active** (you cannot run an EMF Process that is not active).

You can also click the **Save and Run Process** icon on the toolbar to save and run the process.

Note: You must include a scheduler module in your EMF Process to use the **Run Process** or **Save and Run Process** icon.

[Managing EMF Processes](#)

[EMF Process Validation](#)

Setting the Error Handler for an EMF Process

The **Error handling** tab is used to specify which EMF Process you want to act as the Error Handler for the EMF Process you are creating. When you enable error handling, errors encountered while trying to fire an EMF Process instance will be passed on to the selected Error Handler EMF Process. You can then use this Error Handler EMF Process to perform further processing; for example, to email the error output to recipients (see example).

Note: A [default Error Handler](#) EMF Process is located in the **System EMF Process** folder.

To view the Error Handling Properties screen:

1. Right-click in an empty area of the **EMF Process Design Graph**, click **Properties**, and then select the **Error handling** tab.
2. Select the **Enable error handling** checkbox, then browse for and select the EMF Process you want to use as the Error Handler in the **EMF Process folders**.

[Changing General Settings for an EMF Process](#)

[The Advanced tab](#)

[How to Design an EMF Process](#)

[Guide to Building EMF Processes](#)

[Managing EMF Processes](#)

Error Handling

Errors can occur in EMF Process processing; the question is what action to take when they occur. All errors that can occur in EMF Process processing must be handled gracefully by the Server. The **Error Handler** is not intended to define how the Server should cope with errors; it is to provide a proactive method of error notification (i.e. alerting). When an error occurs during EMF Process processing an Administrator must be alerted of the error. For example, if a failure occurs when sending an SMS message to recipient X because his mobile phone number is invalid, send the administrator of the system an email containing the details of the failure.

The **Error Handler** does not define all errors that will be handled during EMF Process processing; any errors will be specific to a particular module and will thus be defined by the requirements of each module. It is intended to produce an **Error Handler** for EMF Process execution. When a critical error occurs within an EMF Process instance, the information is passed to the [Error Handler defined for the EMF Process](#).

To configure an EMF Process to use error handling:

1. Right-click in the **EMF Process Design Graph** and click **EMF Process properties**.
2. Select the **Enable error handling** checkbox in the **Error handling** tab of the **EMF Process properties** screen, and then select the EMF Process you want to use as the **Error Handler**.

Note: The default Error Handler EMF Process is **Error Handler** in the System folder (for more information on the default **Error Handler**, see [Customizing the Error Handler](#)).

[Error Handling Example](#)

[Debugging EMF Processes](#)

[Managing EMF Processes](#)

Customizing the Error Handler

The **Error Handler EMF Process** in the System folder saves the contents of the EMF Process error section to the **errorreportfile.txt** file in the **outputs/errorhandler** subdirectory of the EMF Server.

The content of the default **Error Handler** output is:

```
ERROR REPORT

Server: $DATA('ErrorSource',2,0)$
  Server Repository: $DATA('ErrorSource',0,0)$
Machine: $DATA('ErrorSource',1,0)$
  Alert Repository: $DATA('ErrorSource',3,0)$
Alert: $DATA('ErrorSource',4,0)$
Module: $DATA('ErrorSource',5,0)$
  AlertInstanceId: $DATA('ErrorSource',6,0)$
AlertProfileId: $DATA('ErrorSource',7,0)$
```

```
Error Log:
$MESSAGE('ErrorMessage')$
$DATAFORMAT('ErrorLog','\t','\r\n')$
```

This information is extracted from the error section of the EMF Process object. **ErrorSource** and **ErrorLog** are data sections; **ErrorMessage** is a message section. For more information, see [Error Handler Sections](#).

If you want certain people to be notified when an EMF Process generates errors, you need to configure the default **Error Handler EMF Process** to specify the recipient and output device details, or create a new EMF Process to do this. Creating an **Error Handler EMF Process** is exactly the same as creating other types of EMF Processes. However you do not need to add an initiator module because the EMF Process will be initiated automatically whenever the calling EMF Process generates errors.

Because you can select any EMF Process you want as the **Error Handler** for your EMF Process, you can have a different **Error Handler** for each EMF Process. However, it is likely that you may want one **Error Handler** to deal with errors from a number of EMF Processes, for example:

- **Send customized messages to recipients depending on the EMF Process.** To do this, use [conditional links](#) to compare the ID of the EMF Process that generated the message with the ID of the EMF Process you want to handle in that branch of the Error Handler. Use `$DATA('ErrorSource',7,0)$` as the expression.
- **Send messages to recipients based on their association with specific EMF Processes.** To do this, use aliases to associate recipients with EMF Process IDs. Then use an [aliased recipients](#) module in the Error Handler to compare the alias values against the ErrorSource EMF Process ID (see [example](#)).

[Debugging EMF Processes](#)

[Managing EMF Processes](#)

Error Handling Example

This example shows one possible way of configuring an Error Handler EMF Process to send a message to certain recipients, based on which EMF Process generated an error.

Overview

When an EMF Process with error handling enabled generates errors, email the details to the recipients associated with that EMF Process. If none of the recipients reply within 2 hours, advise the System Administrator via email.

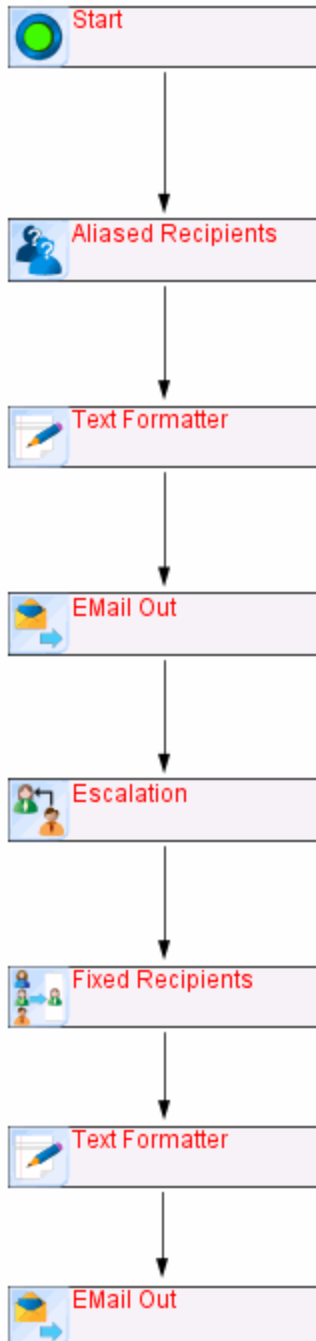
Prerequisites

- [Recipient alias](#) called EMF ProcessID created.
- EMF ProcessID [alias added to recipients' details](#), with the ID of the EMF Process that you want to associate with the particular recipient as the alias value. If you want to associate more than one EMF Process with a recipient, simply add the alias again in the recipient details and enter the new value.

Note: The EMF Processes must have error handling enabled.

- Email out service called SMTP Service configured.

EMF Process structure



Module details

Aliased recipients

ErrorSource is a [data section generated during error handling](#). It contains several fields, one of which is the ID of the EMF Process. **Lookup column** is set to look at this field (column 7).

Text formatter

This contains modifications to the [default error message text](#). The recipient name has been included at the top, and further instructions have been appended. The [\\$ESCALCODE\(\)](#) [dynamic function](#) is used by EMF to associate the reply from the recipient with the EMF Process.

Email out

Set the email out properties in this module. Details about the Email service used, subject of the mail, etc.

Escalation

The **Per EMF Process** cancellation scope means that if any replies are received, [escalation](#) will be cancelled for the EMF Process.

Fixed recipients

In the [Fixed recipients module](#), select the recipient you want to alert about unconfirmed error notifications, for example the Admin user.

Email out (1)

As for the previous Email out module, the service is **SMTP Service**. Select **Text formatter** as the message section.

[The Error Handler](#)

[Debugging EMF Processes](#)

Error Handler Sections

When error handling has been enabled for an EMF Process and an error is generated, the error information is stored in the following sections:

- ErrorSource data section
- ErrorLog data section
- ErrorMessage message section

The **ErrorSource** data section contains the details of the EMF Process that generated the error:

Column ID	Column Name	Description
0	ServerRepository	The ID of the repository in which the EMF

		Process is stored.
1	Machine	The ID of the EMF Server machine on which the error occurred.
2	Server	The name of the server in which the error occurred, for example "WIPQ Server manager".
3	AlertRepository	The ID of the repository in which the EMF Process is stored.
4	Alert	The name of the EMF Process (not the same as the EMF Process API name).
5	Module	The name of the module that generated the error.
6	AlertInstanceID	The ID the instance of the EMF Process in which the error occurred.
7	AlertProfileID	The ID of the EMF Process that generated the error.

The **ErrorLog** data section is used only for recording the details of unresolved aliases. It has the same structure as the data section that you originally used in the [Aliased Recipients module](#). It lists all rows that EMF was unable to resolve against recipients.

If the error was caused by something other than unresolved aliases, then the exception information is stored in the **ErrorMessage** message section.

ErrorLog and **ErrorMessage** are mutually exclusive - an error EMF Process will never contain both these sections. If an Aliased Recipients module in an EMF Process has unresolved aliases, then ErrorSource and ErrorLog will be populated and the Error Handler will run. If any of the modules in the EMF Process generates errors, then ErrorSource and ErrorMessage are populated and the Error Handler will run.

Note: You may also want to look at the [debug log](#) for further details of error that occurred.

[Customizing the Error Handler](#)

[Error Handling](#)

Sent Message Logging

When Sent Message Logging is enabled, a record of messages sent (such as emails and file outs) is recorded to the log (for details, see [Log View](#)). Options are available to record who the message recipient was, as well as the content of the message.

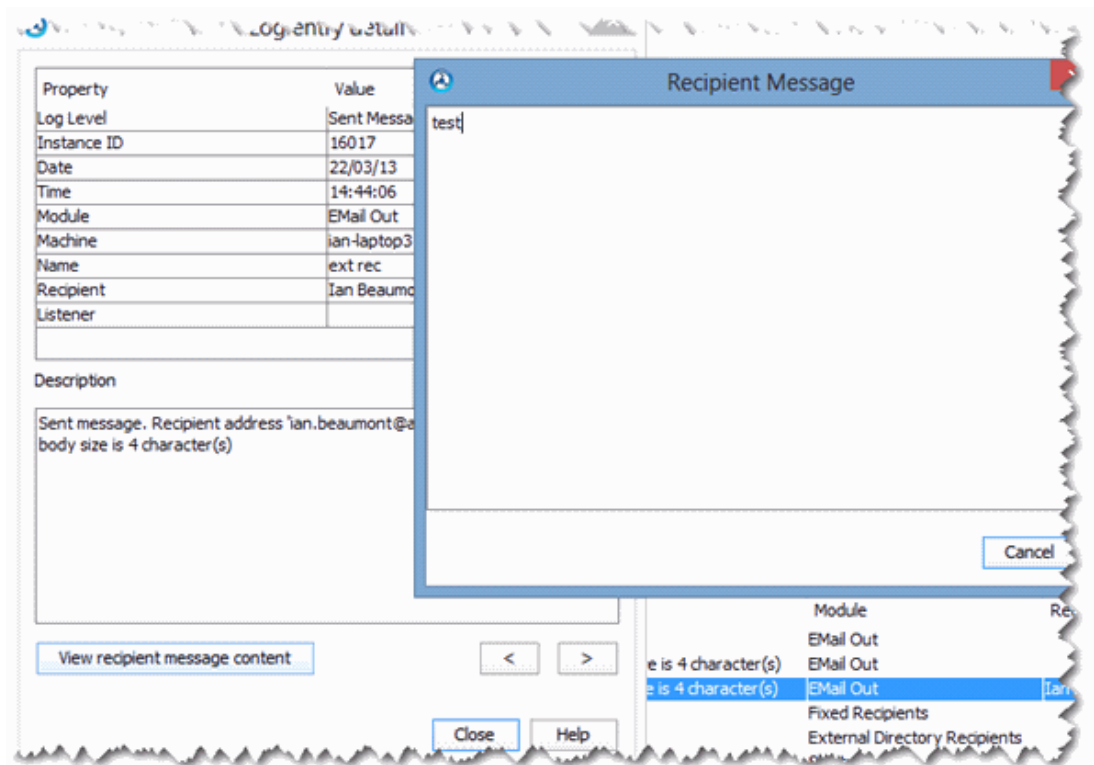
To enable the message logging for a process, on the **Sent Message Logging** tab of Process Properties, check the **Enable message logging** check box. Once enabled, this allows you to select which output modules you want to record the contents they send.

Select the **Log Message Sent** check box to enable a certain output module to log details of every message sent. If you also require that the contents of the message are logged, then select **Log Message Contents** for that module as well.

When the process runs, any logged messages will appear in the log (for details, see [Log View](#)).

Log Level	Instance ID	Date	Message	Module	Recipient
	16017	22-Mar-2013 14:44:06	Process branch executed	EMail Out	Ian Beaumont
	16017	22-Mar-2013 14:44:06	Sent message. Recipient address 'ian.beaumont@aptteam.com', main message body size is 4 character(s)	EMail Out	Ian Beaumont
	16017	22-Mar-2013 14:44:06	Sent message. Recipient address 'ian.beaumont@aptteam.com', main message body size is 4 character(s)	EMail Out	Ian Beaumont
	16017	22-Mar-2013 14:44:05	1 recipient(s) added	Fixed Recipients	Ian Beaumont
	16017	22-Mar-2013 14:44:05	1 recipient(s) added	External Directory Recipients	Ian Beaumont

If the "recipient" was a recipient that exists in the EMF system (i.e., a fixed recipient or aliased recipient module was used), then the recipient name will appear in **Recipient** column in the log view. To view the message contents (if logged), double click on the log entry to display the log entry detail, and then press the **View recipient message content** button.



Old messages are cleared out of the log by the [Sent Message Log Manager](#), so it is important that this is running if you require old messages to be removed.

Setting Advanced Properties for an EMF Process

You can use the **Advanced** Tab of the EMF Process Properties screen to define further settings (e.g. priority, log levels and how many times to retry a failed EMF Process) about the EMF Process you are creating.

To view the Advanced Properties screen:

Right-click in an empty area of the EMF Process Design Graph and click **Properties**, then select the **Advanced** tab.

- **Number of retries:** when an EMF Process instance is being processed, it may encounter problems (e.g. due to the high workload of the EMF system or a congested network). The number that you enter in this field defines how many time the system should try processing the EMF Process instance before placing it on the Dead-letter queue.
- **Debug Log Level:** the level of information that will be added to the Debug Log about any problems encountered. See [Debug Logging](#).
- **Priority:** defines the general location of the EMF Process in the system Queue. For example: if the priority is set to **low** the EMF Process will be placed at the bottom of the queue. If the priority is set to **high** the EMF Process will be placed near the top of the Queue, but underneath any other high priority EMF Processes (in order to ensure that previously fired high priority EMF Processes are processed first).
- **Persist EMF Process When Queued:** when enabled, every time the EMF Process branches (i.e. splits), the EMF Process will be queued.
- **Statistics Logging checkbox:** when this checkbox is enabled, the information processed by an instance of the EMF Process will be stored in the Statistics Log. See [Statistics Logging](#).

[Changing General Settings for an EMF Process](#)

[The Error Handling tab](#)

[How to Design an EMF Process](#)

[Guide to Building EMF Processes](#)

[Managing EMF Processes](#)

Debugging EMF Processes

With the amount of functionality available within EMF Process processes comes the corresponding increase in complexity, which increases the likelihood of errors occurring.

You can debug your completed EMF Process automatically at the time when it runs by inserting a [Debug Output](#) module. In this case, the debug output information will be saved to a file with a name and location that you specify.

Debugging the EMF system

In addition to debugging individual EMF Process, you can also view a log of any errors or other messages generated by the EMF system.

This log is available by clicking **Window > Log** in EMF Administrator. For more details, see [Debug Logging](#).

[Guide to Building EMF Processes](#)

[Go to start of EMF Process Creation and Management](#)

The Create New EMF Process Screen

The **Create new EMF Process** screen allows you to define the name and description of an EMF Process before opening the [EMF Process builder](#). A name must be assigned to an EMF Process before you can proceed to the EMF Process Builder Design graph. Once the EMF Process has been saved from within the EMF Process Builder, the EMF Process is stored in the open folder under the name that you have assigned here.

- **Name** - enter the EMF Process name. This name can be anything you choose, but should not contain apostrophes, for example *Fred's EMF Process*.
- **Description** - this is a field where you can enter a description of the EMF Process which you have created. It is for reference purposes only and is optional.

[The EMF Process Properties Screen](#)

[How to Design an EMF Process](#)

[Guide to Building EMF Processes](#)

[Managing EMF Processes](#)

Debug Logging

Debug logging is a tool that end users, support, and in-house developers can use to help locate system faults. Server managers, modules, EMF Processes and the EMF Server can all log debug information to a central repository for display using the **Debug log table view**.

Note: if for any reason the debug log is unable to write to the repository database, a backup log is created as a file on your computer. You can specify the location of this file using the file named **EMF.properties** in the EMF Java server folder.

The Debug Log displays information about the general EMF system. It does not debug individual EMF Processes; for this you can either [debug the EMF Process](#) as you are creating it or else [insert a Debug Out module](#) to monitor any problems that occur when running the finished EMF Process.

To view the Debug log:

Select the **Windows** and then click **Log** to display the **log table view**, which displays nine columns of information:

- **Type** - the [debug level](#) of the logged entry (i.e. Errors Only, Errors and Warnings, Detailed or Verbose).
- **Importance** - the importance of the logged entry. **5** is **Low**, **10** is **Medium** and **15** is **High**.
- **Event Date** - the date of the logged entry.
- **Event Time** - the time of the logged entry.
- **EMF Process API Name** - the unique API name of the processed EMF Process.

- **Description** - the description of the debug log entry.
- **Component** - the EMF component that produced the log entry.
- **Module** - the EMF module that produced the log entry.
- **EMF Process Instance ID**- the instance identifier of the processed EMF Process. If an EMF Process is fired on ten separate occasions, then there have been ten EMF Process Instances, which each have a unique ID.

You can also view these and other details in the dialog form (see the [Debug entry detail](#) screen) by double-clicking on an entry and using the up and down arrows to scroll through the events.

[Viewing details of an individual entry in the Debug log](#)

[Extracting, saving and deleting information from the Debug log](#)

[Managing the size of the Debug log](#)

[Logging in EMF](#)

[Go to start of EMF Help](#)

Statistics Logging

You can use **Statistics Logging** to monitor how much information is being passed by **Recipient** and **Data read** (SAP and SQL) modules within an EMF Process, as well as information about branch splits. Statistics logging is automatically enabled for all new EMF Processes and, if you need to, you must specifically disable it on a per-EMF Process basis.

To enable or disable Statistics logging for an EMF Process:

1. Right-click in the EMF Process Design Graph and select **Alert Properties**
2. Select the **Properties** tab and enable or disable the **Statistics Logging** option. When statistics logging is enabled, the following information is logged:
 - **Alert started**
 - **Alert branch processed**
 - **no. of data row(s) read**
 - **no. of recipient(s) added**

To view the Statistics entries:

Statistic entries are written to the EMF System log. To view these entries, click **Windows** and then **Log** to view details of each log entry. Statistic log entries are indicated with a "chart" icon.

[Extracting, saving and deleting information from the Statistics log](#)

[Managing the size of the Statistics log](#)

[Logging in EMF](#)

[Go to start of EMF Help](#)

Debug Log Entry Detail Screen

The **Debug entry detail** screen provides a more convenient way to view all the field information returned on a single entry in the Debug log. The information displayed includes everything that is displayed in the Debug log table view, with the addition of some extra fields.

To view the Debug entry detail screen:

Double-click an entry in the list, or right-click it and select **Properties**. You can view details of other events in the list by using the up and down arrows to scroll through them.

- **Type** - the [debug level](#) of the logged entry (Errors Only, Errors and Warnings, Detailed or Verbose).
- **Event Date** - the date of the logged entry.
- **Event Time** - the time of the logged entry.
- **Importance** - the importance of the logged entry. **5, 10, 15** are High, Medium and Low respectively.
- **EMF Process Folder** - the location of the folder containing the EMF Process.
- **EMF Process Profile** - the name of the EMF Process.
- **EMF Process Profile ID** - the EMF Process identifier.
- **EMF Process Instance ID** - the instance identifier of the processed EMF Process.
- **Module** - the EMF module that produced the log entry.
- **Component** - the EMF software component that produced the log entry.
- **Component Version** - the version number of the EMF software component which produced the log entry.
- **Machine** - the name of the machine that logged the entry.
- **Server** - the name of the server associated with the logged entry.
- **Repository** - the name of the repository used for logging.
- **Description** - a description of the event that caused the error log to be generated.

[Overview of Debug logging](#)

[Extracting, saving and deleting information from the Debug log](#)

[Managing the size of the Debug log](#)

[Logging in EMF](#)

[Go to start of EMF Help](#)

Setting a Debug Logging Level

You can control the amount and level of debug information logged when processing EMF Processes by changing the debug log level of the [Server Instance](#) that is processing the EMF Process instance.

To set the debug level from within the EMF Process Properties screen:

You can also set the logging level using the **Properties** screen in the EMF Process builder, but the level cannot be more detailed than the setting on the **Server Instance** screen. If you set a higher level within the EMF Process Properties screen, it will automatically be reduced to that of the Server Instance.

1. Create/Open an EMF Process.
2. Right click in a blank area of the EMF Process Design Graph and select **EMF Process Properties**.
3. Select the **Properties** Tab.
4. Set the **Debug Log** Level.

[Go to start of Debug Logging](#)

[Go to start of Logging in EMF](#)

[Go to start of EMF Help](#)

Folder Organization

Your EMF Processes are stored in folders that you can access from the EMF Tree view. Click on a folder to open it and display the [EMF Processes view](#), showing a list of the EMF Processes in the current folder.

The top level EMF Process folder is called the **Root Folder** and beneath that is another folder (colored red) called **System**. You cannot rename or delete these folders as they are essential to the correct functioning of EMF.

- **System** contains [System EMF Processes](#) that are used by the EMF system. These are used for functions like error handling, SMS EMF Process routing and POP3 EMF Process routing.

Although you can store your EMF Processes in the **System** folder, it is better to create your own folders and keep your EMF Processes there. These should be given meaningful names in order to help you organize your information.

To create a new EMF Process folder:

- Right-click in the folder where you want to create the new folder or on its icon in the Tree view, and click **New folder**. The new folder is created, displayed at the bottom of the list of folders, and opened.

To rename the new folder:

- Right-click on the icon for the folder and select **Rename**, then type the new name.

[Managing EMF Processes](#)

[How To Add Delete and Change Folders](#)

The EMF Processes View

You can display a list of all the EMF Processes in a particular folder by clicking on that folder in the EMF Tree view to open the **EMF Processes View** in the right-hand pane.

- **To select an EMF Process** (e.g. to use administrative tools on it): Click once on the icon for the required EMF Process.
- **To open an EMF Process for editing**: Double-click the icon for the required EMF Process to open it in the EMF Process Builder View.

Customizing the EMF Processes View

The columns in the EMF Processes view are not fixed and you can alter them in a variety of ways (e.g. reordering or hiding them) as you choose.

Adding, Deleting and Renaming Folders

You can add, delete, and rename folders in the EMF Tree view.

You cannot rename or delete the **Templates** or **System** folders, neither can you create new folders in the **Templates** or **System** folders.

Adding Folders

New folders can be added to the EMF Process Folder in order to customize the folder structure. To add new folders:

1. **Right Click** on the parent folder.
2. Select **New Folder**.

A new folder is added to the folder structure.

3. To rename the new folder, right-click on the new folder and select **Rename**.
4. Type the new name and press **Enter**.

Deleting Folders

Note: You cannot delete a folder unless it is empty. If you wish to delete a folder, you must ensure that all EMF Processes within that folder are either deleted or moved.

1. Right-click on the folder that you wish to delete.
2. Select **Delete** from the drop-down list.

Renaming Folders

1. Right-click on the folder that you wish to rename.
2. Select **Rename** from the drop-down list.
3. Type in the new name and press **Enter**.

[Folder Organization](#)

[Managing EMF Processes](#)

Importing and Exporting In EMF

When an EMF Process has been created, it may be necessary to move it from one repository to another. You can do this using the **Import/Export** screen, which allows you to export an EMF Process (or a set of EMF Processes) as an exml file, or to import an existing exml file that contains details of an EMF Process.

You can also export recipients, roles, services, listeners, and data sources from within EMF.

Note: There is a separate migration tool available to import the processes from Xalerts 5.6. For more information, see the Event Management Framework 7 Installation and Configuration Guide.

[Importing EMF Processes](#)

[Exporting EMF Processes](#)

[Go to start of EMF Process Management](#)

[Troubleshooting Import and Export of EMF Processes](#)

Importing EMF Processes

To import items into EMF:

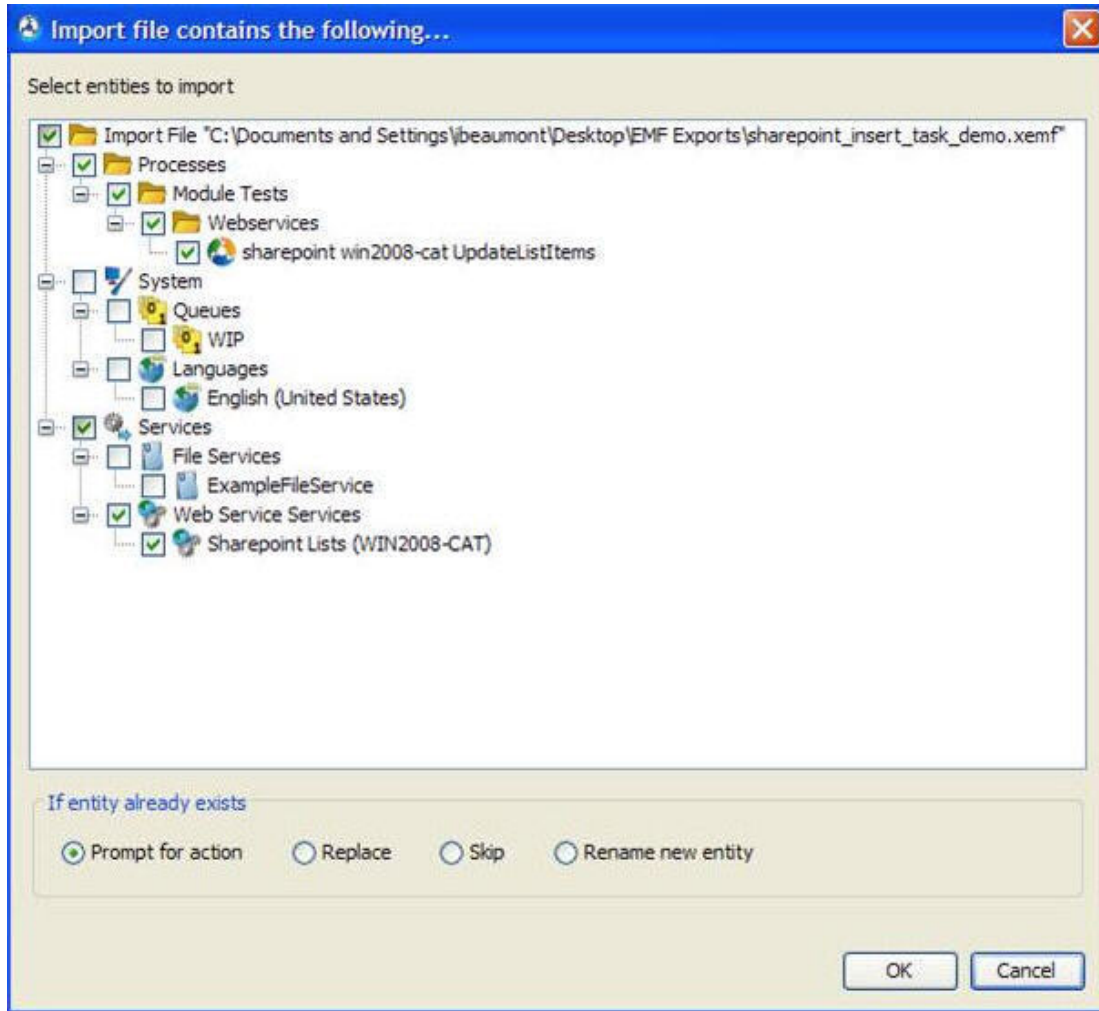
1. Right-click on the area you want to import the item into.

Typically, you can just right click on the **Repository** node, and it will import all the items into the respective locations.

If the export file contains EMF processes and/or folders and you want to put them under your own folder structure, then right-click on the EMF process folder you want to import the items into.

2. Select the file you want to import. By default EMF export files have the extension **.xemf**.

3. On selecting the file, a list of items that can be imported from the file is displayed. Select the items you want to import. By default, all the items are selected. Languages are not selected by default as they usually exist in the EMF repository.
4. **Select** the items you want to import or the option to automatically import, if the item already exists in the EMF repository.
5. Click **OK** to start the import.



Important: To import or export item, you must have full security rights to it and all its dependent objects. Your EMF [license](#) must also allow import and/or export, as appropriate.

Exporting EMF Processes

To export items in EMF:

1. Select the items you want to export in the EMF tree, right-click, and select **Export**.

All items such as processes, recipients, and data sources can be exported. The whole repository can be exported by selecting the **Repository** node.

2. You will be given an option whether or not to export the **Dependencies** for the selected items. For example, if you select to just export an EMF process which has an SQL module, it will have dependencies for the folder it is in and also dependencies on the JDBC Data Source that the SQL Module is using.
3. Type the file name that the exported items will be stored in (this will be the default **.xempf** file that is used for all EMF export files).

Important: To import or export items, you must have full security rights to it and all its dependent objects. Your EMF [license](#) must also allow import and/or export, as appropriate.

Problems Importing or Exporting EMF Processes

Lookup failures on import. These can occur when a lookup of an object referenced by name (eg a JDBCDataSource referenced by a SQL module) has failed. Typically, this is because the referenced object is not in the XML - in which case you must manually create one. Alternatively, the referenced object may not exist yet although it is part of the XML that is to be imported - in which case it will exist once it has been imported. This will be resolved later by another lookup.

Note: References to external business objects are by **name** rather than by ID (because IDs are unlikely to be the same on import as they were originally). Therefore, on export, an ID (eg a **JDBCDataSourceID** used in a SQL module) is converted into a name (the corresponding name of the JDBCDataSource), and this name is added as the value of an attribute in the XML.

If a lookup fails because (for example) a SQL module cannot lookup a data source that does not exist, the EMF Process and the module will be saved - although no data source will be set for the SQL module. If you wish to retry the import after creating an appropriate data source, you should delete the EMF Process first otherwise a unique key violation will result and the module will not get updated with the correct data source.

The following example message indicates the type of object (a UserProfile) that could not be found and its name (Q User 1): ***** (Severity=2, Error code=-2147217502) Unable to lookup object in node 'csUserModuleBo.CbUserModule_2_68484133' using condition " WHERE UserName='Q User 1' " in BO csUserProfileBo.CbUserProfiles.**

Unique Key violations. These occur when an object with the same name as one being imported already exists - e.g. a folder with a particular name already exists within a parent folder. This can be caused by a previous (possibly unsuccessful) import, or by coincidence. If caused by a previous failed import, you may need to delete the object manually before retrying the import.

[Importing and Exporting EMF Processes](#)

Email support at EMF.Support@aptean.com

Process Status Monitor View

The Process Status Monitor view displays the status of those processes that are either currently running, have already run, or are due to run. Only those processes that have the option selected for logging a process start/complete event are shown in this view. (See [Process Properties](#) for how to set this option).

Processes are shown with the following descriptions within the Process Status Monitor view:

- **Processes scheduled to run**
- **Process Completed successfully**
- **Process Completed with errors**
- **Running Process has past its expected completion time:** (See [Process Properties](#) on how to set up the expected completion time).
- **Process is being cancelled:** Shown when the process is selected for cancellation.
- **Process was cancelled:** Also shows the number of branches that were cancelled
- **Process running:** Shows the number of branches being processed.
- **Process started**
- **Process has missed scheduled runs:** Identifies that this instance of the process missed one or more scheduled runs since the last time it ran.
- **Process started late:** See [Scheduled Alerts Retriever](#) on how to set up the late start margin.

The Audit Log view shows in detail the times at which each of the above states occurred. Note that not all processes will show all states. As the Process Status Monitor view gets its information from the Audit Log, truncating the Audit Log will also clear information from the Process Status Monitor view. Use the Audit Log Configuration to manage the entries in this view.

To view the Process Status Monitor:

Click **Windows** and then click **Process Status Monitor** to display the **Process Status Monitor table view**, which displays seven columns of information about all the processes that have run since the Audit Log was last cleared (for information on clearing the log, see the [Audit Log](#) screen):

- **Process name:** The type of the entry.
- **Instance ID:** The instance identifier of the processed EMF Process. If an EMF Process is fired on ten separate occasions, then there have been ten EMF Process Instances, which each have a unique ID.
- **Start time:** The date and time the process instance was queued to run. Shows the next scheduled run time of the process if it has not yet started (The Instance ID column will be blank).
- **Start notes:** The description of the process start. This can have events such as started, missed runs, and late start.
- **End time:** The date and time the process instance ended. Note that for events that happen before the process ends will have a blank end time, for example, cancelling, past expected completion time.

- **End notes:** The description of the process end. This can have events like completed, completed with errors, cancelled.
- **API name:** The API name of the process.

Color coding has been used to make the user aware of any important events within the process cycle. For example, Missed runs, Late start, errors in process run, past expected completion time, process cancelled.

The following options are available on the right-click menu for each entry in the table view:

- **View Log Events for Process:** Shows the log events for the selected process in the standard Log view window.
- **View Log Events for Process Instance:** Shows the log events for the selected process instance in the standard Log view window.
- **Select in Explorer:** Selects the highlighted process in the explorer view.
- **Find Process Status:** For processes that are currently running, shows the number of branches waiting in the queues and what modules the branches are waiting on, in a pop-up dialog box. Double-clicking an entry will also show the process status dialog box.
- **Cancel:** Submits a cancel request to the server to end the highlighted processes. Some modules, which have long running task can be interrupted straight away (for example, SQL Module/SAP data browser batch insert, File Move/Copy modules, Data section tool merge/filter options). Multiple entries in the table view can be selected for cancelling.

The following buttons are available in the view:

- **Refresh:** Refreshes the table view.
- **Export:** Exports the table view to a MS Excel format file. For details, see [Extracting information from the Process Status Monitor view](#).
- **Configure:** Displays a dialog box that allows you to set the interval between each automatic refresh of the table view. Setting too low a value will have an impact on the performance of the EMF UI.
- **Auto refresh ON:** Clicking this button will switch on the Auto refresh, and the table view will be refreshed automatically as per the configured interval. A depressed state of the button shows that the Auto refresh is ON.
- **Auto refresh OFF:** Clicking this button will switch off the Auto refresh. A depressed state of the button shows that the Auto refresh is OFF.

To disable Process start/complete logging:

The process start/complete logging is enabled by default for each process. If you want to turn it off for a process, clear the **Log process start/complete to audit log** option on the [Process Properties](#) screen for that process. This option cannot be turned off globally.

[Setting logging of process start/complete option](#)

[Managing the size of the Audit log](#)

[Logging in EMF](#)

[Introducing EMF](#)

Extracting From Process Status Monitor View

To extract logged entries:

1. Click **Windows**, and then click **Process Status Monitor**.
2. Click **Export**. This displays the **Export to MS Excel format** dialog box.
3. If you want to export the information to a file, enter a location and file name for the file (or select a location).
4. Click **Export** to save the contents of the table view. Exporting Process Status Monitor view entries saves their content to a file and leaves the original entries intact.

[Process Status Monitor view in EMF](#)

[Introducing EMF](#)

The EMF Process Builder Modules

There are different types of modules you can use when building an EMF Process:

- [Initiator Modules](#) are used to define when an EMF Process should be fired - e.g. at a certain time, or in response to an email or other trigger. Each type of Initiator module depends on different EMF system configurations being completed. Links to the relevant system configuration screens are available from the individual initiator Help pages.
- [Start module](#) must be present in every EMF Process, and so it is automatically added whenever a new EMF Process is created. It has no function other than to act as a bridge between the initiator modules and the processing and output modules, and hence has no settings. It therefore has no "Properties" page when double-clicked.
- [Data modules](#) are used to gather and process information from databases, using JDBC and/or SAP connections, and insert this information into a freshly created Data Section. Other modules within the EMF Process can then use this data by referencing the Data Section name.
- [Recipient modules](#) are used to create a list of system recipients and/or groups that can then be used by other modules to define who should receive an EMF Process notification. The same list can be used as many times as necessary, which saves the other modules from having to retrieve the same information more than once.
- [Flow modules](#) are used to influence the order and timing sequence of modules that are executed within an EMF Process.
- [Filter and Transformation modules](#) are used to convert message and data sections between different formats (such as an XML message section into a data section) and to filter the data in different ways.
- [General modules](#) are used to log information, carry out a sequence of commands without any user input and to create a dump of the running process state to help diagnose any problems that occur when the EMF process runs.
- [Formatting modules](#) are used to format data for Output modules. Data from the EMF Repository or from a Data Section generated by a data module earlier in the EMF process can be used. Each formatter module placed in an EMF Process generates a

Message Section, which must have a unique name. You can then select the Message Section in an Output module.

- **Output modules** are the modules that send data to output devices or recipients. You can have as many Output modules as you wish, and generally, you should precede them with an output formatting module.

You must define the output services in the Configuration Database before you can use them (see [configuring services](#)). Some default output services (e.g. **FTP**, **File** and **Run Executable**) are already defined in the EMF system.

- **File Utilities modules** assist in performing operating system file manipulation such as copying or deleting files.
- **Audit modules** assist in the automation of internal and external audit processes.
- **Routers modules** decide which EMF Processes should be fired when a specific Listener server instance receives a message.
- **Event Plan modules** allow Events to be created and processed.

[Creating EMF Processes](#)

Initiator Modules

Initiator Modules are used to define when an EMF Process should be fired - e.g. at a certain time, or in response to an email or other trigger. Each type of Initiator module depends on different EMF system configurations being completed. Links to the relevant system configuration screens are available from the individual initiator Help pages.

When an Initiator module is placed in an EMF Process, it can only be linked to the EMF Process **Start** module.

Note: Most initiator modules also require you to create an appropriate [server instance](#).

- The [Scheduler module](#) allows you to specify the times and days when the EMF Process should be fired.
- The [API In module](#) is required by any EMF Processes that need to be triggered (by database triggers) or run by the EMF Runtime_API.
- The [HTTP In module](#) allows you to specify that an EMF Process should be fired when an HTTP listener receives a request.
- The [Email In module](#) allows you to specify that an EMF Process should be fired when an email message is received either by an IMAP or POP3 listener.
- The [Queue In module](#) allows you to initiate an EMF Process based on an incoming JMS queue message.
- The [SNMP In module](#) allows you to specify that an EMF Process should be fired when a queue message is received by an SNMP listener server instance.

[Go to System Configuration](#)

[Introducing EMF](#)

The Scheduler Module

You can fire an EMF process at a specific time and/or date by adding a **Scheduler module** to your EMF Process. You can use as many Scheduler modules as you wish in an EMF process, or none at all.

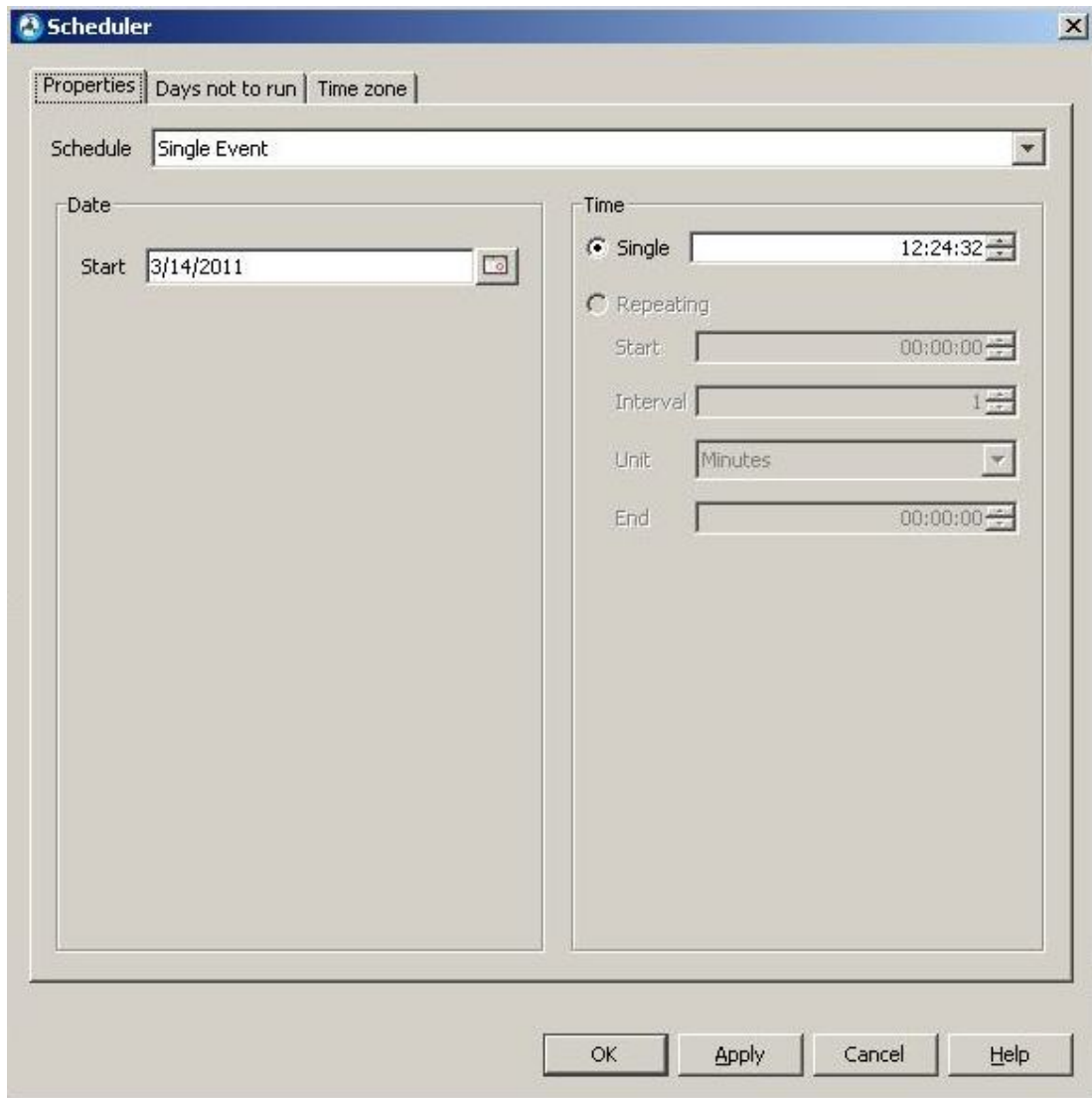
Note: To use a Scheduler module, you must have a [Scheduled Alerts Retriever](#) server instance set up on your machine so that Scheduler modules can be monitored for the times and dates on which EMF Processes should be fired.

To use a Scheduler module:

1. Ensure that you have created at least one **Scheduled EMF Processes Retriever** server instance.
2. Drag a scheduler module icon into your EMF Process and create a link from it to the **Start** module.



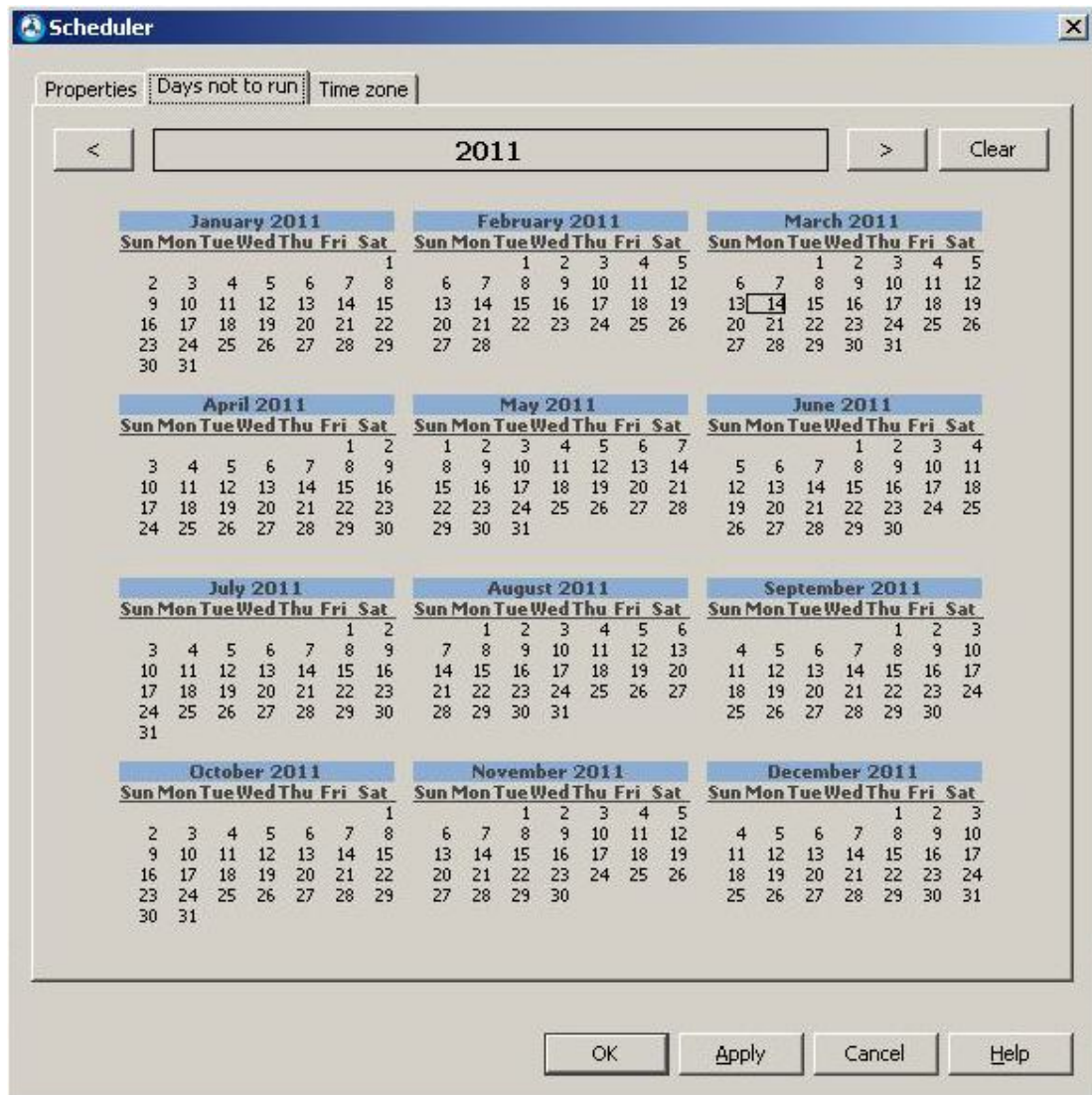
- Double-click the module icon to display the **Scheduler** screen.



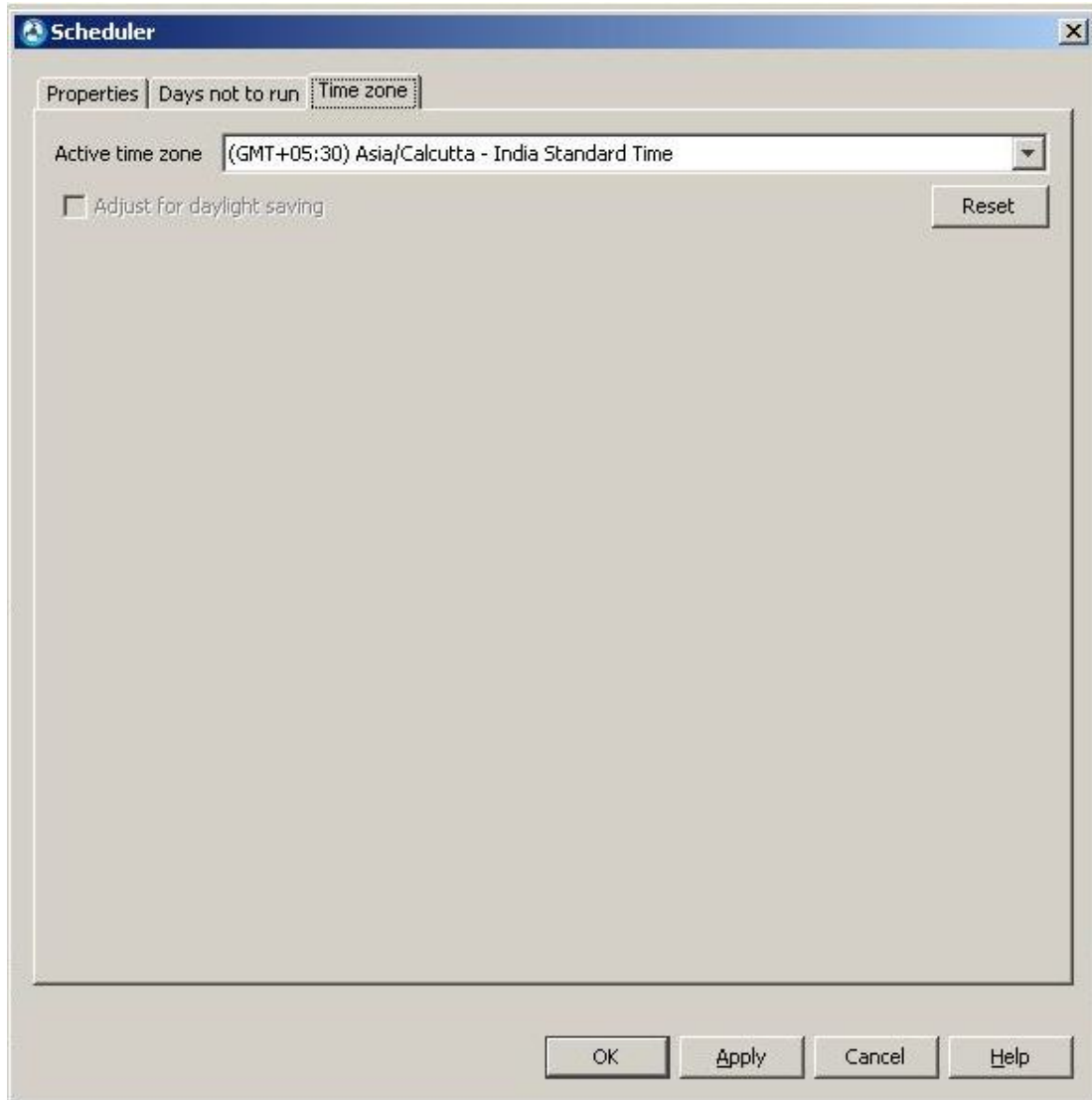
- Select the **Properties** tab and select a **Schedule** from the drop-down list.

The fields and controls on the **Properties** page are different for each of the schedule types, and selecting a schedule will display only the relevant ones. See the appropriate Help topic for your chosen schedule, as follows:

- [Daily](#) - fires the EMF Process every day, or at a specified daily interval.
 - [Weekly](#) - fires the EMF Process every week, or at a specified weekly interval. You can choose the days on which you want the EMF Process to run.
 - [Monthly](#) - fires the EMF Process every month, or at a specified monthly interval. You can specify the date in each month on which the EMF Process should run.
 - [Single Event](#) - fires the EMF Process once only.
- If you want to prevent the EMF Process running on one or more specific days, select the [Days not to run](#) tab and specify the relevant days.



6. If the intended recipients of the EMF Process are in a different time zone, select the [Time zone](#) tab and select the relevant zone.



[Go to Start of Input Modules](#)

[Scheduled EMF Processes Retriever Server Instance Screen](#)

[Introducing EMF](#)

Firing an EMF Process on a Daily Basis

You can use the **Properties** page of the [Scheduler module](#) to define the day(s) and time(s) that an EMF Process should run.

1. To fire an EMF Process on a daily basis, double-click the icon for the Scheduler module and select **Daily** from the **Schedule** field on the main **Properties** page. The other schedule types that are available are [Weekly](#), [Monthly](#), and [Single Event](#).

2. Specify the first date on which the EMF Process should be fired in the **Start** field. You can either type the date manually, use the up and down arrows to increment the numbers singly, or click the drop-down list button and select the required date from a calendar.
3. Enter the required interval between firings in the **Interval (days)** field. To run the EMF Process every day enter 1, to run it every three days enter 3, etc.
4. If you want to prevent the EMF Process from running after a certain date, select the **End** option and enter the required date.

Note: Unless you specify an end date, the EMF Process will continue to fire indefinitely at the specified intervals.

5. Specify the time(s) of day when the EMF Process should run by completing the appropriate fields in the **Time** fields.

Note: The time selectors can only accept two-figure numbers (i.e. they will accept 12:40:45 but not 123:40:45). If you inadvertently enter the wrong number in the hours, minutes or seconds section you will need to click elsewhere, and again in the relevant field to change it again.

- Select **Single** if you only want to fire the EMF Process once per day, and specify the required time.

Note: It is advisable to always specify an **End** time and, if you select **Repeating**, this option is selected by default.

- Select **Repeating** if you want to fire the EMF Process more than once per day. You can then specify the **Start** time (i.e. the first time to fire the EMF Process), **End** time (i.e. when to stop firing the EMF Process), and the **Interval** to wait as a number of **Units**.

[Explanation of Scheduler module](#)

[The Days not to run tab](#)

[The Time Zone tab](#)

[Go to Start of Initiator Modules](#)

Firing an EMF Process on a Weekly Basis

You can use the **Properties** page of the [Scheduler module](#) to define the day(s) and time(s) that an EMF Process should run.

1. To fire an EMF Process on a weekly basis, double-click the Scheduler module icon and select **Weekly** from the **Schedule** field on the main **Properties** page. The other schedule types available are [Daily](#), [Monthly](#), and [Single Event](#).
2. Specify the first date on which the EMF Process should be fired in the **Start** field. You can either type the date manually, use the up and down arrows to increment the numbers singly, or click the drop-down list button and select the required date from a calendar.

3. Enter the required interval between firings in the **Interval (weeks)** field. To run the EMF Process every week enter 1, to run it every three weeks enter 3, etc.
4. Select the days on which you want the EMF Process to fire in the **Days of week** field.
5. If you want to prevent the EMF Process from running after a certain date, select the **End** option and enter the required date.

Note: Unless you specify an end date, the EMF Process will continue to fire indefinitely at the specified intervals.

6. Specify the time(s) of day when the EMF Process should run by completing the appropriate fields in the **Time** fields.

Note: The time selectors can only accept two-figure numbers (i.e. they will accept 12:40:45 but not 123:40:45). If you inadvertently enter the wrong number in the hours, minutes or seconds section you will need to click elsewhere, and then again in the relevant field to change it again.

- Select **Single** if you only want to fire the EMF Process once per day, and specify the required time.
 - Select **Repeating** if you want to fire the EMF Process more than once per day. You can then specify the **Start** time (i.e. the first time to fire the EMF Process), **End** time (i.e. when to stop firing the EMF Process), and the **Interval** to wait as a number of **Units**.
-

Note: It is advisable to always specify an **End** time, and if you select **Repeating**, this option is selected by default.

[Days not to run tab](#)

[Time Zone tab](#)

[Explanation of Scheduler module](#)

[Go to Start of Initiator Modules](#)

[Go to Start of EMF Help](#)

Firing an EMF Process on a Monthly Basis

You can use the **Properties** page of the [Scheduler module](#) to define the day(s) and time(s) that an EMF Process should run.

1. To fire an EMF Process on a monthly basis, double-click the Scheduler module icon and select **Monthly** from the **Schedule** field on the main **Properties** page. The other schedule types available are [Daily](#), [Weekly](#), and [Single Event](#).
2. Specify the first date on which the EMF Process should be fired in the **Start** field. You can either type the date manually, use the up and down arrows to increment the numbers singly, or click the drop-down list button and select the required date from a calendar.

3. Enter the required interval between firings in the **Interval (months)** field. To run the EMF Process every month enter 1, to run it every three months enter 3, etc.
4. To fire the EMF Process on the same date each month, select **Day number** and enter the required date. To fire the EMF Process on the same relative day (e.g. the 2nd Thursday) of each month, select **Occurrence** and select the required day.
5. If you want to prevent the EMF Process from running after a certain date, select the **End** option and enter the required date.

Note: Unless you specify an end date, the EMF Process will continue to fire indefinitely at the specified intervals.

6. Specify the time(s) of day when the EMF Process should run by completing the appropriate fields in the **Time** fields.

Note: The time selectors can only accept two-figure numbers (i.e. they will accept 12:40:45 but not 123:40:45). If you inadvertently enter the wrong number in the hours, minutes or seconds section you will need to click elsewhere and then back in the relevant field in order to change it again.

- Select **Single** if you only want to fire the EMF Process once per day, and specify the required time.
- Select **Repeating** if you want to fire the EMF Process more than once per day. You can then specify the **Start** time (i.e. the first time to fire the EMF Process), **End** time (i.e. when to stop firing the EMF Process), and the **Interval** to wait as a number of **Units**.

Note: It is advisable to always specify an **End** time and, if you select **Repeating**, this option is selected by default.

[Days not to run tab](#)

[Time Zone tab](#)

[Explanation of Scheduler module](#)

[Go to Start of Initiator Modules](#)

[Go to Start of EMF Help](#)

Firing an EMF Process Once Only

You can use the **Properties** page of the [Scheduler module](#) to define the day(s) and time(s) that an EMF Process should run.

1. Double-click the icon for the Scheduler module and select **Single event** from the **Schedule** field on the main **Properties** page. The other schedule types available are [Daily](#), [Weekly](#), and [Monthly](#).
2. Specify the date to fire the EMF Process in the **Date** field. You can either type the date manually, use the up and down arrows to increment the numbers singly, or click the drop-down list button and select the required date from a calendar.
3. Select **Single** to fire the EMF Process once only, then specify the time of day that it should run.

Note: The time selector can only accept two-figure numbers (that is, it will accept 12:40:45 but not 123:40:45). If you inadvertently enter the wrong number in the hours, minutes or seconds section, you will need to click elsewhere and then again in the relevant field in order to change it again.

[Days not to run tab](#)

[Time Zone tab](#)

[Explanation of Scheduler module](#)

[Go to Start of Initiator Modules](#)

[Scheduled EMF Processes Retriever Server Instance Screen](#)

[Go to Start of EMF Help](#)

Setting a Time Zone

The **Time Zone** tab is used to define the standard time to be used for deciding the time when an EMF Process should be fired by the Scheduler module.

- **Active Time zone** - Used to select the time zone to be used when defining the time an EMF Process should be fired by the Scheduler module.
- **Active Time zone Example** - If the EMF machine uses "Greenwich Mean Time", but the EMF Process is for use by customers in a different country, then the Active Time zone can be set to whichever time zone is relevant, and the hourly adjustments will automatically be made from within EMF.
- **Use daylight savings** - If the **Active Time zone** selected uses daylight saving then this checkbox can be activated to adjust the time by an hour when applicable.
- **Reset** - When this button is used, the **Active Time zone** and **Use daylight savings** values are changed to those of the computer that the EMF System is installed on.

[Days not to run tab](#)

[Explanation of Scheduler module](#)

[Go to Start of Initiator Modules](#)

[Introducing EMF](#)

Specifying Days not to run an EMF Process

You can use the **Days not to run** tab to prevent an EMF Process from running on one or more specific days, even if they were scheduled to do so, according to the [Properties](#) tab.

- **To prevent an EMF Process from running on a particular day**, click on the required date. Click again to re-enable the EMF Process for that day.
- **To move back or forward one year**, use the horizontal scrollbar.
- **To cancel all specified "days not to run"**, click the **Clear** button.

[Time Zone tab](#)

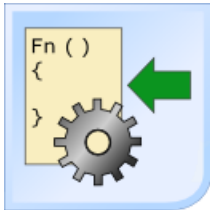
[Explanation of Scheduler module](#)[Go to Start of Initiator Modules](#)[Introducing EMF](#)

The API In Module

If you want to fire an EMF Process using a database trigger (see below), or if you want the EMF Runtime API to fire it, you must include an **API In** module. Without this module, triggers and applications that use the EMF Runtime API will not be able to fire the EMF Process.

To use the API module:

Drag the **API In** icon to the builder screen at the appropriate place.



There are no settings or properties for this module.

Firing a Process using the EMF Runtime API

Methods for calling the EMF Runtime API from an application in order to fire an EMF Process are described in the EMF [Java API](#) guide.

Firing an EMF Process using a SQL Server database trigger

The SQL Server trigger generator is a database trigger generator, available in EMF, which fires an EMF Process based on changes within a SQL Server data source. For the SQL trigger to work, you must first install and [configure the EMF SOAP API and server](#).

When changes to a SQL Server data source occur, the trigger sends information to the SOAP-enabled EMF Runtime API. This in turn passes the request via the SOAP Server to the EMF Server.

Removing a database trigger: Removing a database trigger is a database-level task. The command to use in your command-line or GUI utility is `DROP TRIGGER trigger_name`. You may need specific security rights to drop triggers - check with your database administrator for more information.

[Go to Start of Input Modules](#)[Introducing EMF](#)

The HTTP In Module

You can use the **HTTP Initiator** module to fire an EMF process whenever an [HTTP Listener](#) receives an incoming request.

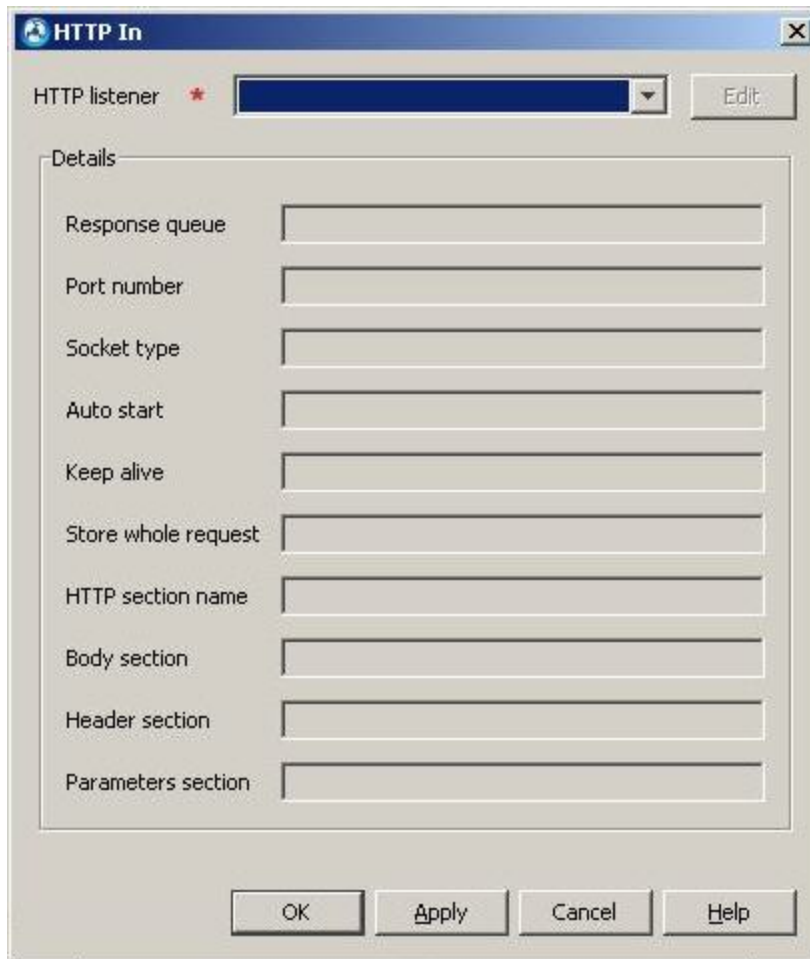
Note: In order to use an HTTP In module, you must have set up an **HTTP Listener** server instance. If you want to be able to send responses back to the client from an HTTP module in an EMF process, ensure that you include and configure an [HTTP module](#) with **Client response** enabled. This HTTP module can be in the same EMF process as the HTTP in module (see [example](#)), or it can be in a slave EMF Process cascaded to by the EMF process containing the HTTP in module (see [example](#)).

To use an HTTP In module:

1. Ensure that you have created one or more [HTTP Listener server instances](#).
2. Drag an **HTTP In** icon into your EMF Process and create a link from it to the **Start** module.



3. Double-click the HTTP In module to display the **HTTP In screen**.



4. Select the required **HTTP Listener** from the drop-down list of those that you have set up.
5. Click **Edit** to modify the properties of the HTTP listener.

[HTTP Router Module](#)

[How HTTP Works in EMF](#)

[Go to Start of Input Modules](#)

The Email In Module

You can use the **Email In** module to fire an EMF Process upon receipt of an email message.

Note: in order to use an Email In module you must have a [POP3 listener instance](#) or an [IMAP listener instance](#) set up on your machine, in order to specify the email account that should be monitored for incoming messages. You must also have an appropriate [System EMF Process](#)

(e.g. the supplied POP3SystemAlert) available in order to check active EMF Processes for Email In modules.

To use an Email In module:

1. Ensure that you have created at least one **POP3 listener** server instance or **IMAP listener** server instance.
2. Drag an **Email In** module icon into your EMF Process and create a link from it to the **Start** module.



3. Double-click the Email In module to display the **Email IN Properties** screen.

4. Select the required **Email listener** from the drop-down list of those that you have set up.
5. Click **Edit** to modify the properties of the Email listener.

[Email Router Module](#)

[POP3 Listener](#)

[IMAP Listener](#)

[Explanation of Email Router](#)

[Introducing EMF](#)

[Go to Start of Input Modules](#)

The Queue In Module

The **Queue In** module allows you to initiate an EMF Process based on an incoming JMS queue message.

Note: To use a Queue In module, you must have set up an [External Queue Provider](#), an [External Queue](#), and a [Queue Listener](#) server instance. You must also have an appropriate [System EMF Process](#) available that contains the [Queue Router Module](#) (e.g. the supplied **QueueInSystemAlert**) in order to decide which EMF Processes should be fired when a queue message is received by a Queue Listener server instance.

To use a Queue In module:

1. Ensure that you have created at least one **External Queue Provider**, one **External Queue** and one **Queue Listener server instance**.
2. Drag a Queue In icon into your EMF Process and create a link from it to the **Start** module.



3. Double-click the Queue In module to display the **Queue In** screen.

4. Select the required **Queue listener** from the drop-down list of those that you have set up.
5. Click **Edit** to modify the properties of the Queue listener.

[Queue Router Module](#)

[Go to Start of Input Modules](#)

The SNMP In Module

The **SNMP In** module allows you to initiate an EMF Process based on an incoming SNMP message.

Note: To use an SNMP In module, you must have set up an [SNMP Listener](#) server instance.

To use an SNMP In module:

1. Ensure that you have created at least one **SNMP Listener**.
2. Drag an SNMP In icon into your EMF Process and create a link from it to the **Start** module.



3. Double-click the SNMP In module to display the **SNMP In** screen.

The dialog box is titled "SNMP In" with a standard Windows window border. At the top, there is a label "SNMP listener" followed by a red star icon and a drop-down menu. To the right of the drop-down is an "Edit" button. Below this is a section titled "Details" which contains five checkboxes: "v1 Traps", "v2c Traps", "v3 Traps", "v2c Informs", and "v3 Informs". Below the checkboxes are four text input fields labeled "Port", "Community", "Data section", and "OID section". At the bottom of the dialog are four buttons: "OK", "Apply", "Cancel", and "Help".

4. Select the appropriate **SNMP listener** from the drop-down list of those that you have set up. The EMF Process will be fired when a message is received on the port being "listened" to.
5. Click **Edit** to modify the properties of the SNMP listener.

[SNMP Router Module](#)

[Configuring an SNMP Listener Server Instance](#)

[SNMP Overview](#)

[Go to Start of Input Modules](#)

Start Module

The Start module forms the link between the initiator and processing modules. It must be present in all EMF processes, and as a result, is automatically added on the design graph whenever a new EMF Process is created.

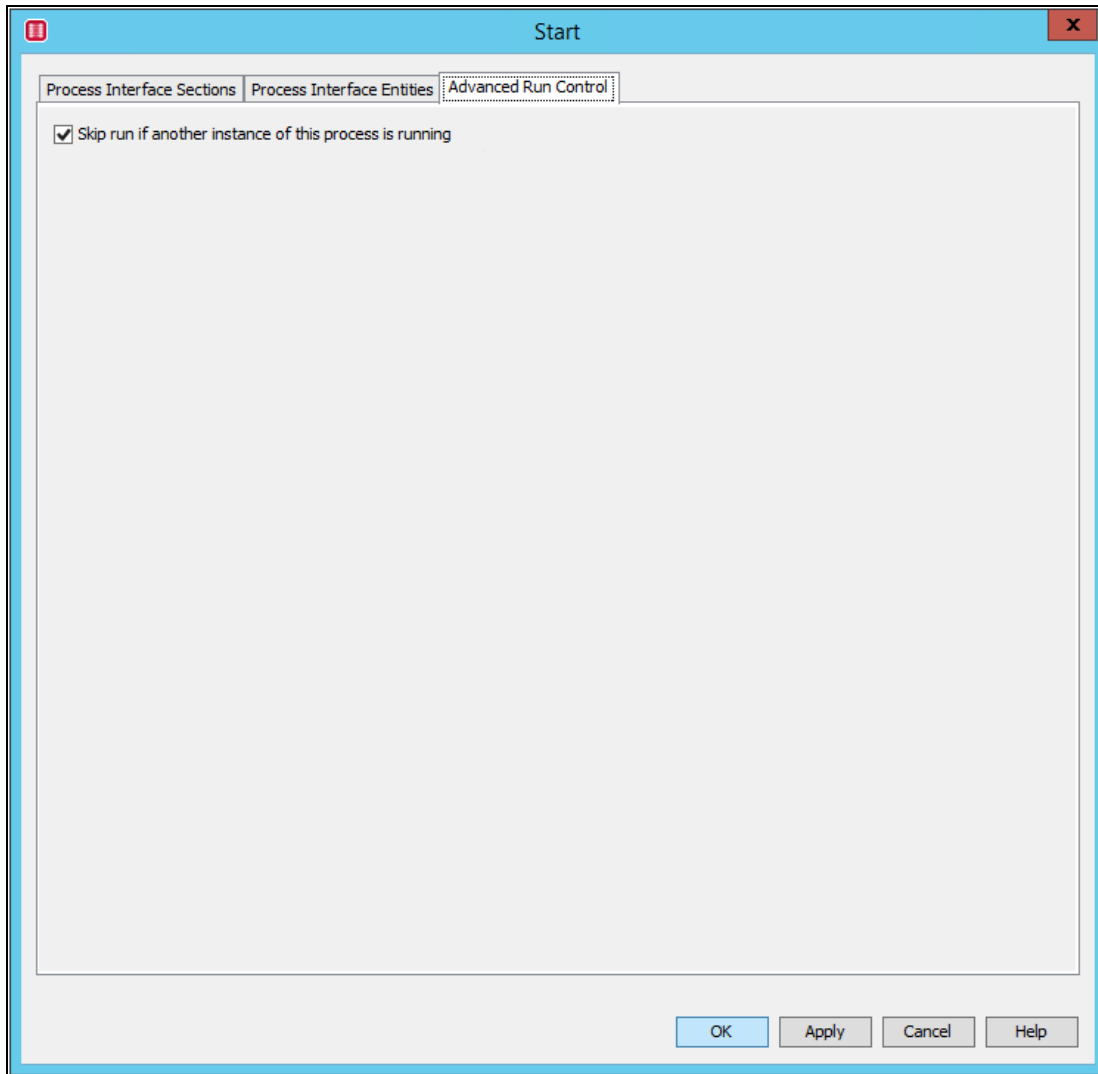
The Start module is also where the Process Interface and Skip settings are defined. Opening up the Start module properties displays the process interface setting and the advanced run control settings.

For information on the Process Interface Sections tab or Process Interface Entities tab in the Start module properties, see: [Process re-use with Process Interface and Call module](#)

For information on the Advanced Run Control tab in the Start module properties, see: [The Advanced Run Control Tab](#)

The Advanced Run Control Tab

The **Advanced Run Control** tab allows a process to run only when no other instance of the process is already running. For this, the checkbox **Skip run if another instance of this process is running** in the **Advanced Run Control** tab and the checkbox **Log process start/complete to audit log** in the **Process properties** window must be checked.



Process properties - Test

Properties | Sent Message Logging | Error handler

Name: Test

API name: Test_1447829919874

Description: for demo

Retries

Number of retries: 0

Delay before retrying: 0 second(s)

Debug log level: Errors and warnings ☒ Log user messages

Priority: Medium

Status: ☐ Active ☒ Hold ☐ Incomplete

☒ Persist Process when queued ☒ Statistics logging ☐ Capture Data

Dead Letter Queue (DLQ)

Delete failed processes from DLQ after: 172800 second(s) ☒ Use default expiry time

Process run time

☒ Log process start/complete to audit log

Expected run time period: 1

Expected run time interval: Hour(s)

OK Apply Cancel Help

If the checkbox **Skip run if another instance of this process is running** is checked, while the **Log process start/complete to audit log** checkbox is not checked, then on clicking **OK/Apply** in the **Start** module, a warning is displayed and the process can only be saved in an "incomplete" state.

How Processes Are Skipped

When a process is run (either manually, initiated by a scheduler or another initiator, or cascaded to/called) then it is placed on a queue. It is only when a server manager then picks up the message from the queue to execute it, is the decision made whether to skip it or not. If another instance of the same process is already executing and the skip option is

selected, then this instance will not execute and will be skipped. A skipped process is shown in the [Process Status Monitor](#) with a status of skipped.

If the process was initiated by a call or cascade module, and it is skipped, then the parent call/cascade module will resume. It is possible to detect if the child process was skipped by creating a message section in the child process that is used to flag that process has run. If that message section then doesn't exist when the process returns, then it can be assumed to have been skipped.

Data Modules

Data modules are used to gather and process information from databases, using JDBC and/or SAP connections, and insert this information into a freshly created **Data Section**. Other modules within the EMF Process can then use this data by referencing the Data Section name.

You can use any number of Data modules in an EMF Process, and each one will place the processed data into a new and separate Data section with a name that you specify.

Important: You must ensure that each Data module creates a Data section with a unique name. If a Data section is given the same name as one that already exists, it will overwrite the original one.

Before you can use a Data module in an EMF Process, you must create a connection to the required databases using the **JDBC connections** and **SAP connections** nodes in the **Data sources** section of the EMF tree view. Once a connection has been established to a data source, all data within it can be accessed by a Data module in an EMF Process.

The following are the Data modules in EMF:

- The [SQL Module](#) gathers and processes data using [JDBC connections](#).
- The [SQL Insert Module](#) allows data to be inserted into a SQL database from an EMF data section.
- The [SAP Module](#) and [SAP Data Browser Module](#) gather and process data using [SAP R/3 Connections](#).
- The [HTTP Module](#) retrieves data from a Web server on the Internet and stores it in a message section. It can also be used to send data back to an Internet address.
- The [JNDI Module](#) retrieves data from a JNDI provider and stores it in a data section.
- The [Aptean Respond module](#) allows Respond Web service calls to be made to Respond (a case management tool). EMF can be used to automate the creation and updating of cases in Respond.
- The [Data Store module](#) allows you to add or update items in the [Data store](#).
- The [Web Service Module](#) allows you to map EMF data and message sections onto the parameters required by the Web service call without having dependencies on the underlying message protocol.
- The [SAP Data Browser Module](#) allows you to connect to SAP R/3 Data Sources and extract information to populate an EMF Process Data section.
- The [File In Module](#) retrieves data from a directory/files from a number of sources (local file system, FTP and so on) and stores them in a data section.

- The [FTP In Module](#) allows you to retrieve information and contents of files and directories from an FTP server.
- The [SAP Authorization Module](#) allows you to assign general and/or finely detailed user authorizations.
- The [Segregation of Duties Module](#) in combination with the SAP Authorization Module allows you to identify users who have conflicting authorizations within their profiles.
- The [SNMP module](#) allows you to send and receive SNMP data.
- The [SAP System DS Data Browser Module](#) allows you to connect to SAP (Systems Applications and Products in Data Processing) R/3 Data Sources and extract information from them in order to populate an EMF Process Data section.
- The [OPC DA Read Module](#) allows you to read values from an OPC server.

SQL Module

You can use a **SQL Module** to connect to JDBC Data Sources, gather data from them (using a SQL statement), and perform certain formatting tasks on that data. The following Help topics explain how to access configured JDBC Data Sources and how to use the data that is in them.

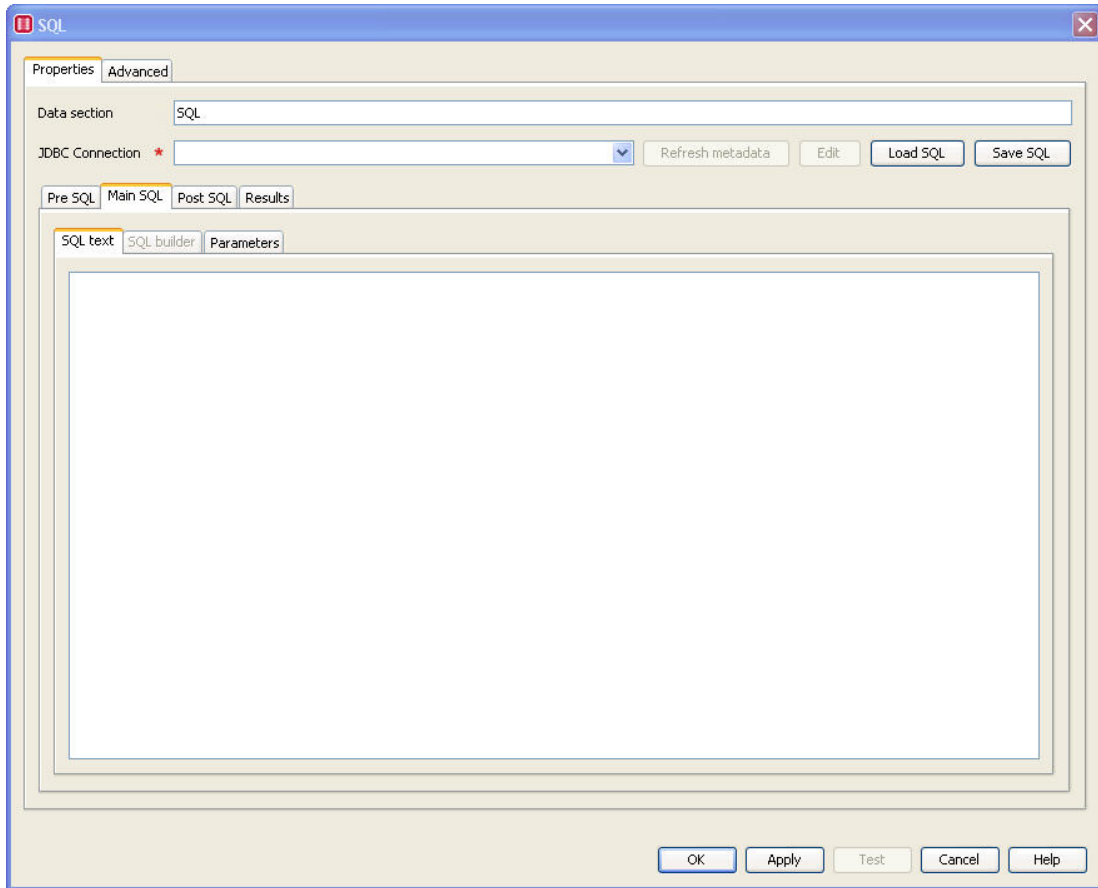
If you are using a SQL Module because you need to know when changes have occurred within a Data Source you should use a Database Trigger to fire the EMF Process. This is done by adding an [API Server In](#) module to the EMF Process and creating a database trigger to check the data source for your specified criteria.

To use an SQL Module:

1. Ensure that you have created a [JDBC connection](#) to any JDBC Data Sources that you want to access (including the EMF Repository or Configuration Databases).
2. Drag the **SQL module** icon into the EMF Process you are creating at the appropriate place.



3. Double-click the module icon to display the **SQL** screen.

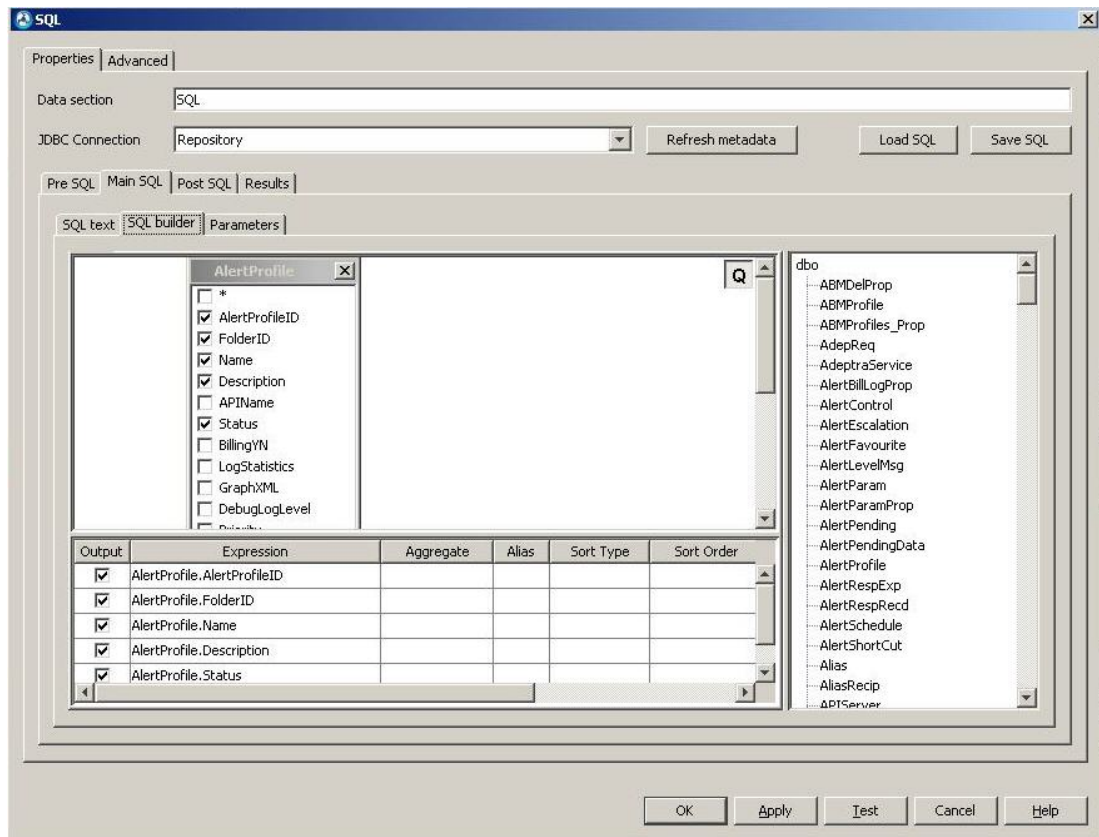


4. Select the required connection from the list of those that you have set up that is shown in the **JDBC connection** drop-down list. This will enable the other tabs.

Metadata provides the names of tables and columns in the database. These tables and columns are displayed on the SQL builder tab, where you can visually build database queries.

Note: The refresh operation happens in the background, but you can continue to work simultaneously.

5. The **SQL builder** tab is updated after the refresh is complete. However, if you think the metadata has changed since the last time it was refreshed (new tables or columns added), click **Refresh metadata** to refresh the metadata saved in the database and then use the visual SQL builder.
6. If you do not want to create the SQL query manually, select the tables and fields that you want to use on the SQL builder tab. This automatically generates your SQL. You can manually refine the SQL on the **SQL text** tab.



7. If you have experience with SQL and feel comfortable creating queries manually, you can enter the SQL statement directly in the **SQL Statement** tab. Select the [Parameters](#) tab to make your query dynamic.

Note: SQL module can capture return values from stored procedures.

8. Select the [Advanced](#) tab to define halt commands and insert data into another database table (that is, perform a data copy between tables in the same or a different database).

Notes for those who create the SQL statement manually:

- You can use the standard SQL language and syntax, although you should note that the word "execute" is not required.
- You can use the **Edit** button to modify the properties of the selected JDBC connection.
- The **Load SQL** button allows you to load a predefined SQL statement from an external source.
- You can enter any Stored Procedures or SQL that you need to run before or after running the main SQL or Stored Procedures by selecting the Pre and/or Post tabs on the Report Statement tab.
- You can use [Dynamic Functions](#) by right-clicking in the SQL statement screen and selecting from those available. However, **you cannot use the RECIPIENT dynamic function in a WHERE clause** in a SQL module, because these are only evaluated in the output module. See [Using Dynamic Functions in SQL Statements](#) for further information.

- When performing string comparisons and assignments in SQL statements, you must put quotation marks (single or double, depending on your database server) around dynamic functions. An example of this (using single quotes) would be:

```
SELECT '$DATA('CSQL',0,0)$', '$DATA('CSQL',1,0)$'
FROM warehousedb.dbo.warehouse_manager
WHERE warehouse_manager.warehouse = ?
```

- You can get an idea of the efficiency of your query by checking the **Time taken to run last SQL** field, which indicates the time (in seconds) the query took to run.
- When you finish, select the **SQL Statement** tab (or click the "Show" button) to display the SQL in full.
- If you wish, click the **Test** button to check the syntax of the SQL statement without having to run the EMF Process and display the information that it collects in a **Results** screen (note that the appropriate JDBC driver must be installed on the PC in order to test the statement).
- Click **Save as sample** to save the data for testing purposes. You can create sample results with data sections that can be accessed in subsequent sections for further processing.

SQL Text Tab

The **SQL text** tab is used to construct and test the SQL statement (note that the appropriate JDBC driver must be installed on the PC in order to test the statement).

Warning: If you want to copy and paste SQL written in Oracle SQL Plus, for example an Oracle function, ensure that you remove the CRLF (carriage return / line feed) characters before you attempt to test and run the SQL.

Note: The other tabs on the SQL builder dialog are not available until you have established a connection to the required database (see step 2 below).

1. Enter a name for the data section that will be created by the SQL Module in the **Data Section Name** field. This name is referenced by subsequent modules further along in the EMF Process when referencing and formatting the data gathered by the SQL statement.
2. Select the data source that you want to use from the **JDBC connection** drop-down list that displays all the [JDBC Connections](#) you have set up, and then click **Open connection**. This will populate the **Report Items** tab with the tables that are available in the data source and enable the other tabs.
3. If you already have a SQL statement that you want to load from an external source, click the **Load SQL** button to browse for it.
4. When you finish, select the **SQL text** tab (or click the "Show" button) to display the SQL in full. If you wish, click the **Test** button to check the syntax of the SQL statement without having to run the EMF Process and display the information that it collects in a **Results** screen (if the JDBC Connection is not set up, you will be prompted for a database).
5. You can get an idea of the efficiency of your query by checking the **Time taken to run last SQL** field, which indicates the time (in seconds) the query took to run.

6. To enter any Stored Procedures or SQL that you need to run before or after running the main SQL or Stored Procedures, select the Pre and/or Post tabs on the Report Statement tab.
7. To add a dynamic function, right-click in the SQL statement field and select from those available.
8. Click **Save as** to export your SQL query as a text file.
9. Select the [Advanced](#) tab to define halt commands and insert data into another database table (i.e. perform a data copy between tables in the same or a different database).

Pre SQL Tab

The **Pre SQL** tab allows you to enter any Stored Procedures / SQL that need to run before those in the **Main SQL** tab.

For example: `data_security_context_pkg.set_userid('UserName')`. This stored procedure could typically be run as `"execute data_security_context_pkg.set_userid('UserName');"` within SQL Plus.

You can use [Dynamic Functions](#) by right-clicking in the SQL statement screen and selecting from those available.

Dynamic Functions Note - The data returned by the SQL module for use within the EMF Process cannot be accessed from within the Pre Tab.

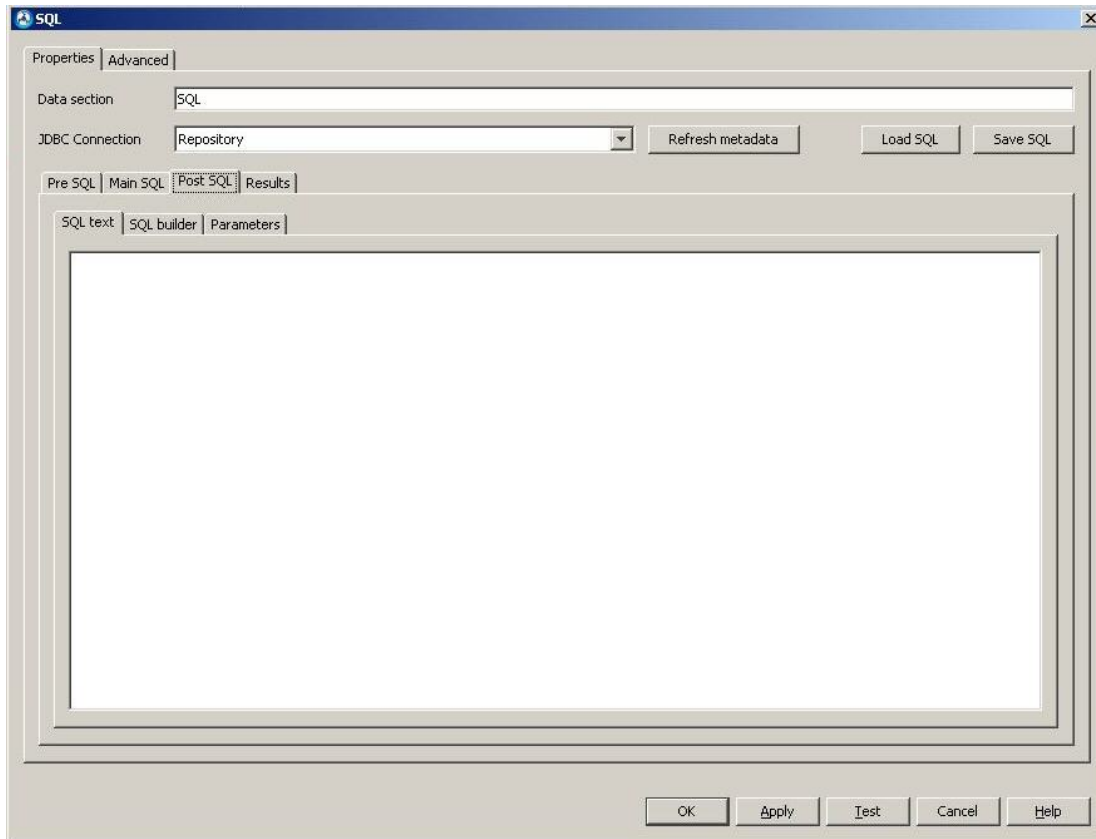
Pre Tab Note - Only the **Main SQL** tab will return data that can be used within the EMF Process.

Example of Use

A user needs to set the Security Context for an Application User, so that the SQL will only run on the tables/columns that the Application User is allowed to access. The user will therefore, set the Security Context in the Pre Tab (this could be a Stored Procedures - e.g. `data_security_context_pkg.set_userid('UserName')`). Then enter SQL in the **Main SQL** tab, which will return.

Post SQL Tab

The **Post SQL** tab is used to enter any Stored Procedures / SQL that you need to run after running the main SQL / Stored Procedures.



You can use [Dynamic Functions](#) (including on the Data Section created in the **Main SQL** tab, as the SQL in this has already run) by right-clicking in the SQL statement screen and selecting from those available.

Post Tab Note - Only the **Main SQL** tab will return data that can be used within the EMF Process.

Example of Use

A user enters SQL in the **Main SQL** tab, which will return the data in a Data Section. If any post processing needs to be done, this can be done in the **Post SQL** tab (e.g. Update a flag in a Table to indicate that **Main SQL** tab processing has been done).

[Connecting to a JDBC Data Source](#)

[Go to start of Data Modules](#)

Advanced Tab

The SQL Module **Advanced** tab provides the following options:

- To select when and if to halt processing of the SQL statement, and also to restrict data that is returned to only that which has changed since the last time the EMF Process ran.

- Allows data that the module will return, to be inserted directly into a database table. This allows a very efficient means of copying data from one table to another in the same/different database.
- Defines the message section that contains the number of rows inserted, updated or deleted during the SQL operation.

The screenshot shows the 'SQL' configuration window with the 'Advanced' tab active. The 'Halt conditions' section has 'Never' selected. The 'Batch insert' section has 'Use batch insert' checked. The 'Destination JDBC connection' is 'Repository' and the 'Table to insert records into' is 'AdepReq'. There are several checkboxes for table and data management, with 'Create table if it does not exist' and 'Create indexes on columns' checked. A list of field names contains 'ABC'. The message section for row counts is configured. Standard 'OK', 'Apply', 'Test', 'Cancel', and 'Help' buttons are at the bottom.

Note: No data section is created and no data is returned from this module if the **Use batch insert** and "Delete data section after batch insert" options are selected. In that case, the data is inserted directly into the database table. If the data then has to be processed immediately after this module, it will need to be queried from the table it was just inserted into.

Halt Conditions

1. Select a **Halt condition** option. The following options allow you to specify whether the returned data section should be checked, and if so, whether the EMF Process processing should be halted:

- Select **On No Data Returned** if you want to stop the EMF Process when the fields selected for return in the SQL statement are empty.
- Select **On Data Returned** if you want to stop the EMF Process when the fields selected for return in the SQL Statement contain any data.
- Select **Never** if you always want to pass the data section from the **SQL** module, irrespective of whether it contains any data.

Batch Insert

1. Select the **Use batch insert** option to insert the data returned by the module into a database table.
2. Select the **Destination JDBC connection** that identifies the database that the data will be stored into.
3. Click the **Open Connection** button.
4. Select the **Table to insert records into** that will store the data returned by the query. This can either be selected from existing tables in the database, or the name of a table can be entered directly (note that this table must be created before this module runs though, otherwise the module will fail). The table needs to have the same column names as the columns that have been selected in the **Table Metadata** tab. The datatypes of the columns must also match or be compatible.
5. **Create table if it does not exist** - if selected will create a new table in the destination database. The value in the **Table to insert records into** field will be the name of the table.
6. **Delete existing entries first** - if selected this will empty the destination table of any existing data before the results of the query are stored.
7. **Delete data section after batch insert** - if selected will delete the whole data section once the batch insert of data is complete.
8. **Drop table if it exists** - if selected, will drop the table first if it exists in the destination database, before creating the new table.
9. **Create indexes on columns** - if selected, will create database indexes on the fields listed. The field names have to be added using the **Add** button.

Note: A database index is a data structure that increases the speed of data retrieval from a database. Creating a database index might result in slower writes and increased storage space.

Count of rows updated/inserted/deleted message section

Enter the name of the message section that will contain the total number of rows that were inserted, updated or deleted during the SQL operation on the **Main SQL** tab of the **SQL Properties screen**. If batch insert is used, then this will be the number of rows inserted by the batch insert operation, otherwise it will be the total from any SQL UPDATE/INSERT/DELETE commands in the **Main SQL** tab of the **SQL Properties screen**.

If just a SQL SELECT statement is executed then the message section will contain **-1** as the count.

[The SQL Module](#)

[Explanation of SQL Modules](#)

[Connecting to a JDBC Data Source](#)

[Go to start of Data Modules](#)

Parameters Tab

1. To specify a parameter in your query, replace the parameter value with a question mark (?). For each question mark, you then have an entry in the **Parameters** tab.

Example: SELECT * FROM AlertProfile WHERE AlertProfileID = ? AND FolderID = ?

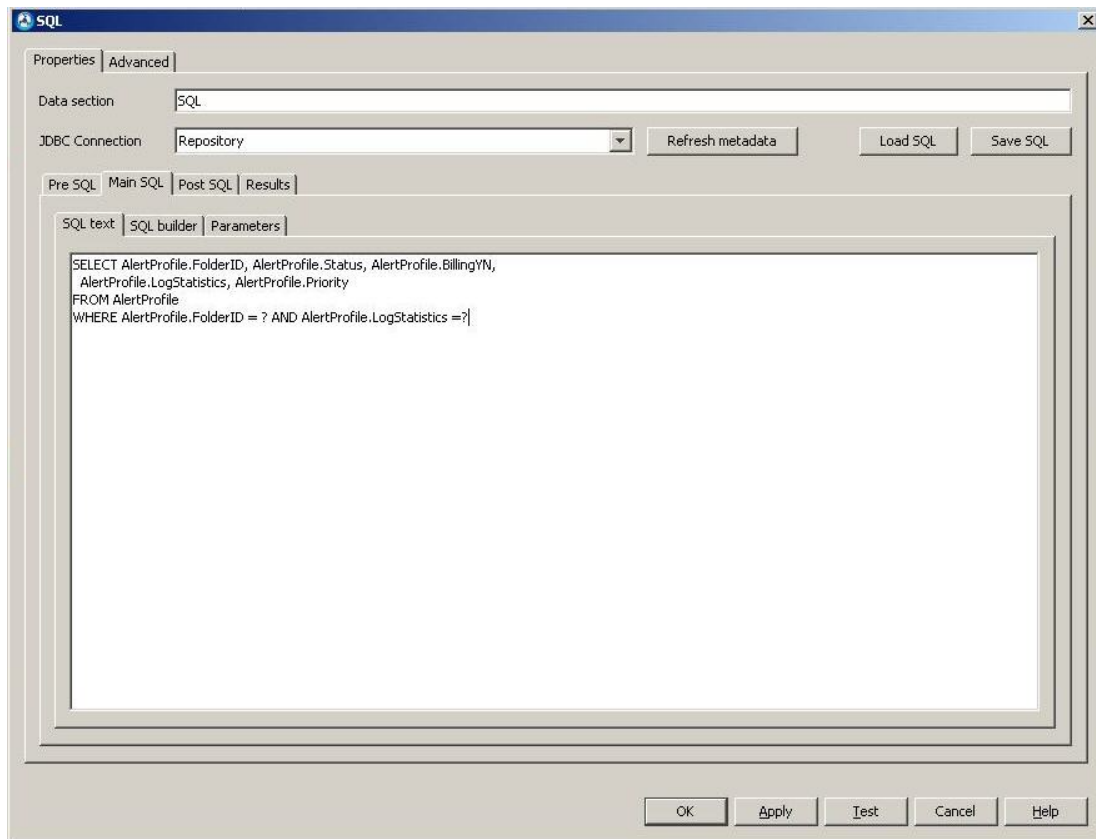
Values for the two question marks are obtained from the values you enter in the Parameters tabs.

Note: The order in which the question marks appear in the query should match the order of the parameters in the Parameters tab.

2. Select the **Out Param** check box if you are calling stored procedures that have output parameters defined within. The corresponding value of the stored procedure will be returned in the output parameter.
3. To change the data type, click on a data type in the **Datatype** field and select a value from the drop-down list.
4. The **Value** field displays the corresponding value that is passed when the process is executed on the server at runtime. Typically, this value is obtained from the preceding modules.
5. For testing purposes, enter a value in the **Test** value field.
6. Select the **Test using parameter values from ’Test value' column (Applies to Pre, Main and Post tabs)** check box if you want to use the value specified in the **Test value** field when you click **Test**. Otherwise, the value displayed in the **Value** field will be used.
7. Additionally, use the following options to add, delete, or change the order of the parameters:
 - Add
 - Remove
 - Remove All
 - Move up/Move down

The **Data section to store results of output parameters** field displays the name of the data section where the results are stored in when you use output parameters (these values are generally returned from stored procedure calls).

Example:



8. Enter values to filter results accordingly.

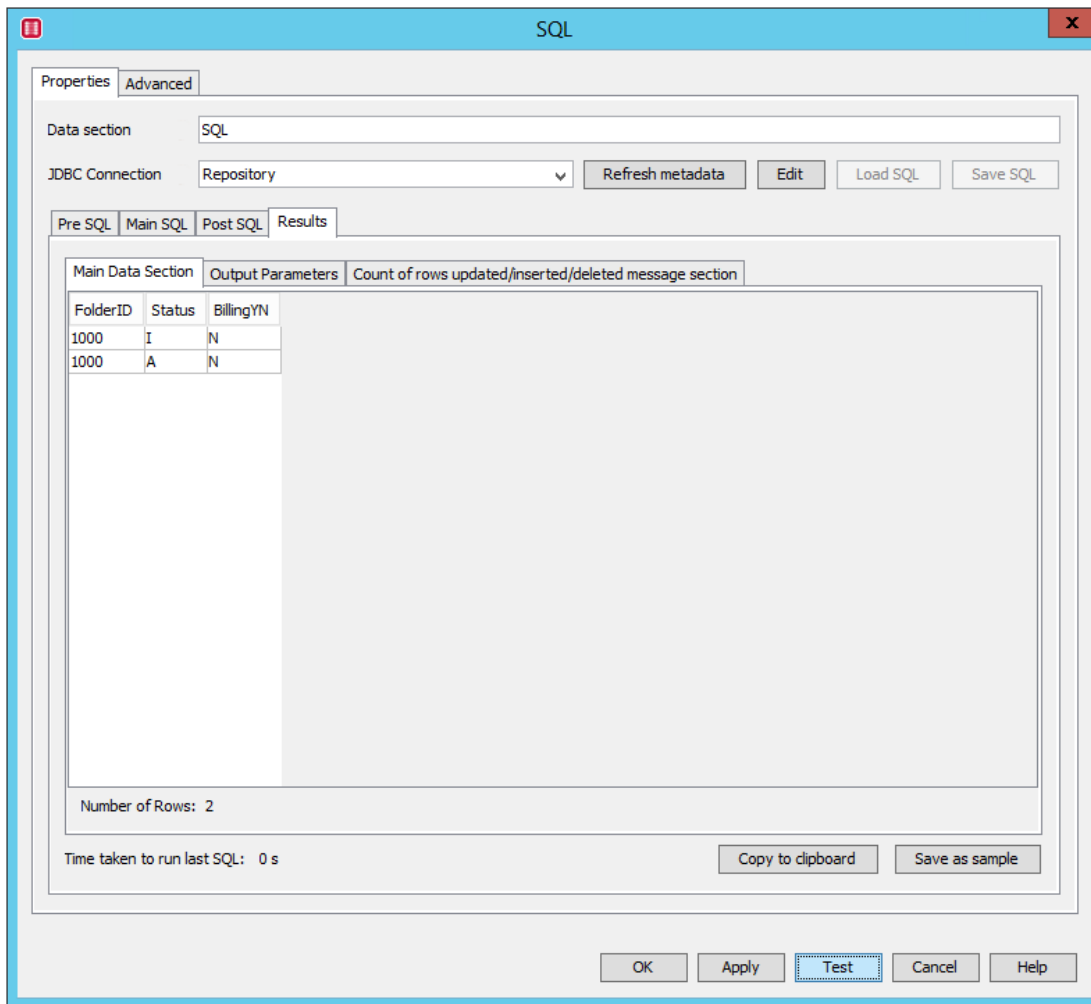
The screenshot shows the 'SQL' Properties dialog box, 'Advanced' tab. The 'Data section' is 'SQL' and the 'JDBC Connection' is 'Repository'. The 'Parameters' section is active, showing a table with two rows: 'Int' with value '1000' and 'String' with value 'y'. The 'Test value' column is empty. The 'Data section to store results of output parameters' is 'Output Param Main'. The 'Test using parameter values from 'Test value' column' checkbox is unchecked.

Out Param	Datatype	Value	Test value
<input type="checkbox"/>	Int	1000	
<input checked="" type="checkbox"/>	String	y	

Buttons: Add, Remove, Remove all, Move up, Move down

Buttons: OK, Apply, Test, Cancel, Help

9. Click **Test** to view the results.



Using Dynamic Functions in SQL Statements

You can insert Dynamic Functions into your SQL statement by right-clicking in the **Pre**, **Main** or **Post** sections of the [SQL statement screen](#) and selecting from the list of available functions.

Note: You cannot use the RECIPIENT dynamic function in a WHERE clause in an SQL module, because these are only evaluated in the output module.

Important: When performing string comparisons and assignments in SQL statements, you must put quotation marks around dynamic functions. These may need to be single quotes or double quotes, depending on your database server.

Example of using (single) quotes:

```
SELECT '$DATA('CSQL',0,0)$', '$DATA('CSQL',1,0)$'
FROM warehousedb.dbo.warehouse_manager
WHERE warehouse_manager.warehouse = '$DATA('CSQL',4,0)$'
```

[The SQL Module](#)

[Using Data Modules to Query Data Sources](#)

SQL Insert Module

The **SQL Insert module** allows data to be inserted into a SQL database from an EMF data section.

Note: The table where the data is to be inserted into need not exist, an option on the module will allow it to be created at runtime.

When a row is inserted, options are available to retrieve any auto-generated row id. This is useful if further data manipulation options need to be performed on the row (such as updating or deleting it later in an EMF process).

To use the SQL Insert Module:

1. Drag the **SQL Insert module** icon to the appropriate location in the EMF Process you are creating.



2. Double-click the module icon to display the **SQL Insert** screen.

The Properties tab

1. Select the required **JDBC Connection** for the database connection to the "table" that data will be inserted into.
2. If the table already exists, click the **Get tables** button to retrieve a list of tables in the database and then select the appropriate table from the **Table to insert into** drop-down list.
If the table does not exist, type a table name to create it. This field will support dynamic functions.
3. Select the **Data section to insert** from the drop-down list of existing data section, that contains the data to be inserted.
If required, enter a section name (that does not currently exist).

4. If an existing table is selected, the **Fields to insert** table is auto-populated with all the fields in the selected table.

You can add/remove fields, if required using the **Add/Remove/Reorder** buttons.

The selected table contains the following columns:

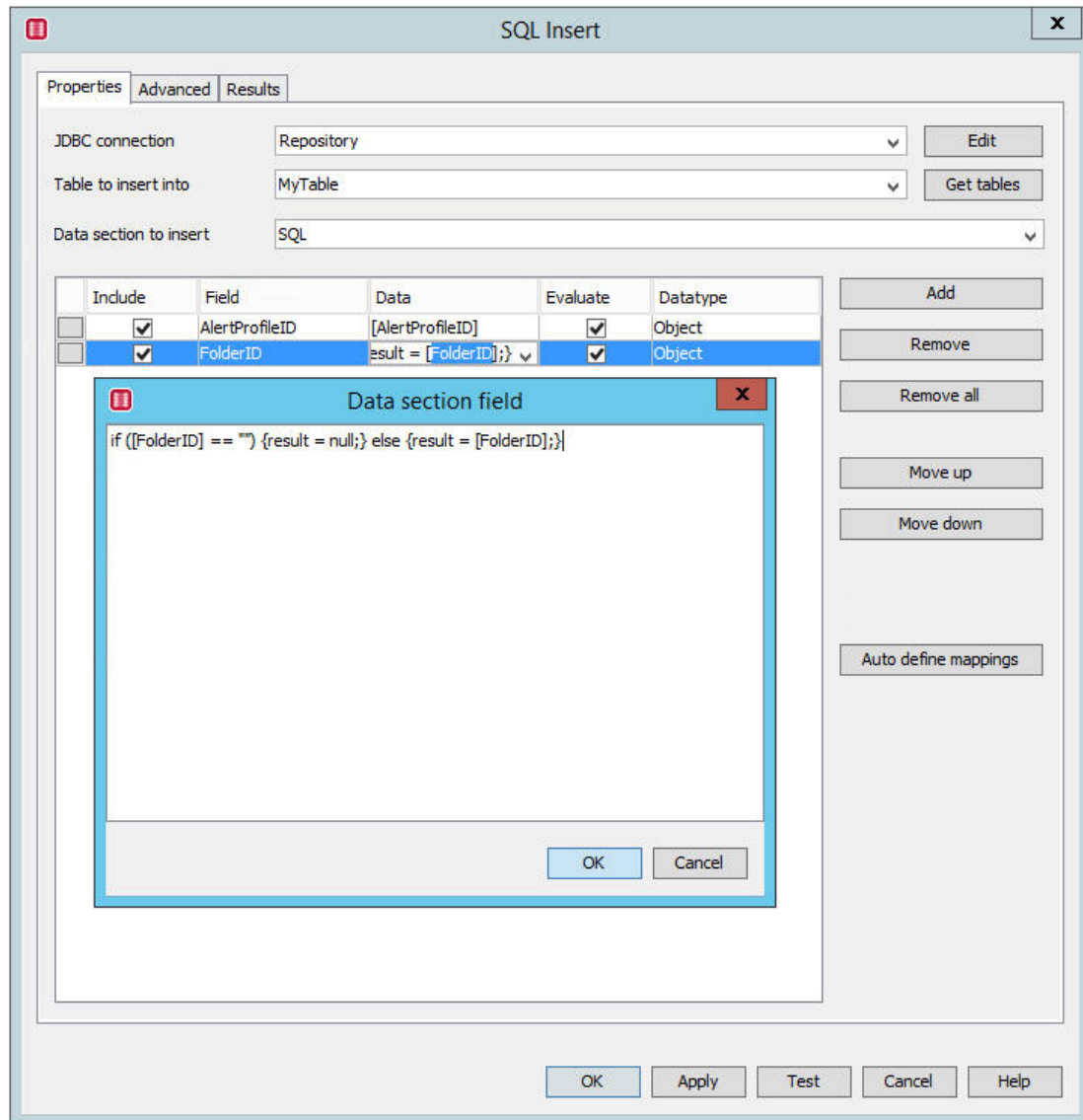
- **Include:** This check box indicates whether this field is included in the insert statement. If it is not selected, then the information about this row is ignored.
- **Field:** Indicates the field name in the database table where data will be inserted. Dynamic functions are supported in this field.
- **Data:** Indicates the expression that will be inserted into the field. This field is an editable combo box. The drop-down list contains all the field names from the data section. You can select a single field from the drop-down list to map data from a data section to the database table. More advanced expressions are allowed to combine multiple fields from the data section to update a single field in the database table.

For example:

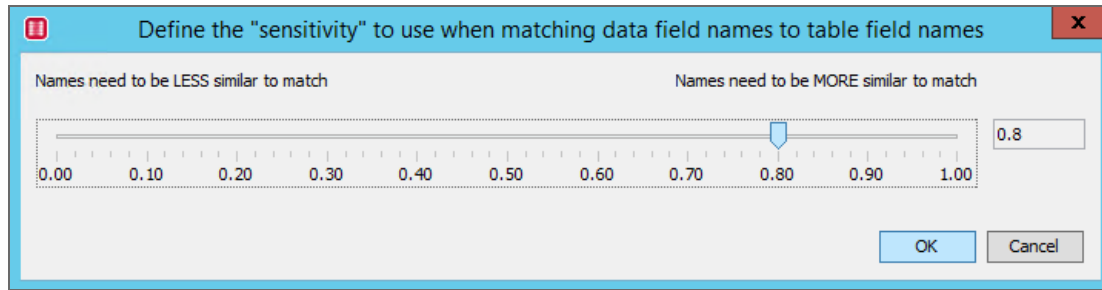
[FirstName] + " " + [LastName]

Points to Note

- It is also possible to assign a constant value to data being inserted, that is, the expression does not have to reference any field in the data section.
- Expressions are entered as Javascript code. A right-click pop-up menu gives a shortcut to any of the data section field names (as well as the standard **Dynamic Function** menus).
- Data section field names are enclosed in square brackets such as **[ExampleField]**
- Double-click action on the field invokes a larger text editor. This allows more complex and multi-lined expressions to be entered easily.



- **Evaluate:** This check box indicates that any field names and script expressions are evaluated at runtime and if this check box left unchecked, then the text entered will be passed through the dynamic function resolver to resolve any dynamic functions, and the result inserted directly into the database – in this case all records inserted will have the same value.
 - **Datatype:** Select the data field type from a non-editable drop-down list. At runtime, the data field is attempted to be converted to the specified type before being inserted. Selecting the "Object" type will allow the server to automatically try to best determine the datatype. This field also is used to calculate the data type field used when the "create table" option is used.
5. Click the **Auto define mappings** button to map the Data column names from the data section to insert to the table Field column.



If you click the **Auto define mappings** button, you will get a dialog box where you can define the sensitivity of the match from 0.01 to 1. If you increase the sensitivity from 0.01 to 1, then the Data column name needs to be more similar to the Field name and should exactly match if the sensitivity increases to 1. For example, if the Field name is **FirstName** and the Data column name is **First** and if you select the sensitivity as 1, the Data column will be blank as the Data column does not match exactly with the Field column. If 0.8 is selected then the Data column displays **First** which is seen as similar enough to match the Field name. If the Data columns do not match with the Field columns, then the Data column will be blank with no change in Datatype column.

If the Fields do not exist (because the tables does not exist/ cannot be connected to), then the Auto Map will create one to one mapping between the Data column and the Field column to create a valid SQL field name for each Data column. In this case the Datatype column will be set based on the Data type of the Data column.

Note: Bulk updates to the **Datatype**, **Include** and **Evaluate** columns can be performed by selecting multiple rows and using the right click menu.

Advanced Tab

The **Advanced** tab on the SQL insert module allows to specify options that are less frequently used.

SQL Insert

Properties Advanced Results

INSERT expression

INSERT INTO EventPublisher(EventID, Category)
VALUES (?, ?)

☐ Enable editing Regenerate

Insertion and result options

☐ Drop table if it exists

☐ Create table if it does not exist

☐ Delete existing entries first using SQL TRUNCATE TABLE

☐ Delete data section after batch insert

☐ Transact each insertion

☐ On error continue, copying failed rows to data section Failed Insertions

Batch size 5000

☐ Create indexes on columns (These will only be created when a table is created)

	Field name
<input type="checkbox"/>	EventID
<input type="checkbox"/>	Category

Results Message Section Rows Inserted Count

☐ After insertion append any database auto-generated IDs to the input data section

OK Apply Test Cancel Help

INSERT Expression options

INSERT expression field displays the actual SQL expression that will be run. By default the field is read-only and the contents are auto updated when any options on the **Properties** tab are changed.

Selecting the **Enable editing** check box makes the field editable. An expression can be manually edited to fine tune it. For example, the user can modify the field to the following:

```
INSERT INTO MyTable (FullName, PostCode, NewCustomer, DateAdded)
VALUES (?, ?, ?, GetDate())
```

The **INSERT expression** field accepts dynamic functions and these are evaluated before the first insertion (not for each insertion).

Note: No validation is performed to check if the expression is valid or includes the correct number of parameters (which, if incorrect, will lead to an error at runtime).

When **Enable editing** is selected, the **Regenerate** button is enabled. Click this button to generate the SQL Insert statement based on the items selected on the **Properties** tab.

Insertion and results options

- **Drop table if it exists:** If selected, will drop the table first if it exists in the destination database, before creating a new table. If selected, then **Create table if it does not exist** is auto selected and disabled.
- **Create table if it does not exist:** If selected, will create a new table if it does not exist. The data types used to create the new table use the following rules:
 - If the "Datatype" column is set to Object, then it will try to derive the actual data type by using the metadata of the column in the data section (direct mapping) or if an expression is evaluated, it will determine what kind of Object has been resolved.
 - Else, the "Datatype" column will be used to derive the type of column to create.

Note: The column types created will not necessarily be exactly the same as those in the existing data section. For example, if the data type is a string, then on MS SQL server it will create a field of NVARCHAR(MAX) (no attempt is made to try to calculate the required string length). If the table created doesn't use the desired field types/lengths then it is recommended to use the "SQL Module" to create the table first, and not rely on the "Create table" functionality of the "SQL Insert module".

- **Delete existing entries first using:** If selected, this will empty the destination table of any existing data before the new data is inserted. If enabled, the combo box next to the check box will also be enabled giving two possible options of how the data should be deleted, using either "SQL TRUNCATE TABLE" or "SQL DELETE". It defaults to "SQL TRUNCATE TABLE".

Note: Using "TRUNCATE" is faster. However, it has certain limitations. Hence, you must be aware of those limitations for your specific database.

- **Delete data section after batch insert:** If selected, will delete the whole data section once the insert of data is complete.
- **Transact each insertion:** If selected, then each insertion will run in its own transaction, and no bulk data insertion options will be used. This will decrease performance, but may be necessary for particular insertions that cause triggers to fire. It may also be necessary if auto-generated row ids are required. If selected, further error handling options are available, **On error continue, copying failed rows to data section** – and a data section name should be entered. This option is useful when inserting rows into a database where they may already exist – and therefore failures are expected. Further processing then can take

place on the failed rows to update the database.

- **Batch size:** Indicates the size of the batches when bulk inserting data. Dynamic functions can be entered in this field.
- **Create indexes on columns:** If selected, will create database indexes on the fields selected in the grid below it. Indexes are only created if the table is (re-)created. The grid will list all fields that have been selected for the INSERT, and any can be selected to have an index created.

Note: Table and index creation is only tested on Oracle and SQL Server. While standard SQL is used to generate these commands, there is no assurance that they will work on other database types.

- **Results Message Section:** Indicates the message section name that will be created at runtime containing the result of the number of rows that were inserted.
- **After insertion append any database auto-generated IDs to the input data section:** If selected, then any auto-generated IDs will be appended to the input data section. The exact behaviour of this field is JDBC driver dependent.

Example using the jTDS JDBC driver on SQL Server database: If the table "MyTable" used in the above example had a field which was a SQL Server auto generated number, then after insertion a new column will be added to "Data Section 1", containing the IDs for each row inserted. This field will always be called "ID" (not the name of the actual database field).

Example using the Oracle JDBC driver on Oracle database: With any insertion of a row in Oracle, the "ROWID" is returned. This is an internal Oracle ID. It can then be used to later reference a row in the database by using a syntax similar to:

```
select * from MyNewTable
where rowid = ?
```

And setting the rowID as the parameter in the query.

Results Tab

The standard **Test** button is available on the dialog, which will run the SQL Insert. The **Results** tab will display the results.

SAP Module

You can use a **SAP module** to connect to SAP (Systems Applications and Products in Data Processing) R/3 Data Sources and extract information from them to populate an EMF Process Data section.

The **SAP module** can also be used to update data in the R/3 system. Whether data is retrieved or updated depends upon the BAPI/RFC that is selected and executed.

Note: EMF requires SAP Java Connector 2 (SAP JCo 2) or SAP Java Connector 3 (SAP JCo 3) to connect to the SAP backend. SAP JCo supports connections to 4.0B, 4.5B, 4.6B, 4.6C, 4.6D, 4.7, 6.20 and higher.

To use a SAP Module:

1. Ensure that you have created a [SAP Profile Connection](#) to the Data Source you wish to use.
2. Drag a **SAP module** icon into the EMF Process you are creating at the appropriate place.



3. Double click on the module icon to display the **SAP screen**.

SAP Properties

Data section: SAP

SAP Connection: [Dropdown] [Edit]

☒ Use RFC [Get RFC metadata] ☐ Use BAPI [Get BAPIs]

Halt conditions: ☒ Never ☐ On no data returned ☐ On data returned

Available BAPIs: [Empty list box]

RFC properties table:

Name	...	Re...	Description

Selected parameter table:

Name	...	Value	Description

Buttons: OK, Apply, Cancel, Help

The **Properties** tab is used to specify the required SAP connection, and the information required for the EMF Process.

4. Select the connection that you wish to use from the **SAP connection** drop-down list, which shows all the SAP connections you have set up in the **Data sources** section of the EMF tree view.

You can click **Edit** to modify the properties of the selected SAP Connection. For more information, see [How to create a SAP connection](#).

5. When you have selected the required connection, press the **Get BAPIs** button and the **Available BAPIs** panel will display the BAPIs (Business Programming Application Interfaces) that you can select from the selected connection. Selecting **Use RFC** allows the name of an RFC (remote function call) to be entered directly, if known.
6. Enter an appropriate name for the connection in the **Data Section** field. This will be used to identify information that is collected from the SAP connection for other modules further on in the EMF Process.
7. Select the **Use BAPI** radio button and then select the BAPI that you want to populate the data section with, from the list displayed in the BAPIs available panel. When one is chosen its name is displayed above the BAPI Properties panel, which displays all information held within the BAPI.

The information displayed in the **BAPI Properties** panel is of four types:

- **SAP R/3 System** represents the root item nodes in the EMF tree view. They indicate each SAP R/3 System that your company has access to and that has been defined in the EMF system (via SAP Connections).
- **Application areas** are child items of the SAP R/3 system nodes in the tree view. The areas of data that are available will depend on your access rights to the SAP R/3 systems.
- **Business objects** are child items of the Application areas. They consist of related groups of BAPIs.
- BAPIs are child items of the Business objects. There will normally be a group of them, and when you select one, its details are displayed in the panel to the right.

Select the **Description** tab to view BAPI description.

8. When a BAPI is executed, it must always return some data. The check boxes in the BAPI properties grid allow the user to select the data that they want returned in the **Data Section**. Select the check boxes against the data items you want returned.
9. It is possible to enter parameters into the query that is about to be executed. These parameters may be the values that the backend system will be updated with, or they may be criteria used to filter the result set that will be returned. Parameters are entered in the value fields - these values can be dynamic functions (right click on the field to access the dynamic function menu).

Note: There is no need to select the check box on a row if entering a value in the row - the check boxes should only be selected if that row's value should be returned in the resultant Data Section.

10. Select the Properties tab to define filters and halt conditions.
11. The SAP module's **Properties** tab allows you to select when and if to halt processing of the SAP request. It is also to restrict data that is returned to only that which has changed, since the last time the EMF Process ran. Normally all data is returned, and the EMF Process is stopped if no data is returned from the SAP provider.
12. Select a **Halt condition** option. These allow you to specify whether the returned data section should be checked, and if so, whether the EMF Process processing should be halted:

- Select **On No Data Returned** if you want to stop the EMF Process when the data sections selected on the **BAPI Browser** tab are empty.
- Select **On Data Returned** if you want to stop the EMF Process when the data sections selected within the **BAPI Browser** tab are populated (i.e. when they contain any data).
- Select **Never** if you always want to pass the data section from the **SAP** module, irrespective of whether or not it contains any data.

[Data Sources \(Configuring a SAP Profile\)](#)

[SAP Module Examples](#)

[Go to start of Data Modules](#)

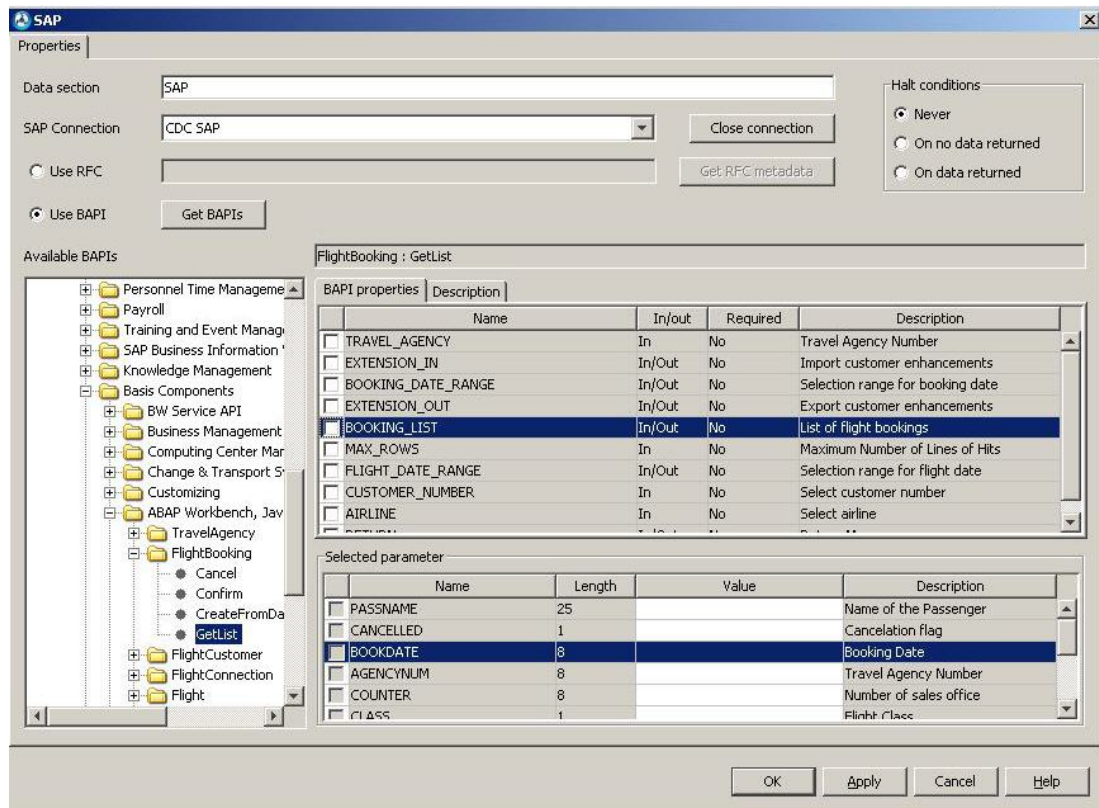
SAP Module Examples

The following examples demonstrate how to use the SAP module to extract and update data in the R/3 system. The examples use the example Flight Booking System data provided as part of the SAP BASIS system and should therefore be present with most installations.

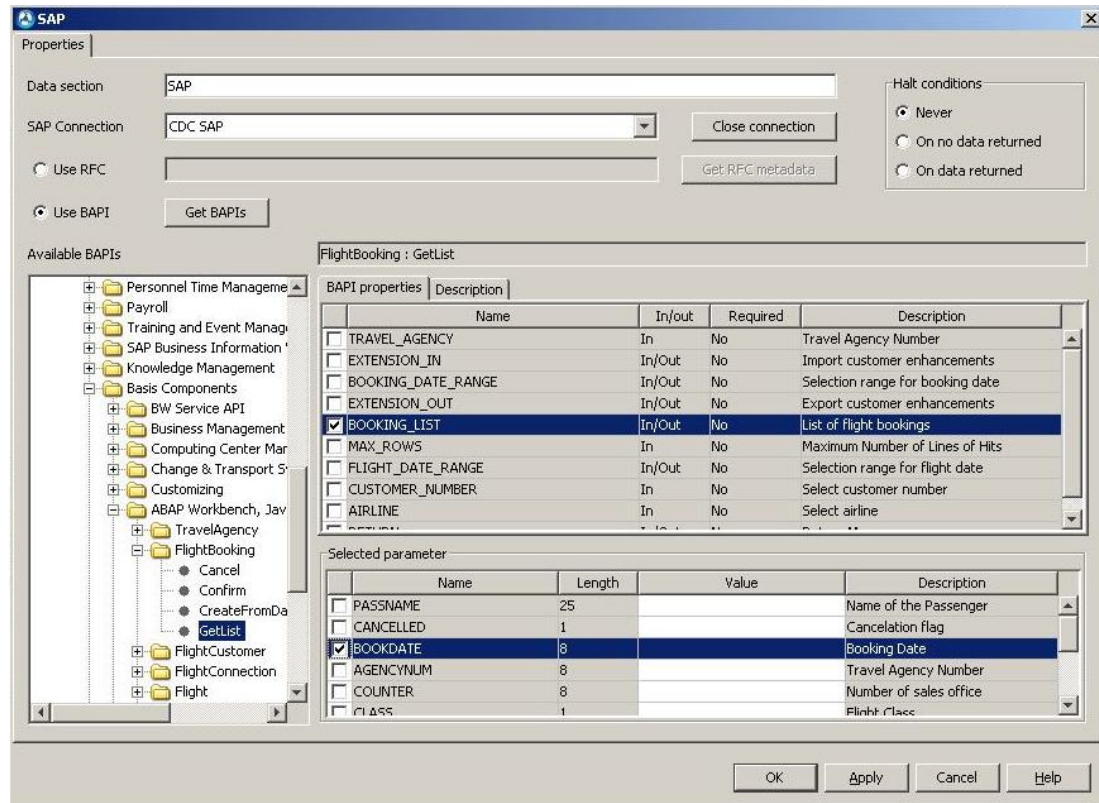
Example 1 - Returning a list of flight bookings

This example will demonstrate how to configure the SAP module to return a list of flight bookings in a Data Section.

1. Select your SAP connection from the drop down box and click **Open Connection** to populate the list of BAPIs available.
2. From the list of BAPIs available, select **Application Components->Basis Components->ABAP Workbench->FlightBooking->GetList**. On selecting the **GetList** BAPI, the **BAPI properties** grids should populate.



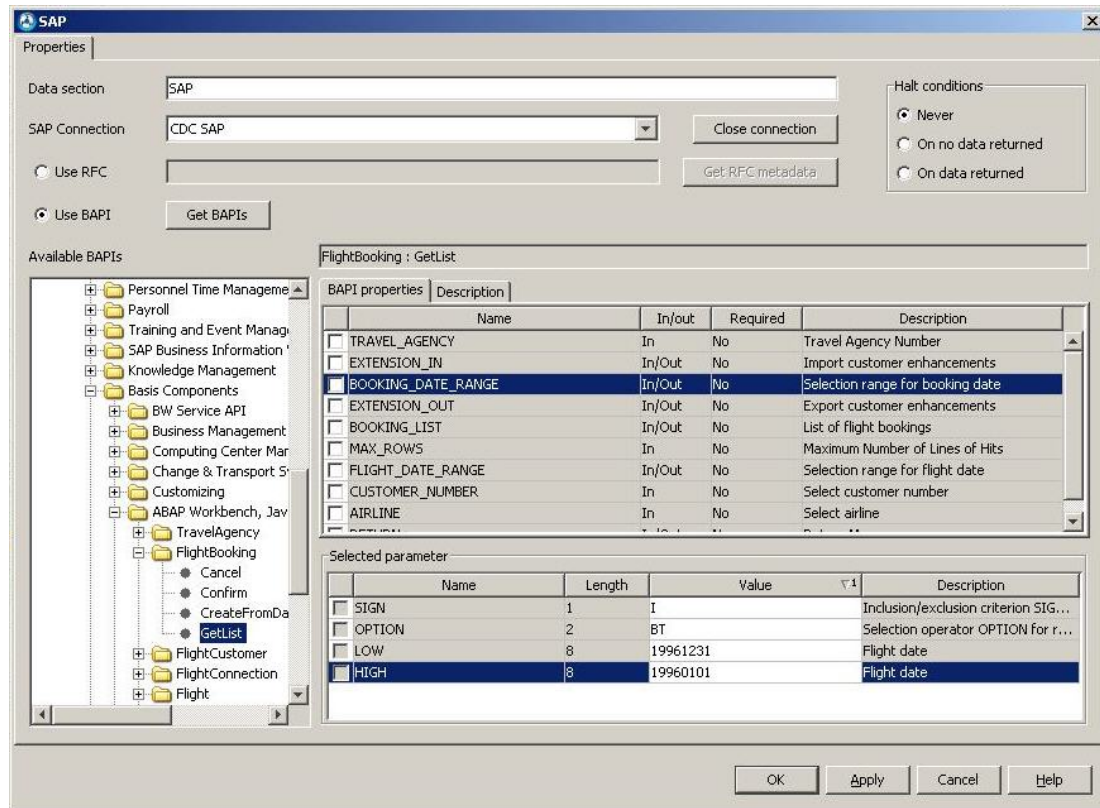
- Now the data to be returned must be selected. In this example, the flight date, booking date, customer id and airline id will be returned. To return this information, first select the **BOOKING_LIST** property that contains this information. Check the box next to the **BOOKING_LIST** to indicate that data will be returned from this property. Now in the grid below (now called **BOOKING_LIST**) we can see all the sub properties of **BOOKING_LIST**. Check the items **FLIGHTDATE**, **BOOKDATE**, **AIRLINEID** and **CUSTOMERID**.



Example 2 - Returning a list of flight bookings in a specified range

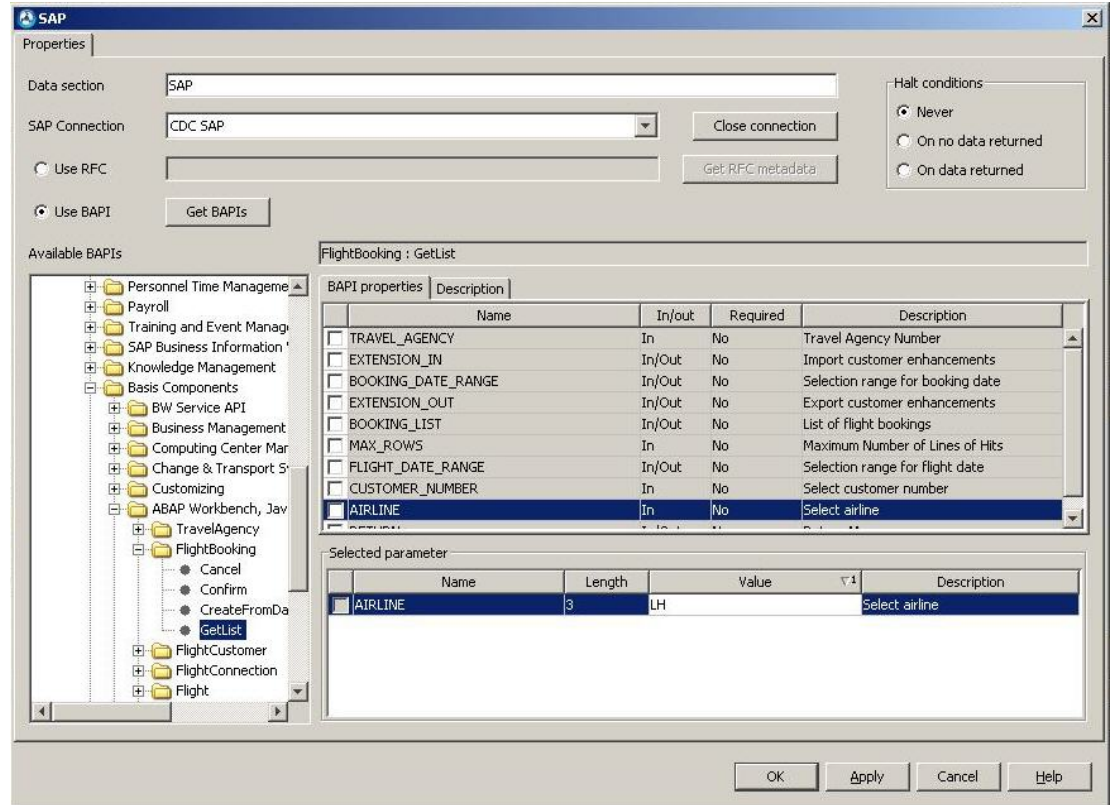
This example builds on the previous example and will demonstrate how to set criteria for the data that is to be returned. It will filter the data so that only bookings in 1996 for the Lufthansa airline are returned.

1. Build/reuse the SAP query that was created in Example 1.
2. Select **BOOKING_DATE_RANGE** in the BAPI properties (don't select the checkbox - this should be left unchecked). In the booking date range table enter the following values:
 - **SIGN = I.** This means the date range will be **inclusive**.
 - **LOW = 19960101.** This is the start date of the period of interest.
 - **OPTION = BT.** This means that all values **between** the high and low date are required.
 - **HIGH = 19961231.** This is the end date of the period of interest.



3. Select **AIRLINE** in the BAPI properties (do not select the check box - this should be left unchecked). In the airline table enter the following value:
 - **AIRLINE = LH**. LH is the airline code for Lufthansa airline.

Note: Any of the values entered could have been from dynamic functions. For example, the airline name filtered for could have been extracted from another datasource and the dynamic function **\$DATA('SQL',0,0)\$** used in place of **LH**.

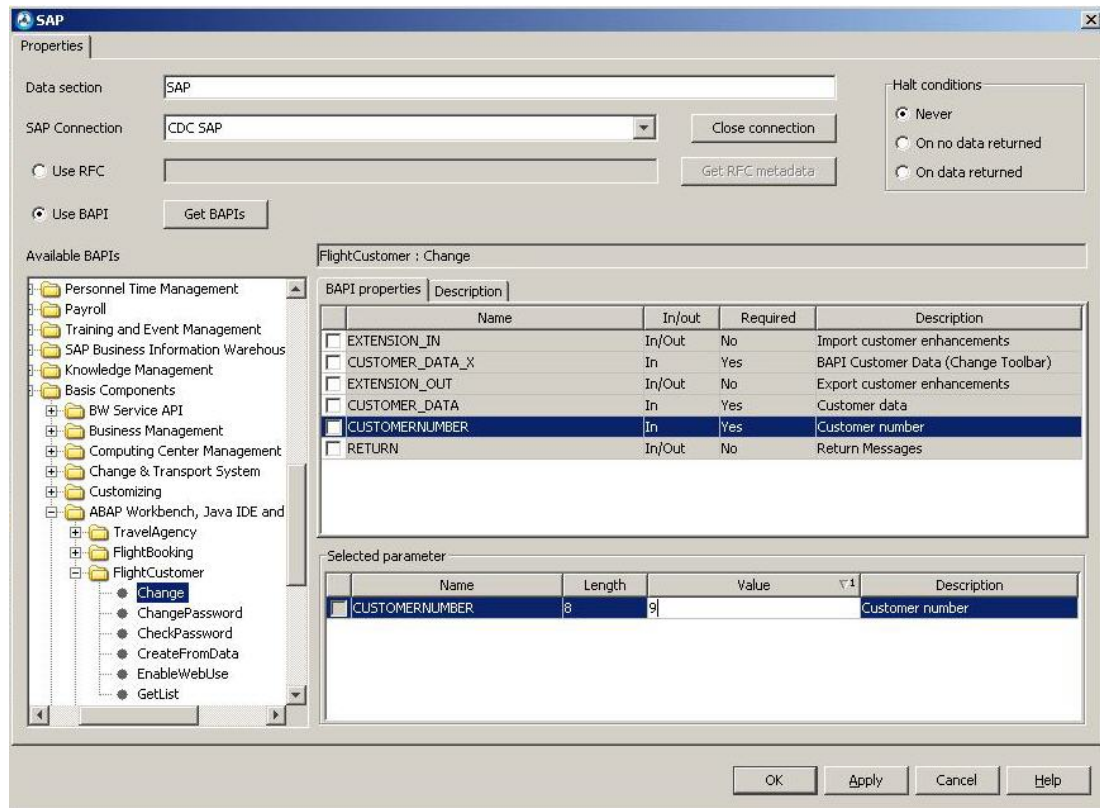


Example 3 - Updating a customers contact details

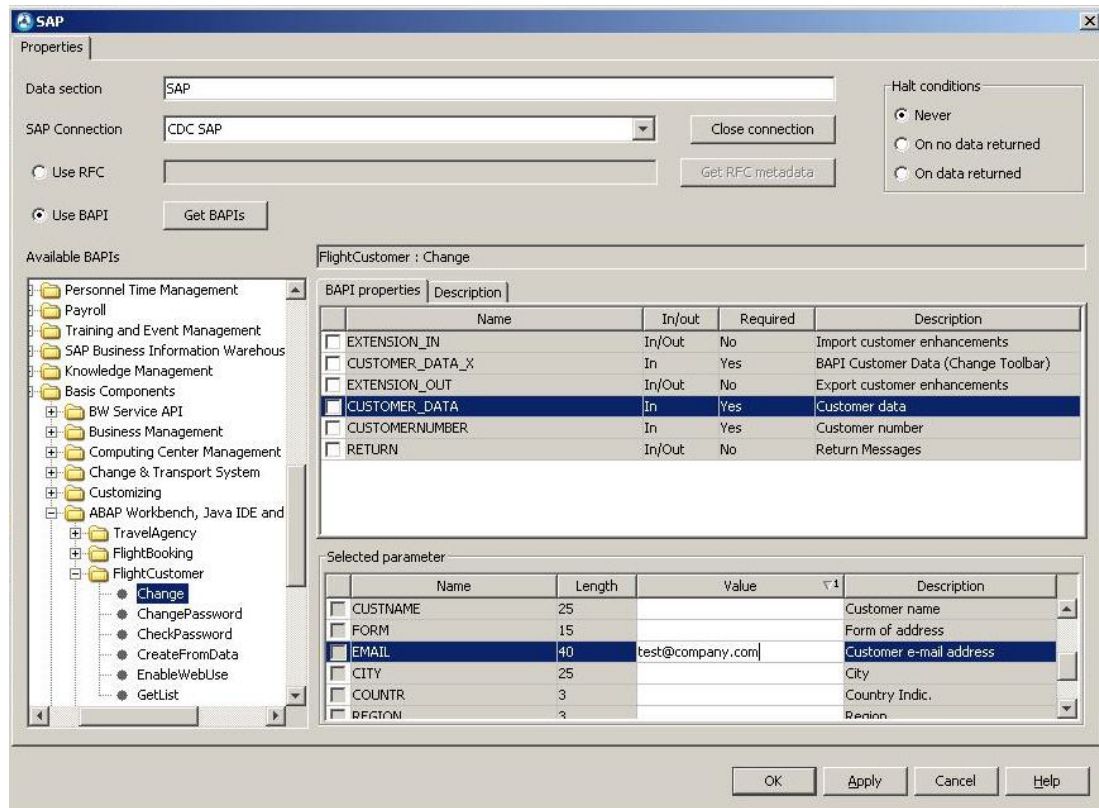
This example will demonstrate how to update a customers email address in the R/3 system. There is no difference in what is configured between a query and an update, it is all decided by the BAPI that is selected and the fields that are populated.

1. Select your SAP connection from the drop down box and click **Open Connection** to populate the list of BAPIs available.
2. From the list of BAPIs available select **Application Components->Basis Components->ABAP Workbench->FlightCustomer->Change**. On selecting the **Change** BAPI the **BAPI properties** grids should populate.
3. Select the **CUSTOMERNUMBER** property (don't select the checkbox - this should be left unchecked). In the **CUSTOMERNUMBER** grid enter the id of the customer to be updated. In this example customer number **9** will be updated.

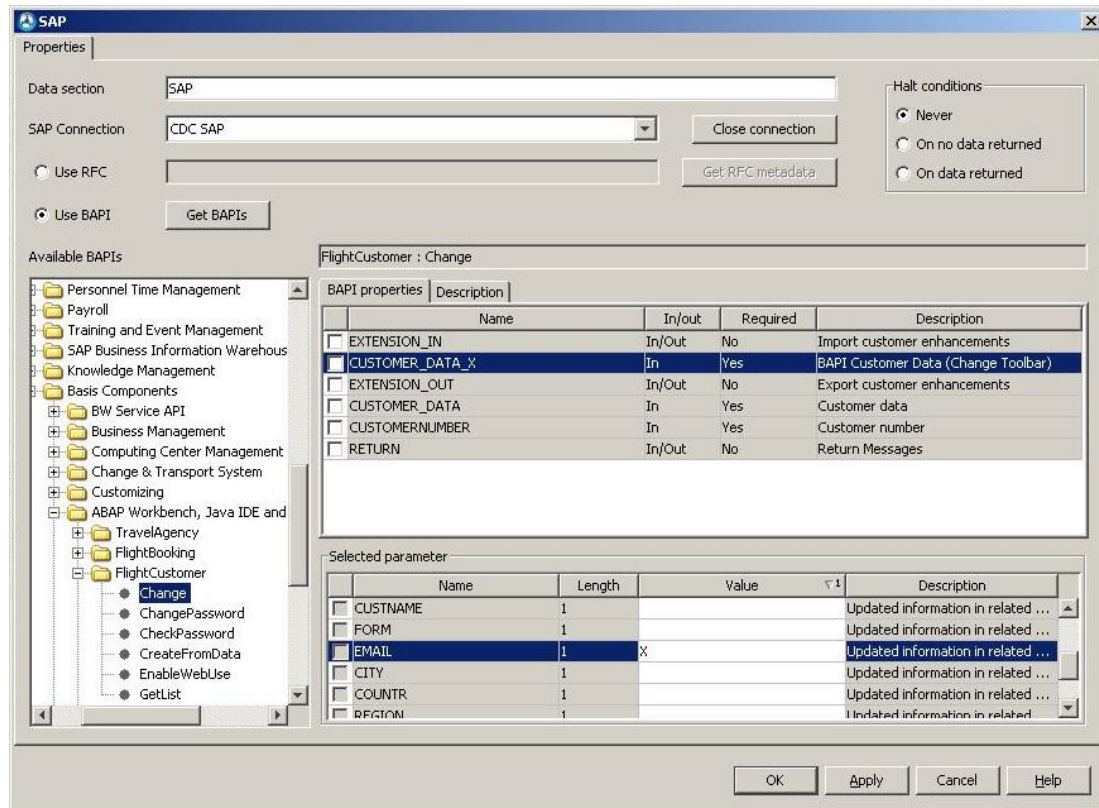
Note: In a real world case, the customer number could have been extracted in a previous SAP Module by performing a **GetList** call on the Flight customers and entering the customers name as a filter criteria. A dynamic function could then be used in this query to enter the customer number.



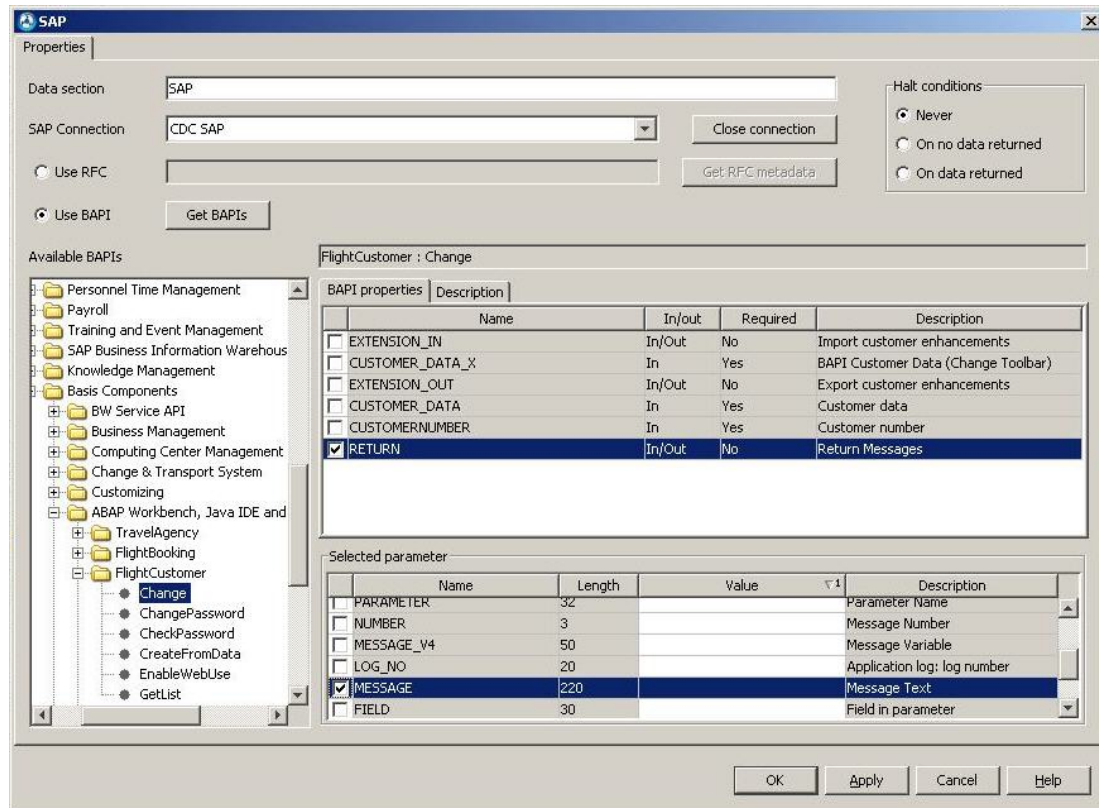
4. Select the **CUSTOMER_DATA** property and in the **CUSTOMER_DATA** table, enter the email address that this customer should have in the **EMAIL** row.



- With this BAPI, it is also required to indicate fields that will be updated. To do this, select **CUSTOMER_DATA_X** property and in the **CUSTOMER_DATA_X** table, enter an **X** for **EMAIL** row value.



6. It is required in EMF that all BAPIs return some data. As this query is performing an update, in the Data Section it is recommended that the update result is returned. This can be found from the **RETURN** property. Select the **RETURN** property and then select the check box next to it (as this is a value that must be returned). In the **RETURN** table, select the **MESSAGE** row.



[SAP Module](#)

[Data Sources \(Configuring a SAP Profile\)](#)

[Go to start of Data Modules](#)

HTTP Module

You can use the **HTTP module** to retrieve data from a Web server at a given location (e.g. on the Internet, an Intranet, etc.) dynamically for use in an EMF Process and store it in a message section. Additionally, although the HTTP module is not classed as an [Output module](#) (because it cannot use recipients), you can use it to send data to an HTTP address.

If the HTTP module is in an EMF Process that was initiated by an [HTTP Listener](#) - or is in an EMF Process cascaded to from an EMF Process initiated by an HTTP listener - you can use it to send responses back to the HTTP client that the requests came from. For more information on using the module in this way, see [How HTTP Works in EMF](#).

Information is retrieved by making a request using an HTTP 'GET' request, and sent by using an HTTP 'POST' request and specifying the data to be sent as part of the request.

To use an HTTP module:

1. If you want to use the HTTP module to send responses back to an HTTP client, ensure that the EMF Process also includes an [HTTP in](#) module, or is cascaded to by an EMF

Process that includes an HTTP in module.

2. Ensure that you have set up one or more [HTTP services](#) (in the **Services** section of the EMF tree view).
3. Drag the icon for the **HTTP module** into your EMF Process at an appropriate place.



4. Double-click the icon to open the HTTP screen. This screen allows you to select the required HTTP Service and configure the most common HTTP options.

5. Select the required **mode**:
 - **If you want the HTTP module to act as an HTTP client**, select **HTTP request** as the **Mode** and then select the required [Service](#) and **Method** in the **HTTP request** section (e.g. **GET** if you want to request a document or submit a small form, or **POST** if you want to submit a form of any size). The HTTP module

also supports the HEAD, PUT, OPTIONS, and DELETE methods.

If you want to edit the Service selected, click **Edit** to display the **HTTP Service** window, where you can edit the properties and Internet Security settings of the Service. For more information see, [HTTP Service](#).

- **If you want to send the HTTP request to a client HTTP device**, select **Use recipient based address**. This will automatically take the URL(s) of the destination device(s) from the properties of the HTTP devices specified for each of the recipients in the Recipient module, and use these as the base of the destination URL(s). Anything in the **Resource URI** field, or the default URI (if selected, see below) will be appended to this.

Next, specify the required **Resource URI** (or select the **Use Default URI** option to use the URI that is specified in the relevant HTTP service).

Finally, if you do not want the HTTP module to automatically follow any redirection commands it encounters, deselect the **Handle redirection automatically** option.

- **If you want the HTTP module to send back responses to a client that has come through the HTTP Listener**, select **Client response** and then select the required **Code** to return the response from the drop-down list in the **Client response** section. You can select any of the standard HTTP codes from the drop-down list, or enter a custom code and a description in the **Information** field. You can also enter [dynamic functions](#) by right-clicking in the **Information** field.

Note: If you select **Client response**, the HTTP module uses the details configured in the [HTTP Listener](#). The "HTTP request" option and the "Request Result" tab are not applicable and will not be available.

6. The [Message body/ Headers](#) tab provides two options:
 - To select the **Message section** to be sent in the body of a message.
 - To select the HTTP headers to be sent with the request/response.
7. **If you want to extract the body text and/or headers from responses** and store them in message sections, select the [Request Result](#) tab.
8. Select the [Advanced](#) tab to set [Message state logging](#), [Escalation filtering](#) and Target device options.
9. Click **Test** to view the connection.

[Basic HTTP example](#)

[HTTP Cascade example](#)

[Go to start of Output Modules](#)

[How to create a self-signed certificate](#)

The Message body/Headers tab

The Message body section

You can use the **Message body** section of the **HTTP** module's **Properties** page to specify whether or not a message should be sent in the body of a request.

The screenshot shows the 'HTTP' module's 'Properties' dialog box, specifically the 'Message body / Headers' tab. The dialog has a title bar with a close button. Below the title bar are five tabs: 'Properties', 'Message body / Headers' (selected), 'Request result', 'Advanced', and 'Preview'. The 'Message body' section contains a checkbox for 'Append message body' (unchecked). Below this are three drop-down menus for 'Message section', 'Content type', and 'Content coding'. The 'Headers' section contains a checkbox for 'Send headers' (unchecked) and a table with two columns: 'Header' and 'Value'. To the right of the table are three buttons: 'Add', 'Remove', and 'Remove all'. At the bottom of the dialog are five buttons: 'OK', 'Apply', 'Test', 'Cancel', and 'Help'.

1. If you want to send a message along with the request to the specified URI, select **Append message body**. If you do not select this option the other options will not be available.
2. Select the required **Message section** from the drop-down list of those in the EMF Process, or enter the name of a section.
3. Select the appropriate MIME Content type from the drop-down list, or enter a custom type. If the charset is specified, then the content will be encoded using the character set specified. The following default types are supported:
 - text/plain
 - text/html

- text/html; charset=ISO-8859-1
- text/html; charset=windows-1252
- text/html; charset=utf-8
- text/xml
- text/xml; charset=utf-8
- image/jpeg
- image/png
- image/gif
- application/x-www-form-urlencoded
- application/user-form-data
- application/json
- application/json; charset=utf-8
- application/octet-stream
- multipart/related
- multipart/mixed

Note: **application/user-form-data** is an EMF-specific MIME type that makes it easier for you to submit form data in an HTTP Request. Using application/user-form-data, you can enter the form parameters in a message section, irrespective of whether you are using the GET or POST method. In other words, you do not need to enter the parameters as part of the GET Resource URI.

4. If the content type specified above is encoded, select the appropriate MIME **Content coding** type from the drop-down list, or enter a custom type. The following default types are supported:
 - Gzip
 - Compress
 - Deflate
 - None

The Headers Section

You can use the **Headers** section of the **HTTP** module's **Properties** page to define any header information that you want to send with a request.

To define headers to send with a request:

1. Double-click the appropriate **HTTP** module and select the **Request Headers** tab.
2. To send header information with a request, select the **Send Headers** option.
3. Click in a cell in the **Header** column and either select from the drop-down list of available headers or enter your own custom header name. The following headers are available in the list:

In **HTTP response** mode:

In **Client response** mode:

- Accept-Ranges

- Accept
- Accept-Charset
- Accept-Encoding
- Accept-Language
- Accept-Ranges
- Authorization
- Cache-Control
- Connection
- Cookie
- Content-Length
- Content-Type
- Date
- Expect
- From
- Host
- If-Match
- If-Modified-Since
- If-None-Match
- If-Range
- If-Unmodified-Since
- Max-Forwards
- Pragma
- Proxy-Authorization
- Range
- Referer
- TE
- Upgrade
- User-Agent
- Via
- Warning
- Age
- **Allow**
- Cache-Control
- Content-Encoding
- Content-Language
- Content-Length
- Content-Location
- Content-Disposition
- Content-MD5
- Content-Range
- Content-Type
- Date
- ETag
- Expires
- Last-Modified
- Location
- Pragma
- Proxy-Authenticate
- Refresh
- Retry-After
- Server
- Set-Cookie
- Trailer
- Transfer-Encoding
- Vary
- Via
- Warning
- WWW-Authenticate

4. For each header specified in the **Header** column, click in the corresponding cell in the **Value** column and enter an appropriate value. This value can be plain text or a dynamic function (you cannot use recipient-based functions).

[HTTP Module](#)

[HTTP Module Request Result tab](#)

[HTTP Module Advanced settings](#)

[Using Data Modules to Query Data Sources](#)

The Request Result tab

You can use the **Request Result** tab of the **HTTP** module's **Properties** page to specify whether or not to store the headers and/or body text with a message response.

To define headers to send with a request:

1. Double-click the appropriate **HTTP** module and select the **Request result** tab.

The screenshot shows the 'HTTP' dialog box with the 'Message body / Headers' tab selected. The 'Response' section contains the following options and values:

- ☐ Store message content
- ☐ Fully qualify links
- ☐ Store as binary
- Message section name: MESSAGE_BODY
- ☐ Store response headers
- Data section name: RESPONSE_HEADERS
- ☒ Store response code
- Message section name: RESPONSE_CODE

The 'Advanced' section contains the following options and values:

- ☐ Store request stream
- ☐ Store response stream
- Message section name: WHOLE_RESPONSE_STREAM

At the bottom are buttons for OK, Apply, Test, Cancel, and Help.

- To store the entire message, along with any headers, in a single section, select **Store whole response stream** and then enter an appropriate **Message section name**.
- To parse messages and store the body text in a separate section, select the **Store message body** option and then enter an appropriate **Message section name**. A message section of this name will be created in the EMF Process, when required, to store the message body. If you do not select the **Store message body** option, the body text of messages will be ignored.

Note: When you view the stored body text of an HTTP resource, any images or other resources that were included on the page will not normally display, this is because the links to these resources tend to reference other local files on the original server. If you want to convert these "relative" links into "absolute" links in order to be able to view these items, select the **Fully qualify links** option. This could, however, result in the server taking longer to process the HTML page.

- To parse messages and store the headers in a separate section, select the **Store response headers** option and then enter an appropriate **Data section name**. A data section of this name will be created in the EMF Process, when required, to store the message headers. If you do not select the **Store response headers** option, the

headers of messages will be ignored.

Important: If your EMF Process contains more than a single HTTP module you should ensure that the Message and Data section names in each are different, in order to avoid them being overwritten.

[HTTP Module](#)

[Message body/Headers tab](#)

[HTTP Module Advanced settings](#)

How HTTP Works in EMF

There are two main ways to use the HTTP functionality in EMF:

- Closed loop
- Standalone

With a closed loop system, you can receive a request from an HTTP client (for example, a browser, trigger an EMF Process and send a response back to the original client from the EMF Process).

If you want to use HTTP this way, ensure that you configure the following:

- HTTP response queue. This is pre-configured in EMF as **HTTPQ**, but you can change this if you like. The HTTP response queue must not be used by any other parts of the EMF system.
- [HTTP listener](#) to listen for incoming client requests. The HTTP listener must be located on the same machine as the HTTP response queue.
- [HTTP service](#) to send the response back to the client.
- An EMF Process containing an [HTTP in](#) module that uses the listener you configured.
- An EMF Process containing an [HTTP module](#) that uses the HTTP service you configured. The HTTP module should be set to **Client response** mode.

Note: The HTTP in module and HTTP module can be in the same EMF Process, but they do not have to be. This is because the client connection information is passed automatically to the EMF Process. **However**, the EMF Process with the HTTP in module **must contain a Cascade module** that either initiates the EMF Process containing the HTTP module, or allows users to specify this EMF Process dynamically (from an HTTP client).

See [Closed loop HTTP in action](#) for instructions on how to quickly configure and test the EMF closed loop HTTP functionality. Also, see [HTTP Cascade example](#) for details on how to initiate a chosen EMF Process dynamically via a Web browser.

When using the HTTP components standalone

- To launch an EMF Process based on an incoming HTTP request, configure an [HTTP](#)

[listener](#) and include an [HTTP in](#) module in the EMF Process. The HTTP listener must be listening to an HTTP response queue. This is preconfigured in EMF as **HTTPQ**, but you can change this if you like. The HTTP response queue must not be used by any other parts of the EMF system and must be located on the same machine as the HTTP listener.

- To send an HTTP request to a Web server, configure an [HTTP service](#) and include an [HTTP](#) module set to **HTTP Request** mode in the EMF Process.

The [HTTP diagram](#) illustrates how different components of the HTTP functionality in EMF work together.

[Go to start of Output Modules](#)

Calling RESTful Web Services

Representational State Transfer (REST) Web services is an alternative to SOAP and WSDL (Web Services Description Language) Web services. RESTful Web services is built upon the concepts of HTTP, and though EMF has the **Web Service Module** for calling SOAP based Web services, the [HTTP module](#) is best suited to make the calls.

This section provides a quick overview of how to call a RESTful Web service. For detailed information, refer to the documentation for the appropriate Web service.

REST creates a one-to-one mapping between the create, read, update, and delete operations using HTTP methods:

- To create a resource, use the HTTP POST method.
- To retrieve (read) a resource, use the HTTP GET method.
- To update a resource, use the HTTP PUT method.
- To delete a resource, use the HTTP DELETE method.

Making these calls using the HTTP module is a direct process and only requires setting the appropriate method on the HTTP request. For example, to read information about product `abc` in company `xyz`, the url can be as follows:

`http://www.xyz.com/restapi/product/abc`

The screenshot shows the 'HTTP' dialog box with the 'Properties' tab selected. The 'Mode' section has 'HTTP request' selected. The 'HTTP request' section includes a 'Service' dropdown set to 'HTTP', a 'Method' dropdown set to 'GET' (highlighted in yellow), and a 'Resource URI' text field containing 'http://www.xyz.com/restapi/product/abc' (also highlighted in yellow). There are checkboxes for 'Use recipient based address', 'Use default URI', and 'Handle redirection automatically' (checked). The 'Client response' section has a 'Code' dropdown set to '200' and an 'Information' text field containing 'OK'. At the bottom are buttons for 'OK', 'Apply', 'Test', 'Cancel', and 'Help'.

Set the response to store the message content and the response code.

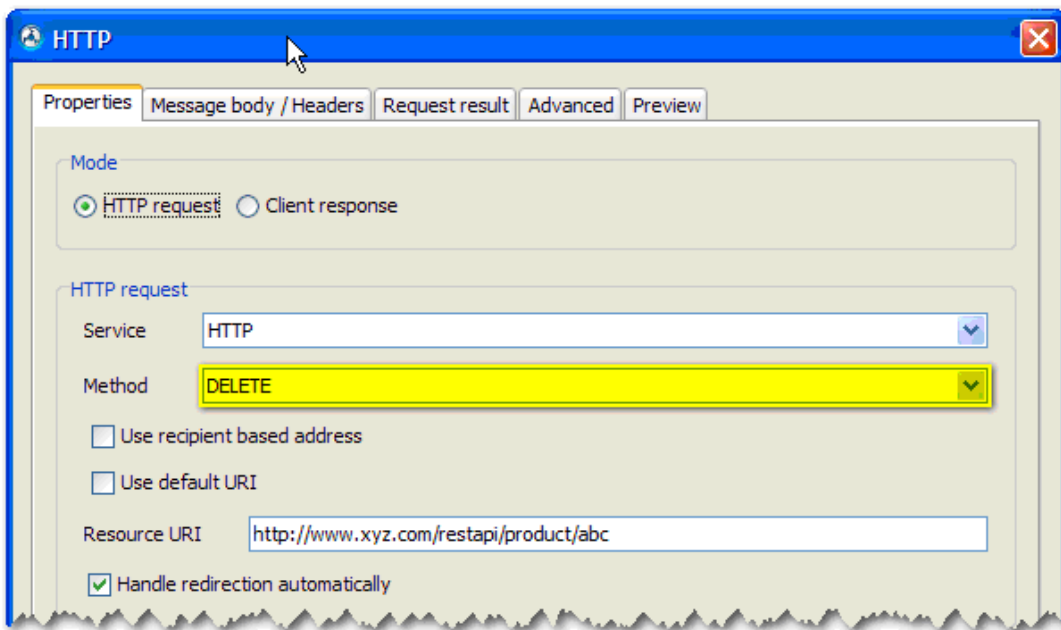
The screenshot shows the 'HTTP' dialog box with the 'Request result' tab selected. The 'Response' section has checkboxes for 'Store message content' (checked), 'Fully qualify links', and 'Store as binary'. The 'Message section name' text field contains 'MESSAGE_BODY'. There is an unchecked checkbox for 'Store response headers'. The 'Data section name' text field contains 'RESPONSE_HEADERS'. The 'Store response code' checkbox is checked, and the 'Message section name' text field below it contains 'RESPONSE_CODE'. The dialog box has a decorative, torn-edge bottom border.

When the call is made, it will return an XML document (typically) in the MESSAGE_BODY, with information about the product (assuming that the call was successful. You should check the RESPONSE_CODE value to ensure this).

Example:

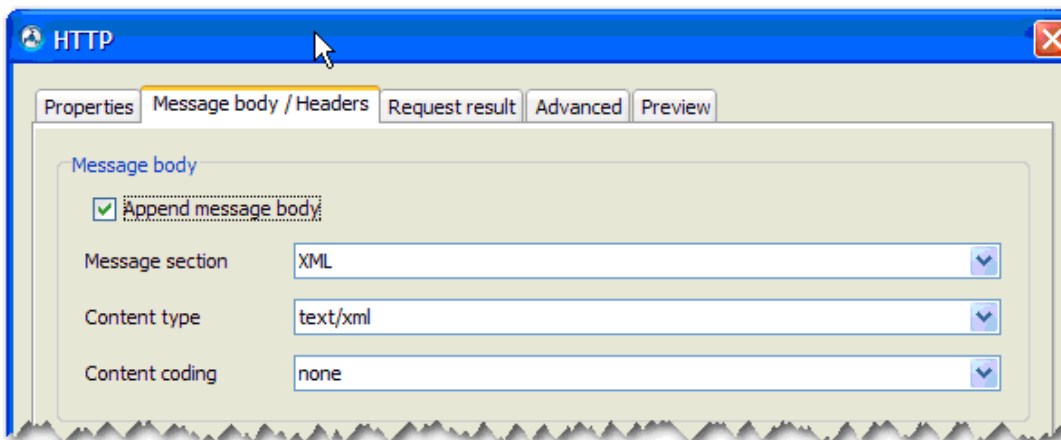
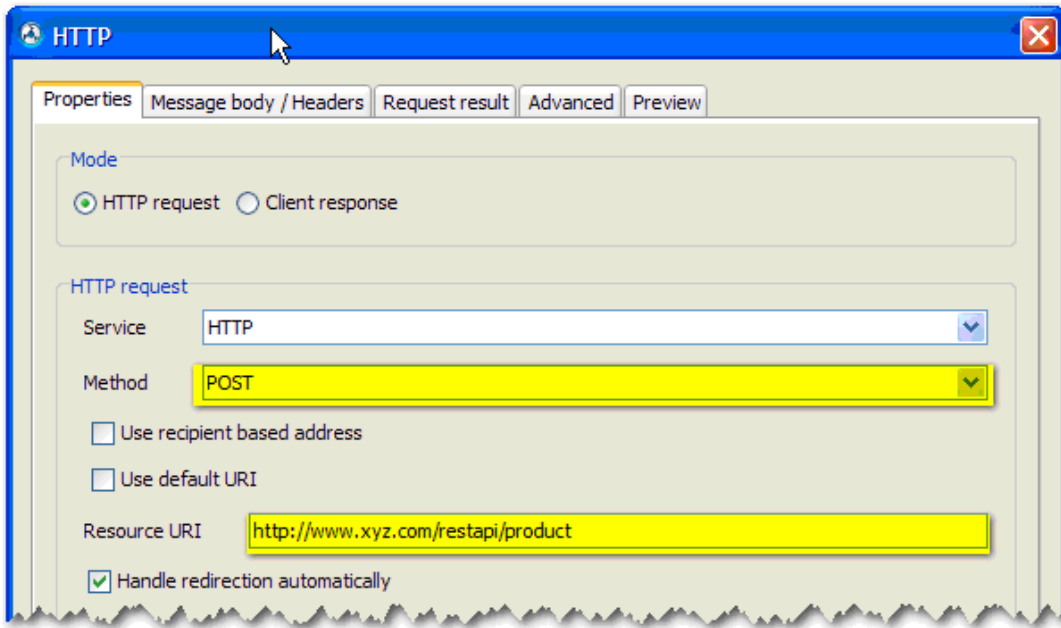
```
<?xml version="1.0"?>
<PRODUCT>
<ID>abc</ID>
<NAME>This is a product abc</NAME>
<PRICE>99.9</PRICE>
</PRODUCT>
```

If you had to delete this product, you would have had to use the HTTP DELETE method instead of GET. The URL would be the same.



To create an item, POST the XML for the new item to the URL:

<http://www.xyz.com/restapi/product>

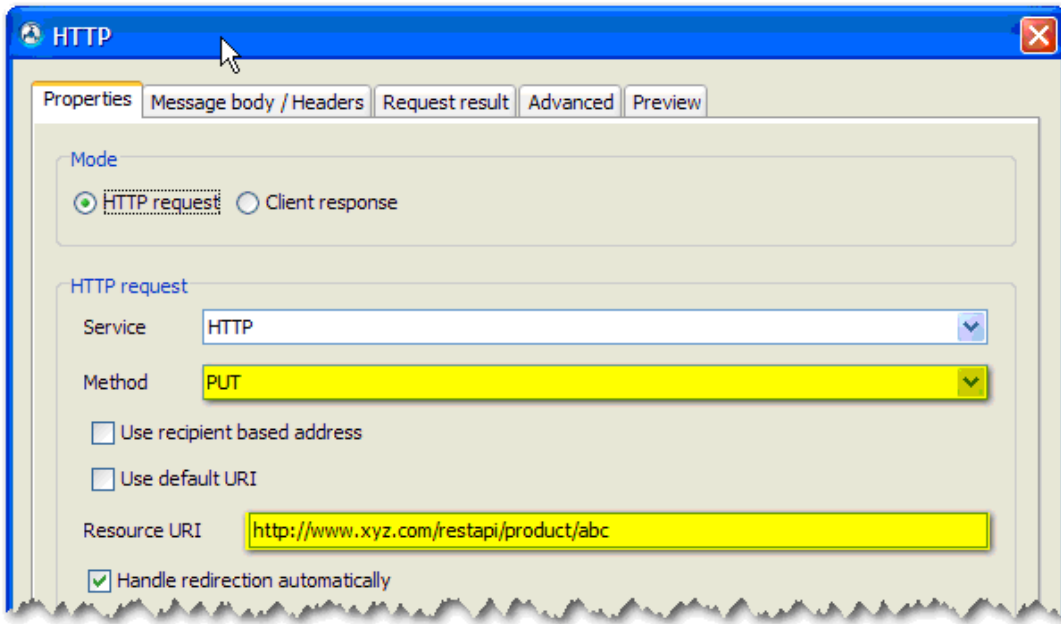


The XML message section will contain the XML of the new product you wanted to create.

Example:

```
<?xml version="1.0"?>
<PRODUCT>
  <ID>def</ID>
  <NAME>This is a new product</NAME>
  <PRICE>8.1</PRICE>
</PRODUCT>
```

Typically, a response code value of 201 will be returned if the item was successfully created. To update an item, use the PUT method and send the XML for the updated product.



Examples

Following are two examples of the actions that can be performed using the HTTP functionality:

- [Closed Loop HTTP in Action](#): Describes how to set up a basic closed loop HTTP system within EMF
- [HTTP Cascade Example](#): Describes how to trigger an EMF Process specified as a parameter in the HTTP request from an HTTP client.

Closed Loop HTTP in Action

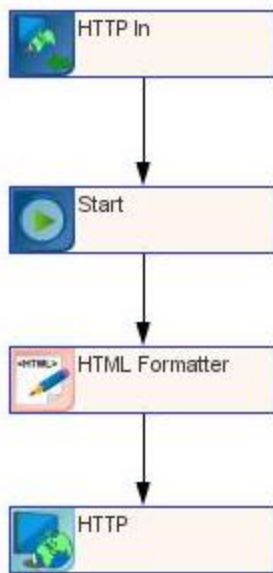
This topic provides instructions for setting up a basic closed loop HTTP system within EMF.

Important: Before following these instructions, ensure that you have installed EMF and created an EMF configuration and repository database using the EMF Processes configuration wizard.

This example EMF Process will take the parameters you enter in a Web browser URL, and trigger an EMF Process to return these to you in an HTML page.

1. [Create an HTTP listener](#): in the **Properties** tab, ensure that **EMF Process** is set to "HTTPSystemAlert" and the **Response queue** to "HTTPQ". Give the listener a name, and leave all other settings as default.
2. [Create an HTTP service](#): Give the HTTP service a name and leave all other settings as default.

3. [Create a new EMF Process](#) as follows:



4. Configure the modules as follows:

- **HTTP in:** select the HTTP listener you created.
- **HTML formatter:** insert the following text between the body tags (right-click to access the [DATA](#) dynamic function):

```

<h1>Response</h1>
<p>First parameter: $DATA('HTTPIN_PARAMS','Param1',0)$</p>
<p>Second parameter: $DATA('HTTPIN_PARAMS','Param2',0)$</p>
  
```

- **HTTP:** select **Client response** mode in the **Properties** tab. In the **Message body** tab, select the **Append message body** checkbox. Then set the **Message section** to "HTML formatter" and the **Content type** to "text/html". Leave all other settings as default.
5. [Save the EMF Process](#) and set it to **Active**.
6. [Start the EMF Server](#).
7. Open a Web browser on your EMF Server machine and enter **http://localhost:80/?Param1=Hello&Param2=World**. Your browser should display the parameters you typed in an HTML table.

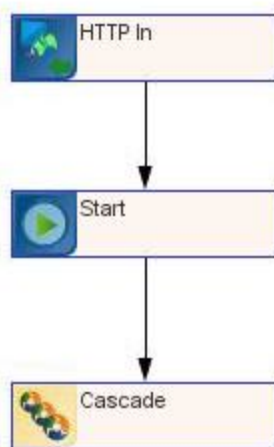
[Go to start of Output Modules](#)

HTTP Cascade Example

You can use the HTTP functionality in EMF to dynamically trigger a named EMF Process. The following example shows how to trigger an EMF Process specified as a parameter in the HTTP request from an HTTP client.

Important: Before following these instructions, ensure that you have installed EMF and created an EMF configuration and repository database using the EMF Processes configuration wizard. It is also recommended that you test your HTTP functionality using the example in [Closed loop HTTP in Action](#).

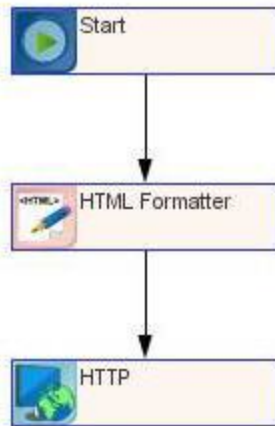
1. [Create an HTTP listener](#): in the **Properties** tab, ensure that **EMF Process** is set to "HTTPSystemAlert" and the **Response queue** to "HTTPQ". Give the listener a name, and leave all other settings as default.
2. [Create an HTTP service](#): Give the HTTP service a name and leave all other settings as default.
3. [Create a new master EMF Process](#) as follows:



4. Configure the modules as follows:
 - **HTTP in**: select the HTTP listener you created.
 - **Cascade**: select **Dynamic**. Right-click in the **Dynamic EMF Process text** textbox and select **DATA > HTTPIN_PARAMS**. Then edit the second field and enter the parameter name, for example, "EMF ProcessID". You can use any name you like. Users enter this name in front of the parameter value when they type the URL.

Important: you cannot accept the default index-based value because there is no guarantee in which order users will type the parameters. Therefore, you should specify the parameter by name.

- Select **EMF Process ID** from the **Dynamic resolution** drop-down list.
 - a. Create a new slave EMF Process as shown here:



- b. Configure the modules as follows:

- **HTML formatter**: insert the following text between the body tags:

```
<h1>Response</h1>
<p>Your EMF Process has run.</p>
```
- **HTTP**: select **Client response** mode in the **Properties** tab. In the **Message body** tab, select the **Append message body** checkbox. Then set the **Message section** to "HTML formatter" and the **Content type** to "text/html". Leave all other settings as default.

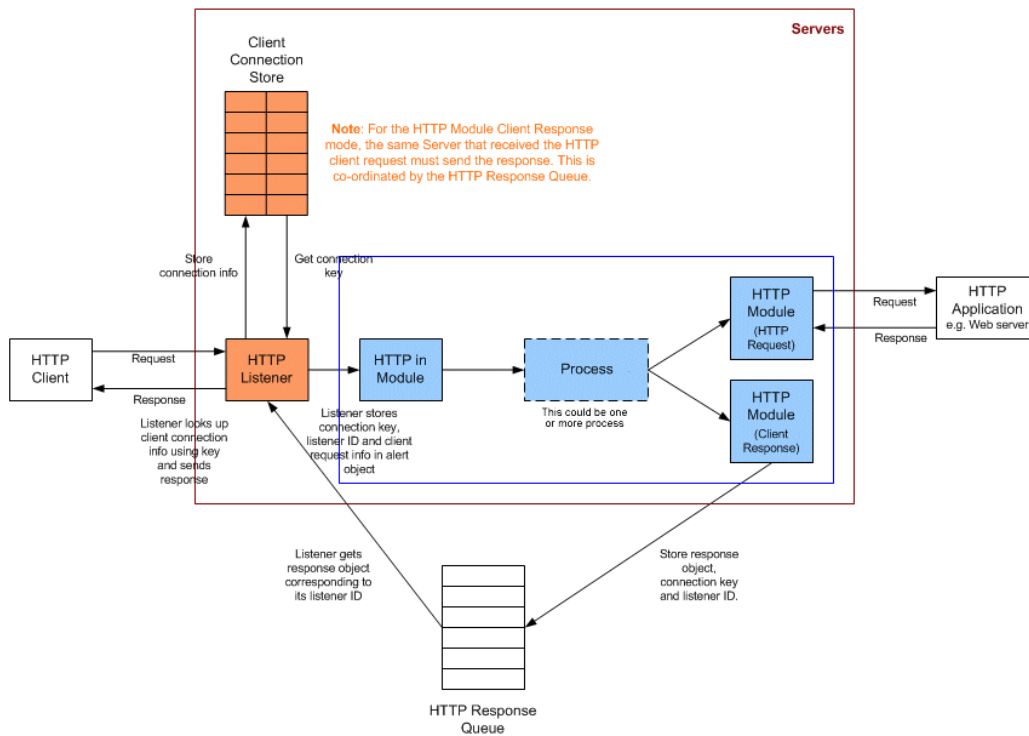
- c. [Save both EMF Processes](#) and set them to **Active**.

5. Make a note of the **Identity** of the slave EMF Process in the EMF Administrator treeview. For example, 1005.
6. Start the EMF Server.
7. Open a web browser on your EMF Server machine and enter **http://localhost:80/?AlertID=<Identity>**, for example **http://localhost:80/?AlertID=1005**. Your browser should display the text you typed in the HTML formatter.

[Go to start of Output Modules](#)

HTTP Diagram

This diagram illustrates how the different components of the HTTP functionality in EMF work together.



JNDI Query Module

You can use the JNDI query module to retrieve information from a JNDI directory service and save it to a data section for further processing in your EMF Process.

EMF supports the version of JNDI in Java 2 SDK 1.6 or later. See the [Sun web site](#) for more information on JNDI.

How to use a JNDI Query Module

1. Ensure that you have created a [JNDI service](#) for connecting to the JNDI service provider and directory service you wish to use.
2. Drag a **JNDI query module** icon into the EMF Process you are creating at the

appropriate place.



3. Double-click on the module icon to display the **JNDI query screen**.

The screenshot shows a dialog box titled "JNDI query" with a blue title bar and standard window controls. It has three tabs: "Properties" (selected), "Attributes", and "Preview". The "Properties" tab contains the following fields:

- "Data section": A text box containing the value "JNDI".
- "JNDI service": A dropdown menu with a red asterisk to its left and an "Edit" button to its right.
- "Query": A large, empty text area.
- "Entry count limit": A text box containing the value "0".
- "Timeout (ms)": A text box containing the value "1000".
- "Search scope": A dropdown menu containing the value "One level".
- "Suggested page size": A text box containing the value "1000".

At the bottom of the dialog are five buttons: "OK", "Apply", "Test", "Cancel", and "Help".

4. Type a name for the **Data section** in which the returned JNDI information will be saved.
5. Select the connection that you wish to use from the **JNDI service** drop-down list, which shows all of the JNDI services that you have set up in the **Services** section of the EMF

tree view.

You can click **Edit** to modify the properties of the selected **JNDI service**.

6. Type the query to run against the selected attributes in the **Query** textbox. This allows you to specify the directory information that is returned. You can also use [dynamic functions](#) in this textbox. **You must enter a query** - an error will be returned if you leave this text box blank.

LDAP Query Examples

<code>(ObjectClass=*)</code>	This will retrieve all common names.
<code>(cn=*)</code>	This will retrieve all common names.
<code>(&(objectClass=*)(cn=*))</code>	This will retrieve all common names within all objects
<code>(&((cn=a*) (cn=b*)) (objectClass=OrganizationalPerson))</code>	This will retrieve directory information for users of type OrganizationalPerson with names beginning with "a" or "b".
<code>(&(objectCategory=person) (objectClass=user) ((cn=jo*) (cn=stephen) (cn=david)))</code>	This will retrieve directory information for users with a common name that starts with jo, or users with the common name stephen or david.
<code>(&(!(physicalDeliveryOffice=United Kingdom)) (!(cn=S*)))</code>	This will retrieve all entries where the physical delivery office is not United Kingdom and the common name does not begin with S

7. To limit the number of records returned, type a maximum number in the **Entry count limit** textbox. The default "0" indicates that there are no restrictions on the number of records returned.
8. In the **Timeout (ms)** textbox, specify how long you want the EMF Server to wait for a response from the JNDI directory service before timing out and continuing the EMF Process processing. The default is 1000 milliseconds.
9. If you want to search the context root (specified in the [JNDI Service](#)) and all objects beneath the root, select **Subtree** level from the **Search scope** drop-down list. Otherwise, if you only want to search the context root, leave the default of **One level** selected.

10. The **Suggested page size** is used to limit the number of records that are brought back by the JNDI query, per page, when run against LDAP Servers. For example, setting it to 1000 will bring back 1000 records from the LDAP Server per page, until all records (that satisfy the query) are read from the LDAP Server. If the page size is set to a value greater than the LDAP Server paging size, then it will be ignored and the LDAP Server paging size will be used instead. If the Entry Count Limit is set to anything other than "0", then this value will be used to limit the maximum number of records. Setting the Page Size has no effect on non-LDAP Servers.
11. Click the [Attributes](#) tab to select the attributes you want to retrieve.
12. If your JNDI directory service is accessible from your machine and functioning correctly, click the **Test** button to display the **Preview tab** containing the data that will be retrieved.
13. Click **Save as sample** to save the data for testing purposes.

Tips for using the JNDI data section

- The JNDI query module can return the data back in a different order to the order you used for the selected [Attributes](#) when you run the EMF Process. Therefore, when you use the data in the resulting data section, you should reference by the column name rather than the column index.
- Attributes may be returned with a different name to the name originally shown in the **Available** list. To check the actual column names that will be returned, click the **Test** button to display the test results.
- If the selected attribute does not exist - that is, it has not been defined in the JNDI directory service - then it will not be included in the data returned. This may happen if you enter the attribute name manually.
- Duplicate attributes will not be returned. In other words, if you define a custom attribute that is the same as an attribute already selected, the resulting data section will only contain one attribute (usually the first one).

[Connecting to a JNDI Server](#)

[Go to start of Data Modules](#)

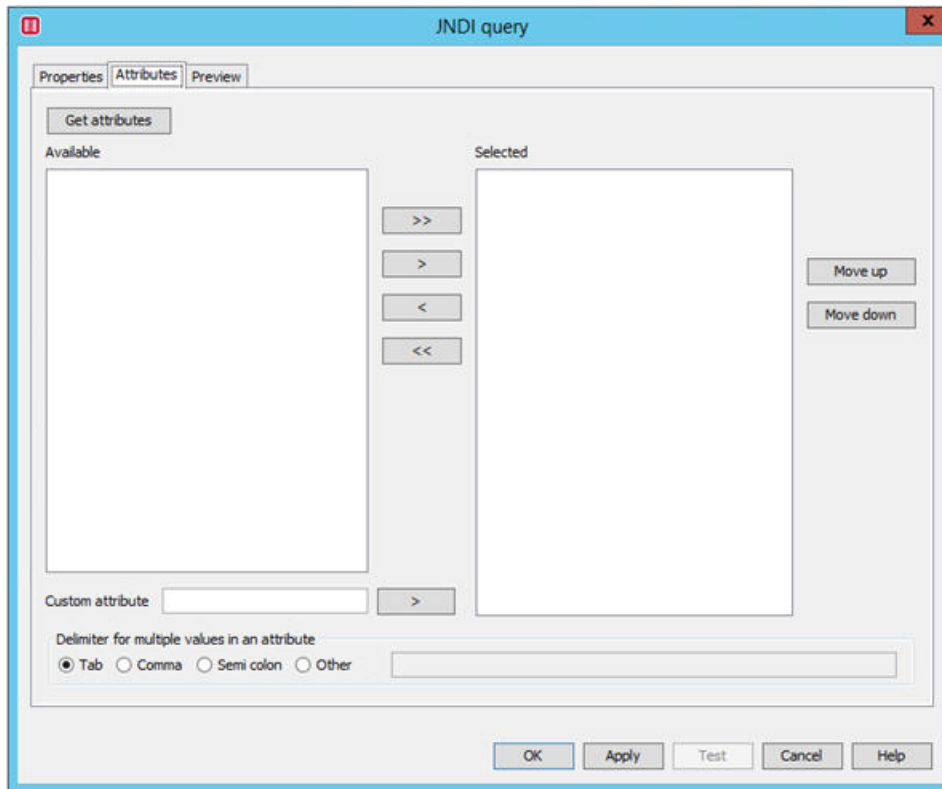
JNDI Query Module Attributes Tab

You can use the **Attributes** tab of the **JNDI query** module's **Properties** page to specify the attributes you want to search for the query entered in the **Properties** tab.

Important: Entering attribute information is optional. If you do not select any attributes, then all directory information under the context root is returned.

To select attributes to search:

1. Double-click the appropriate **JNDI query** module and select the **Attributes** tab.



2. If your JNDI server supports retrieving attributes, you can automatically retrieve the list of attribute names for the context root by clicking the **Get attributes** button. The list of attributes is displayed in the **Available** panel.
3. Select an attribute you want to query and click **>** to add the attribute to the **Selected** list. Repeat until all the attributes you want to query have been added.

Tip: You can select several attributes at once by clicking on the first one in the list and then holding down the SHIFT key while clicking on the last one. You can also add all the attributes in the list by clicking the **>>** button, and remove one or all of the selected attributes by clicking the **<** or **<<** buttons.

4. If your JNDI server does not support retrieving attributes, you can enter the attributes you want to query manually in the **Custom attribute** text box. Then click **>** to add the attribute to the **Selected** list. Continue until all the attributes you want to query have been added to the list.

Note: You can enter [dynamic functions](#) in the **Custom attribute** text box. Dynamic functions are only evaluated at run time. If you click the **Test** button in this screen, you will only see partial results from the query.

5. In the **Delimiter for multiple values in an attribute** area, select the delimiter you want to use. If you want to use a delimiter other than Tab, Colon, or Semicolon, type the delimiter character in the **Other** text box. Any printable characters are supported. You cannot specify carriage returns, linefeeds, or spaces as delimiters.
6. You can use the **Move up** and **Move down** buttons to change the order of the selected

attributes.

7. If your JNDI server is accessible from your machine and functioning correctly, click the **Test** button to display a window containing the data that will be retrieved. If no data is displayed, see [Troubleshooting JNDI queries](#) for help on diagnosing the problem.

[JNDI Query Module](#)

[Go to start of Data Modules](#)

Web Service Module

The Web Service module is used to make direct calling of SOAP based Web services that have a supporting Web Service Definition Language (WSDL) file. This module allows EMF data and message sections to be mapped onto the parameters required by the Web service call without having dependencies on the underlying message protocol.

The Web Service module can be used to retrieve information, or pass information to another system, or both, depending on the call being made.

Note: It is also possible to use the HTTP module to make Web service calls, but you will need to understand and format the XML message payload manually and set the required HTTP headers, which can be complex. This may be necessary with certain complex XML schema definitions that the Web service module cannot currently interpret.

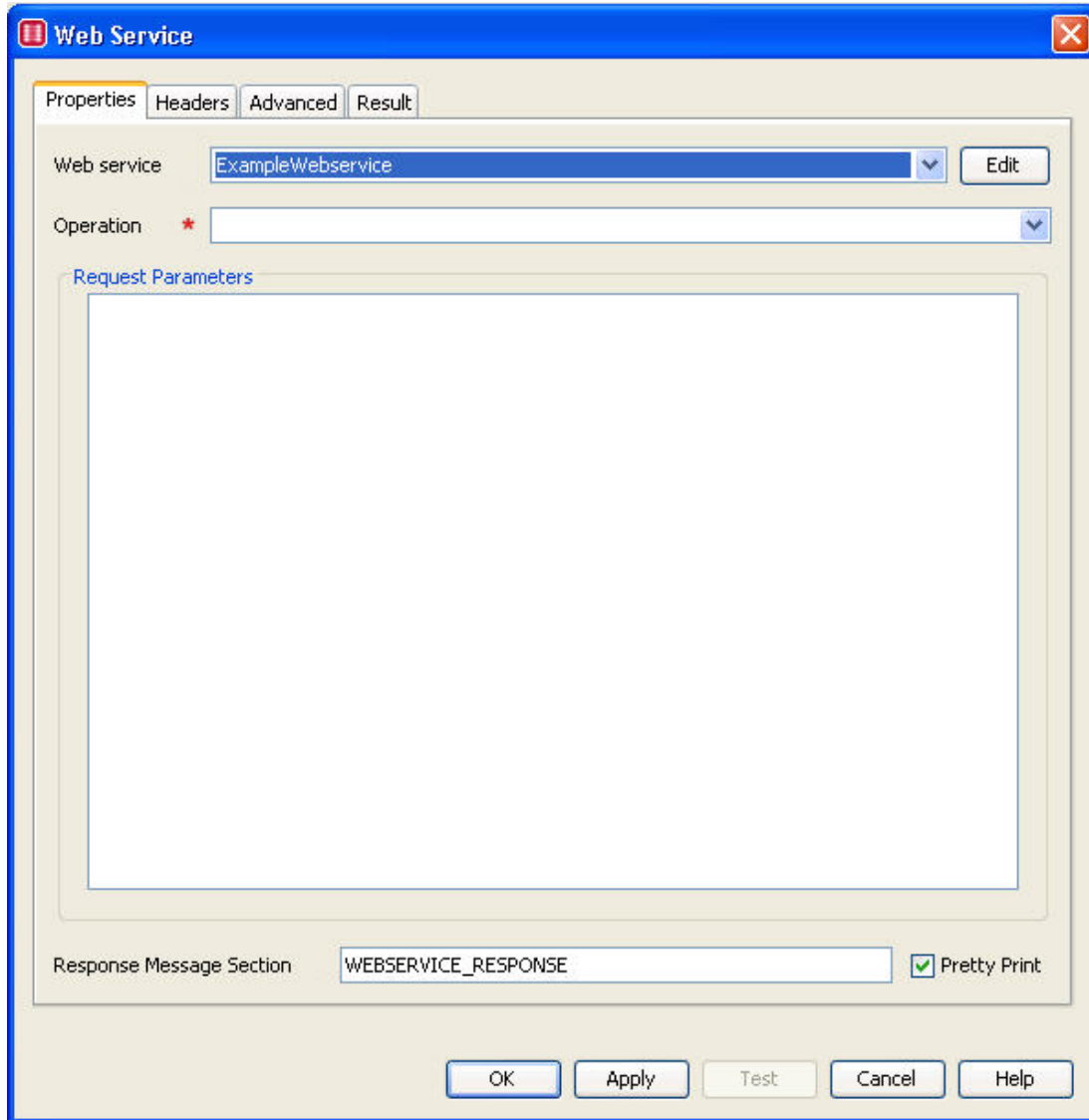
Note: The Web Service module only supports WSDL 1.1; WSDL 2.0 is not supported. Additionally, this module does not support old style SOAP-RPC calls, these calls will fail when attempted.

To use a Web Service Module

1. Ensure that you have set up one or more [Web Service services](#) (using the **Services** section of the EMF tree view).
2. Drag the icon for the Web Service module into your EMF Process at an appropriate place.



3. Open the Web Service module to display its properties. By default, you need to double-click on the icon to open the **Web Service** properties window. Alternatively, you can configure mouse-click actions using the **Options** window. To do this, from the **Tools** menu, select **Options**, **EMF**, and then select the **Process Builder Behaviour** tab.

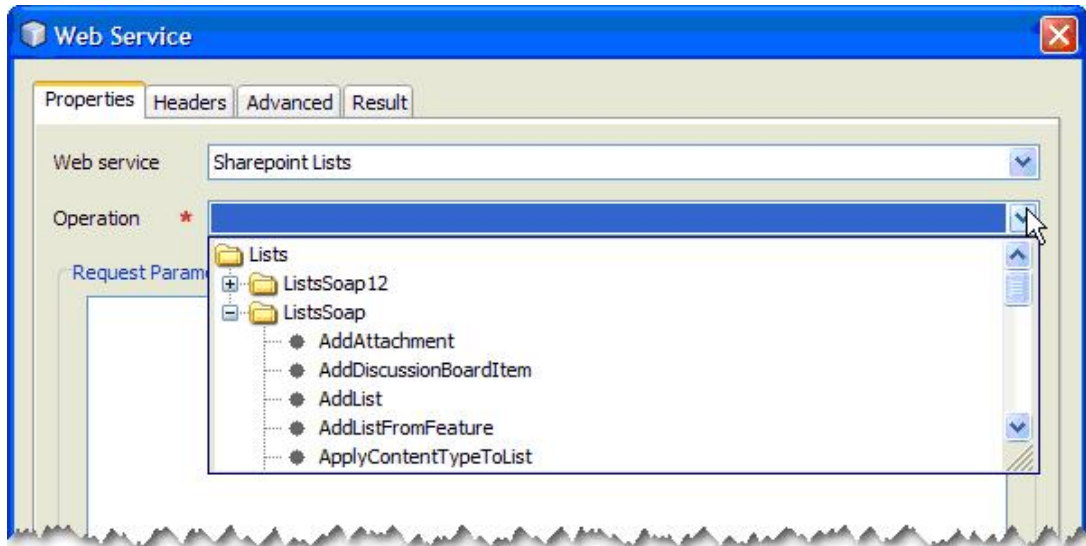


4. Select a Web service from the **Web service** drop-down list. The Web service services are those created in the **Web Service Services** section of the EMF tree view. You can click **Edit** to modify the properties and Internet security settings of the selected Web Service. For more information, see [Web Service Services](#).

The services will then populate the **Operation** list of the **Web Service properties** window with a list of available calls that can be made.

The operations are presented in a tree view. The service names are displayed at the top level, and under each service, the different Ports that the service provides for connecting are displayed. Finally, under each Port, the available Operations that the port provides are displayed.

For example, in the following figure, there is a single service called **Lists**, which has two **Ports** for connection - ListsSoap12 and ListsSoap. ListsSoap in turn has operations AddAttachment, AddDiscussionBoardItem, and so on.

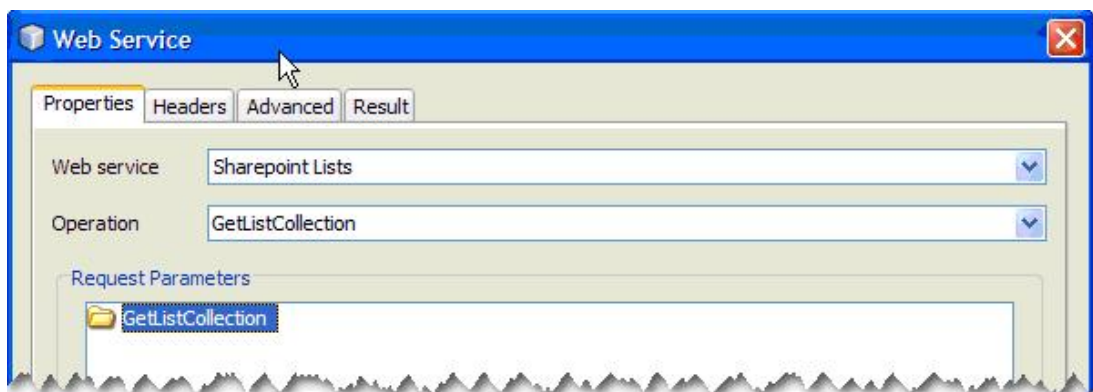


In most of the cases, the same list of Operations will be listed under all the Ports. Hence in the above example, if ListsSoap12 was expanded, the same operation list would be displayed.

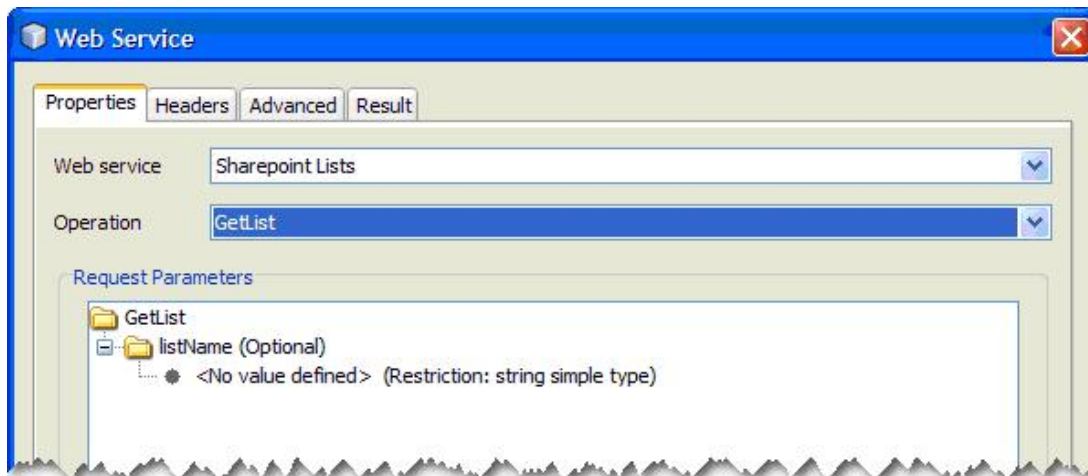
When selecting the operation to use, you must select it from the correct Port. To do this, you will need to refer to the documentation for the Web service you are trying to use. In the above example, one port would use the SOAP 1.1 protocol and the other, SOAP 1.2. In this case, either of the ports could be used. For a different Web service, for example, you may find one port using a secure HTTPS connection and another different port, a standard HTTP connection. Hence you will need to select the desired operation from the list.

5. After you select the operation, the **Request Parameters** section will be populated with the required structure that must then be filled to successfully make a call. The parameters required depend on the operation being called.

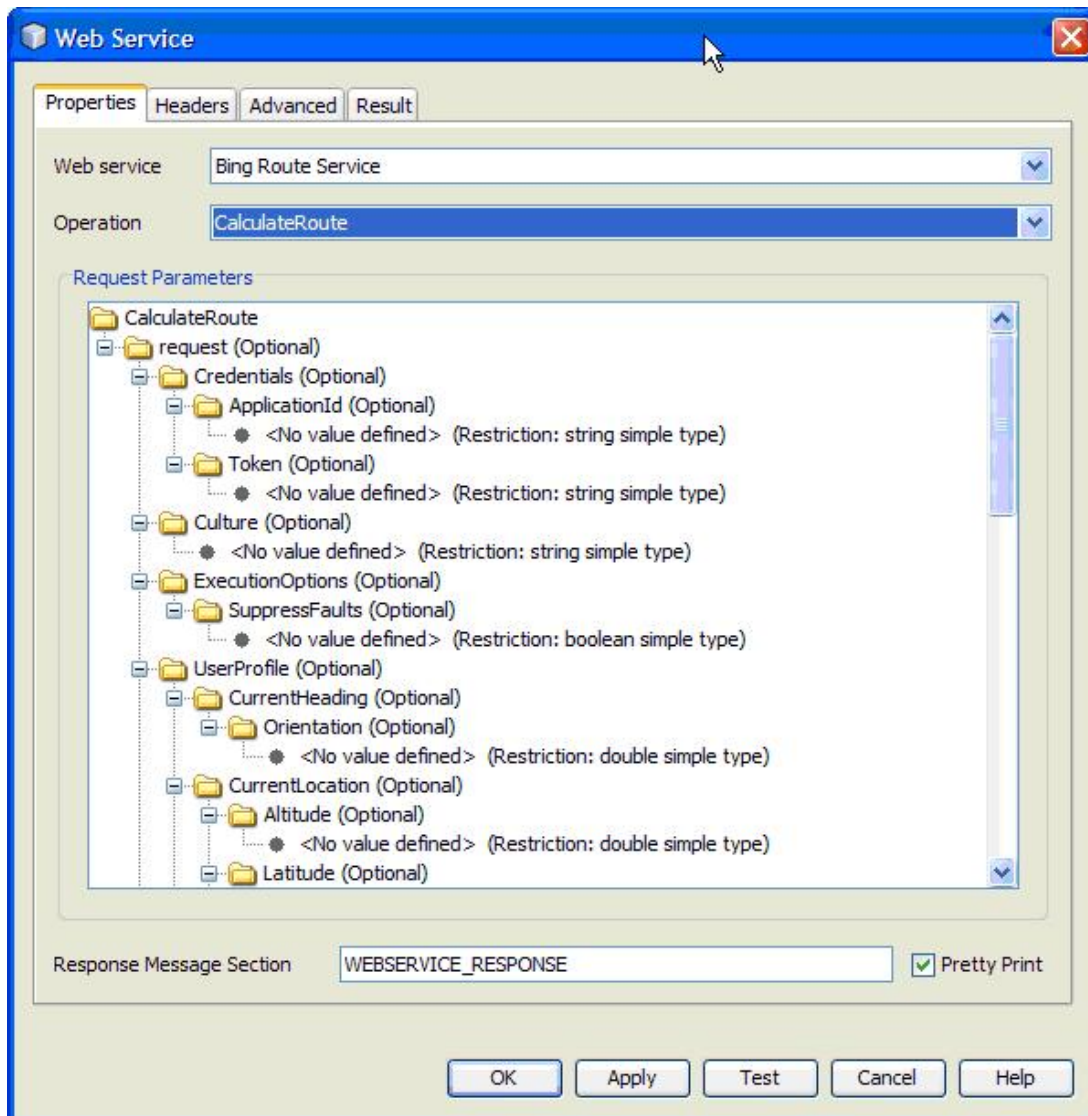
For example, the following Web service call requires no parameter.



The following Web service call requires a single parameter (the listName)



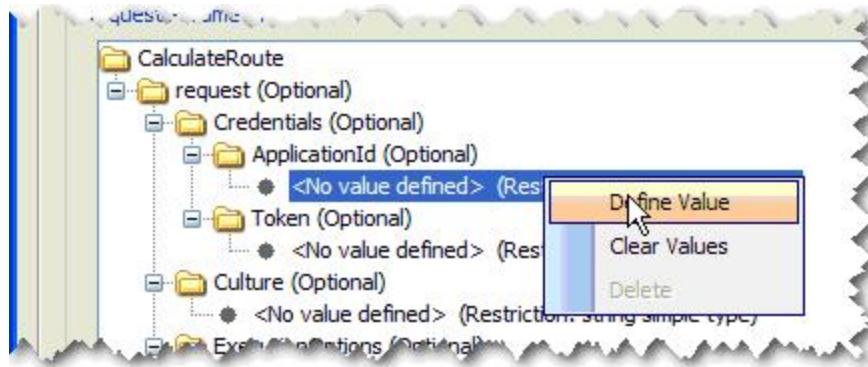
The following Web service requires many parameters to be set:



Each parameter that can have a value set will be displayed initially as `<No value defined>`, and its expected XML Schema datatype will be displayed next to it (see <http://www.w3.org/TR/xmlschema-2/#built-in-datatypes> for a list of possible datatypes, and the format of values should take).

To define a value, either:

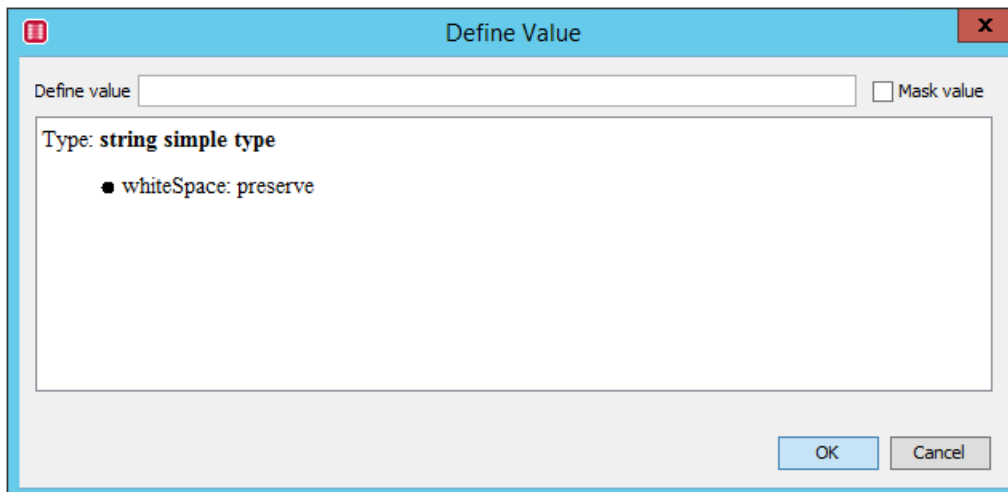
- Double-click on the item for which you wish to define the value, OR
- Right-click on the item to display the context menu and select **Define Value**.



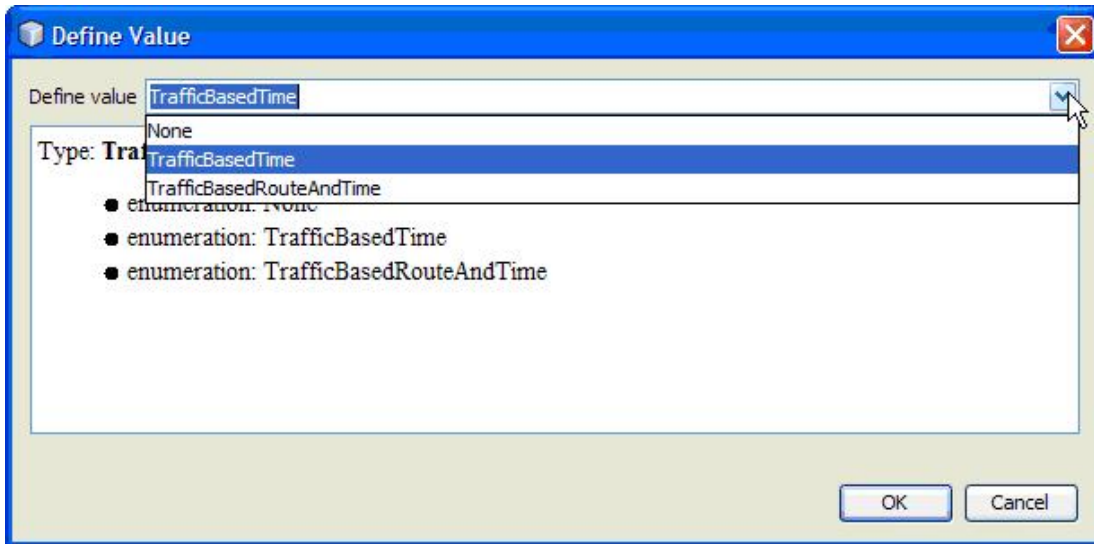
6. The **Define Value** dialog allows you to set the value for the parameter to be passed when the Web service call is made. Hence, type the value in the **Define Value** field, or right-click in the field to access the dynamic function field and enter a dynamic function. Select the **Mask value** check box to encrypt the defined value. This is useful if the field is a password field and its value should not be exposed to other users of the system.

The area below the **Define Value** field displays hints on the type of value expected and the expected formatting.

Note: No validation is performed to check whether the value you enter meets the server requirements. If you enter an invalid value, the Web service call will fail.

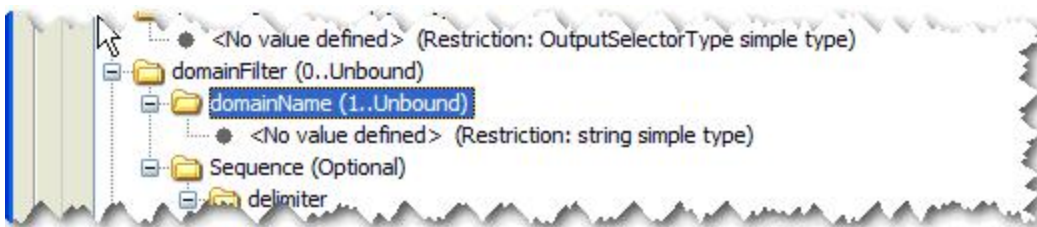


The **Define Value** field may sometimes appear as a drop-down list, containing possible values where a finite list is available.

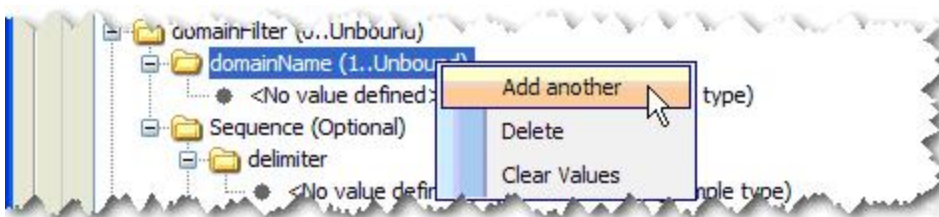


Press **OK** to accept the value, and the request parameters will be updated with the value.

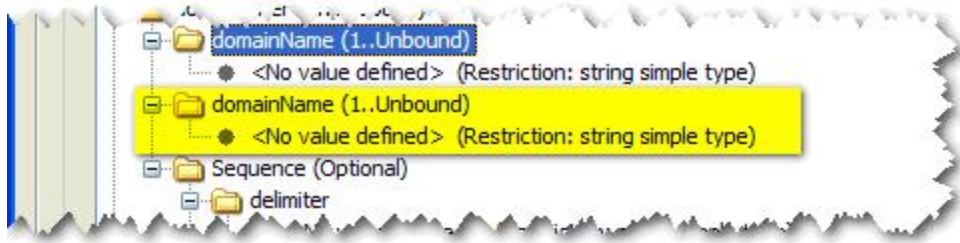
7. If you have set a value and then realize you do not want to send it as part of the call, then right-click on the value and select **Clear Values** to clear it. Some of the parameters are optional, and in this case, it will display **Optional** next to the parent element. In these situations, you may have set a value and then realized that you need to remove it.
8. Some items in the request parameters represent lists and may accept multiple values. These are indicated by a range setting next to them. For example, the highlighted item below, (1..Unbound), indicates that the operation expects a value of at least 1 to be defined, but could accept an unlimited number of values. (0..3) would mean that the operation expects up to 3 of these items to be defined.



In the above example, to define multiple domainNames, you need to right-click on the **domainName** node to access the context menu, and select **Add another**.



This will duplicate the item below, so that a second domainName could be defined. You will need to repeat this for every domainName required.



If you have added multiple items, and need to delete some, right-click on the item and select **Delete**.

Note: When you have only one item left in a list, the item cannot be deleted. If you do not want to define values for it, select **Clear Values** on the node, so that no values are sent in the request.

9. Nodes with the datatype **Any type** indicates that they accept any data type. You will need to see the documentation for the web service operation you are calling to understand what type of value is expected.

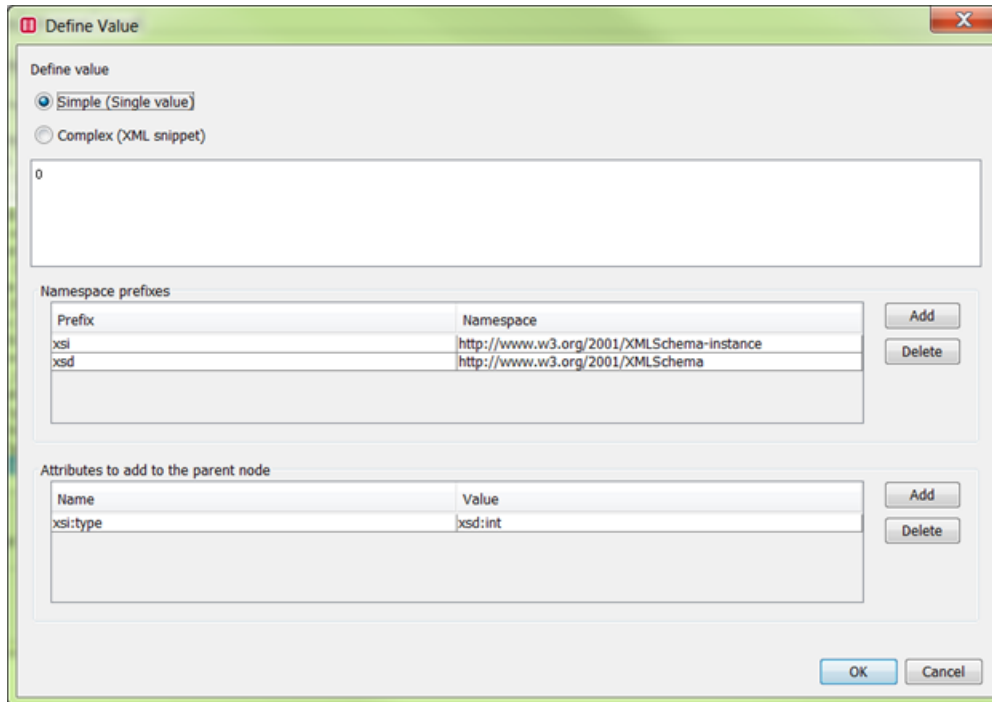


Double-clicking the entry presents the dialog for entering the value. Select the **Complex (XML snippet)** option if the value is a XML fragment that should be added to the final payload. If the **Any type** field requires a single value select the **Simple (Single value)** option.

The **namespace prefixes** table defines the namespaces that are used by any XML snippet and attributes defined for the parent element.

The **Attributes to add to the parent node** table allows attributes to be defined that will be added to the parent element.

The image displayed below shows a simple value being defined with parent attributes and namespace prefixes.



The 'Define Value' dialog box is shown with the following settings:

- Define value:**
 - ☒ Simple (Single value)
 - ☐ Complex (XML snippet)
- Value:** 0
- Namespace prefixes:**

Prefix	Namespace
xsi	http://www.w3.org/2001/XMLSchema-instance
xsd	http://www.w3.org/2001/XMLSchema
- Attributes to add to the parent node:**

Name	Value
xsi:type	xsd:int

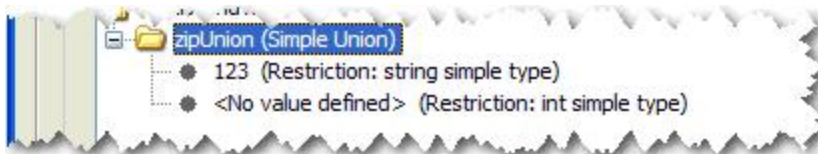
Buttons: Add, Delete, OK, Cancel.

Assuming the parent tag was called "ns1:anyType" the settings above would resolve to the following element in the payload xml:

|

Note: Dynamic functions can be entered as field values in the XML snippet, but a dynamic function cannot be used to define the XML structure. For example, you cannot have a single message section containing the XML, and then enter only a single dynamic function expression in the dialog. The XML in the dialog must be valid and should be able to be parsed without having to resolve the dynamic function expressions. For simple types, dynamic functions can be freely used.

- Nodes with the datatype **Simple Union** means that the operation is expected to have only one of the child items to have a value defined (see <http://www.w3.org/TR/xmlschema-0/#UnionDt>).



- Nodes with a datatype of **Simple List** allow multiple items of the same simple datatype to be added to the list (see <http://www.w3.org/TR/xmlschema-0/#ListDt>). Right-click on the node to add more items to the list.



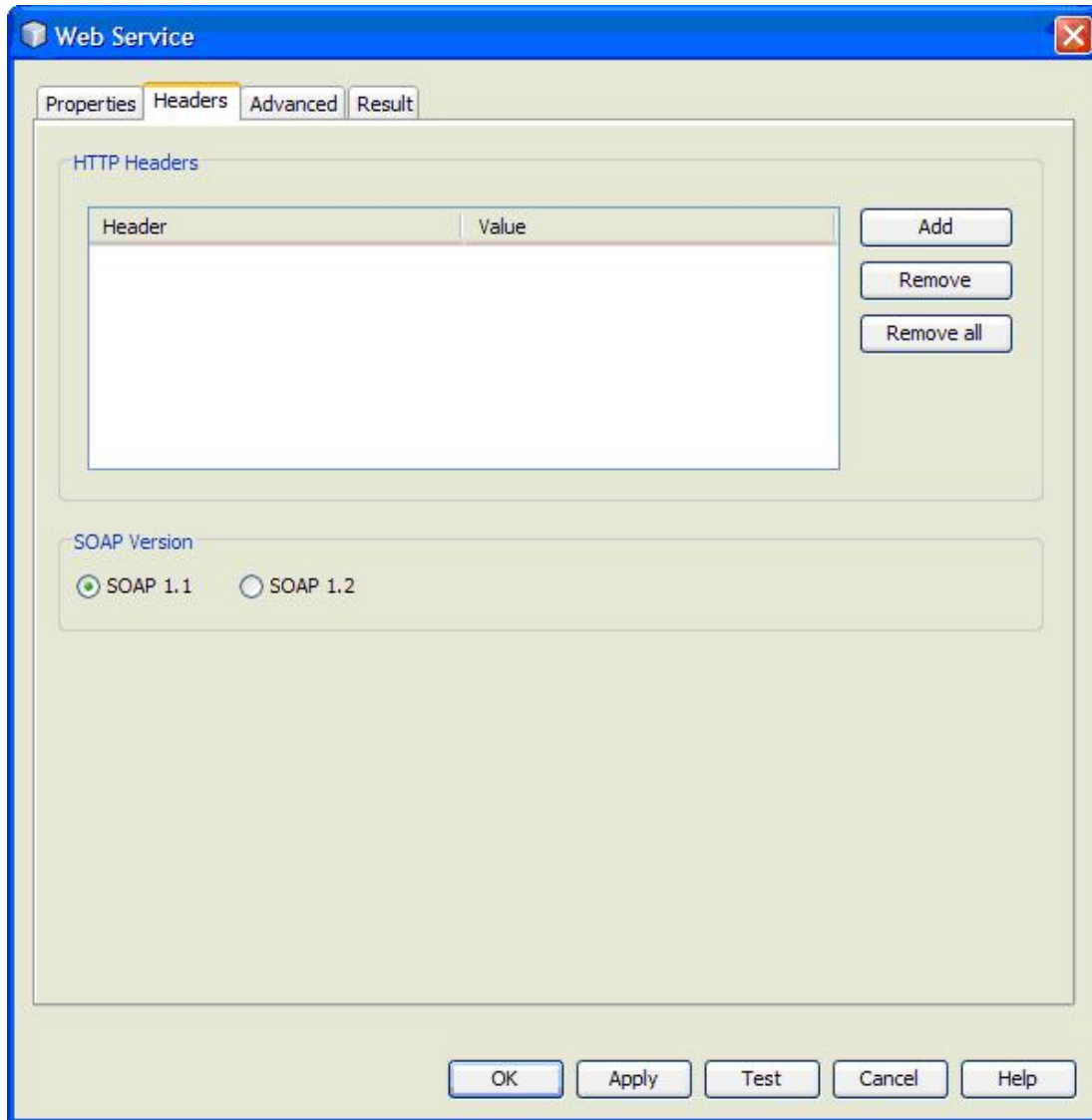
12. In the **Response MessageSection**, enter the name of the message section containing the results that will be created when the operation is run.
13. By selecting the **Pretty Print** check box, the response message section will be formatted well to make it easier to read (line breaks and indentation).
14. To test the Web service operation, press the **Test** button. The result of the call will be displayed in the **Result** tab.

Note: HTTP redirection is not supported; so if your Web service end point automatically redirects elsewhere, then this will cause issues (this can be overcome by setting the endpoint URL to the required site).

Tip: When accessing Sharepoint Web services, use the SOAP 1.1 ports and not 1.2. For example, ListsSoap and not ListSoaps12. This will allow error messages to be correctly retrieved.

Headers Tab

The Web Service module **Headers** tab allows you to define additional HTTP Headers and the SOAP version.



HTTP Headers

Additional HTTP Headers that may need to be sent with the Web service request can be defined. For example, the eBay SOAP Web services need additional items, such as an application security key, set in the HTTP headers. The documentation for the Web service that you are calling should have documented any additional headers that need setting.

Use the **Add/Remove/Remove All** buttons to add or remove the headers from the table. Dynamic functions may be used for the Header and/or Value.

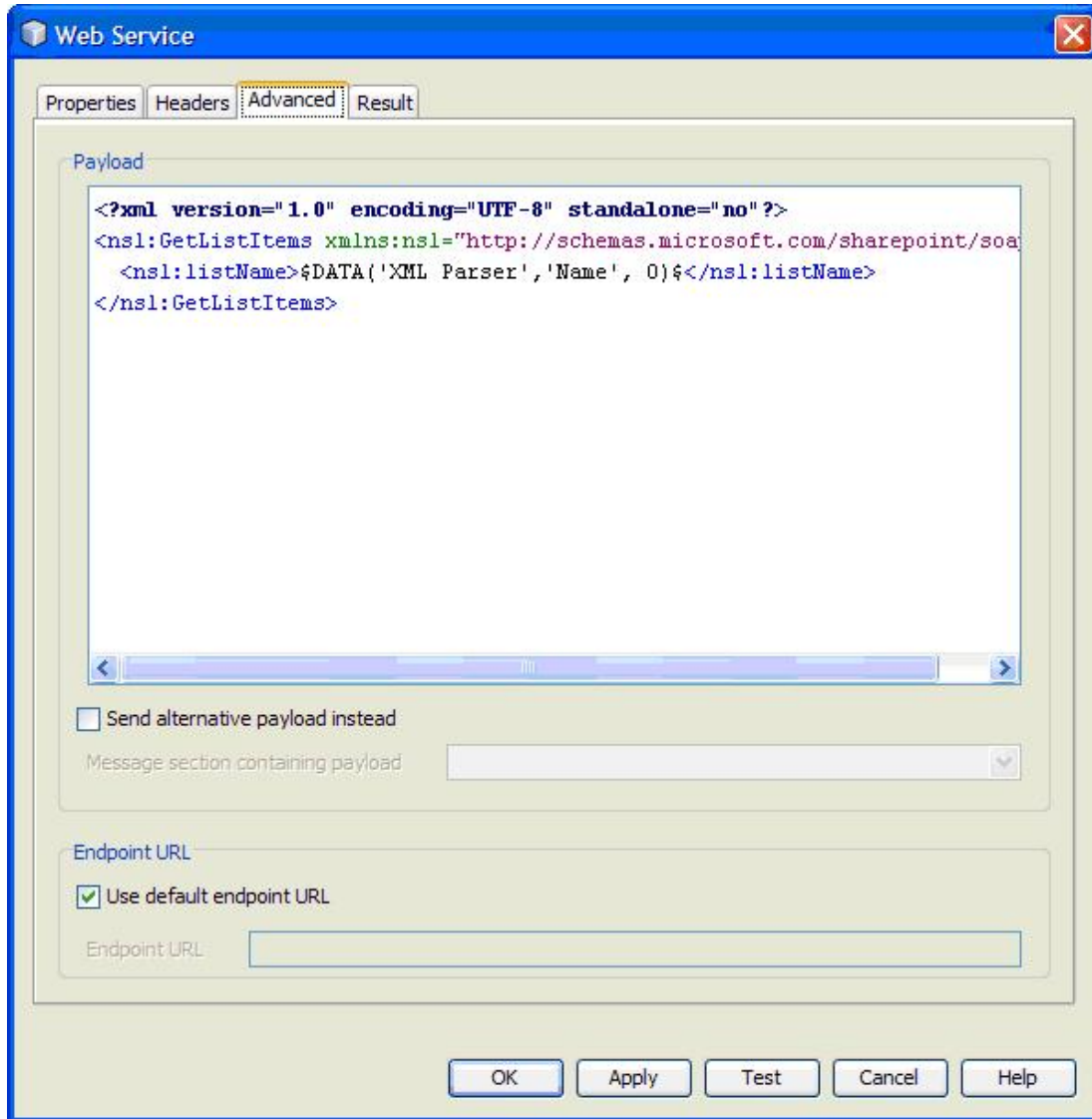
SOAP version

The **SOAP version** radio buttons are used to control the SOAP version that will be used to make the Web service request. When a Web service operation is selected in the **Properties** tab, an attempt is made to automatically detect the SOAP version it should

use, and the radio button will be selected appropriately. However, you can change this setting after the operation has been selected.

Advanced Tab

The following figure shows the Web Service module **Advanced** tab (with an example payload included):



Payload

The payload section displays the XML message that will be sent as the message to the web service. Note that it does not show dynamic functions resolved. They will not be resolved until the message is sent.

Some complex payloads cannot easily be constructed using the web service parameters module (e.g. where the payload needs to be built by iterating over a data section building it up line at a time). Therefore, it is possible to build the payload outside of the web service module using standard text formatters, and then checking the Send alternative payload instead and selecting the Message section containing the payload that contains the XML that should be sent to the web service.

Endpoint URL

The endpoint URL is the web address that the web service call will be made to. These are defined in the WSDL file that was selected in the web service service. Typically, you will use the default URL that is defined. However, it is possible to override this and use a different URL. Clear the **Use default endpoint URL option** and then enter a new URL in the **Endpoint URL** field. The server will then use this new URL when calling the web service.

SAP Data Browser Module

You can use a **SAP Data Browser module** to connect to SAP (Systems Applications and Products in Data Processing) R/3 Data Sources and extract information from them in order to populate an EMF Process Data section.

Note: EMF requires SAP Java Connector 2 (SAP JCo 2) or SAP Java Connector 3 (SAP JCo 3) to connect to the SAP backend. SAP JCo supports connections to 4.0B, 4.5B, 4.6B, 4.6C, 4.6D, 4.7, 6.20 and higher. The SAP authorizations required for running the RFCs in the SAP Data browser and the SAP Authorization modules are:

Auth Object: S_RFC

ACTVT: 16

RFC_NAME: SYST, SUSO, RFC1, STUW, SDTX, SUSR , BAPT, SDIFRUNTIME, SEU_COMPONENT

RFC_TYPE: FUGR

Auth Object: S_TABU_DIS

ACTVT: 03

DICBERCLS authorization group for all tables that must be downloaded: &NC&, SC, SS

SUSR : This is required to make authorization checks from the EMF server (**Show missing authorizations** is not selected in SAP auth Module). If you are using only the local copies of the UST , USR tables, then it is not required.

SEU_COMPONENT: To get a list of tables in the databrowser

&NC& : This is required as it allows access to UST tables, unless you already have user-defined authorization groups which give access to UST tables only

How to use a SAP Data Browser Module:

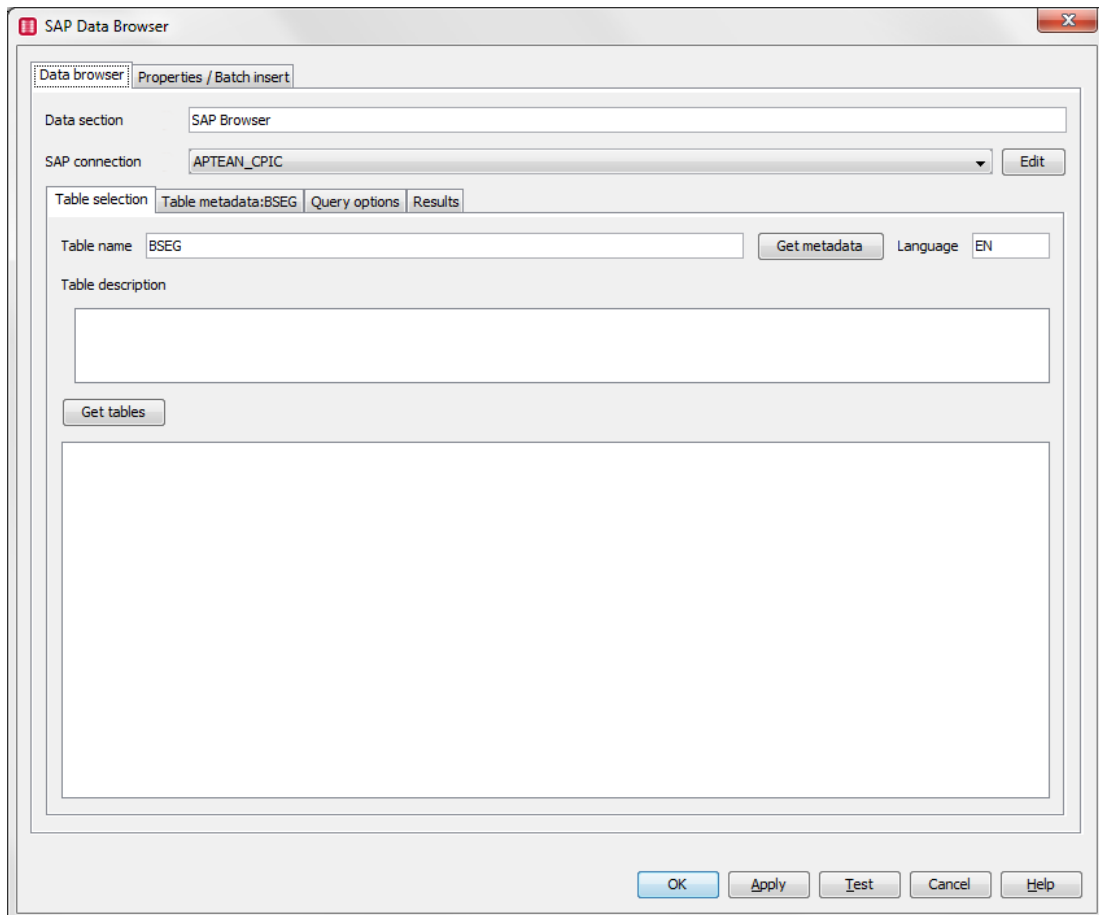
1. Ensure that you have created a [SAP Profile Connection](#) to the Data Source you wish to

use.

2. Drag a **SAP Data Browser module** icon into the EMF process you are creating at the appropriate place.



3. Double-click on the module icon to display the SAP Data Browser screen.



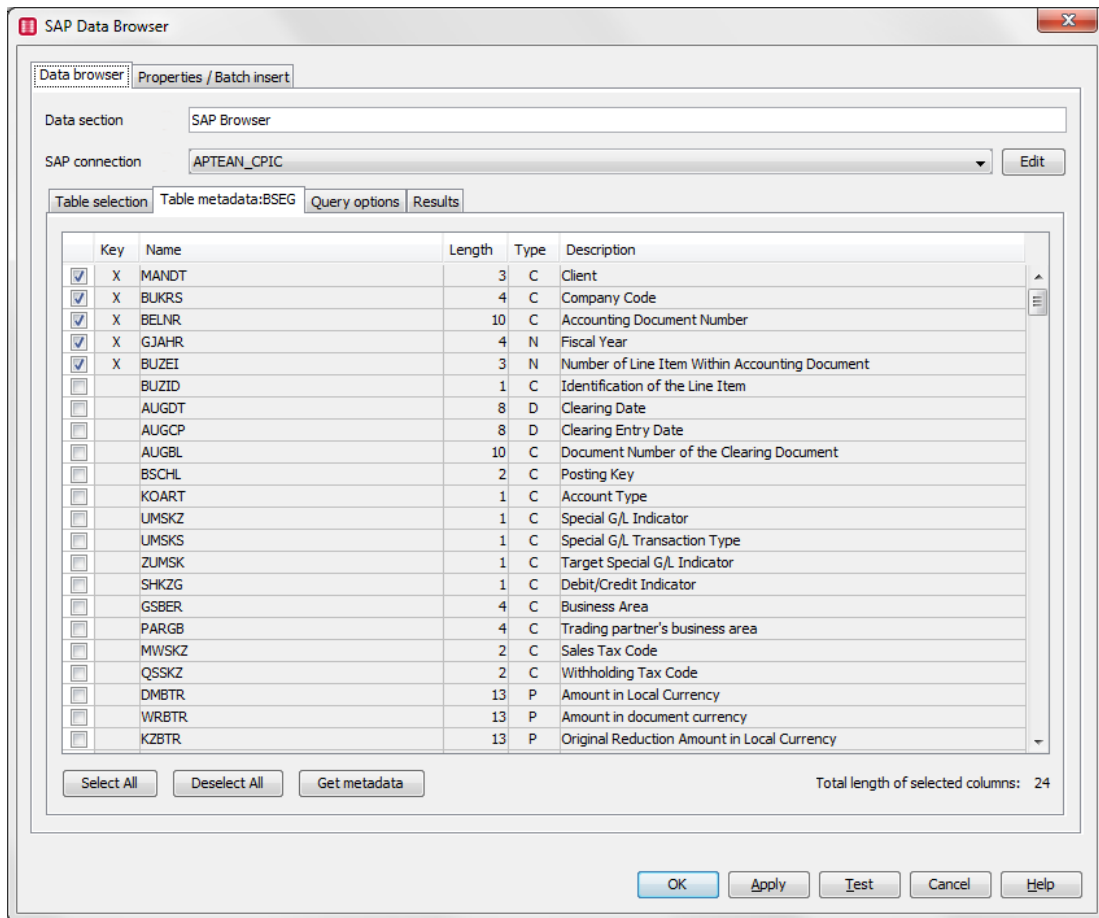
The **Data Browser** tab is used to specify the required SAP connection, and which information is required for the EMF process.

4. Select the connection that you wish to use from the **SAP connection** drop-down list, which shows all of the SAP connections that you have set up in the **Data sources** section of the EMF tree view.
You can click **Edit** to modify the properties of the selected SAP Connection.

5. Enter an appropriate name for the connection in the Data Section field. This will be used to identify the information that is collected from the SAP connection for other modules further on in the EMF process.
6. Select the table that you want to populate the data section with from the list displayed in the Table Selection panel, which displays the tables that can be selected from the SAP Connection. When one is chosen its name is displayed in the **Table name** box.

There are four different types of information displayed in the **Table Selection** panel:

- **SAP** represents the root item node in the EMF tree view. This indicates the SAP R/3 System that your company has access to and that has been defined in the EMF system (via SAP Connections).
 - **Application components** are the child items of the SAP system node in the tree view. The components that are available will depend on your access rights to the SAP R/3 systems.
 - **Object components** are the "child" items of the Application components. These can contain further Object components.
 - **Tables** are child items of the object components. When you double click on a Table item, its name is displayed in the **Table name** box and its description is displayed in the panel below the table name.
 - Enter the **Language** for the table to be queried. This is normally the same as the Language in the SAP Connection.
 - The **Table name** field is either populated from the Table Selection panel or a name can be entered manually in it (if the table name is known). Once the Table name is entered, press the **Get metadata** button to retrieve the metadata for the selected table. The metadata will be displayed in the **Table Metadata** tab.
7. When a Table is queried it must always return some data. The check boxes in the Table Metadata grid allow the user to select the data that they want returned in the **Data Section**. Select the check boxes against the data items you want returned. The **Total length of selected columns** is an indicator of the length of each row in the returned data.



Note: There is no need to select the check box on a row if entering a value for that field - the check boxes should only be selected if that row's value should be returned in the resultant **Data Section**.

- The **Query options** tab allows criteria to be specified about the amount and type of data to return. Parameters are entered in the **Query options** tab - these values can be **dynamic functions** (right click on the **Value** field to access the dynamic function menu). Multiple criteria can be entered by using the **And into query** and **Or into query** buttons. The final Query is shown in the **Query** box. The query can be edited manually in the **Query** box, if required.

The screenshot shows the 'SAP Data Browser' window with the 'Properties / Batch insert' tab selected. The 'Data section' is set to 'SAP Browser' and the 'SAP connection' is 'APTEAN_CPIC'. Under the 'Table selection' tab, a query is defined: 'MANDT EQ 700'. The 'Query' text area displays 'MANDT EQ '700''. Below the query, there are two checked options: 'Return all rows' and 'Auto trim'. The 'Batch size' is set to 5000, and the 'Number of rows to skip' is set to 0. At the bottom, there are buttons for 'OK', 'Apply', 'Test', 'Cancel', and 'Help'.

- Return all rows - If selected, then all data that matches the specified query will be returned. Due to memory limitations the data needs to be pulled back in batches. The size of those batches is specified by the Batch size entry field. If **Return all rows** is not selected then the Batch Size entry box becomes **Maximum number of rows** to be returned. This allows the maximum number of rows that should be returned to be defined. This data is returned as one batch, so there will be a limit on the amount of data that can be returned (dependant on the amount of memory available).
 - To skip a certain number of rows from the results returned, enter the number in the **Number of rows to skip** field. The data returned will not contain the first x number of rows that have been specified in that field.
 - **Auto Trim** - If selected, any trailing spaces on the returned data will be removed.
9. Select the **Properties** tab to define filters and halt conditions.
 10. Select the **Batch Insert** tab to copy the returned data directly into a database table.
 - Create a table if it does not exist.
 - Delete the data section after batch insert.
 11. When you have finished, click the **Test** button to view the data returned from the Table.
 12. Click the **Results** tab to preview the output.

[SAP Data Browser Module Properties/Batch insert tab](#)

[Data Sources \(Configuring a SAP Profile\)](#)

[Go to start of Data Modules](#)

File In Module

You can use the **File In** module to retrieve information and contents of files and directories from the following sources:

- Local files
- FTP sites
- Zip, Jar, and Tar compressed files
- HTTP or HTTPS sites

How to use a File In Module

1. Drag a **File In module** icon into the EMF Process you are creating, at the appropriate location:



2. Double-click on the module icon to display the **File In** properties.

3. Type a name for the **Data section** in which the returned file/directory content/information will be saved. See [Returned Data Section Description](#) for information on the structure of returned data.
4. Select the file service that you wish to use from the **File service** drop-down list. If you are running in a single server environment, then selecting the "Default File Service" will be ok. In a clustered server environment, the selected service will give you control over which server this module will execute on. This may be important if the file/directory to read is located on a specific server. You can click **Edit** to modify the properties of the selected **File Service**.
5. **File Path** - Enter the file or directory of interest. If you wish to bring back the contents of a directory, then select a directory. If you wish to bring back a single file, then the file may be selected directly. You can use [dynamic functions](#) in this text box. Do not enter any wild cards in the name. The contents of a directory can be filtered using the File Filter option.
6. Click **Test** to test the connection.

If the file of interest is not a local file, then it is necessary to specify the required protocol in the file path. The following describes the available protocols:

File type	Format	Description	Examples
Local Files	[file://] absolute-path	Access the files/directories on the	file:///home/someuser/somedirfile:///C:/Documents and Settings file:///somehost/someshare/afile.txt

			/home/someuser/somedir
		local physical file system	c:\program files\some dir
			c:/program files/some dir
FTP	ftp://[username[: password]@]hostname[: port][absolute-path]	Access the files/directories on an ftp server	ftp://user:password@host/pub/file.tgz
zip, tar and jar compressed archives	zip:// arch-file-uri[! absolute-path]	Access the contents of a zip, jar, tar archive file.	
	jar:// arch-file-uri[! absolute-path]	arch-file-uri refers to a file of any supported type, including other zip files. Note: if you would like to use the ! as normal character it must be escaped using %21. tgz and tbz2 are convenience for tar:gz and tar:bz2.	
	tar:// arch-file-uri[! absolute-path]		
			jar:../lib/classes.jar!/META-INF/manifest.mf
			zip:http://somehost/downloads/somefile.zip
			jar:zip:outer.zip!/nested.jar!/somedir
HTTP and HTTPS	http://[username[: password]@]hostname[: port][absolute-path]	Access the files/directories on an http/https server	http://somehost:8080/downloads/somefile.jar
	https://[username[: password]@]hostname[: port][absolute-path]		http://myusername@somehost/index.html

7. **File Filter** - This is only relevant if what is selected in File Path points to a directory rather than a file. It will filter the returned files from the directory that match only the filter entered. The filter uses the characters '?' and '*' to represent single or multiple wildcard characters. This is the same as found on Dos/Unix command lines. The testing is always case-sensitive. For example, to return only files that have the extension txt, enter ***.txt**.

8. **Read File Contents** - Select this check box if the contents of the file should be read and stored in the resultant data section. If this check box is not selected, then only the details of the file are retrieved (that is file name, size, last modified time etc). See [Returned Data Section Description](#) for more details.
9. **Store contents as** - These options are only available if **Read File Contents** has been selected. By default, files will be read and stored as **Text**. This allows the contents of the file to be used in creating messages in the same way as other text data in EMF. If **Binary** is selected, then files are read in and stored in their raw form. This allows limited special functionality to be performed on the content (e.g. it can be stored or written out again in exactly the same format). See [Binary Data](#) for more information on the **Binary** option.

Returned Data Section Description

Once the File In module runs, it creates a data section using the name entered in the module properties. If the file path points at a directory then the data section will contain a row for every item in that directory (files/sub directories) after the File Filter has been applied. If the file path points to a file, then the data section will contain a single row with the file details in.

If the file/directory cannot be found then an error will be logged and no data section will be created. The error can then be handled by an error link, if required.

The columns in the returned data section are as follows:

Column Name	Description
NAME	The name of the file/directory
TYPE	Either file or folder
LASTMODIFIEDDATE	The time the file/directory was last modified
SIZE	The size of the file - no value if it is a directory
ISHIDDEN	If the file/directory has a hidden attribute set
ISREADONLY	If the file/directory has a read only attribute set
URI	The full URI (path) to the file/directory
CONTENTS	This column will only exist if the Read File Contents check box is selected. The contents of the file are stored in this column. To output the contents of file in a message use the \$DATA\$ dynamic function e.g. \$DATA('File in','CONTENTS',0)\$

Tip: You can use a File In module followed by a Data Filter module to monitor a directory and take action on files changing in a directory. For example, if invoices are stored as separate files in a directory, you can monitor the directory and send an e-mail to notify another person that a new invoice had been raised.

[Go to start of Data Modules](#)

FTP In Module

You can use the **FTP In** module to retrieve information and contents of files and directories from an **FTP** server.

How to use an FTP In Module

1. Drag an **FTP In module** icon into the EMF Process you are creating, at the appropriate location:



2. Double-click on the module icon to display the **FTP In** properties.

3. Type a name for the **Data section** in which the returned file/directory content/information will be saved. See [Returned Data Section Description](#) for information on the structure of returned data.
4. Select the FTP service that you wish to use from the [FTP service](#) drop-down list. You can click **Edit** to modify the properties of the selected [FTP service](#).
5. **FTP Path** - After selecting the **FTP service**, the FTP path will automatically display the default directory. You can use [dynamic functions](#) in this text box. Do not enter any wild cards in the name. The FTP Path should either be a path to a directory, or to a single file on the FTP server. The contents of a directory can be filtered using the File Filter option.
6. **File Filter** - This is only relevant if what is selected in FTP Path points to a directory rather than a file. It will filter the returned files from the directory that match only the filter entered. The filter uses the characters '?' and '*' to represent single or multiple

wildcard characters. This is the same as found on Dos/Unix command lines. The testing is always case-sensitive. For example, to return only files that have the extension txt, enter ***.txt**.

7. **Read File Contents** - Select this check box if the contents of the file should be read and stored in the resultant data section. If this checkbox is not selected, then only the details of the file are retrieved (that is file name, size, last modified time etc). See [Returned Data Section Description](#) for more details.
8. **Store contents as** - These options are only available if **Read File Contents** has been selected. By default, files will be read and stored as **Text**. This allows the contents of the file to be used in creating messages in the same way as other text data in EMF. If **Binary** is selected, then files are read in and stored in their raw form. This allows limited special functionality to be performed on the content (e.g. it can be stored or written out again in exactly the same format). See [Binary Data](#) for more information on the **Binary** option.
9. Click **Test** to test the module settings. This will attempt to connect and read the directory/files using the settings given. A sample data section can then be saved for use in later modules.

Returned Data Section Description

Once the FTP In module runs, it creates a data section using the name entered in the module properties. If the FTP path points at a directory then the data section will contain a row for every item in that directory (files/sub directories) after the File Filter has been applied. If the FTP path points to a file, then the data section will contain a single row with the file details in.

If the file/directory cannot be found then an error will be logged and no data section will be created. The error can then be handled by an error link, if required.

The columns in the returned data section are as follows:

Column Name	Description
NAME	The name of the file/directory
TYPE	Either file or folder
LASTMODIFIEDDATE	The time the file/directory was last modified
SIZE	The size of the file - no value if it is a directory
ISHIDDEN	If the file/directory has a hidden attribute set
ISREADONLY	If the file/directory has a read only attribute set
URI	The full URI (path) to the file/directory
CONTENTS	This column will only exist if the Read File Contents check box is selected. The contents of the file are stored in this column. To output the contents of FTP in a message use the \$DATA\$ dynamic function e.g. \$DATA('FTP in','CONTENTS',0)\$

Tip: You can use an FTP In module followed by a Data Filter module to monitor a directory and take action on files changing in a directory. For example, if invoices are stored as separate files in a directory, you can monitor the directory and send an e-mail to notify another person that a new invoice had been raised.

[Go to start of Data Modules](#)

SAP Authorizations Module

Overview

To meet the high demands of data protection and security, SAP provides the following R/3 security mechanisms:

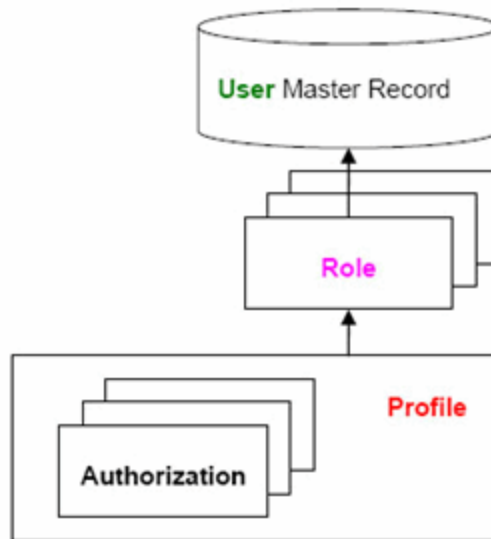
- Access protection and authentication outside of R/3
- Authorization concept (Discussed and used in this document)
- Secure network communication
- Activity logging

We will only deal with the Authorization concept as that is what the SAP Authorization module is designed for. SAP Authorizations are designed for security and to assign appropriate privileges for end users to fulfill their job duties. The SAP R/3 authorization concept permits the assignment of general and / or finely detailed user authorizations. These assignments can reach down to the transaction, field and field value level. These authorizations are centrally administered in user master records and most allow the handling of certain R/3 components applicable to specific operations. Actions by a user may require several authorizations. For example, to change a material master record, authorizations are required for the:

- Transaction "change"
- Specific material
- General authorization to work within the company code

The resulting relationships can become very complex. To meet these requirements, the R/3 authorization concept has been implemented as a form of pseudo-object-oriented concept with complete authorization objects. Each authorization object is a combination of authorization fields. An authorization always refers to an authorization object and can contain ranges for the field values. Authorization checks protect the functions or objects that you choose.

Authorizations are created by administrators, which are then assigned to users in collections called profiles. These profiles are contained within Roles.



Authorization Object

An authorization object is composed of a number of authorization fields that represent the smallest component against which authorization checks may be programmed. The authorization object contains a maximum of ten such fields per object. Users may only conduct an activity if they satisfy the authorization check for each field in the authorization defined on a specific authorization object.

Authorization Object Fields

The fields in an authorization object are linked to data elements in the SAP ABAP Dictionary. The permissible values constitute an authorization. When an authorization check takes place, the system checks the values you have specified in an authorization against those required to carry out the action. Users may only carry out the action if they satisfy the conditions for every field defined for a specific authorization object.

Authorizations

An authorization allows you to carry out an R/3 task based on a set of field values in an authorization object. Each authorization refers to exactly one authorization object and defines the permitted value range for each authorization field of this authorization object. Authorizations are utilized in the user master record as profiles. By themselves, authorizations do not exist. They only have meaning inside a profile. Example:

Field	Value
Customer type (CUSTTYPE)	* (all possible values)
Activity (ACTVT)	02 (display)

The authorization above allows a user to display records for all customer types (Note that the EMF SAP Authorization module allows you to be flexible with the way * is handled. It allows you to define how * is to be interpreted within each individual SAP Authorization module).

Authorizations are used to specify permitted values for the fields in an authorization object. There may be one or more values for each field. Authorizations allow you to determine the number of specific values or value ranges for a field. Changes to authorizations affect all users whose authorization profile contains that authorization.

Authorization Profiles

User authorizations are assigned to users as authorization profiles. Authorization profiles can either be single or composite. Composite profiles are collections of many single profiles. Example: A profile Z_CUSFI01 (maintain customization in FI area) may have the following authorizations:

Profile	Object	Authorization
Z_CUSFI01	S_TABU_DIS	Z_CUSFI01_01
Z_CUSFI01	F_T011	F_ALL
Z_CUSFI01	F_BKPF_VW	F_ALL

Roles

Roles are assigned to users so that they can perform certain tasks. These roles contain authorization profiles to limit what the user can do within the R/3 system. This allows the user to use only the functionality that is within his role. Example: A Role Z_CUST_FI (Customization for FI area) could contain the profile Z_CUSFI01, among other profiles, to enable the user with this role to perform certain activities within the FI area:

Role	Profile
Z_CUST_FI	Z_CUSFI01
Z_CUST_FI	Z_CUSFI02
Z_CUST_FI	ZZ_RFCACL

Introduction

The SAP Authorization module is able to evaluate authorizations at the transactional level and/or at the authorization object level, to generate information about users who are authorized to perform activities based on specified criteria.

The following sections detail the setting up of the SAP Authorization module (Authorizations check module) within processes.

Before building an alert within processes using the SAP Authorization module, it is important to understand the requirements for a successful implementation of the module. The SAP Authorization module allows you to run against either a SAP System or a local

Database that contains the necessary tables to run the authorizations checks. An EMF data section is populated with the results of the authorization check. This data section can be used for further data processing by other modules within processes.

Running against a SAP System will take longer to run the queries/authorization checks in EMF and may take up valuable resources on the SAP System. On the other hand, provided the local database is up to date, running against it will be more efficient and takes less time.

Note: EMF requires SAP Java Connector 2 (SAP JCo 2) or SAP Java Connector 3 (SAP JCo 3) to connect to the SAP backend. SAP JCo supports connections to 4.0B, 4.5B, 4.6B, 4.6C, 4.6D, 4.7, 6.20 and higher.

Setting up the local database

EMF requires certain SAP tables to be available in order to run the authorization queries. All fields are not necessarily required in all of the tables stored locally. At the time of writing, the following SAP tables are required by EMF:

UST12, UST04, UST10S, UST10C, USR02, AGR_1016, USREFUS.

Template processes are available, that will create these tables and populate the data from the SAP system.

How to use a SAP Authorizations Module

1. Ensure that you have created a SAP Profile Connection to the Data Source you wish to use. If you wish to use a local datasource, then ensure you have created a JDBC connection to any JDBC Datasource you want to access.
2. Drag a **SAP Authorizations** module icon into the process you are creating at the appropriate place.



3. Double-click on the module icon to display the SAP Authorizations screen.

The **Properties** tab is used to specify the required SAP connection, Data section and the authorization objects to check.

4. Select the connection that you wish to use from the **SAP connection** drop-down list, which shows all of the SAP connections that you have set up in the Data sources section of the EMF tree view. When you have selected the required connection, the **SAP client** field will automatically display the client that you are connecting to.
5. If using a local database, select the **JDBC connection** from the drop-down list, and enter the SAP client manually in the **SAP client** field. You can click **Edit** to modify the selected JDBC connection.
6. Enter an appropriate name for the module in the **Data Section** field. This will be used to identify the information that is collected from the authorization check, for other modules further on in the process. In addition to the named **Data Section** (`<DataSection>`), there are two other data sections created; `<DataSection>_HEADER` and `<DataSection>_CHECKS`.
7. Transaction codes can be added in the **Transaction** column, using the **Add** button below the **Transaction** column. To delete a transaction code, select it and press the **Delete** button below the **Transaction** column. Empty rows are ignored and will not be saved. If more than one transaction code is specified, the logical OR relationship is used. For example, Transaction codes SU10, PFCG would bring back authorizations

which had either SU10 or PFCG transaction codes associated with them. The Transaction codes use a logical AND with the Authorization objects to bring back the list of users.

8. Authorization objects to look for can be entered in the right hand pane, using the **Add** button below the Authorizations grid. To delete an Authorization object entry, select the row and press the **Delete** button below the Authorization grid. If more than one authorization object is specified, then a logical AND relationship is used. If the same field is entered for the same Authorization object more than once, an OR relationship will be used for those object/field entries. For example, in the diagram above, only those authorizations will be returned that have either activity 02 OR activity 22 for the S_USER_GRP AND have activity 02 for the S_USER_AGR object.

[Advanced options Tab](#)

[Domains tab](#)

Segregation of Duties (SoD) Module

Segregation of duties is a basic, key internal control and one of the most difficult to achieve. It is used to ensure that errors or irregularities are prevented or detected on a timely basis by employees in the normal course of business. Segregation of duties provides two benefits:

- A deliberate fraud is more difficult because it requires collusion of two or more persons, and
- It is much more likely that innocent errors will be found.

At the most basic level, it means that no single individual should have control over two or more phases of a transaction or operation.

For example, no one individual should be able to set up a new supplier, raise a purchase order for that supplier, post and approve the invoice from that supplier, create, approve and record the payment to that supplier. This is because giving a single individual the ability to perform all of the above operations increases the risk of fraud or error.

A SoD project aims to ensure that conflicting activities that cannot be assigned to the same individual are appropriately segregated. For example, this is a key part of achieving SOX/MiFID/ISO2100 compliance.

Segregation of Duties must be performed in a way that it reduces the risk associated with a function/process that can be malfunctioned to practice any fraudulent exercises. If proper SoD does not exist in an organization, then:

- There are ineffective internal access controls.
- There is improper use of materials, money, financial assets and resources.
- Estimation of financial condition may be wrong.
- Financial documents produced for audits and review may be incorrect.

There are circumstances where proper Segregation of Duties cannot be implemented. In such cases there should be a mitigating control designed in order to keep a check on the unresolved SoD.

One of the ways that the segregation can be checked is to identify users who have conflicting authorizations within their profiles. EMF can check for these conflicting authorizations by using one or more SAP Authorization modules in combination with the Segregation of Duties module in a process. The Segregation of Duties module applies the necessary rules (i.e. logical OR/AND) on data sections (created by the SAP Auth module) to create one Segregation of Duties result showing the conflicts.

How to use a Segregation of Duties Module

1. Drag a **Segregation of Duties** module icon into the process you are creating at the appropriate place.



2. Double-click on the module icon to display the **Segregation of Duties** screen.

 The screenshot shows a window titled "Segregation of Duties". At the top, there is a "Data section" text box containing "Seg of Duty". Below this is a section titled "Datasections to perform action on" which contains a table with three columns: "Auth datasection", "Operator", and "Group". The table is currently empty. To the right of the table are buttons: "Add", "Remove", "Remove all", "Move up", and "Move down". Below the table is a section titled "Column to match on" with two radio buttons: "Specify by index (zero-based)" and "Specify by name". The "Specify by name" radio button is selected. Below this is a text box labeled "Column ID / Name" containing the text "USER". At the bottom of the window are two text boxes labeled "Short description" and "Long description", both of which are empty. At the very bottom are four buttons: "OK", "Apply", "Cancel", and "Help".

3. Type the name of the data section in the **Data section** text box.

4. Click on **Add** to add the data sections on which the segregation actions are to be performed. You have to add the authorized data section, the name of the operator and the group for the same.
5. You can specify the criteria based on which the columns are to be matched. Select either of the two mentioned below:
 - Specify by index (zero-based)
 - Specify by name (if you select **Specify by name** then provide the Column ID or name in the **Column ID/Name** text box.)
6. You can provide a description for the segregation and click **OK**.

SNMP Module

The SNMP module allows you to send and receive SNMP data in an EMF Process. You can use the SNMP module as either an SNMP agent or an SNMP network manager. For example, you can use it to send traps (agent role), or you can use it to request information from a network device (network manager role). Data returned to the SNMP module is saved to a data section and can be used by other modules in the EMF Process.

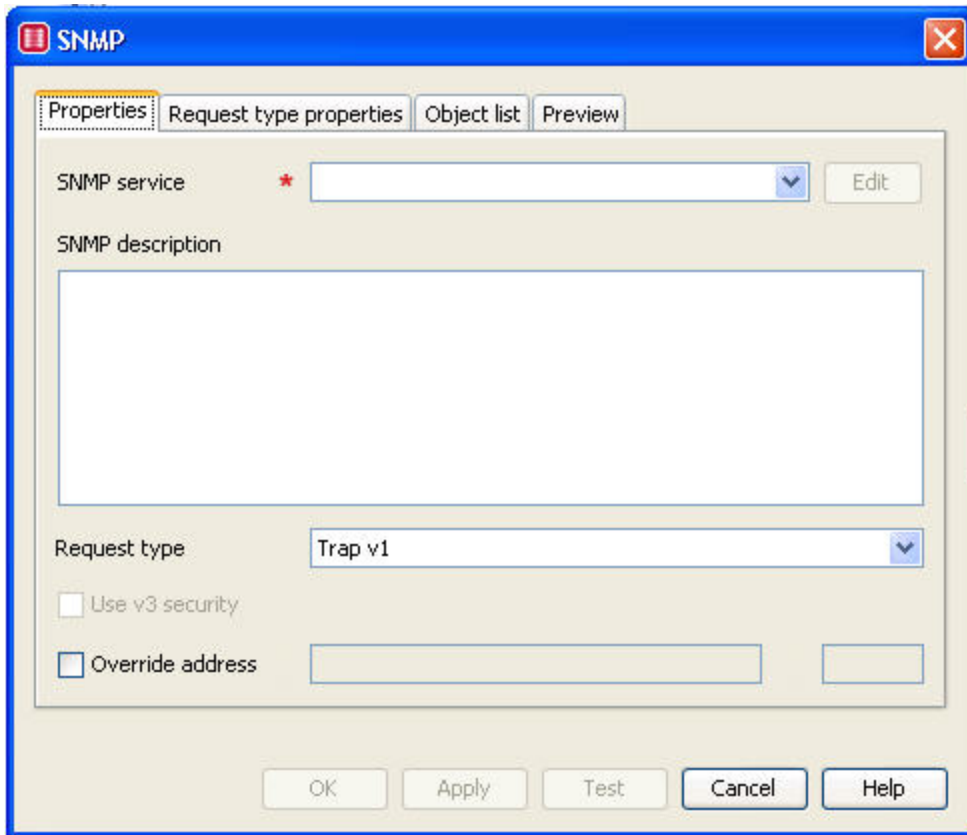
Before you use the SNMP module, it is recommended that you have an understanding of SNMP. For more information on SNMP, see [SNMP Overview](#).

To use an SNMP module:

1. If you have not already done so, [configure the SNMP service](#) that you want the SNMP module to use.
2. Drag the **SNMP** icon onto the EMF Process Design graph at the appropriate place.



3. Double-click the SNMP icon to display the **SNMP Properties** screen:



4. Select the required **SNMP Service** from the drop-down list of those that you have set up.
You can click **Edit** to modify the properties and security settings of the selected **SNMP Service**.
5. Enter an **SNMP description** - this is purely for information purposes to remind you what the module is for.
6. Select the type of message that you require from the **Request type** drop-down list. The selection that you make determines what additional properties (if any) are available in the [Request type properties](#) tab. You must specify the object(s) that the request type should reference on the **Object list** tab (not necessary for the Trap request type).

The options available are **Trap** (v1 or v2c), **Get** (v1 or v2c), **Get next** (v1 or v2c), **Get bulk** (v2c only), **Set** (v1 or v2c) and **Inform** (v2c only). For more information on the message types, see [SNMP Overview](#).

Note: message types can be v1 or v2c compatible. If you are using SNMPv1, select a v1 option. If you are using SNMPv2c or SNMPv3, select a v2c option.

7. If you want to use the security features of SNMPv3, select **Use V3 security**. This checkbox is only available if you selected a v2c request type.

8. If you wish to override the network address and port number specified in the selected SNMP service, select **Override address** and enter the required details in the textboxes. You can also use a [dynamic function](#) to obtain the network address.
9. Select the [Request type properties](#) tab if you need to define any additional properties as a result of you selected **Request type** (see above).
10. Select the [Object list](#) tab to enter the details of the object(s) that should be referenced by the request type.
11. Click **Test** to view the output data in the **Preview** tab.
12. Click **Save as sample** to save the data for testing purposes.

[Request type properties tab](#)

[Object List tab](#)

[Data Modules](#)

Aptean Respond Module

Respond is a case management tool sold by Aptean. EMF can be used to automate the creation and updating of cases in Respond. For example, a process can be built in EMF to take customer complaint records from one system and automatically create case records in Respond for each complaint. A minimum version of Respond v5.9 is needed to operate with EMF. The user should have read and be familiar with the "Respond Web Services User Guide.pdf" available with Respond.

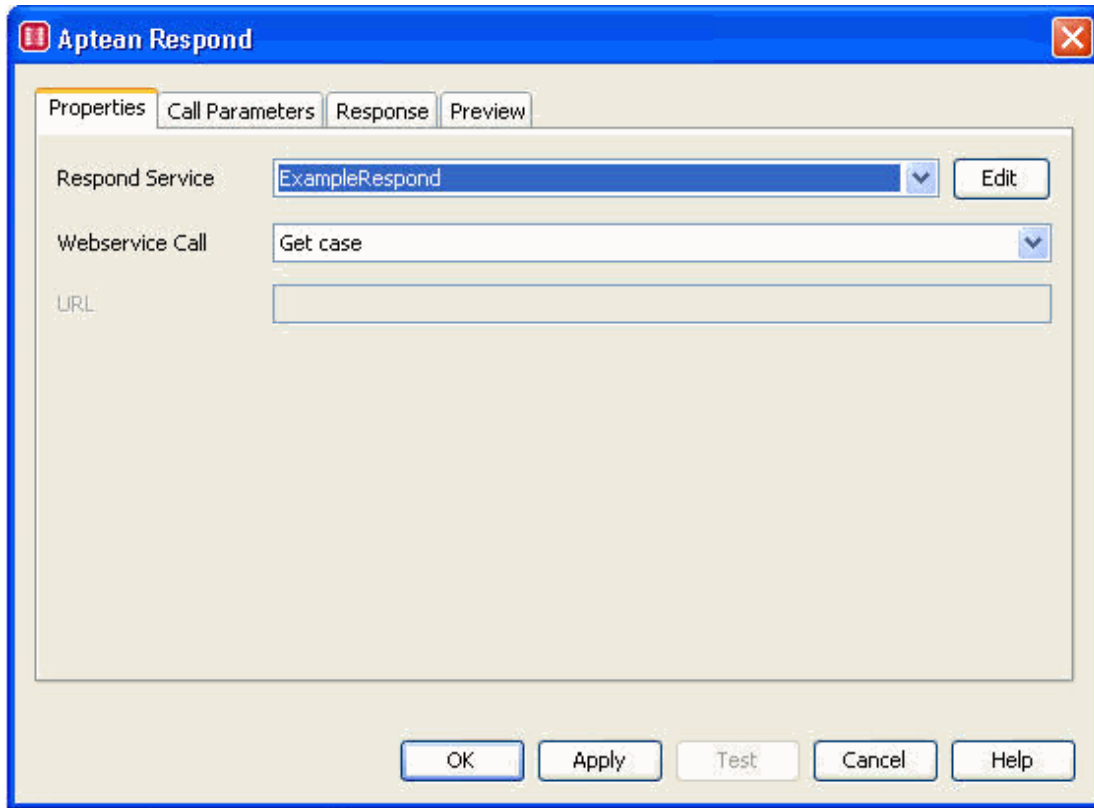
The Aptean Respond module allows Respond Web service calls to be made to Respond. It simplifies the calling process, taking care of the security and formatting of messages. However, the payload for the Web service calls to create and update cases must still be created in XML first.

To add an Aptean Respond module to an EMF Process:

1. In the EMF Process builder, drag the icon for the Respond module to the appropriate place in your EMF process.



2. Open the Respond module to display its properties. By default, you need to double-click the icon to open, but you can configure mouse-click actions on the Options window. From the Tools menu, select **Options -> EMF-> Process Builder Behaviour** tab.



3. Select a **Respond Service** – this defines the Respond database and security settings to which you want to make the Web service call.
Click **Edit** to modify the properties of the selected Respond Service. For more information, see [Aptean Respond Service](#).
4. Select the type of **Webservice Call** that is to be made. The following options are available. These options are described in more detail in the "Respond Web Services User Guide.pdf".
 - **Get case**: The Case XML will be returned for the retrieved Case.
 - **Update case**: An existing Case can be updated, fields set and new child Entities created by providing the properly formatted XML document.
 - **Create case**: A Case and its child Entities can be created and the field values set by providing the properly formatted XML document.
 - **Run search (GET)**: This will allow searches that have been pre-configured to be executed and their result returned. Filter values can be specified. This version is via HTTP GET.
 - **Run search (POST)**: This will allow searches that have been pre-configured to be executed and their result returned. Filter values can be specified in an XML document provided. This version is via HTTP POST.
 - **Other (GET)**: This allows any "GET" Web service call to be called by specifying the URL. It allows for future expansion, to call new Respond Web service types that may be added.
 - **Other (POST)**: This allows any "POST" Web service call to be called by specifying the URL. An XML document can also be specified to send as the body of

the message. It allows for future expansion, to call new Respond Web service types that may be added in.

- **Get Metadata:** Returns the XML metadata for a particular Web service call. Useful to find out the structure and field names of the data.

Depending on the **Webservice call** selected, different fields will be available to set the required values on the **Properties** and **Call Parameters** tabs.

- **URL:** This field is only enabled for Other (GET) and Other (POST). If using these methods, then the full URL of the Web service call needs to be specified.

Call Parameters tab

The Call Parameters tab allows you to specify the parameters required for each type of Web service call. Different fields will be enabled, depending on the Web service call selected.

The screenshot shows the 'Apteian Respond' dialog box with the 'Call Parameters' tab selected. The dialog has four tabs: 'Properties', 'Call Parameters', 'Response', and 'Preview'. The 'Call Parameters' tab contains the following fields and controls:

- Definition ***: A text input field.
- Identifier ***: A text input field.
- Lookup Field**: A section containing a checkbox labeled 'Identifier requires lookup field' and a dropdown menu labeled 'Lookup Field' with 'CRM Case ID' selected.
- Message Body**: A dropdown menu.
- Page Number**: A text input field.
- Filter**: A text input field.

At the bottom of the dialog are five buttons: 'OK', 'Apply', 'Test', 'Cancel', and 'Help'.

- **Definition:** The name given to the Web service definition created in Respond Configuration Manager. For example, for a search Web service call, the definition identifies the service search to run.
- **Identifier:** The case identifier for update and get case Web service calls. It can be one of several values:
 - Case Unique Identifier (Guid)
 - Case Reference Number
 - Some other field, such as Case CRM Id

- When using an identifier other than Case Unique Identifier or Case Reference Number, the **Identifier requires lookup field** check box should be selected and the appropriate Lookup Field value selected from the drop-down list.
- **Message Body:** Identifiers the message section that should be sent as the request body for the Web service call. This should be an XML document. So, for example, with a create case Web service call, the message body will contain the XML that represents the case to create.
- **Page number:** Used by a Search (GET) request to specify the page of the search results required. For a Search (POST), this is specified in the message body XML.
- **Filter:** Used by a Search (GET). Any quick expression configured within the searches filter can be set using this field. Multiple values can be set using the format: `<field schema name>=<value>;<field schema name>=<value>`

Response Tab

The Response tab defines the section names that will contain the results returned from the Web service call.

The screenshot shows the 'Aptean Respond' dialog box with the 'Response' tab selected. The dialog has four tabs: 'Properties', 'Call Parameters', 'Response', and 'Preview'. The 'Response' tab contains the following fields and options:

- 'Message section to contain response content' with the text 'Respond Response Contents'.
- 'Message section to contain response code' with the text 'Response Code'.
- An unchecked checkbox labeled 'Store whole request/response HTTP stream (for diagnostics)'.
- 'Whole stream message section name' with the text 'Respond Whole Stream'.

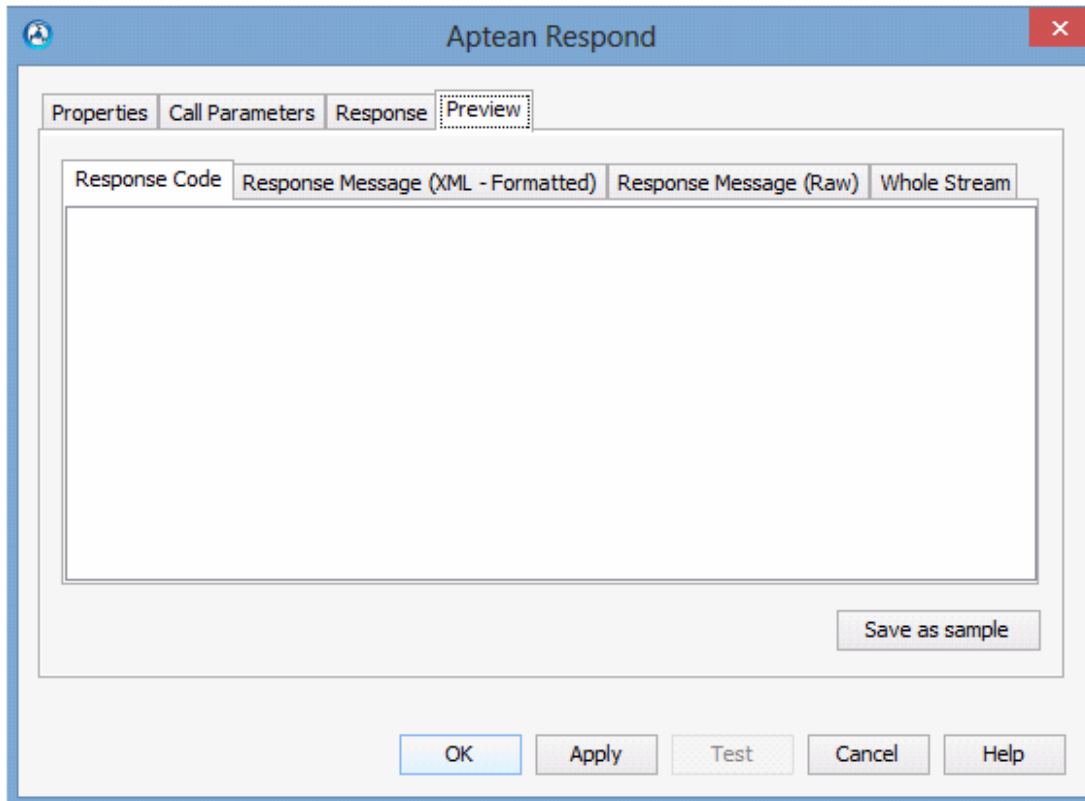
At the bottom of the dialog are five buttons: 'OK', 'Apply', 'Test', 'Cancel', and 'Help'.

- **Message section to contain response content:** This is the name of the message section that will contain the XML (assuming a successful call) returned from the Respond server.
- **Message section to contain response code:** This is the name of the message section that will be created containing the HTTP response code of the call. A successful call will return 200. The result of this message section can be checked on a conditional link to see if processing should continue or some error action should be taken.

- **Store whole request/response HTTP stream:** This check box should be selected if the whole HTTP stream is needed. This is usually only needed to diagnose why a call is not working as expected.

Preview Tab

After all values have been entered as required, the Test button can be used to test the Web service call to the Respond server. The Preview tab displays the resultant message sections and their content that were created during the testing.



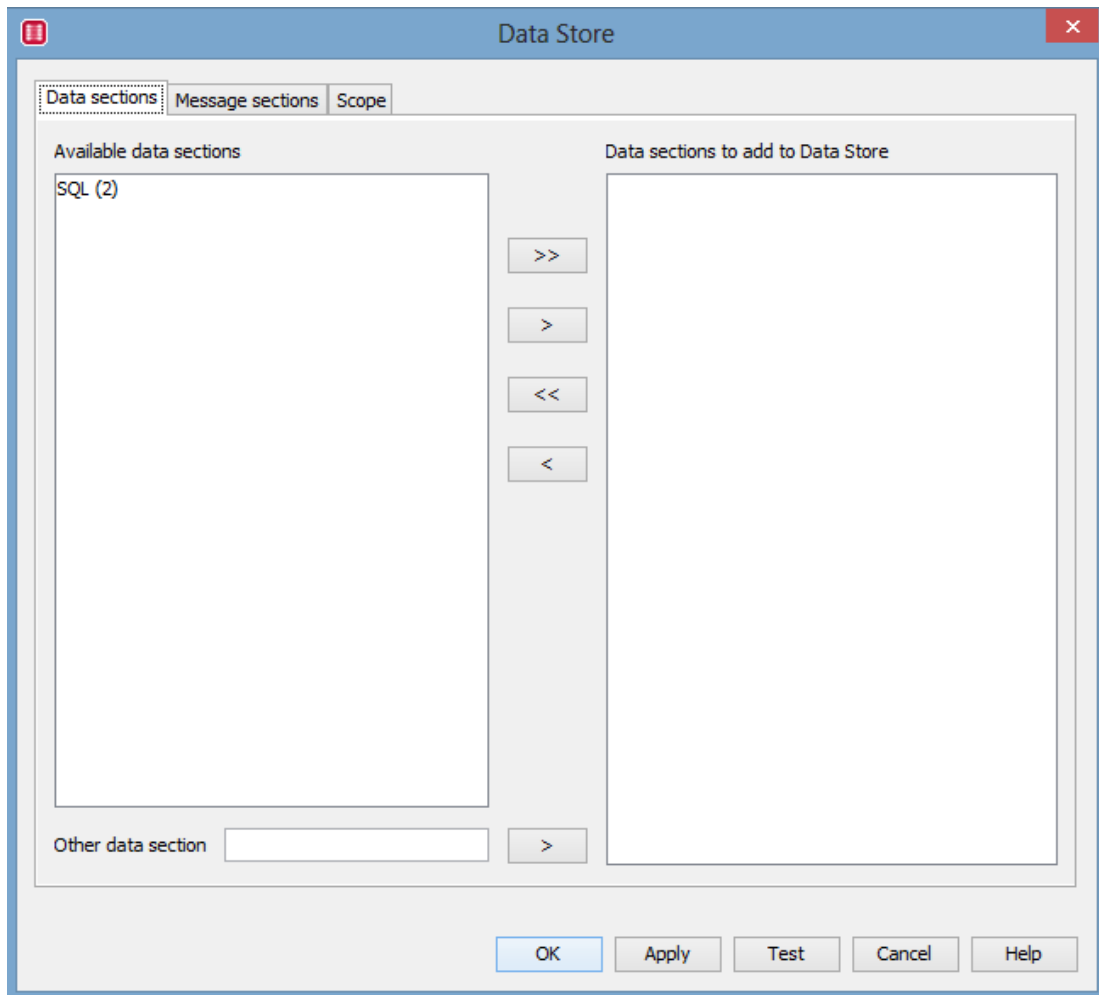
[Apteian Respond Service](#)

Data Store Module

The data store module is used to add or update items in the data store (data and message sections). For an overview of the data store, see [EMF Data store](#).

To use the Data store module:

1. Drag a **Data store** module icon into the EMF Process you are creating at the appropriate place.
2. Double-click the module icon to display the **Data Store** screen.



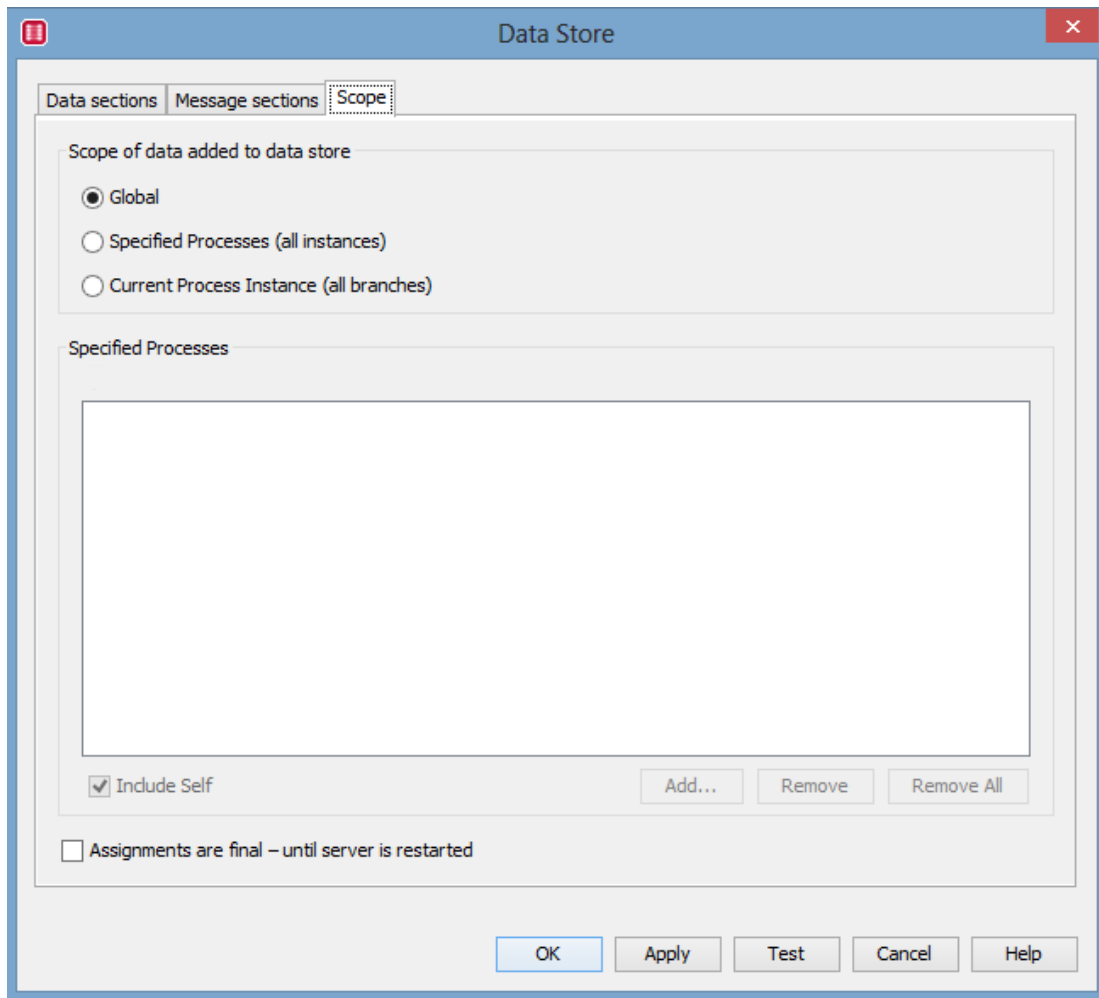
3. On the **Data sections** tab, select the data sections that you wish to add to the data store as follows:
 - a. Select the section name in the **Available data sections** list.
 - b. Click the **>** button to move it to the **Data sections to add to Data Store** list.

If the section name is not listed (for example, because it is created in a script module), then type the name in the **Other data section** field, and move it to the **Data sections to add to Data Store** list.

4. Select the **Message sections** that you wish to add to the data store in the same way as done for data sections.

The **Message sections** tab works in exactly the same way as the **Data Sections** tab – except it lists message sections instead of data sections.

5. Select the **Scope** tab, to specify the scope of the sections that should be added. For more information on scope see [Data Store Scope](#).



6. Selected the appropriate **Scope of data added to the data store**.
 - **Global** – The sections added to the data store will be available to all processes.
 - **Specified Processes (all instances)** - The data added to the data store will be available to all processes listed in **Specified Processes** list. If you wish to give the current process access to the sections being added, select **Include Self**. To give access to any other process, press the **Add** button and select the appropriate processes.
 - **Current Process Instance (all branches)** - The data added to the data store will be available to all branches of the current process instance. Each instance of a process will have their own set of data, and it is not shared between process instances.
7. Select **Assignments are final – until server is restarted** to inform the server that once a data or message section is assigned to the data store in a particular scope, its value will not change again. If its value is changed, then the changes will be ignored until the server is restarted.

The purpose of this option is to allow the server to optimise in a multi-machine environment to improve performance. Selecting this option is particularly useful when using the data store to hold "configuration information". If a "final" data store section is updated, then a warning is logged.

8. Clicking the **Test** button will cause different actions to happen depending on scope selected:
 - **Global** or **Specified Processes** scope – the data store is updated with the values that the data/message sections resolved to when the test button was pressed (that is, there is no "sample" value). Live data store values will be used when other modules are tested that reference data store sections.

Note: Updating the data store may or may not affect the values that the server is running with at that, as it depends on the server configuration and the optimisations being made. The process should be run through the server if it is important for the server to use the new values, or the server restarted.

- **Process instance** scope – the data store will store sample values for this process based on what the current data/message sections selected resolve to. These values will be different to the values that the server will use, as they cannot be calculated until the process instance runs.

Note: If Specified Processes (all instances) is selected, the following scenario occurs:

A message section called "Configuration" with a value of "red" is added to the data store that is in scope for processes A and B. A different message section, but also called "Configuration", but with a value of "yellow", is also added to the data store but in scope for processes A and C. This will overwrite the "Configuration" value of process A, and now process A and B will have different values in the data store for the message section "Configuration".

The latest version written to the data store for the specified process is always used.

Note: Due to the way that the caching works to improve performance, if a section is assigned to a specific process in the **Specified Processes** scope and run through the server, and then the section is assigned to a different process in the same data store module in the EMF UI – the first process may still access the original section as it may be cached. A server restart will clear out the cache.

Similar issues occur if a section is renamed/removed from a data store module when using a specified process or global scope. The original section name may still be cached by the server if it has previously been run, and therefore available to other processes. Again, restarting the server will clear the cache.

SAP System DS Data Browser Module

You can use a **SAP System DS Data Browser** module to connect to SAP (Systems Applications and Products in Data Processing) R/3 Data Sources and extract information from them in order to populate an EMF Process Data section.

Note:

- EMF requires SAP Java Connector 2 (SAP JCo 2) or SAP Java Connector 3 (SAP JCo 3) to connect to the SAP back-end.
 - The SAP System DS Data Browser is only supported on SAP ERP 6.0 (Technology Platform SAP NetWeaver 7.0) and higher.
-

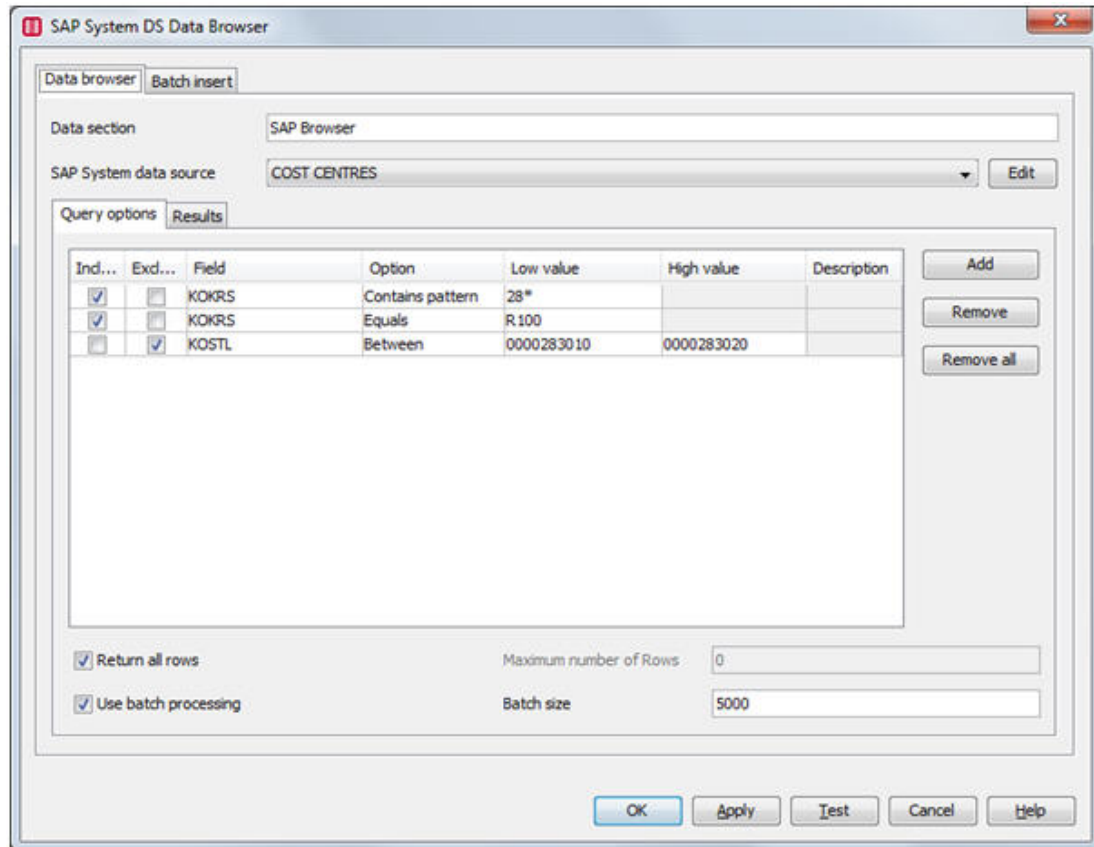
To use a SAP System DS Data Browser module:

1. Ensure that you have created a SAP System Data Source for the table data that you want to query.
2. Drag and drop a **SAP System DS Data Browser** module icon into an EMF process. Create a link to it from a module already in the EMF process instance.



3. Open the **SAP System DS Data Browser** module to display its properties. By default, double-click on the icon to open.

Tip: You can configure mouse-click actions on the **Options** window. That is, from the **Tools** menu, select **Options > EMF > Process Builder Behaviour** tab.



The **Data browser** tab is used to specify the required SAP System data source, Data section and the query criteria.

- **Data section:** Enter an appropriate name in the **Data section** field. This will be used to identify the information that is collected from the **SAP System DS Data Browser** for other modules further on in the EMF process.
- **SAP System data source:** Select the [SAP System data source](#) that you want to use from the SAP System data source drop-down list, which displays all of the SAP System data sources that you have set up in the SAP System data sources section of the EMF tree view.

When you have selected the required data source, the **Query** options tab will automatically display all the query fields available for the data source.

- Unused query fields can be removed if required.
- Multiple query fields can also be added.
- If more than one query field is specified, then a logical AND relationship is used.
- If the same query field is specified more than once, an OR relationship will be used for those field entries.

For example, in the diagram above, the query says: Include all records that have the KOKRS='R100' OR KOKRS contains values that begin with "28" AND Exclude those records that have a KOSTL value

between "0000283010" and "0000283020".

- **Edit:** This button allows you to edit the selected SAP System data source. The data source will be launched in a new dialog. Once the data source has been edited, clicking **OK** will return you back to the module and if there have been any changes to the data fields selected for Use in Query, they will be reflected in the **Query options** tab.

Note: The newly added fields will appear in the drop-down lists, and any query fields that have been removed from the data source will also be removed from the **Query options** tab.

Newly added fields will only be automatically added into the drop-down list of the **Field** column; if these newly added query fields are to be used in the **Query options**, then a new row(s) needs to be added and the relevant field selected from the **Field** drop-down list.

Additionally, the index columns in the **Batch insert** tab will be updated to reflect new/removed fields selected for Return.

- **Include:** Select this check box to include data matching the query criteria entered in the results.
- **Exclude:** Select this check box to exclude data matching the query criteria entered in the results.
- **Field:** Select the query field from the drop-down list. These are the fields that are selected for **Use in Query** in the SAP System Data Source.
- **Option:** Select from one of the pre-defined comparator options; Equals, Not Equal To, Contains Pattern, Between, Greater Than, and so on.
- **Low value:** Enter the value for the query field.
 - The **Low value** is used for all options that require only one value (Example: Equals, Greater than, and so on) and is also used as the lower value for ranges (Example: Between).
 - Dynamic functions (right-click on the **Low value** field to access the dynamic function menu) can be specified, if required. Leave this field blank to check for empty values for the field.
 - **Wildcards** (only valid for Contains pattern, Does not contain pattern option) can be specified; "*" - any value, "+" - any character.
Example: 28* means any value starting with 28, 2+00 means the second digit can be any value providing the first digit is a 2 and the other digits are 00.
- **High value:** Enter the value for the query field.
 - The **High value** is used for all options that require a high value for ranges (Example: Between).
 - Dynamic functions (right-click on the **High value** field to access the dynamic function menu) can be specified, if required.
 - **Wildcards** (only valid for Contains pattern, Does not contain pattern option) can be specified here; "*" - any value, "+" - any character.

Example: 28* means any value starting with 28, 2+00 means the second digit can be any value providing the first digit is a 2 and the other digits are 00.

- **Description:** The field description is displayed.
- **Return all rows:** If selected, then all data that matches the specified query will be returned. If **Return all rows** is not selected, then the **Maximum number of Rows** must be specified. This allows the maximum number of rows that should be returned to be defined.
- **Use batch processing:** Due to memory limitations, the data may need to be pulled back in batches. Select this option if large amounts of data is to be returned. The size of those batches is specified by the **Batch size entry** field.

[Batch Insert](#)

[Example: How to Set Up a Query in the SAP System DS Data Browser Module](#)

[Creating EMF SAP System Data Source](#)

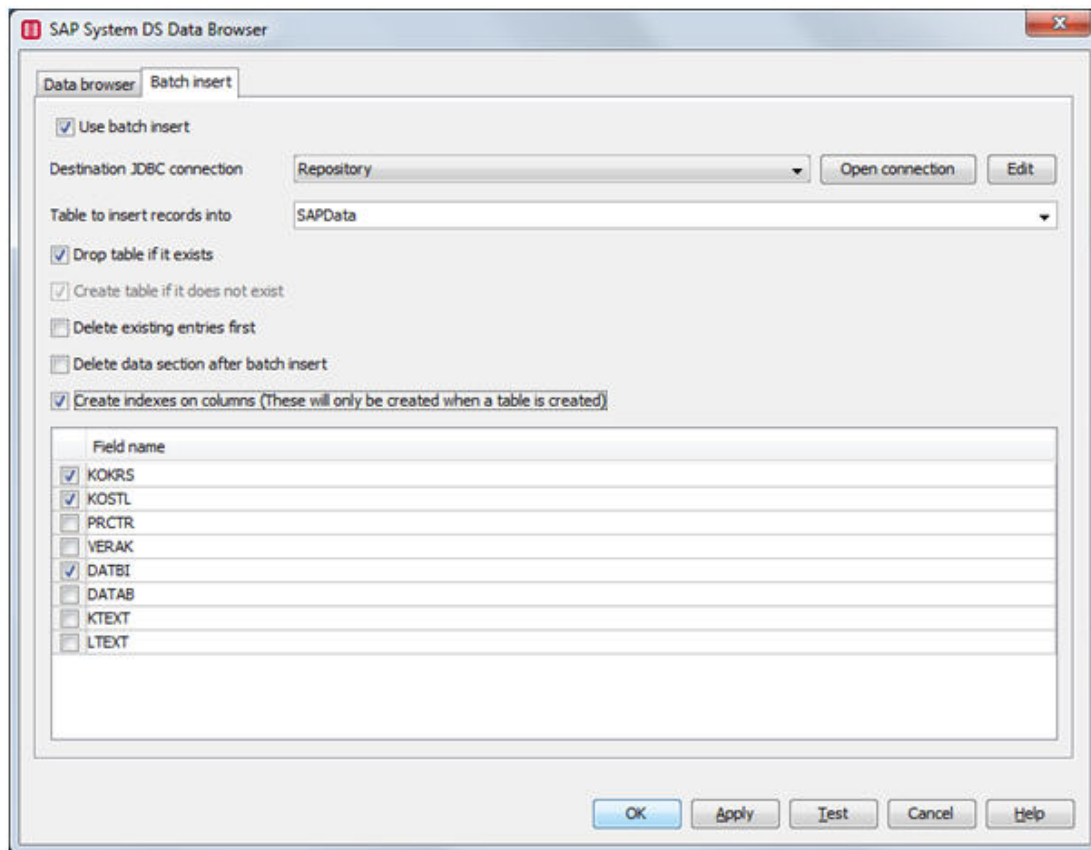
[Creating SAP System Data Sources](#)

[SAP System Data Source Examples](#)

[SAP Connection](#)

Batch Insert

Batch insert allows data that the module will return, to be inserted directly into a database table.



- **Use batch insert:** Select this option to insert the data returned by the module into a database table.
- **Destination JDBC connection:** Select a connection that identifies the database that the data will be stored into from the drop-down list. Once you select the connection, click the **Open Connection** button.
- **Table to insert records into:** Select the tables that will store the data returned by the query. This can either be selected from existing tables in the database, or the name of a table can be entered directly

Note: This table must be created before this module runs, else the module will fail if the **Create table if it does not exist** option is not selected. The table needs to have the same column names as the **Return** field columns that have been selected in the **Data fields** tab of the SAP System Data Source. The datatypes of the columns must also match or be compatible.

- **Edit:** This button allows you to edit the selected JDBC connection. The JDBC connection will be opened in a new dialog. Once the JDBC connection has been edited, click **OK** / **Cancel** to return to the **SAP System DS Data Browser** module **Batch insert** tab.
- **Drop table if it exists:** If selected, will drop the table first if it exists in the destination database, before creating the new table.

- **Create table if it does not exist:** If selected, will create a new table in the destination database. The value in the **Table to insert records into** field will be the name of the table.
- **Delete existing entries first:** If selected, this will empty the destination table of any existing data before the results of the query are stored.
- **Delete data section after batch insert:** If selected, will delete the whole data section once the batch insert of data is complete.
- **Create indexes on columns:** If selected, will create database indexes on the fields listed. The field names listed are the ones that are included in the data output (obtained from the **Return** field column in the SAP System Data Source **Data fields** tab). Select the fields to include in the Index.

Note: A database index is a data structure that increases the speed of data retrieval from a database. Creating a database index may result in slower writes and increased storage space.

- **Test:** This button can be used to test that the output is generated as expected. The **Test** button will only return the first batch of data if the **Use Batch Processing** option is selected.

[Example: How to Set Up a Query in the SAP System DS Data Browser Module](#)

[Creating EMF SAP System Data Source](#)

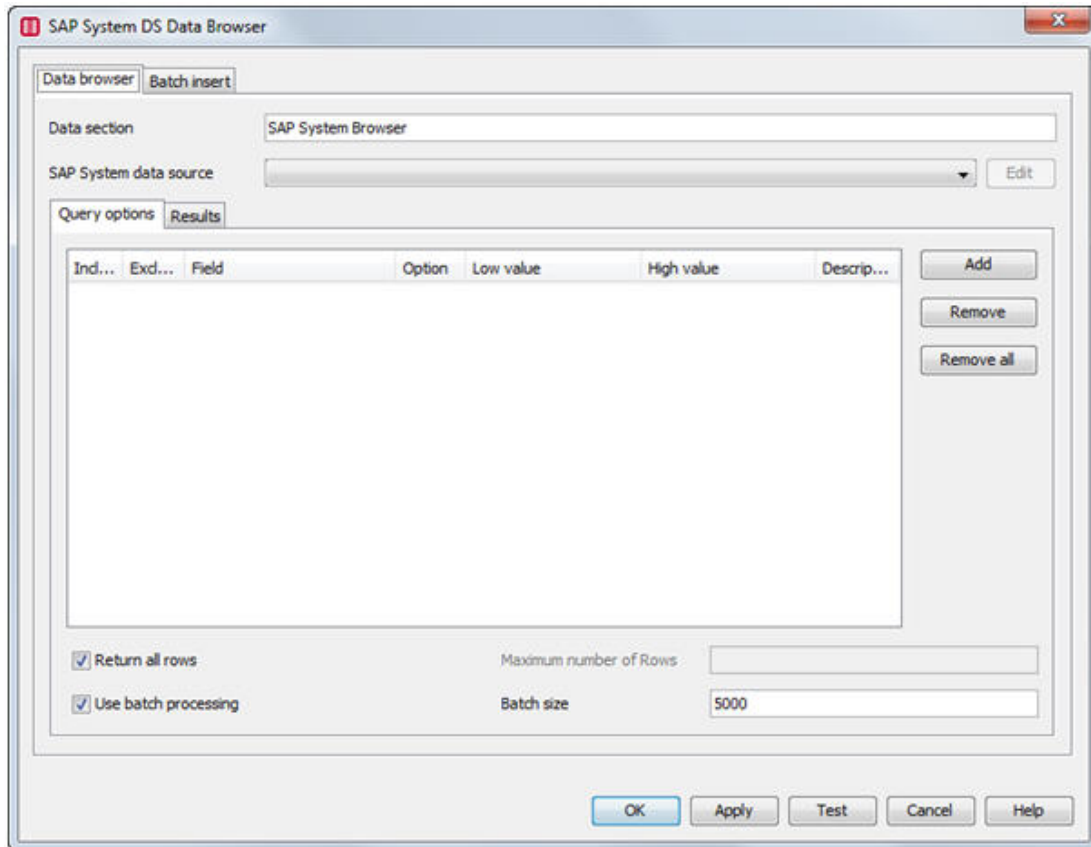
[Creating SAP System Data Sources](#)

[SAP System Data Source Examples](#)

[SAP Connection](#)

Example: How to Set Up a Query in the SAP System DS Data Browser Module

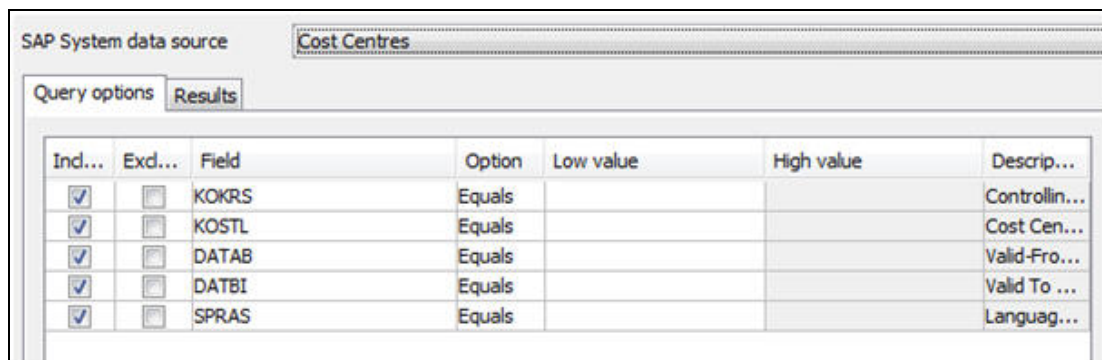
1. Open the **SAP System DS Data Browser** as described in the Help. The following dialog is displayed:



2. Select the **SAP System data source** from the list of available data sources.

Note: You must have a **SAP Connection**. If it has not been set up, click **Cancel** to exit from this dialog and create a [SAP Connection](#) and ensure that you can connect to a SAP System.

In this example, the **SAP System data source** set up in the [SAP System Data Source](#) example will be used. If the set up is valid, the following dialog will be displayed when the "Cost Centres" SAP System data source is selected:



3. In this example, the **DATAB**, **DATBI**, **SPRAS** and **KOKRS** fields are not required. Select each of these to remove the unwanted query fields and click the **Remove** button to remove them one by one.

4. Once these fields are removed, the **Query options** tab will display as illustrated in the following image:

Ind...	Excl...	Field	Option	Low value	High value	Descrip...
<input checked="" type="checkbox"/>	<input type="checkbox"/>	KOSTL	Equals	0000001210		

5. In the **Low value** field, type a valid value to retrieve matching data.
6. Clear the **Return all rows** check box, so that the test in the UI brings back a subset of data.
7. Click **Test**. If there are any data, it will be displayed in the **Results** tab.

KOKRS	KOSTL	BUKRS	KTEXT	DATAB	DATBI	VERAK	LTEXT
1000	0000001210	1000	Telephone	1994-01-01	9999-12-31	Nisch	Telephone
2000	0000001210	3000	Telephone	1994-01-01	9999-12-31	Barton	Telephone
4700	0000001210	4700	Telephone	2004-01-01	9999-12-31	Nisch	Telephone
1000	0000001210	1000	Telephone	1994-01-01	9999-12-31	Nisch	Telephone
2000	0000001210	3000	Telephone	1994-01-01	9999-12-31	Barton	Telephone
4700	0000001210	4700	Telephone	2004-01-01	9999-12-31	Nisch	Telephone
1000	0000001210	1000	Telephone	1994-01-01	9999-12-31	Nisch	Telephone
2000	0000001210	3000	Telephone	1994-01-01	9999-12-31	Barton	Telephone
4700	0000001210	4700	Telephone	2004-01-01	9999-12-31	Nisch	Telephone
1000	0000001210	1000	Telefon	1994-01-01	9999-12-31	Nisch	Telefon
2000	0000001210	3000	Telefon	1994-01-01	9999-12-31	Barton	Telefon
1000	0000001210	1000	Telephone	1994-01-01	9999-12-31	Nisch	Telephone
2000	0000001210	3000	Telephone	1994-01-01	9999-12-31	Barton	Telephone
4700	0000001210	4700	Telephone	2004-01-01	9999-12-31	Nisch	Telephone
1000	0000001210	1000	Téléphone	1994-01-01	9999-12-31	Nisch	Téléphone
2000	0000001210	3000	Téléphone	1994-01-01	9999-12-31	Barton	Téléphone

Number of Rows: 33

8. To further limit the results, in the **Query options** tab, add in the **SPRAS** field.

Ind...	Excl...	Field	Option	Low value	High value	Descrip...
<input checked="" type="checkbox"/>	<input type="checkbox"/>	KOSTL	Equals	0000001210		
<input checked="" type="checkbox"/>	<input type="checkbox"/>	SPRAS	Equals	4		

9. Click **Test** and notice that there are only a few records displayed.

Query options		Results					
KOKRS	KOSTL	BUKRS	KTEXT	DATAB	DATBI	VERAK	LTEXT
1000	0000001210	1000	Telephone	1994-01-01	9999-12-31	Nisch	Telephone
2000	0000001210	3000	Telephone	1994-01-01	9999-12-31	Barton	Telephone
4700	0000001210	4700	Telephone	2004-01-01	9999-12-31	Nisch	Telephone

10. Click **Edit** to update the selected SAP System Data Source.
11. From the **Data fields** tab, remove the **VERAK** field using the **Select Fields** button. The following image illustrates the **Data** fields:

Data Source		Tables to join		Data fields		Fields for join	
Return field	Use in Query	Key	Table	Field	Description		
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKS	KOKRS	Controlling Area		
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKS	KOSTL	Cost Center		
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKS	BUKRS	Company Code		
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKT	KTEXT	General Name		
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	CSKS	DATAB	Valid-From Date		
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKS	DATBI	Valid To Date		
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKT	LTEXT	Description		
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKT	SPRAS	Language Key		

12. Click **Update** to update the SAP System Data Source in the SAP System. Click **OK** to return back to the **SAP System DS Data Browser**. Click the **Test** button to view the results in the **Results** tab.

Query options		Results					
KOKRS	KOSTL	BUKRS	KTEXT	DATAB	DATBI	LTEXT	
1000	0000001210	1000	Telephone	1994-01-01	9999-12-31	Telephone	
2000	0000001210	3000	Telephone	1994-01-01	9999-12-31	Telephone	
4700	0000001210	4700	Telephone	2004-01-01	9999-12-31	Telephone	

As a result of this update, the **VERAK** column is no longer in the output.

13. Change the **Low value** for the **KOSTL** field to a valid matching pattern and the **Option** to "Contains pattern".

Query options		Results				
Ind...	Excl...	Field	Option	Low value	High value	Descrip...
<input checked="" type="checkbox"/>	<input type="checkbox"/>	KOSTL	Contains pattern	*4+0		
<input checked="" type="checkbox"/>	<input type="checkbox"/>	SPRAS	Equals	4		

14. Click the **Test** button and view the results in the **Results** tab.

Query options		Results				
KOKRS	KOSTL	BUKRS	KTEXT	DATAB	DATBI	LTEXT
1000	0000002410	1000	RE Vacant apart...	1950-01-01	9999-12-31	RE Vacant apartment
1000	0000002420	1000	RE Vacant office	1950-01-01	9999-12-31	RE Vacant office
1000	0000002430	1000	RE Vacant, others	1950-01-01	9999-12-31	RE Vacant, other business obj...
1000	0000004400	1000	Quality Assurance	1994-01-01	9999-12-31	Quality Assurance
1000	H96-1400	1000	Painting works	2003-01-01	9999-12-31	Painting works
2000	0000004400	3000	Quality Assurance	1994-01-01	2001-12-31	Quality Assurance
2000	0000004400	3000	Quality Assurance	2002-01-01	9999-12-31	Quality Assurance
2000	0000004400	3000	Quality Assurance	1994-01-01	2001-12-31	Quality Assurance
2000	0000004400	3000	Quality Assurance	2002-01-01	9999-12-31	Quality Assurance
2000	0000004410	3000	QA check vendor	2001-01-01	2001-12-31	QA check vendor
2000	0000004410	3000	QA check vendor	2002-01-01	9999-12-31	QA check vendor
2000	0000004410	3000	QA check vendor	2001-01-01	2001-12-31	QA check vendor
2000	0000004410	3000	QA check vendor	2002-01-01	9999-12-31	QA check vendor
2000	0000006400	3000	Outsourcer	2002-01-01	9999-12-31	Outsourcer
2000	0000093400	3000	Plant Denver Admin	2003-01-01	9999-12-31	Plant Denver Admin
2000	0000093410	3000	Dev.Smart Homeb...	2003-01-01	9999-12-31	Dev.Smart Homebanking

Number of Rows: 39

Copy to clipboard

Notice that the KOSTL values are the ones that match the pattern. In the illustration shown above,

- it matches anything before the last 3 characters
- the third last character is a 4
- the penultimate character is any character and the last character is 0.

[Batch Insert](#)

[Creating EMF SAP System Data Source](#)

[Creating SAP System Data Sources](#)

[SAP System Data Source Examples](#)

[SAP Connection](#)

Generic Trap Types

If you select the Trap v1 request type in the **Properties** tab of the SNMP module screen, you need to select the trap type in the **Specific properties** tab.

Following are descriptions of each trap type:

Trap Type	Description
Enterprise specific	The trap is enterprise-specific. The trap type is specified in the Specific trap code textbox.
Cold start	The agent has rebooted. All management variables will be reset, for example, Counters and Gauges. For more information on the Counter and Gauge object types, see Object Data Types .
Warm start	The agent has reinitialized itself. No variables will be reset.
Link down	A network interface on the managed device has gone down.
Link up	A network interface on the managed device has come back up.
Authentication failure	The agent has been queried with an incorrect community string.
egp Neighbor loss	An Exterior Gateway Protocol (EGP) neighbour has gone down. EGP is a protocol used by neighboring gateway hosts to exchange routing and other information.

[Specific Properties tab](#)

[SNMP Module](#)

[Back to Start of Data Modules](#)

How to import a self-signed certificate into EMF

1. In Internet Explorer, access a HTTP Secure page (https). Right-click on the page and select **Properties**.
2. Click **Certificates** and select the **Details** tab.
3. Click **Copy to File....** to launch the **Certificate Export Wizard**.
4. Click **Next** and select **DER encoded binary X.509 (.CER)** (should be the default option). Click **Next** and enter a filename (for example, c:\MyCertificate.cer).
5. Click **Next** and then **Finish** to complete the export process.
6. From the command prompt, enter "C:\Program Files\Java\jre6\bin\keytool.exe" -v -importcert -file c:\MyCertificate.cer -keystore MyKeyStore.jks

7. Enter a new password (at least 6 digits). Enter the same password again and press Enter.
8. Enter **Yes** next to the **Trust this certificate? [no]** field and press Enter. This generates a file called MyKeyStore.jks.
9. In EMF, select **System->Net Security Key Stores**. Right-click and select **New** to create a new Key Store.
10. Enter a name (for example, My Key Store). Select **JKS** from the File type drop-down list and **SunX509** from the Certificate type drop-down list. Browse and select MyKeyStore.jks that you created. Enter the password you created in **Step 7**. Leave the Private key password field blank.
11. Go to the HTTP service you set up, select the **Internet Security** tab. Click **SSL Settings** and select **Client Authentication Required**. Select the key store you created from the drop-down list. This is now called a Web service.

Object Data Types

For each object listed in the [Object list tab](#) of the **SNMP module** screen, you need to select an object type. Following are descriptions of each of the eight object types supported in EMF.

Object Type	Description
String	A string of zero or more octets (bytes).
Integer	A 32-bit number.
Counter	A 32-bit number with a minimum value of zero and a maximum value of 2 to the power of 32, minus 1. When the maximum is reached, it wraps around to zero. A counter object should never decrease, although it can be reset to zero.
Gauge	A 32-bit number with a minimum value of zero and a maximum value of 2 to the power of 32, minus 1. Unlike the Counter type, a gauge value can increase or decrease, and it never wraps around to zero.
IPAddress	A 32-bit number, representing an IP address.
ObjectID	An object identifier of a managed object, written in decimal dot notation.
Opaque	A string of zero or more octets (bytes), containing any ASN.1 format data.
Timeticks	A 32-bit number with a minimum value of zero and a maximum value of 2 to the power of 32, minus 1. This type is used to measure time in hundredths of a second (from a predetermined start time). For example, you would use this type to measure the uptime of a device.

[Object List tab](#)

[SNMP Module](#)[Back to Start of Data Modules](#)

Object List Tab

You can use the **Object list** tab of the SNMP module to enter the details of the object(s) that should be referenced by the request type selected in the **Properties** tab.

There should be at least one object in the object list, and the **Object ID** and **Type** are mandatory for all objects.

For each object you want to add, fill in the details as follows:

- **Object ID:** (mandatory unless Trap request type) Type the object identifier for the object in the **Object ID** field. This is the dotted notation number that uniquely identifies the object in the MIB tree.
- **Description:** (optional) Type a description for the object that will allow users to easily identify it
- **Type:** (optional) Select a type for the object. For more information on the available types, see [Object Data Types](#).
- **Value:** (optional unless Set request type) Enter the value to associate with the object. The value should be an allowed value for the object type selected from the **Type** drop-down list. For example, if you selected **IPAddress** as the object type, you should not enter an alphanumeric text string as the value.

[Back to Start of Data Modules](#)

SAP Authorizations Domains Tab

The **Domains** tab allows you to run the same authorization checks multiple times by varying one or more of the criteria specified in the **Authorizations** grid in the **Properties** tab. For example, the main criteria might contain authorization checks for one company and domains (or variants as they are sometimes known as) have been defined for different companies with slightly different criteria (for example, company code may be different). Selecting the domain to run against will use the same checks as the main domain, substituting some of the authorization values to look for with the ones from the selected domain(s).

SAP Authorizations

Properties Advanced options Domains

☒ Use static domains

Select the domains to run against

Name	Description
<input type="checkbox"/> 01	Activity 01
<input type="checkbox"/> 01(1)	Activity 01
<input type="checkbox"/> 01(2)	Activity 01
<input type="checkbox"/> S-DEVELOP - ACTIVITY 02	activity 02
<input type="checkbox"/> 01(3)	Activity 01
<input type="checkbox"/> S-DEVELOP - ACTIVITY 02(1)	activity 02
<input checked="" type="checkbox"/> SOD_MM-IM	Inventory count for activity 02

☐ Use dynamic domains

Data section containing domains:

Column index (one-based):

OK Apply Cancel Help

If a domain (static or dynamic) has been selected, then the Authorization module will only run the selected/specified domain.

- Use static domains

This allows you to run against the selected domains. The grid shows the available domains (name and description of domain). To select a domain, select the check box next to the domain you wish to run against.

- Use dynamic domains

This allows you to run against domains (which will already have been defined in the EMF system and will be visible in the static list of domains) that are selected at run time. A data section and a column index (one-based) must be specified in which to look for the domain names. Dynamic Functions can be used to specify the data section.

For information on managing domains, refer [Manage Domains](#).

SAP Authorizations Advanced options Tab

The **Advanced options** tab is used to specify additional options that can be used to filter some of the data that will be brought back.

The screenshot shows the 'SAP Authorizations' dialog box with the 'Advanced options' tab selected. The dialog has three tabs: 'Properties', 'Advanced options', and 'Domains'. The 'Advanced options' tab contains the following fields and options:

- Short description:** A text input field.
- Long description:** A large text area.
- Use UST tables:** A checked checkbox.
- Show missing authorizations:** A checked checkbox.
- Ignore blank user groups:** An unchecked checkbox.
- Wildcard (*) processing:** A group box containing two options:
 - ☐ Match any value against authorization object values
 - ☒ Match against any user profile authorization object values
- Part wildcard (*) processing (e.g. CE*):** A group box containing two options:
 - ☐ Match any value against authorization object values
 - ☒ Match against any user profile authorization object values
- Use local cache for table storage:** An unchecked checkbox.
- Cache storage interval (minutes):** A text input field with the value '0'.
- Show authorizations for users:** An unchecked checkbox.
- Profiles to include (only valid for non-wildcard matches):** A text input field.

At the bottom right of the dialog are four buttons: 'OK', 'Apply', 'Cancel', and 'Help'.

- **Short description** - Use this field for any short descriptions that may need to be shown in the output.
- **Long Description** - Use this field for describing in detail, the authorization checks that are being performed within the SAP Authorizations module.
- **Use UST Tables** - This allows you to run the authorizations check using the UST tables from the SAP system, as this allows for faster processing. Selecting this option relies on the UST tables being up to date and that they have been synchronized with the USR

tables (see your SAP Administrator regarding synchronizing the UST tables). This option is always selected when running against a local database. If this option is not selected, USR tables will be used to check for authorizations. Using the USR tables will take a long time to run the authorization checks and may impact on system performance.

If the USR tables are used, the standard RFC_READ_TABLE needs to be changed to allow it to bring back data of up to 4000 characters (instead of the standard 512 characters). After changing the WA field (to allow 4000 characters) in the user copy of RFC_READ_TABLE (for example, Z_EMF_RFC_READ_TABLE), the following needs to be added/changed to the config.xml file

```
<auth>
<rfcreadtable>Z_EMF_RFC_READ_TABLE</rfcreadtable>
</auth>
```

- **Show missing authorizations** - When running the authorization checks in the SAP system, the logged-in user may not have rights to check for details of a particular user in the SAP system and therefore not all user profile results will be returned. Selecting this option will not check if the logged-in user has authorization to check the details of the user in the results, therefore displaying all user profiles for the check being made. This option is selected by default when running against a local database.
- **Ignore blank user groups** - Some SAP systems do not use the Class/Group field of the User. This means that the group field is left blank in the SAP database. If you want to include those users who do not have any user group specified for them, then this option should be selected. This option is always selected when running against a local database, as a user authority check cannot be performed on local databases.
- **Use local cache for table storage** - Selecting this allows you to hold some common data in memory for the period defined (Cache storage interval), when running the authorization checks for the alert in the EMF Server. It, therefore, reduces the amount of time it takes the checks to run. This is connection-based and other SAP Authorization modules using the same JDBC/SAP connection will reuse this cache. Use with caution, as it can consume a high amount of memory.
- **Show authorizations for users** - Selecting this option will show the authorization against the profile for the user, in the output. This can be useful for cross-checking of authorizations for users.
- **Profiles to include** - This option is only available if wildcards have not been selected. Entering profile names in the box (separated by commas, e.g. SAP_NEW, SAP_ALL) will include those authorizations that belong to the entered profiles. These authorizations will be in addition to those already obtained using the selection criteria in the main properties tab.
- **Wildcard (*) processing** - This allows you to specify how to treat wildcards (*) in the user input and the user authorizations. There are 2 options available:
 - Match any value against authorization object values.

Authorization object	Field	Value
F_LFA1_BUK	BUKRS	*

Selecting this option will allow the SAP Authorization module to include all authorizations for object L_LFA1_BUK and all values for field BUKRS. If you leave

the check box un-selected, the system will only look for authorizations that exactly match object L_LFA1_BUK and field BUKRS with a value of *.

- Match against any user profile authorization object values

If the following value is entered

Authorization object	Field	Value
F_LFA1_BUK	BUKRS	032

Selecting this option will allow the SAP Authorization module to include all authorizations for object L_LFA1_BUK and field BUKRS with a value of * or a value of 032. If you leave the check box un-selected, the system will only look for exact matches for the specified value (032 in the above example).

- **Part wildcard (*) processing** - This allows you to specify how to treat part wildcards (e.g. CE*) in the user input and the user authorizations. There are 2 options available:

- Match any value against authorization object values

Authorization object	Field	Value
S_USER_GRP	CLASS	MM*

Selecting this option will allow the SAP Authorization module to include all authorizations for object S_USER_GRP and field CLASS, where the value in the authorizations starts with MM. If you leave the check box un-selected, the system will only look for authorizations that exactly match object S_USER_GRP and field CLASS with a value of MM*.

- Match against any user profile authorization object values

If the following value is entered

Authorization object	Field	Value
S_USER_GRP	CLASS	MM:0030

Selecting this option will allow the SAP Authorization module to include all authorizations for object S_USER_GRP and field CLASS that contain MM*, M*, MM:003*, etc. Leaving the check box unselected will only look for exact matches for the specified value (MM:0030 in the above example).

- **Wildcard character for "all" values** - This option allows an authorization with a wildcard value specified in the **Properties** tab, to be used to bring back all values for that particular authorization object/field combination. For example, if the following is entered:

Authorization object	Field	Value
F_LFA1_BUK	ACTVT	02
F_LFA1_BUK	BUKRS	**

and the wildcard character is set to **, all values in the SAP system for the F_LFA1_BUK/BUKRS will be used for the authorization check, regardless of what the other wildcard settings are. This behavior for this row will be the same as the Match any value against authorization object values wildcard setting. The other wildcard settings will still be used for the authorization checks on other rows.

SAP Data Browser Module Properties/Batch Insert tab

The SAP Data Browser Module's **Properties/Batch Insert** tab allows data that the module will return to be inserted directly into a database table.

SAP Data Browser

Properties / Batch insert

Halt conditions

☒ Never

☐ On no data returned

☐ On data returned

Batch insert

☒ Use batch insert

Destination JDBC connection: <<JDBC Batch insert connection>> Open connection

Table to insert records into: AGR_1016

☒ Drop table if it exists

☒ Create table if it does not exist

☒ Delete existing entries first

☒ Delete data section after batch insert

☒ Create indexes on columns (These will only be created when a table is created)

Field name	
MANDT	<input checked="" type="checkbox"/>
AGR_NAME	<input checked="" type="checkbox"/>
PROFILE	<input type="checkbox"/>
PSTATE	<input type="checkbox"/>

OK Apply Test Cancel Help

This allows a very efficient means of copying SAP data to another database. For analysis of data this is often required as it reduces the load on the SAP system, makes processing faster and gives access to the power of SQL to be used when building queries.

Note: No data section is created and no data is returned from this module if the **"Use batch insert"** and **"Delete data section after batch insert"** options are selected. In that case, the data is inserted directly into the database table. If the data is then needed to be processed immediately after this module, it will need to be queried from the table it was just inserted into.

Halt Conditions

You can specify the Halt Conditions for the SAP Data Browser. The options available are:

- Never
- On no data returned
- On data returned

Batch Insert

1. Select the **Use batch insert** option to insert the data returned by the module into a database table.
2. Select the **Destination JDBC connection** that identifies the database that the data will be stored into.
3. Click the **Open Connection** button.
4. Click **Edit** button to modify the selected JDBC connection.
5. Select the **Table to insert records into** that will store the data returned by the query. This can either be selected from existing tables in the database, or the name of a table can be entered directly (note that this table must be created before this module runs though, otherwise the module will fail). The table needs to have the same column names as the columns that have been selected in the **Table Metadata** tab. The datatypes of the columns must also match or be compatible.
6. **Create table if it does not exist** - if selected will create a new table in the destination database. The value in the **Table to insert records into** field will be the name of the table.
7. **Delete existing entries first** - if selected this will empty the destination table of any existing data before the results of the query are stored.
8. **Delete data section after batch insert** - if selected will delete the whole data section once the batch insert of data is complete.
9. **Drop table if it exists** - if selected, will drop the table first, if it exists, in the destination database, before creating the new table.
10. **Create indexes on columns** - if selected, will create database indexes on the fields listed. The field names have to be added using the **Add** button.

Note: A database index is a data structure that increases the speed of data retrieval from a database. Creating a database index might result in slower writes and increased storage space.

[SAP Data Browser Module](#)

[Data Sources \(Configuring a SAP Profile\)](#)

[Go to start of Data Modules](#)

Overview of SNMP

The following topics are covered in this overview:

- [Introduction](#)
- [Management Information Base \(MIB\)](#)
- [Message Formats](#)
- [Security](#)
- [SNMP Standards](#)

Introduction

The Simple Network Management Protocol (SNMP) is a protocol used for remote management of network devices. It is part of the TCP/IP protocol suite. There are currently three main versions of SNMP: SNMPv1, SNMPv2 (several varieties) and SNMPv3.

Note: EMF supports all three versions of SNMP (SNMPv1, SNMPv2c and SNMPv3). However if you want to use SNMP v3, you will need to have version 1.2.1 of the Java Cryptography Extension (JCE) installed.

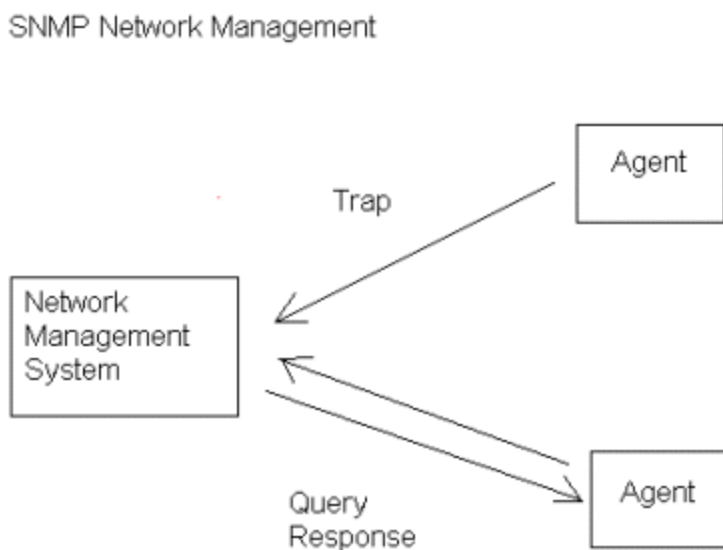
SNMP defines a client/server relationship. In an SNMP-managed network, one or more network managers - the clients - use SNMP to communicate with SNMP agents - the servers - on network devices. Network devices can include printers, routers, hosts and bridges.

The network manager is usually incorporated in a network management system.

There are two types of communication across an SNMP-managed network:

- Query/Response - the network manager sends a query for information to an agent and receives a response.
- Traps - the agent asynchronously sends status information to a network manager.

Diagram of SNMP communication



Management Information Base (MIB)

Each SNMP agent has a MIB. A MIB is a database that defines the network information that can be monitored on the network device. MIB information is stored in a text file, written in an adapted subset of ASN.1 notation.

The structure of the MIB, the allowable data types and the data representation are defined by the SNMP protocol in the Structure of Management Information (SMI).

The information in a MIB is arranged hierarchically in a treelike structure (see diagram below).

There are three types of elements in a MIB tree:

- Objects
- Tables
- Groups

Nodes without children, also known as leaf nodes, are objects. In the diagram below, **sysDescr** is an example of an object. Each object is a variable that can take a value or list of values. Each object has an associated data type. For example, an object could be of data type `IpAddress`, which is defined as a string of four bytes. For more information on data types, see [Object Data Types](#).

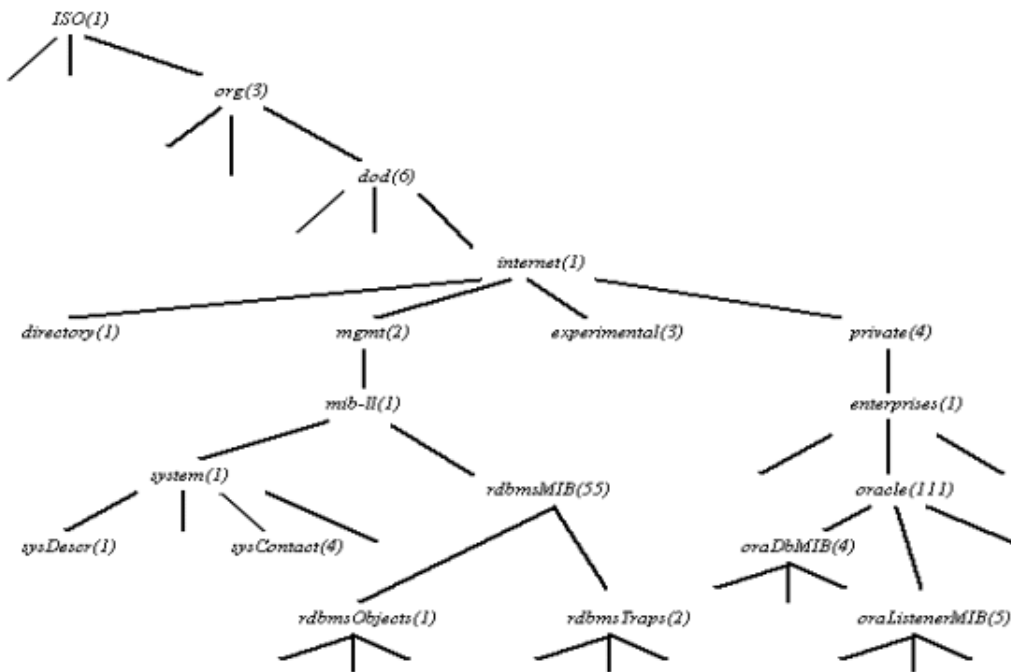
Some objects are part of a "table". Tables are a convenient way to organize related objects that can have more than one value at a time (that is, a list of values). The columns in the table are defined by the objects in the table, while the number of rows can vary. The rows in the table are referenced using an index column (object). A table is defined in a similar way to an object and has the data type `Sequence Of`.

Groups only have a structural function within the MIB and do not have information that can be queried by SNMP network managers.

Objects, tables and groups can be identified by a text or numeric identifier written in dot notation. Both of these describe the path taken through the tree to get to the object or group. For example, the numeric identifier for **sysDescr** (see diagram below) is **1.3.6.1.2.1.1.1**. The text identifier is **iso.org.dod.internet.mgmt.mib-II.system.sysDescr**.

The numeric identifier is called the object identifier.

Diagram of part of the MIB structure



All nodes under the **mgmt** branch are considered standard and are defined by the Internet Engineering Task Force (IETF). However, the SNMP protocol is extensible and the **private** branch allows the addition of proprietary object definitions. For example, if you add a network device that supports SNMP to your network, the vendor should provide the required definition information for you to add the objects to the MIB.

Message Formats

SNMPv1 defines four types of messages that can be sent between a network manager and an agent: Get, Get Next, Set and Trap. SNMPv2 defines an additional two message types: Get Bulk and Inform. SNMPv3 does not define any additional message types.

The following table describes the message types.

Message Type	Description
Get	Returns the value of a named object
Get Next	Returns the name and value of the "next" object in the MIB tree. This can be used to traverse the entire tree to determine all the objects which a network device supports.
Get Bulk	Returns large blocks of data, for example, multiple rows in a table
Set	Sets the value of a named object

Trap	Asynchronous notification generated by a network device and sent to a network manager
Inform	Allows one network manager to send trap information to another network manager

Security

SNMPv1 and SNMPv2c

SNMPv1 and SNMPv2c have a primitive community-based security model. The model specifies weak authentication, and does not provide for message integrity or privacy.

Authentication is performed using "community strings", which are passwords that control access to node information. If an SNMP message from a network management system in the community access list includes the correct community string in the message header, then the message is considered authentic by the receiving network agent.

The community string is not encrypted, so anyone monitoring the network packets would be able to discover it.

SNMPv3

SNMPv3 provides considerably better security than SNMPv1 and SNMPv2c. It allows for strong authentication, message integrity verification and privacy.

The network manager and the agent both share a secret authentication key. The key is stored locally together with the "username" of the manager or agent. Unlike SNMPv1 or SNMPv2c, the key is never sent in a message. Instead, the message includes a fingerprint, derived from the key and the message using a hashing algorithm.

The username is also sent with the message for authentication purposes.

A hashing algorithm takes a message and produces a digest version of the message, called a fingerprint. What makes this important for security is that it is computationally infeasible to produce the same fingerprint from two different messages. In addition, it is practically impossible to reverse the process - the original message cannot be reconstructed from the fingerprint.

SNMP supports two hashing algorithms:

- HMAC-MD5-95, based on MD5 (128-bit key)
- HMAC-SHA-96, based SHA-1 (160-bit key)

MD5 is faster, while SHA-1 is stronger. The SNMPv3 protocol specifies that a network device must support MD5. Additional support for SHA-1 is optional.

When an SNMP network element (either a manager or agent) receives a message, it uses the locally stored authentication key associated with the sender's username to generate a fingerprint from the message. If this fingerprint matches the fingerprint included with the message then the message has been authenticated and the message integrity has been verified.

For privacy, 56-bit DES encryption can be used to encrypt messages. Encryption is not mandatory. Similarly to authentication, a privacy key (generated from a privacy password) is used to encrypt and decrypt the message.

Security functions are performed by each SNMP element's SNMP engine.

SNMPv3 has three levels of security. The following table gives a description of each level.

Security Level	Description
NoAuthNoPriv	No authentication or privacy. This is equivalent to the security offered by SNMPv1 and SNMPv2c. Instead of a community string, it uses a username.
AuthNoPriv	Authentication, but no privacy.
AuthPriv	Authentication and privacy. You will need to installed a JCE1.2.1 compliant provider is you wish to use this security level. See http://java.sun.com/products/jce for more information.

Because of the overhead associated with providing increased security, if security is not required you should choose either SNMPv1 or SNMPv2c.

SNMP Standards

The following are SNMP-related RFCs (standard specifications from the IETF):

- SNMPv1: 1155, 1157, 1213
- SNMPv2c: 1901, 1905, 1906
- SNMPv3: 2273-75

You can access these RFCs at the IETF web site <http://www.ietf.org/rfc.html>.

[SNMP Module](#)

[SNMP In Module](#)

[SNMP Service](#)

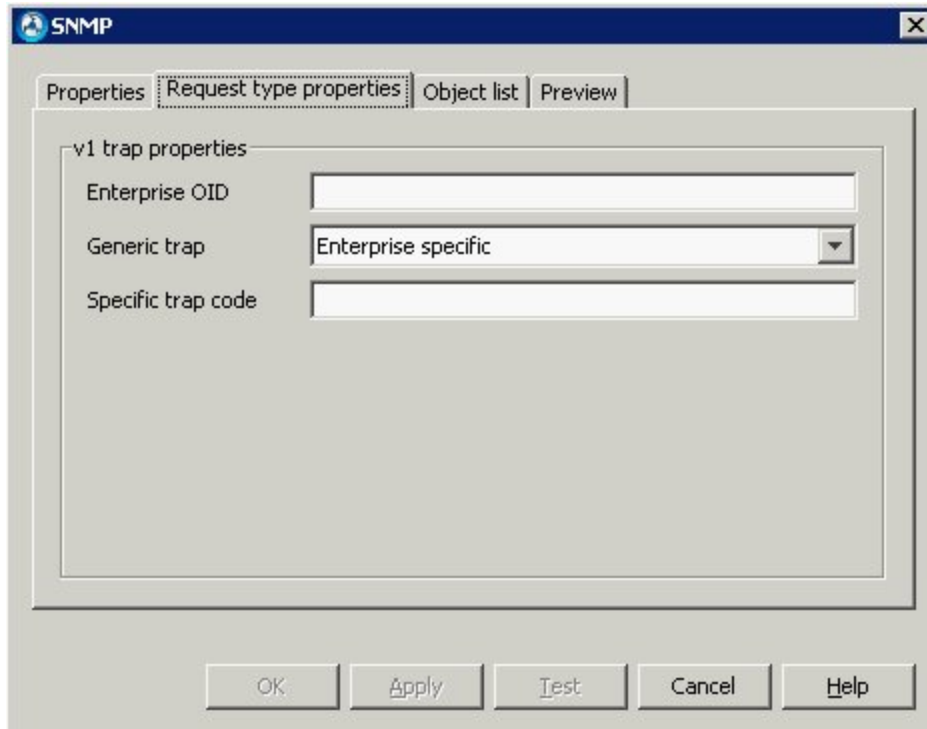
[Back to Start of Data Modules](#)

Request Type Properties tab

You can use the **Request type properties** tab to set additional properties (if any) for the request type selected in the SNMP module's Properties tab.

The options available in this tab will differ depending on which request type you selected. There are no specific properties to set for the Inform v2c, Set v1 and Set v2c request types.

Example of the Request type properties tab (Trap v1 options shown)



Trap v1 Specific Properties

- **Enterprise OID:** Type the enterprise object identifier in this textbox. This is the dot notation numeric identifier associated with the enterprise in the MIB.
- **Generic trap:** Select the type of trap being sent. In SNMPv1 there are seven trap types. The available options are: Cold Start, Warm Start, Link Down, Link Up, Authentication failure, egp Neighbor loss and Enterprise specific. If you select **Enterprise specific**, then you must also enter a trap code in the **Specific trap code** textbox. For more information on trap types, see [Generic Trap Types](#).
- **Specific trap code:** This textbox is only available if you selected the **Enterprise specific** generic trap type. Enter the trap code here. The trap code has specific meaning within the enterprise. You can also use a dynamic function to obtain the trap code.

Trap v2c Specific Properties

- **Trap value:** Enter the object identifier of the trap being sent (in dot notation).

Get v1 and Get v2c Specific Properties

- **Data section:** Enter the name of the data section in which the returned values should be placed.

Get next v1 and Get next v2c Specific Properties

- Select one of **Walk until end of sub-tree** or **Walk for set number of repetitions** to determine when EMF should stop doing a Get next. If you select **Walk until end of sub-tree**, EMF will keep performing Get next requests until the parent of the retrieved object identifier is different to that of the starting object identifier (as specified in the **Object list** tab).

Note: The first object in the **Object list** tab is the only object used for this comparison. All other objects in the list are ignored.

If you select **Walk for set number of repetitions**, EMF will perform Get next requests the specified number of times or until the end of the MIB, whichever is reached first.

- **Data section:** Enter the name of the data section in which the returned values should be placed.

Get bulk v2c Specific Properties

- **Non repeaters:** Type the number of objects in the object list which are scalar, that is, have only one variable.
- **Max repetitions:** For tables, type the number of rows in the table that should be returned.

The total number of variables that the agent will attempt to return will be: (non repeaters) + (number of objects in tables * max repetitions).

- **Data section:** Enter the name of the data section in which the returned values should be placed.

[Object List tab](#)

[Back to Start of Data Modules](#)

OPC DA Read

OPC DA (Open Process Communications Data Access) support in EMF allows the EMF server to act as an OPC client and read/write data to OPC servers. OPC is a standard typically used to allow industrial devices and control applications to communicate. For more information on OPC see <http://opcfoundation.org>. EMF supports OPC DA v2.0. OPC DA v3.0 and OPC UA (unified architecture) are not currently supported.

You can use the **OPC DA Read module** to retrieve data from an OPC Server. OPC Items (tags) are selected that should be read, and the values for the items are then returned in an EMF datasection.

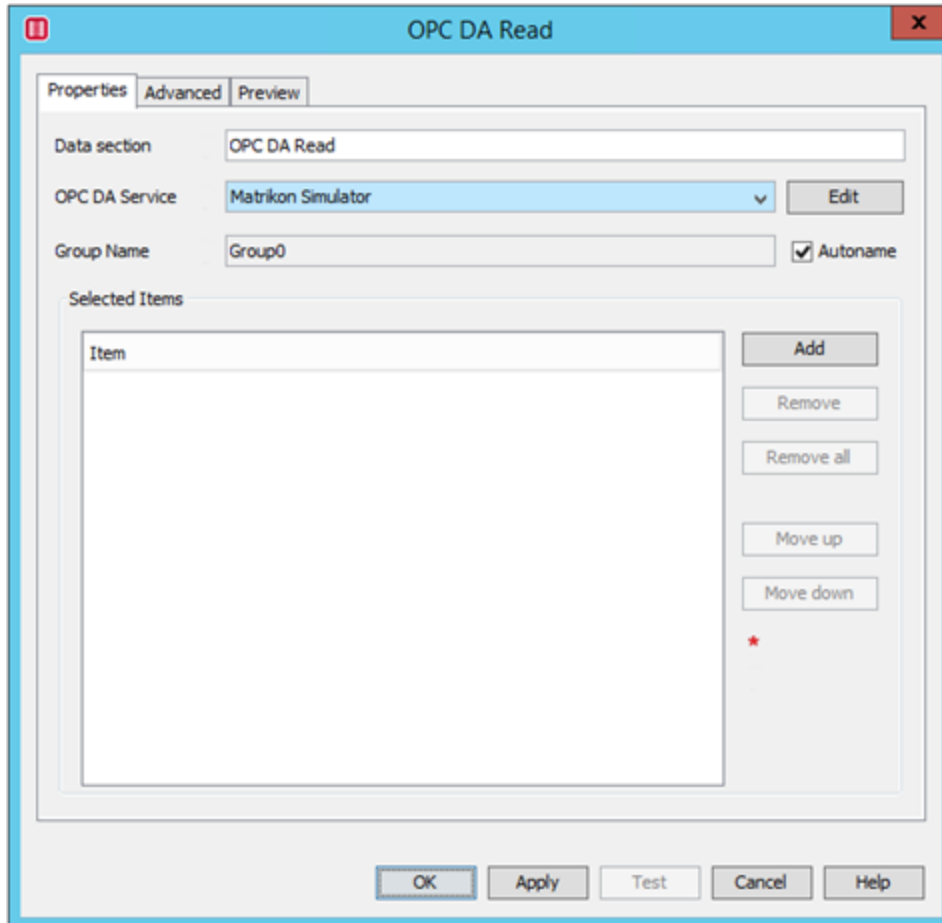
To use an OPC DA Read Module

1. Drag an **OPC DA Read module** icon into the EMF Process you are creating, at the

appropriate location:



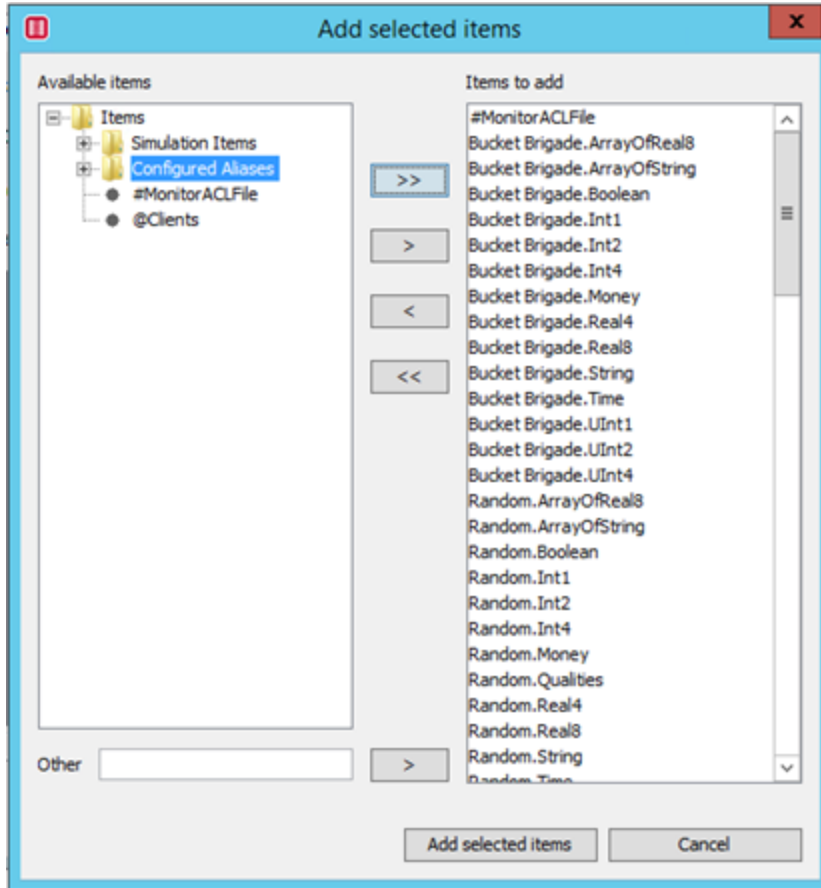
2. Double-click on the module icon to display the **OPC DA Read** properties.



3. Enter an appropriate name for the data section that will be created in the **Data Section** field.
4. Select the OPC DA service that contains the OPC server settings from the **OPC DA Service** drop-down list.
You can click **Edit** to modify the properties of the selected **OPC DA Service**.
5. Enter a **Group Name** to which the OPC items are added. Group names are unique per client connection. By default, the **Autoname** check box is selected which allows the group name to be automatically assigned.
6. In the **Selected Items** section, click **Add** to add the items that will be queried from the OPC server. This will open the **Add Selected items** window. This displays all

available items that the server has. Select the items to add from the available items and click **Add selected items**. If you do not currently have a connection to the OPC server, or an item you know exists is not displayed, or you wish to define the item dynamically at runtime using dynamic functions then you can manually enter the item in the **Other** field and add it.

You can also delete, or change the order of the items added using the **Remove**, **Remove all**, **Move up**, and **Move down** options.



7. Click **Test** to view the output data in the **Preview** tab. You can click **Copy to clipboard** button to copy the data to clipboard for future use. You can also click **Save as sample** button to save the data for testing purposes.

Item	Value	Timestamp	Quali
Bucket Brigade.ArrayOfReal8	[]	2015-09-10 02:04:07.441	192
Bucket Brigade.ArrayOfString	[]	2015-09-10 02:04:07.441	192
Bucket Brigade.Boolean	false	2015-09-10 02:04:07.441	192
Bucket Brigade.Int1	0	2015-09-10 02:04:07.441	192
Bucket Brigade.Int2	0	2015-09-10 02:04:07.441	192
Bucket Brigade.Int4	0	2015-09-10 02:04:07.441	192
Bucket Brigade.Money	0.0	2015-09-10 02:04:07.441	192
Bucket Brigade.Real4	0.0	2015-09-10 02:04:07.441	192
Bucket Brigade.Real8	0.0	2015-09-10 02:04:07.441	192
Bucket Brigade.String		2015-09-10 02:04:07.441	192
Bucket Brigade.Time	Sat Dec 30 00:00:00 EST 1899	2015-09-10 02:04:07.441	192
Bucket Brigade.UInt1	0	2015-09-10 02:04:07.441	192
Bucket Brigade.UInt2	0	2015-09-10 02:04:07.441	192
Bucket Brigade.UInt4	0	2015-09-10 02:04:07.441	192
Random.ArrayOfReal8	[]	2015-09-10 02:04:07.441	28
Random.ArrayOfString	[]	2015-09-10 02:04:07.441	28
Random.Boolean	false	2015-09-10 02:04:07.441	28
Random.Int1	0	2015-09-10 02:04:07.441	28
Random.Int2	0	2015-09-10 02:04:07.441	28
Random.Int4	0	2015-09-10 02:04:07.441	28

The data section created will contain the following columns:

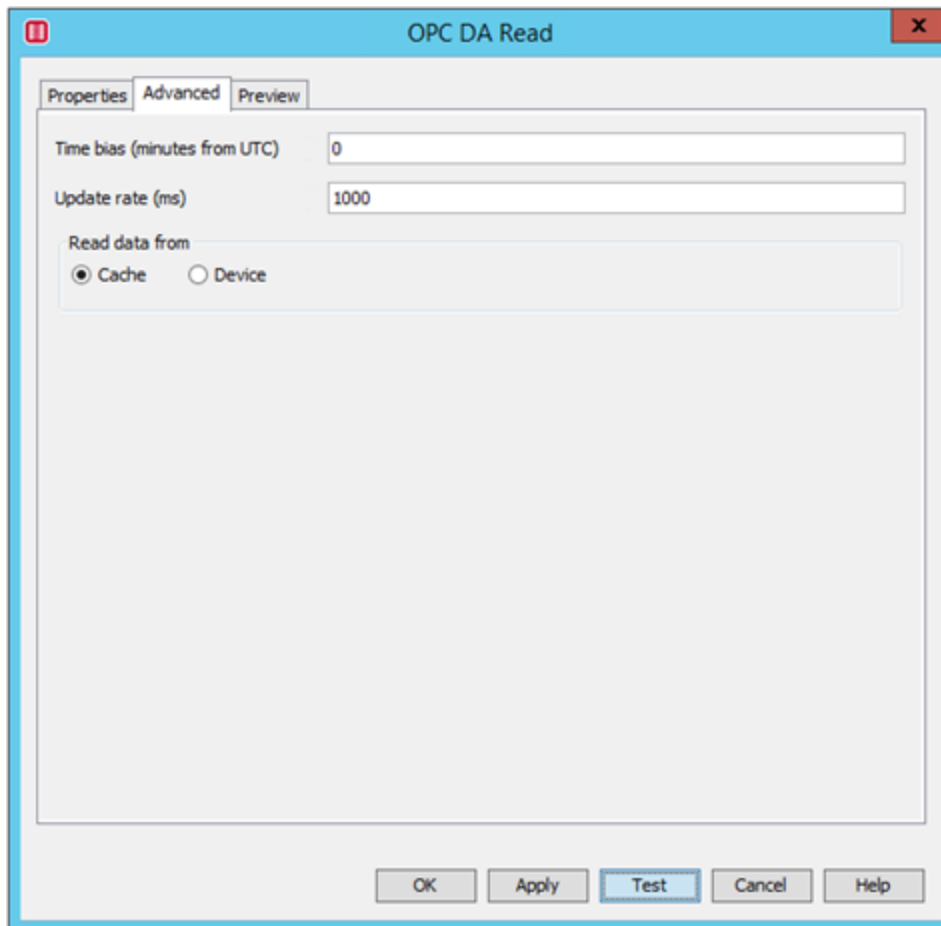
- **Item:** the item (tag) name.
- **Value:** the value read from the server
- **Timestamp:** the time when the OPC server last read the value from the device
- **Quality:** the quality code indicates the quality of the data returned from the OPC server. The high 8 bits of the quality value are vendor specific. The lower 8 bits map to a standard OPC quality code. See `QualityReason` column for a textual representation of the standard OPC quality code.
- **QualityReason:** gives a textual representation of the quality of the data.
- **ErrorCode:** the error code reports any issues with trying to communicate with the OPC server. A successful call will return 0.
- **ErrorCodeReason:** the textual representation of the error code.

Tip – When reading multiple values, the [DATAMAP](#) dynamic function is useful to access any of the returned item values. For example, if the item was called "opctest.myItem", then entering the following will return the value of that particular item:

```
$DATAMAP('OPC DA Read','Item','Value','opctest.myItem')$
```

8. Click the **Advanced** tab to set the advanced options.

Advanced Tab



The **Advanced** tab is used to define the following options:

- **Time bias:** This is the number of client minutes offset from UTC. Timestamps reported by the server may be adjusted by this option (this depends on the OPC server implementation).
- **Update rate:** This is the period that the OPC server will update the values from the device. The OPC server can then hold these values in its own cache. This means that for devices that take a long time to read, up to date values will be in the OPC server cache and instantly available preventing the module from blocking for a large amount of time (if reading from the cache). If reading data from the Device then this setting has no affect.
- **Read Data from:** Defines where the values read come from, select any of the following option:
 - **Cache:** Select this option to read data from the server cache.
 - **Device:** Select this option to read data from the actual device.

[Go to start of Data Modules](#)

Recipient Modules

You can use **Recipient modules** to create a list of system recipients and/or groups that can then be used by other modules to define who should receive an EMF Process notification. The same list can be used as many times as necessary, which saves the other modules from having to retrieve the same information more than once.

The Recipient modules

When the processing of an EMF Process reaches a Recipient module, the system recipients are filtered according to the criteria you specify, and then a **Recipient section** is created from a subset of them. The information contained in the resulting Recipient section can then be used by other modules further along in the EMF process, rather than have each module recreate its own list of recipients from scratch.

The Recipient modules use various techniques to gain information on recipients, and from that information define who requires an EMF Process to be sent to. There are three types of Recipient modules in EMF:

- **[Fixed Recipients module](#)** - Selects a list of recipients and groups that are already within the EMF system to create a recipient section in an EMF Process. Use this module if you know exactly who should receive the EMF Process output at the time of building the EMF Process.
- **[Aliased Recipients module](#)** - Takes a data section created in a previous module and uses a column of this data to look up recipient Alias values in order to determine the list of recipients to create a recipient section. Use this module when the recipients are EMF recipients but you do not know who they are at the time of building the EMF Process.
- **[Dynamic Recipients module](#)** - Takes a data section that was created in a previous module and uses the columns of this data to set recipient property values in a recipient section. Use this module when the recipients are not EMF recipients and you do not know who they are at the time of building the EMF Process.
- **[External Directory Recipients module](#)** - Used to select recipients from an external LDAP store (such as Microsoft Active Directory) to send EMF output to. Use this module to select a user (or group) if you want to send that user (or group) in Active Directory an email to notify them of some event.
- **[Escalation module](#)** - You can use Escalation to ensure that if an EMF Process recipient does not receive the information sent, the problem can be detected and steps be taken to rectify the situation. For example, another recipient can be sent the information, or the original recipient can be re-sent the message via another delivery method.
- **[Reply Module](#)** - You can use the Reply module to pause the processing of an EMF Process until the recipient (can be human, for example, via email, or from an external system, for example, using message queueing) has replied to the EMF Process output. When the EMF Process processing resumes you can perform further processing on the data received in the reply. An escalation code is used to link the recipient's reply to the original EMF Process, so the output module must include the dynamic function **ESCALCODE**.

A Recipient module can be placed anywhere before an output module, but it is generally more efficient to make it the module that immediately precedes an output module in order to save having to move the recipient section around.

[Go to start of EMF Help](#)

The Fixed Recipients Module

The **Fixed Recipients** module is used to select which of the existing EMF system Recipients and/or Groups should be included in a Recipient Section, so that this information can then be passed to subsequent modules. When the EMF Process is fired, Groups that have been chosen to populate the Recipient section are resolved into individual Recipients, so a Recipients Section is always effectively a list of individual recipients.

You can also define which of the existing information (for example, names, addresses, email addresses) associated with the recipients or groups should be included in the Recipients section by using the [Advanced](#) tab of the Fixed Recipients screen.

- You can create system recipients using the **Recipient Administration\Recipients** section of the EMF administrator (see [Creating a System Recipient](#)).
- You can create groups of recipients, and groups of groups, using the **Recipient Administration\Groups** section of the EMF administrator (see [Creating Recipient Groups](#)).

[How to use a Fixed Recipients Module](#)

[Go to start of Recipient Modules](#)

[Explanation of Output Modules](#)

[Explanation of Recipient Alias Values](#)

[Explanation of EMF Groups](#)

How to use a Fixed Recipients Module

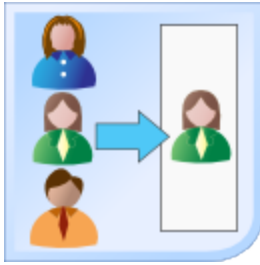
Fixed Recipient modules generate a Recipient Section from the existing EMF Recipients available. You can use the **Fixed Recipients** screen to define which Recipients are to be sent EMF Process information by selecting the required recipients or groups from the **Available recipients** list and adding them to the **Selected recipients** list. The selected Recipients can then receive information from the EMF Process being created.

A recipient module must be preceded by at least one **Initiator Module** (used to fire the EMF Process) and **Data Module** (used to collect information from databases), but does not have to be immediately after either.

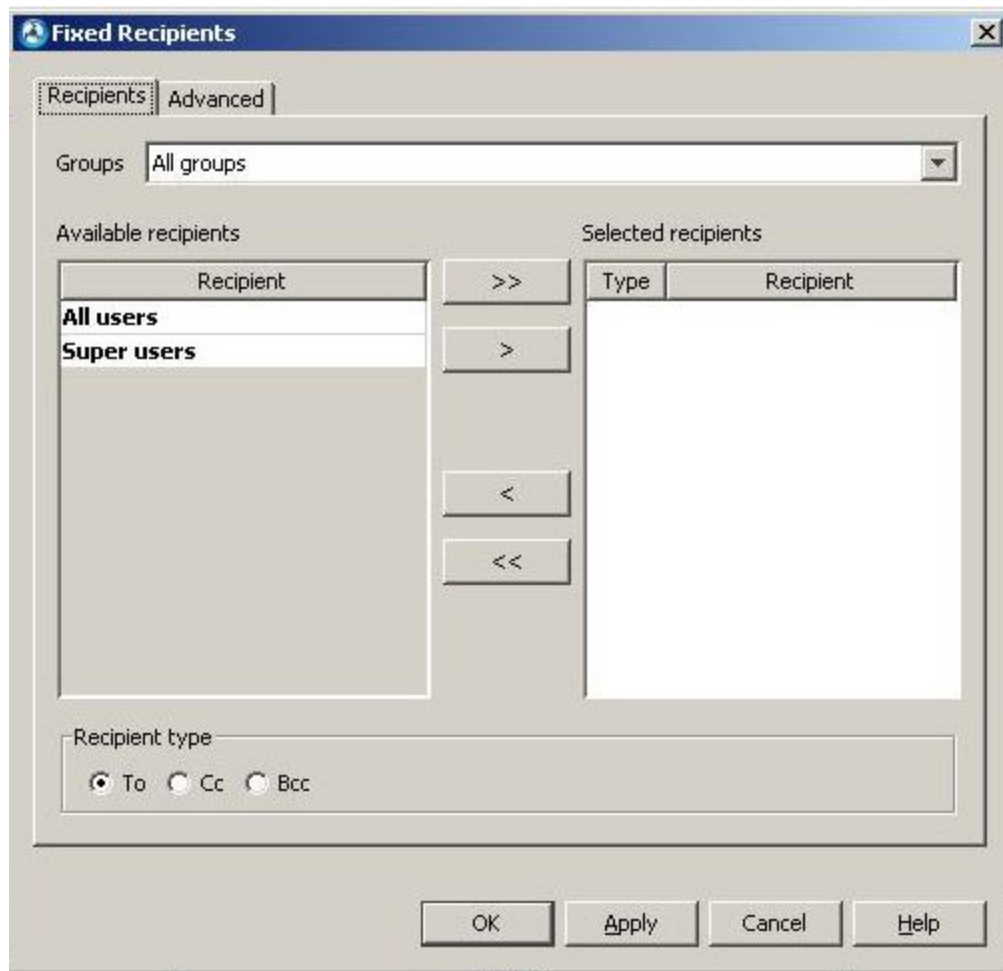
To add a Fixed Recipient Module to an EMF Process:

1. Drag and drop a **Fixed Recipient** module into an EMF Process and create a link from

a preceding module. If there are already subsequent modules in the EMF Process, create a link *from* the Fixed Recipient Module *to* the next module.



2. Double-click the Fixed Recipient module icon to display the **Fixed Recipients** screen.



3. Click the **Groups** drop-down list and select a set of system recipients for display in the **Available Recipients** list. You can select:
 - **All recipients** to display a list of all the individuals that are known to the system.
 - **All groups** to display a list of all the existing groups.

- Any of the other **groups** (for example, Super users, or any groups that you have created yourself) that exist in the EMF system to display a list of the members of that group.
4. Select a name from those displayed in the **Available recipients** pane.
 5. If the recipient is the main recipient of the mail, leave the **Recipient type** set to **To**. If the recipient is someone who should receive a copy of a message to the main recipient, or someone who should receive a copy of the message without the knowledge of the main recipient, select **Cc** or **Bcc** respectively.

Note: The Recipient Type is only applicable for certain types of message, for example, email. However, you must still specify a recipient type, even if the primary method of delivery does not support this feature, in case a later escalation module should require an email to be sent.

6. Click **>** to add the recipient to the **Selected recipients** list.

Tip: You can select several recipients at once by clicking the first one in the list and then holding down the SHIFT key while clicking the last one. You can also add all the recipients or groups in the list by clicking the **>>** button, and remove one or all of the selected recipients by clicking the **<** or **<<** buttons.

7. Repeat steps **4** to **6** until all the Groups and Recipients that should receive the EMF Process have been added to the **Selected recipients** panel.
8. Click the [Advanced](#) tab, where you can choose the individual items of information (name, address, email address, and so on) that should be retrieved for the listed recipients. You can also decide whether the information retrieved should add to or replace that in any previous Recipient sections, and whether to split the recipients into blocks for optimal processing.

[The Fixed Recipients screen Advanced tab](#)

[Explanation of Fixed Recipients Module](#)

[Go to start of Recipient Modules](#)

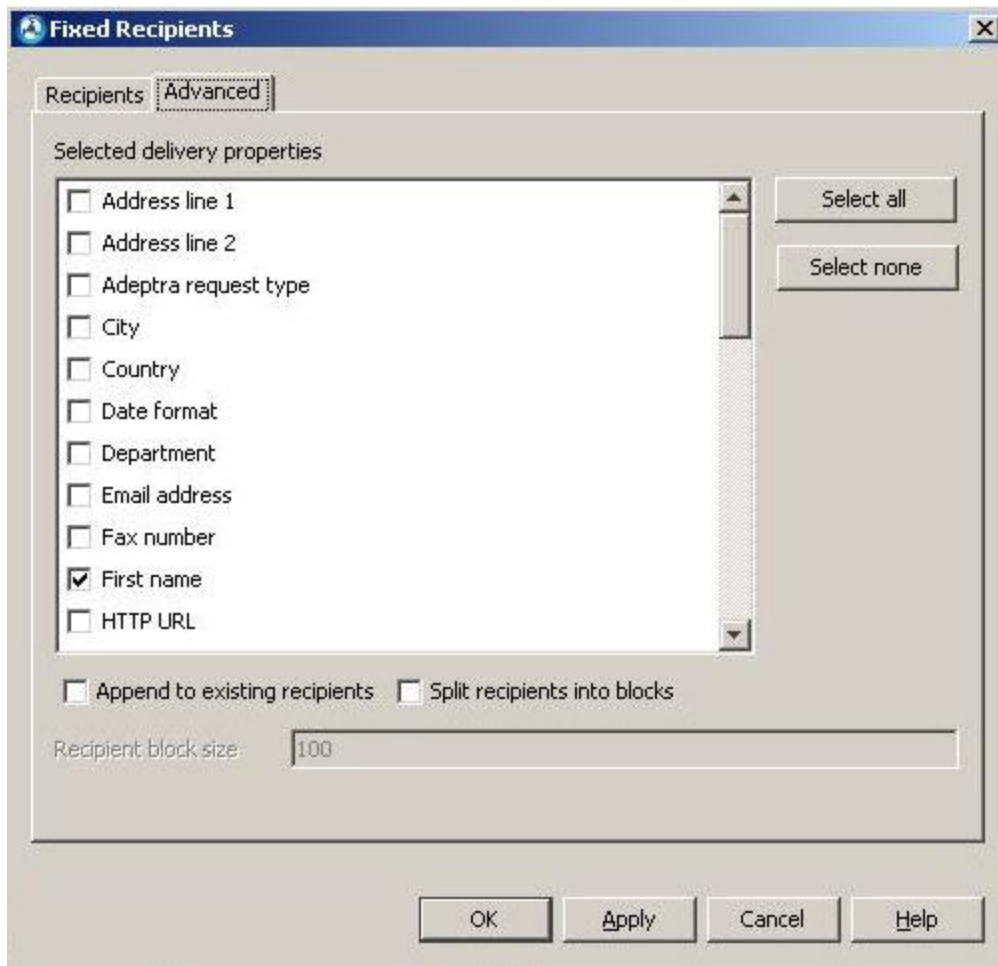
[Explanation of Recipients](#)

[Explanation of System Groups](#)

[Introducing EMF](#)

The Fixed Recipients Advanced tab

The **Advanced** tab of the **Fixed Recipients** module contains a list of properties that can be retrieved from each Recipient. If, in the EMF Process thread, a property is required for each recipient and that has not been selected for retrieval by a Fixed recipients module, then the subsequent module will have to retrieve the property for each recipient. Selecting properties here is therefore an optimization, since properties are fetched in bulk rather than individually.



- **Selected delivery properties** - Displays a list of all the types of Recipient details. When a check mark is placed next to a list item, the details for that field will be retrieved and placed in the Recipient Section for all recipients in that section.
- **Append To existing recipients** - Select this option if you want to add Recipients in this Recipient section to those in a previous Recipient Section in the EMF Process. If this option is not selected, Recipients in this section will replace those from any previous Recipient Sections.
- **Split recipients into blocks** - Select this option if you want to divide recipients in the recipient section into a number of smaller and more manageable blocks for your Output server to deal with. If there are a lot of recipients, the Output server may be slowed down by the amount of processing required to complete the delivery of the EMF Process information.

If you select the "Split Recipients Into Blocks" option, you can enter the number of recipients to be placed in each block in the **Recipient block size** field. The number entered here defines the maximum number of recipients in each block. Note that the last block will contain less than the others, unless the total number of recipients is exactly divisible by the **Recipient Block Size** chosen.

[Explanation of Fixed Recipients](#)

[Explanation of Fixed Recipients screen](#)

[Go to start of Recipient Modules](#)

[Explanation of Recipients](#)

The Aliased Recipients Module

Sometimes, you may need to create and run an EMF Process where the recipients cannot be known in advance (for example, you may need to send a monthly reminder to all your clients who have an outstanding balance of £1000 or more), or may change over time. In such a situation, instead of using a [fixed recipients](#) module (which defines a specific set of recipients) or a [dynamic recipients](#) module (which gets unfiltered recipient details from an existing data section), you could use an **Aliased Recipients** module.

The Aliased Recipients module takes customer details from an existing Data section and matches them against a specific requirement, then creates a recipients section from those that match. The resulting list is then resolved against a predefined alias that maps the data to a recipient in the repository. The list of recipients that should receive the EMF Process is therefore automatically created afresh each time the EMF Process is run.

Example usage of Aliased Recipients

Aliased recipients are generated by selecting an Alias type, selecting a specific column within a data section (for example, "account number"), and then comparing its contents to the equivalent Alias type values held by recipients within the EMF System (see [Recipients Administration](#)). If a data section column value matches an Alias type value for a system recipient, they will be brought into the Recipient Section.

For the example given below, we will assume that the customer database contains a table as follows:

ID	First Name	Last Name	Account No	Location	Credit Limit	Length of Account	Balance
1	John	Green	10216	London	£5000	6.3 years	£942
2	Samantha	Forsyth	14395	Cambridge	£3000	3.7 years	£2871
3	Terry	White	18630	Stevenage	£2000	2.9 years	£2351
4	Jo	Bentham	21763	Weybridge	£1000	1.9 years	£528
5	Martin	Armstrong	26839	London	£1000	1.4 years	£1823

6	Chris	Achilles	31472	Oxford	£1000	0.3 years	£387
---	-------	----------	-------	--------	-------	-----------	------

Each month, you want to send an EMF Process to all those who are overdrawn by more than £1000.

1. [Create a new Alias](#) called "Account number".
2. [Add this alias to all your recipients' details](#), in each case entering their account number as the **alias value**.

Note: You can assign as many alias values as you like to each recipient.

3. Create an EMF Process and add a [Data module](#) that will retrieve all the **account numbers, credit limits** and **balances** from the table shown above where the **balance** figure exceeds the **credit limit**.
4. Add an [Aliased recipients module](#) and specify **Account number** (or whatever you named it in step 1) as the *alias type* and **column 3** as the *lookup column*.

Note: The numbering starts from zero, so column 3 is the "Account number" column in the table above.

5. Add a [Formatter module](#) and write a message to send to the overdrawn people explaining the situation.

Note: You can use standard recipient-based [dynamic functions](#) (for example, `FIRSTNAME`, `LASTNAME`, and `ADDRESS`) to personalize the messages by retrieving the appropriate information from the recipient's details, but you can also use the [RECIPIENT](#) and/or [RECIPTABLE](#) functions (referencing the column number in the table above) to enter each individual's *specific* overdrawn amount. This is because the alias effectively adds new columns to the Recipient section containing the specified information retrieved from the Data section.

6. Add a [Scheduler module](#) to the EMF Process and set it to run monthly.

Each time that the EMF Process runs, the specified column in the data section (in this case, the account number) is compared to each individual's alias value and, if they match on the alias value, that recipient is added to the list of recipients for the EMF Process.

Other uses for Aliased Recipients

There are countless situations where using Aliased Recipients can improve and facilitate your alerting system, for example:

- Matching sales people to accounts, in order to send "new business" EMF Processes. Each sales person could have a set of alias values mapping account names that they are responsible for. This is matched and an email sent when new business is logged to a particular account.
- Matching sales people to accounts, in order to send "new business" EMF Processes. Each sales person could have a set of alias values mapping account names that they are responsible for. This is matched and an email sent when new business is logged to a particular account.

- Matching sales people to accounts, in order to send "new business" EMF Processes. Each sales person could have a set of alias values mapping account names that they are responsible for. This is matched and an email sent when new business is logged to a particular account.

[How to use an Aliased Recipients Module](#)

[Explanation of Recipient Alias Values](#)

[Go to the start of Recipient Modules](#)

How to use an Aliased Recipients Module

Aliased recipients are generated by selecting an Alias type (see [Aliases](#)), selecting a column within a data section, and then comparing its values to the specified Alias type values held by recipients within the EMF System (see [Recipient Administration](#)). If a data section column value matches an Alias type value for a recipient, they will be brought into the Recipient Section.

To create an Aliased recipients section:

1. Drag and drop an **Aliased recipient** module into an EMF Process at some point after a Data Module and create a link to it. If there are already subsequent modules in the EMF Process, create a link *from* the Aliased Recipient module *to* the next module.



2. Double-click the Aliased Recipient Module to display the **Aliased recipients module Properties** screen.

3. Select the required **Data section** from the drop-down list of those created by existing [Data Modules](#) in the EMF Process (for example, an SQL module).
4. Select the **Create unresolved alias list** option to add an error section to the EMF Process if any unresolved rows are found.

The name of the unresolved alias data section is **UnresolvedAlias**. If error handling is enabled for the EMF Process, then this information is also stored in the [ErrorLog data section](#) that is passed to the [Error Handler](#) for the EMF Process.

5. Select the **Case sensitive** option if you want the matching of the data section column value to the alias value to be case sensitive.
6. Define the column in the Data section that is to be used by entering a number in the **Lookup Column** field.

Note: The column numbering starts from zero, so the first column would be number 0 and the second column would be 1, and so on.

7. Select the **Alias type** - this is the Alias name (for example, Bank Account). The Alias types that appear in the list are those that have previously been defined using the [Aliases screen](#).

Note: The **System** alias is an alias automatically defined for each recipient by the system. It is always the same as the recipient name.

8. Click the [Advanced](#) tab, where you can choose the individual items of information (name, address, email address, etc.) that should be retrieved for the listed recipients. You can also decide whether the information retrieved should add to or replace that in any previous Recipient sections, and whether to split the recipients into blocks for optimal processing.
9. Click **Test** to preview the recipient section that will be created using the available sample data.

Note: Use the [RECIPTABLE Dynamic Function](#) in the message section of the relevant [Output formatting module](#) to access all the data from multiple rows in the data section.

[The Aliased Recipients Advanced Tab](#)

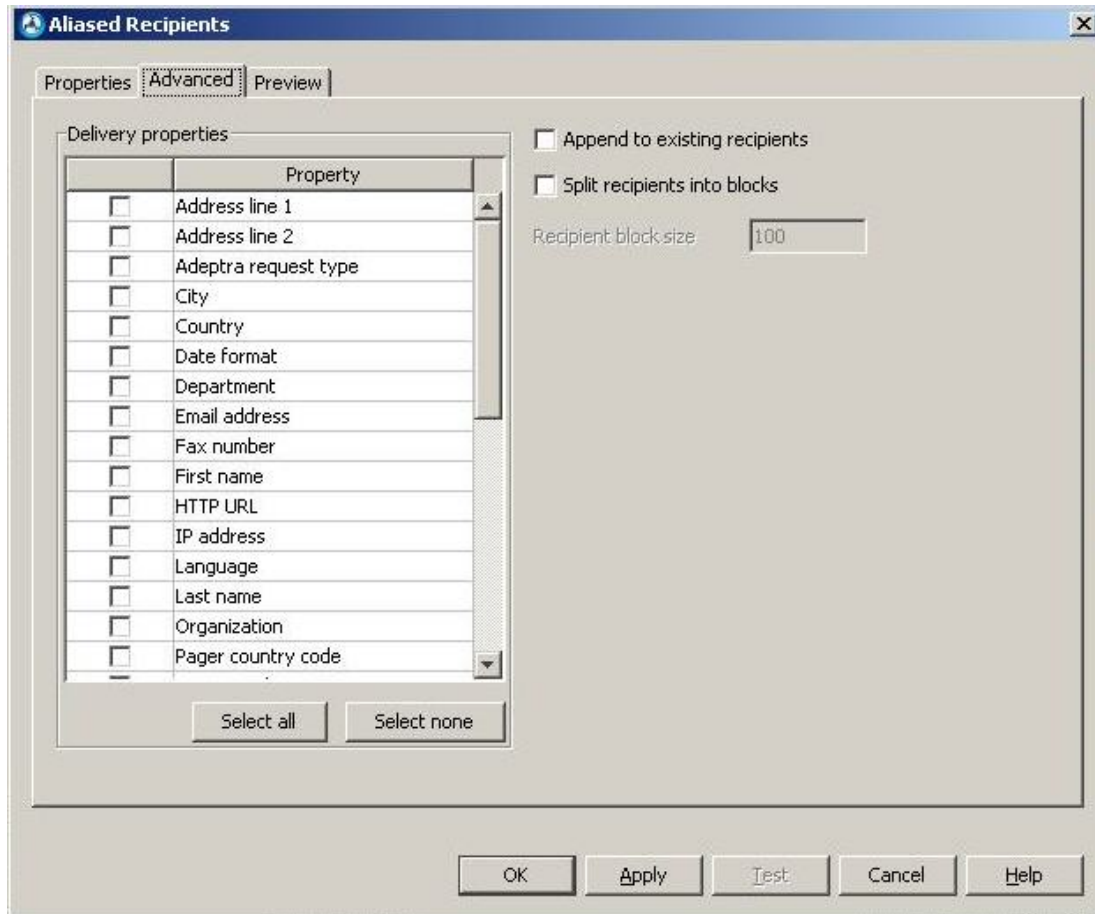
[Go to start of Aliased Recipients](#)

[Go to start of Recipient Modules](#)

[Go to start of EMF Help](#)

The Aliased Recipients Advanced Tab

The **Advanced** tab of the **Aliased Recipients** module contains a list of the properties that can be retrieved from each System Recipient. If, further on in the EMF Process thread, a property is required for each recipient and that has not been selected for retrieval by a Fixed Recipients module, then the subsequent module will have to retrieve the property itself for each recipient. Selecting properties here is therefore an optimization, since properties are fetched in bulk rather than individually.



- **Delivery properties** - Displays a list of all types of System Recipient details. When a check mark is placed next to a list item, the details for that field will be retrieved and placed in the Recipient Section for all recipients in that section.
- **Append to existing recipients** - You should select this option if you want to add the Recipients in this Recipient section to those in a previous Recipient Section in the EMF Process. If this option is not selected, Recipients in this section will replace those from any previous Recipient Sections.
- **Split recipients into blocks** - You should select this option if you want to divide recipients in the recipient section into a number of smaller and more manageable blocks for your Output server to deal with. If there are a lot of recipients, the Output server may be slowed down by the amount of processing required to complete the delivery of the EMF Process information.

If you select the "Split Recipients Into Blocks" option, you can enter the number of recipients to be placed in each block in the **Recipient block size** field. The number entered here defines the maximum number of recipients in each block. Note that the last block will contain less than the others unless the total number of recipients is exactly divisible by the **Recipient Block Size** chosen.

[Explanation of Aliased Recipients](#)

[Explanation of the Aliased Recipients screen](#)

[Go to start of Recipient Modules](#)

The Dynamic Recipients Module

The **Dynamic Recipients** module takes a Data Section created previously in the EMF Process and uses the columns of this data to map data values to recipient properties in a Recipient Section.

For example, if you have a database containing email addresses and user names, then you can query this information in a SQL module and then map the returned fields to recipient properties using the dynamic recipient module. This then allows you to send email messages to those recipients.

When using dynamic recipients, all the necessary values must exist in the recipient section for a subsequent module to function correctly. This is because these recipients are dynamic and not in the EMF Repository, so no extra information is stored about them. Thus, if a module further on in the EMF Process should need an additional property for one of these recipients, it would not be able to retrieve it (such as Email out requires an email address - so this field must be mapped).

The **Dynamic Recipients screen** is used to define a Recipient section based on the contents of a data section created by a previous data module in the EMF Process (such as SQL). These columns of data are used to set up recipient property values in an EMF Process recipient section.

A Dynamic Recipients module requires a Data Section.

Data Sections

In respect to Dynamic Recipients, a Data Section should contain Customer/Recipient details, such as User Name, Email Address, SMS Provider and Number etc. These columns are resolved by number to represent the various delivery properties for each recipient.

[How to use a Dynamic Recipients Module](#)

[Go to start of Recipient Modules](#)

How to use a Dynamic Recipients Module

Dynamic recipients are generated by selecting columns from within a data section to represent different delivery properties for recipients. Each row in the data section will be resolved into a recipient.

To create a Dynamic Recipients section:

1. Drag and drop a **Dynamic Recipients** module into an EMF Process and create a link from a previous data module in the EMF process. If the data module is already in the middle of an EMF Process thread, create a link *from* the Dynamic Recipients module to the next module.



2. Double-click the Dynamic Recipients icon to display the **Dynamic Recipients Properties** screen.

Dynamic Recipients

Properties Preview

Data section *

Delivery properties

☒ Use column names

☐ Use column indexes (zero-based)

Property	Column names
Address line 1	
Address line 2	
Adepra request type	
City	
Country	
Date format	
Department	
Email address	
Fax number	
First name	
HTTP URL	
IP address	

Clear all

☒ Discard data section after use

☐ Append to existing recipients

☐ Split recipients into blocks

Recipient block size: 100

OK Apply Test Cancel Help

3. Click the arrow next to the **Data section** field and select from the drop-down list of those in your EMF Process.

4. Select the **Delivery properties** that you wish to gather from the data section by entering *either* the column indexes (i.e. the reference numbers) *or* the column names (you can find out the information that is contained within the data section columns by clicking the **Test** button within the [Data module](#) that generates it).

For best results, map each column in the data section to a single delivery property - if a **Column** field is blank, the property is not included.

You can also enter [dynamic functions](#) (not recipient-based) by right-clicking in the **Column no./name** field and selecting from those available.

- To specify column indexes, select the **Use Column indexes** option and then enter them in the **Column indexes** fields. **Note:** column numbering starts from zero, so to specify the fourth column, you would enter the number 3.
- To specify column names, select the **Use Column names** option and then enter them in the **Column names** fields.

Note: You can tell the Dynamic Recipients module to interpret your input as *either* names *or* indexes. Whichever you choose, you can still enter text characters (because you might want to insert a dynamic function) and/or numerals (because your column name might be "3") into the fields, but these will be interpreted differently. If you later change the column type, it is likely that much of the information that you have entered so far will no longer be valid.

Note: The property *ToCcBcc type* is a special property that allows you to define how a dynamic recipient receives emails (whether they are sent TO, CCed or BCCed). To send emails via TO, the property must map to a data field that contains *0* (this is the default setting if no mapping is specified for this property). To send emails via CC, the property must map to a data field that contains *1*, and *2* for BCCed.

Important: If no recipient is specified to receive an email by TO, then no email out is sent in EMF. To BCC a list of recipients, also include one dummy recipient who will receive the message as a TO.

5. Select the **Discard data section after use** option if you do not want to keep the data section that is used to create the dynamic recipients after this module has executed. This can improve performance but may mean that you will need to recreate the same data later in the EMF process.
6. Click **Test** to view the results.

Note: If you are using all the columns in a data section (i.e. there are no unmapped columns), you may notice a considerable performance increase by enabling the **Discard data section after use** option.

7. Select the **Append to existing recipients** option if you want to add the recipients in this recipient section to those in a previous recipient section in the EMF Process. If this option is not selected, Recipients in this section will replace those from any previous recipient sections.
8. Select the **Split recipients into blocks** option if you want to divide recipients in the recipient section into a number of smaller and more manageable blocks for your output server to deal with. If there are a lot of recipients, the output server may be

slowed down by the amount of processing required to complete the delivery of the EMF Process information.

If you select the "Split recipients into blocks" option, you can enter the number of recipients to be placed in each block in the **Recipient block size** field. The number entered here defines the maximum number of recipients in each block, note that the last block will contain less than the others unless the total number of recipients is exactly divisible by the **Recipient block size** chosen.

9. Click **Test** to preview the recipient section that will be created using the available sample data.

[Explanation of Dynamic Recipients Module](#)

[Go to start of Recipient Modules](#)

[Introducing EMF](#)

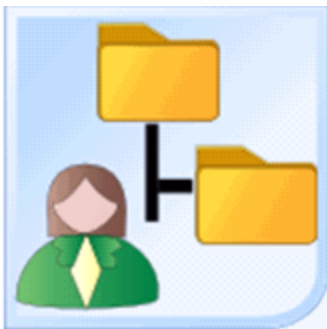
External Directory Recipients Module

The External Directory Recipients module is used to select recipients from an external LDAP store (such as Microsoft Active Directory) to send EMF output to. For example, if you wanted to send a user (or a group) in Active Directory an email to notify them of some event, then you would use this module to select that user (or group). Before using the module, a [JNDI service](#) needs to be configured with the appropriate connection details to the LDAP store.

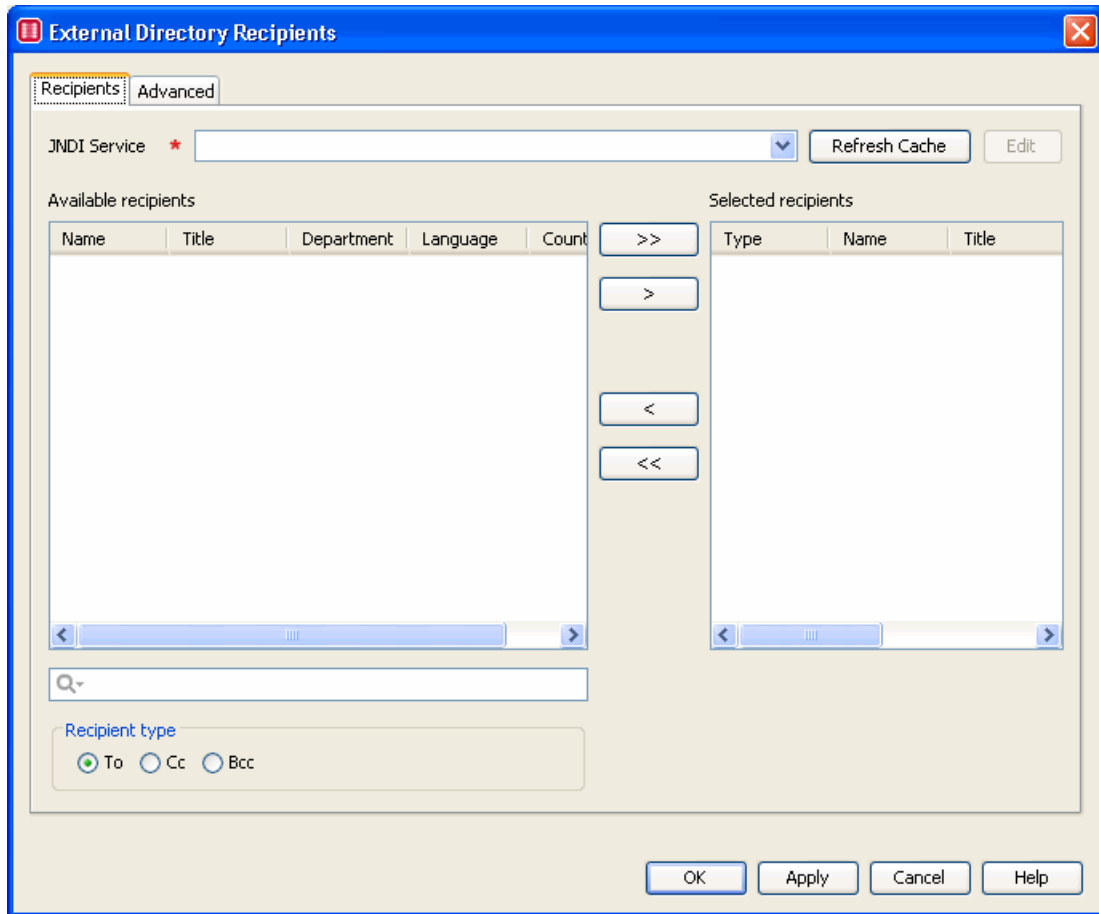
Note: This module is only tested against Microsoft Active Directory, but other LDAP stores may work.

To add an External Directory Recipients module to an EMF Process:

1. In the EMF Process builder, drag the icon for the External Directory Recipients module to the appropriate place in your EMF process.



2. Open the External Directory Recipients module to display its properties. By default, you need to double-click the icon to open, but you can configure mouse-click actions on the Options window. From the Tools menu, select **Options -> EMF-> Process Builder Behaviour** tab.



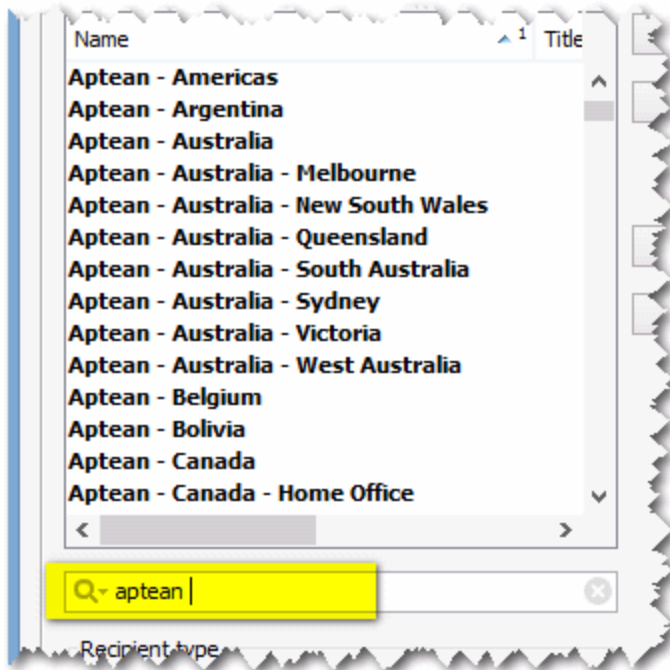
3. Select the **JNDI Service** to which you want to add white listed items. The first time a new JNDI service is selected in an external recipients module, all the queries defined in the service are run to read in the available users and groups. This may take some time. This information is cached locally, so subsequently when that service is selected, there is only a small delay to read in the information locally. If the contents of the LDAP store have changed (new recipients added or removed), or the queries in the service have been modified, click **Refresh Cache** to renew all the information in the cache for that service.
You can click **Edit** to modify the properties of the selected JNDI Service.

Note: At runtime, the server does not use this cached information and will access the LDAP store to find the update to date information on users/groups.

4. After a service has been selected, then the **Available recipients** list will be populated with all available users and groups. Groups are shown in bold in the list. Double-click an item in the list or select items and press the > button, to copy the selected items to the **Selected recipients** lists. The selected recipients list contains the list of recipients that will receive any message.
5. If the recipient is the main recipient of the mail, leave the **Recipient type** set to **To**. If the recipient is someone who should receive a copy of a message to the main recipient, or someone who should receive a copy of the message without the knowledge of the main recipient, select **Cc** or **Bcc** respectively.

Note: The Recipient Type is only applicable for certain types of messages, for example, email. However, you must still specify a recipient type, even if the primary method of delivery does not support this feature, in case a later escalation module should require an email to be sent.

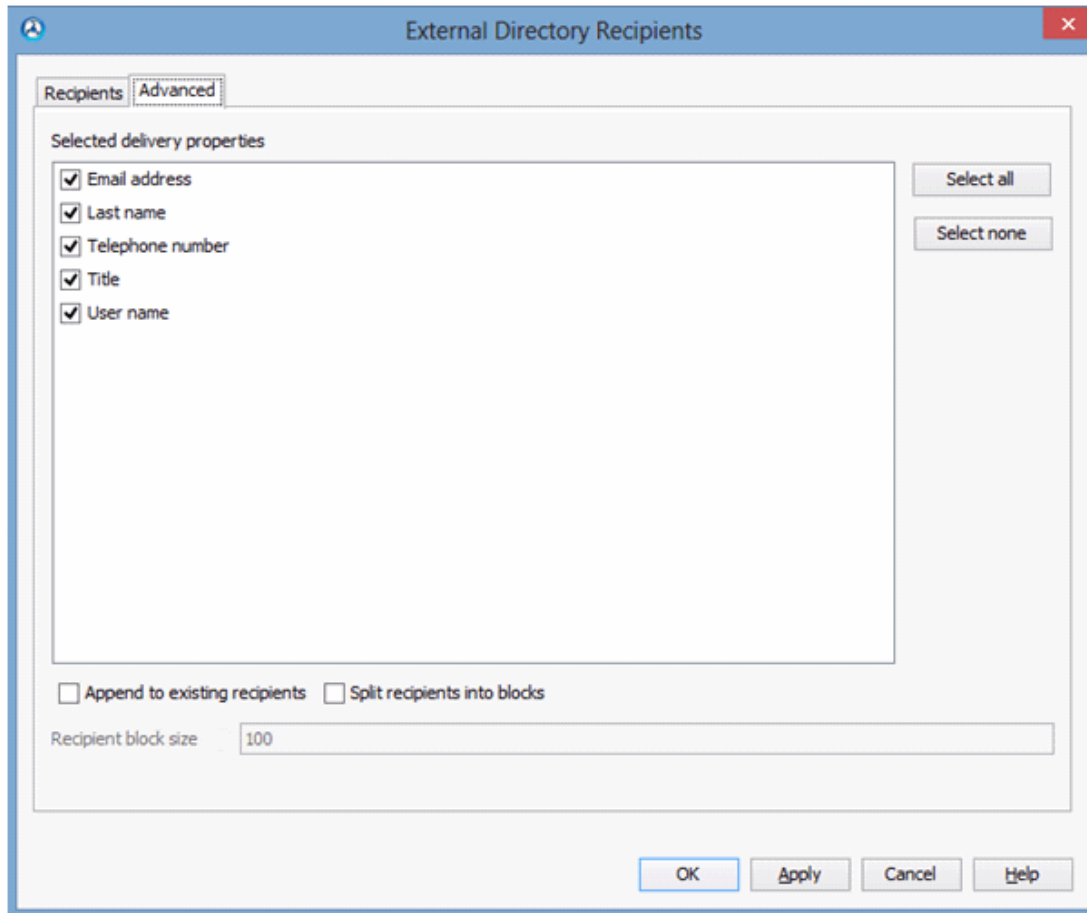
6. The Available recipients list can be filtered to make it easier to find the user/group required. Just type the name in the filter box.



[The External Directory Recipients Advanced tab](#)

The External Directory Recipients Advanced Tab

The **Advanced** tab of the **External Directory Recipients** module contains a list of properties that can be retrieved from each Recipient. The available properties are dependent on the mapping created in the JNDI Service on the [External recipient/operator LDAP attribute mappings](#) tab. Any properties that will be needed by later modules (such as email address) must be selected in the list (they cannot be retrieved later, as is the case with the other recipient modules).



- **Selected delivery properties:** Displays a list of all the types of Recipient details. When a check mark is placed next to a list item, the details for that field will be retrieved and placed in the Recipient Section for all recipients in that section.
- **Append to existing recipients:** Select this option if you want to add Recipients in this Recipient section to those in a previous Recipient Section in the EMF Process. If this option is not selected, Recipients in this section will replace those from any previous Recipient Sections.
- **Split recipients into blocks:** Select this option if you want to divide recipients in the recipient section into a number of smaller and more manageable blocks for your Output server to deal with. If there are a lot of recipients, the Output server may be slowed down by the amount of processing required to complete the delivery of the EMF Process information.

If you select the **Split recipients into blocks** option, you can enter the number of recipients to be placed in each block in the **Recipient block size** field. The number entered here defines the maximum number of recipients in each block. Note that the last block will contain less than the others, unless the total number of recipients is exactly divisible by the **Recipient block size** chosen.

[External Directory Recipients Module](#)

Escalation Module

You can use **Escalation** to ensure that if an EMF Process recipient does not receive the information sent, the problem can be detected and steps be taken to rectify the situation. For example, another recipient can be sent the information, or the original recipient can be re-sent the message via another delivery method.

Every EMF Process that contains an Escalation module is assigned an **escalation code**. This is a unique code that can be returned to the EMF Process by the recipient to prove receipt.

Important: The appropriate **Listener** ([POP3](#)) and [Escalated EMF Processes Retriever \(EAR\)](#) server instances must be configured and activated before the escalation will function correctly (the **listener** monitors for return receipts from EMF Processes and when it detects a receipt, it passes the necessary information to the EMF system. The **EAR** polls the EMF system for EMF Processes that need to be escalated and, when it determines that escalation is required, triggers the EMF Process to continue processing).

To add an Escalation module to an EMF Process:

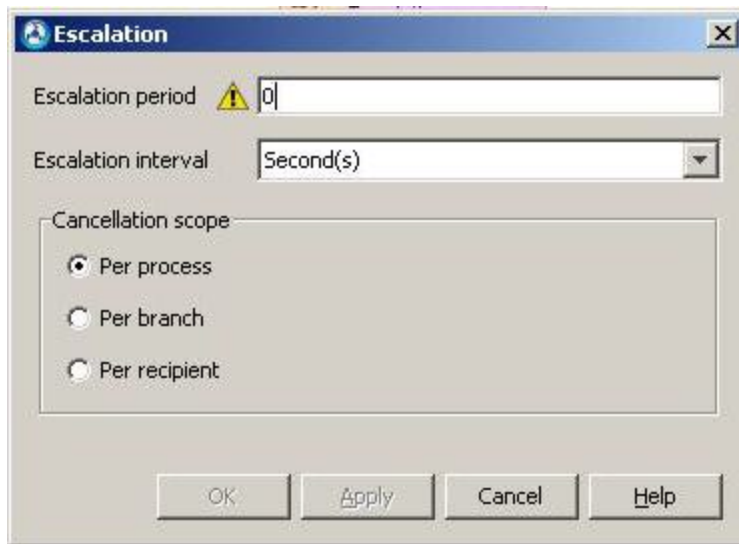
1. Ensure that you have already added and configured all the other modules in the EMF Process (you should always do this before adding an Escalation module). Also, ensure that the **Output formatting module** that defines the text for the Output module contains the [ESCALCODE Dynamic function](#).
2. Drag and drop an **Escalation module** into an EMF Process in an appropriate place (i.e. after an [Output Module](#) that can send a reply, e.g. SMS Out).



Note: you can use as many Escalation modules as you like in a single EMF Process, but you can only place a single Escalation module after each Output module.

3. Enable [Message State Logging](#) in the **Advanced Properties** of the **Output module** preceding the Escalation module (this is essential in order for escalation to work).

- Double-click the Escalation module icon to open the **Escalation** screen.



The **Escalation** screen allows you to specify when and how an EMF process should continue if recipients do not reply. The EMF process will only continue if a return receipt from recipients is not detected by EMF in the time specified within this screen.

- Enter a number for the **Escalation period** and select a unit for the **Escalation interval** (the default interval is **Seconds**). Escalation will occur if the recipient does not reply within the specified number of seconds, minutes, hours, days, weeks or months.
- Select a **Cancellation Scope** in order to define when, and if, to cancel the escalation. The available options are:
 - Per EMF Process** - If a single recipient replies to the message from the Output module, escalation is cancelled. Any other Escalation modules within the EMF Process that are waiting for replies are also cancelled. [Example](#).
 - Per Thread** - If a single recipient replies, escalation is cancelled in that thread of the EMF Process. The EMF Process Escalation Module instance is cancelled, but no others in the EMF Process Instance. [Example](#).
 - Per Recipient** - If all recipients reply, escalation is cancelled in that thread of the EMF Process. If escalation does occur, all the recipients that replied will be removed from the Recipient section to ensure that all recipients receive the information. [Example](#).

You can combine multiple Escalation modules, with different cancellation scopes, within the same EMF Process - see this [example of combining of cancellation scopes](#).

- Continue the EMF Process thread after the Escalation Module to specify what should be done if Escalation occurs (the simplest thread to generate would be an additional Output Module linked to the Escalation module).

[Go to Start of Recipient Modules](#)

Reply Module

You can use the Reply module to pause the processing of an EMF Process until the recipient (can be human, for example, via email, or from an external system, for example, using message queueing) has replied to the EMF Process output. When the EMF Process processing resumes you can perform further processing on the data received in the reply. An escalation code is used to link the recipient's reply to the original EMF Process, so the output module must include the dynamic function **ESCALCODE**.

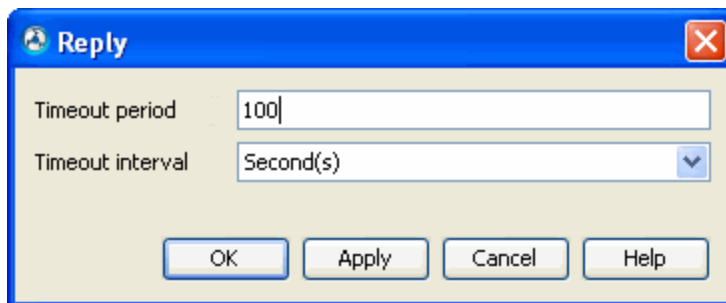
You can also insert a [Conditional Halt module](#) after the reply module and then continue processing the EMF Process (or not) depending on whether or not a reply has been received.

To use a Reply Module:

1. If you haven't already done so, configure the following:
 - An [Escalated EMF Processes Retriever server instance](#).
 - The [System EMF Process](#) that you want to use to process the incoming response from the device listener.
 - A device listener to monitor incoming responses.
2. Insert an [Output module](#) that includes the ESCALCODE [dynamic function](#) into the EMF Process. Select the [Advanced](#) tab of the Output module and set **Message State Logging** to full.
3. Drag and drop a **Reply** module into the EMF Process at the appropriate place (i.e. at some point after the Output module).



4. Double-click the icon to display the **EMF Process reply Properties** screen.



5. Set the desired timeout period: **Reply period** is the number of time units that you want the EMF process processing to pause before timing out (if a reply has not been received), and **Reply interval** defines the units - i.e. seconds, minutes, hours, days, weeks or months.

To check whether a reply has been received:

If a reply is received, it will create a Data section called [REPLY_DATA](#). You can then use conditional links and DATAEXISTS dynamic function to test for its existence and modify the behavior of the EMF Process accordingly.

For example, you can branch the processing of the EMF Process after the Reply module and place a Conditional Halt module on each branch. You can then set one to halt processing if the REPLY_DATA section exists, and the other to halt processing if the REPLY_DATA section does not exist.

[The REPLY_DATA Data Section](#)

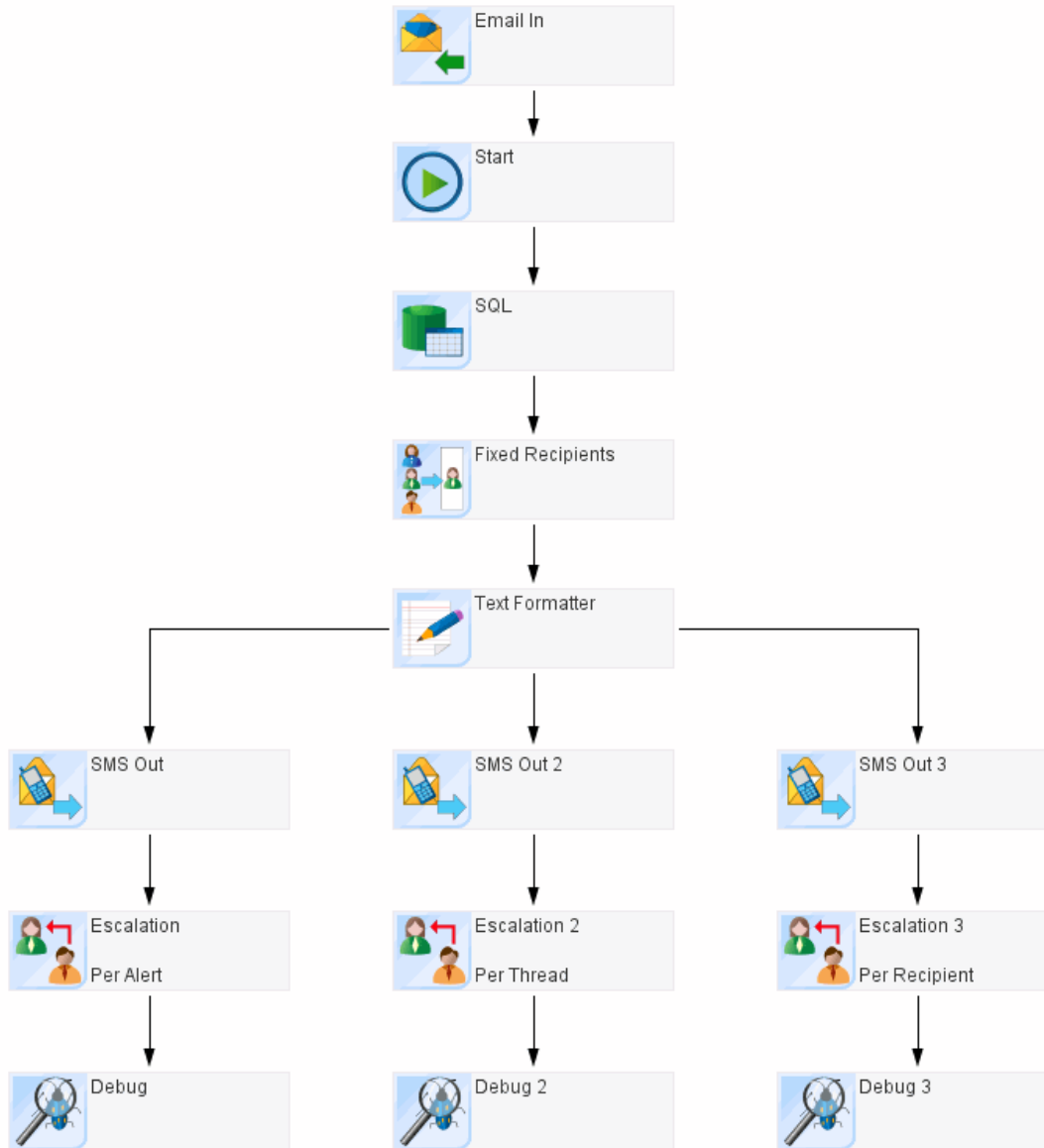
[How the Reply Module Works](#)

[Go to Start of Recipient Modules](#)

Example of Combined Cancellation Scopes

You can combine Escalation modules with different cancellation scopes within the same EMF Process.

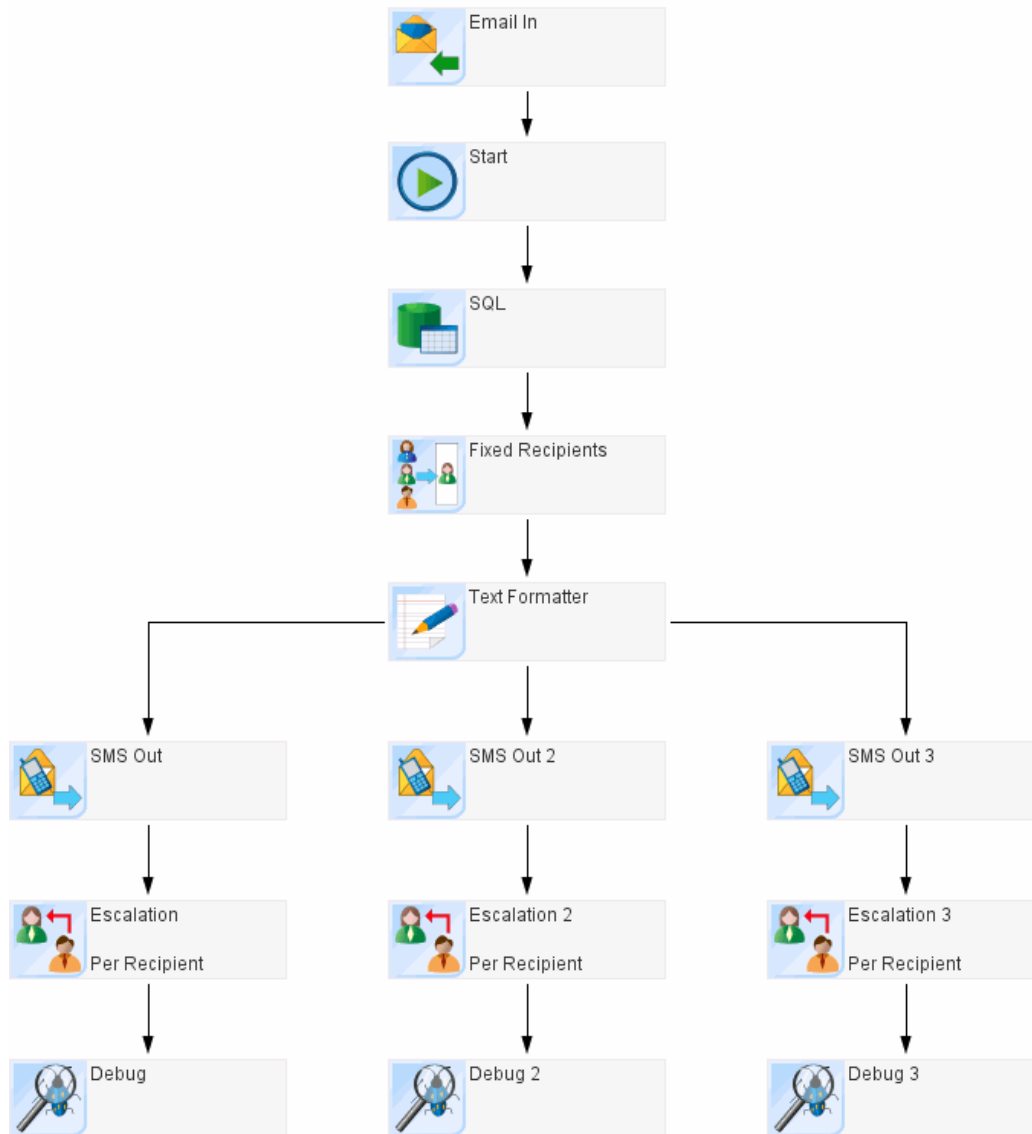
In this example, if one recipient for SMS Output reply's to the SMS message, Escalation, Escalation (1) and Escalation (2) are *all* cancelled; if one recipient for SMS Output (1) reply's to the SMS message, Escalation (1) is cancelled; if all recipients for SMS Output (2) reply to the SMS message, Escalation (2) is cancelled.



[How to use an Escalation Module](#)

Example of Per Thread Cancellation

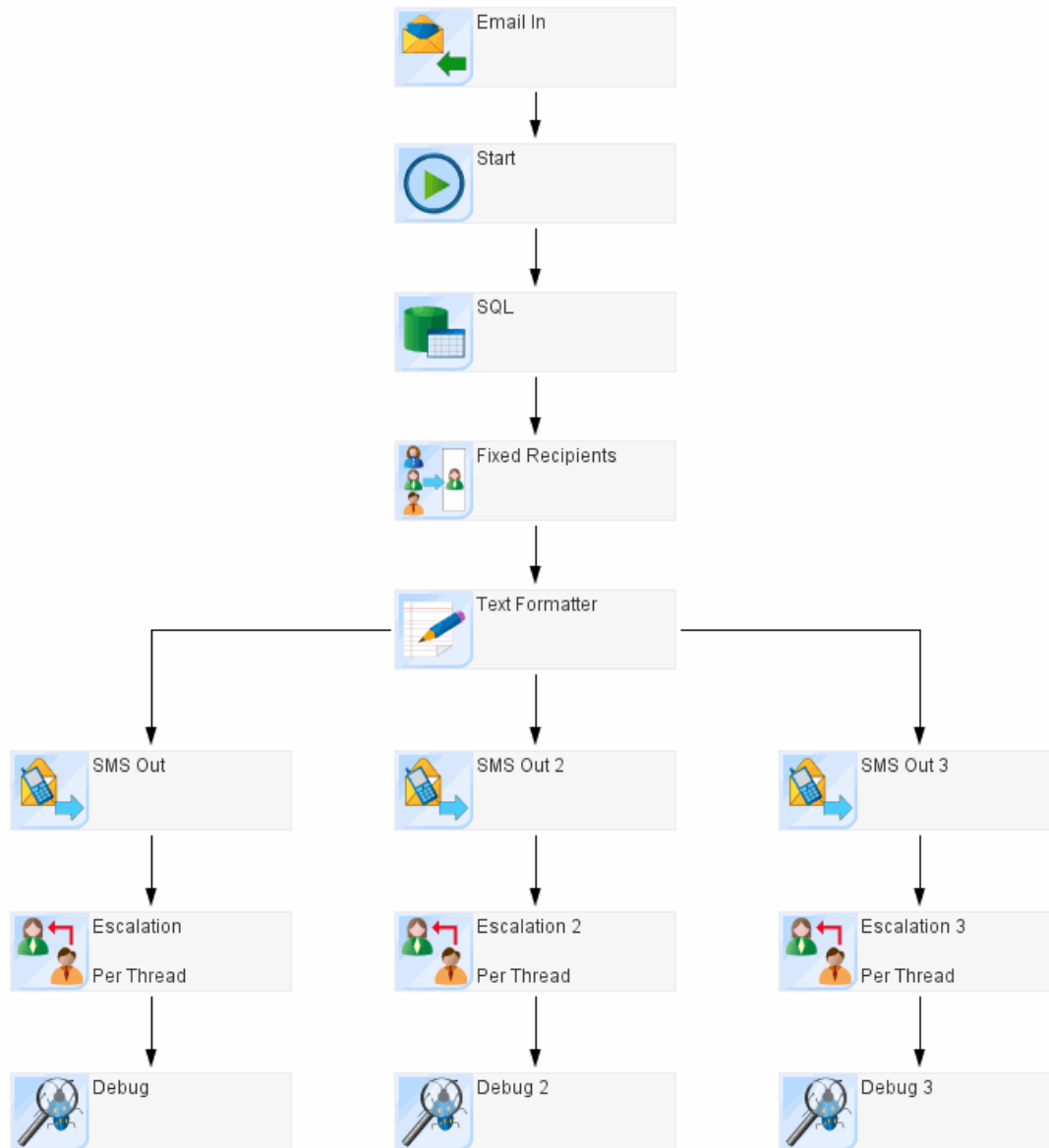
In this example, the recipients for SMS Output must *all* reply to the SMS message, for Escalation to be cancelled; the recipients for SMS Output (1) must *all* reply to the SMS message, for Escalation (1) to be cancelled; the recipients for SMS Output (2) must *all* reply to the SMS message, for Escalation (2) to be cancelled.



[How to use an Escalation Module](#)

Example of Per Recipient Cancellation

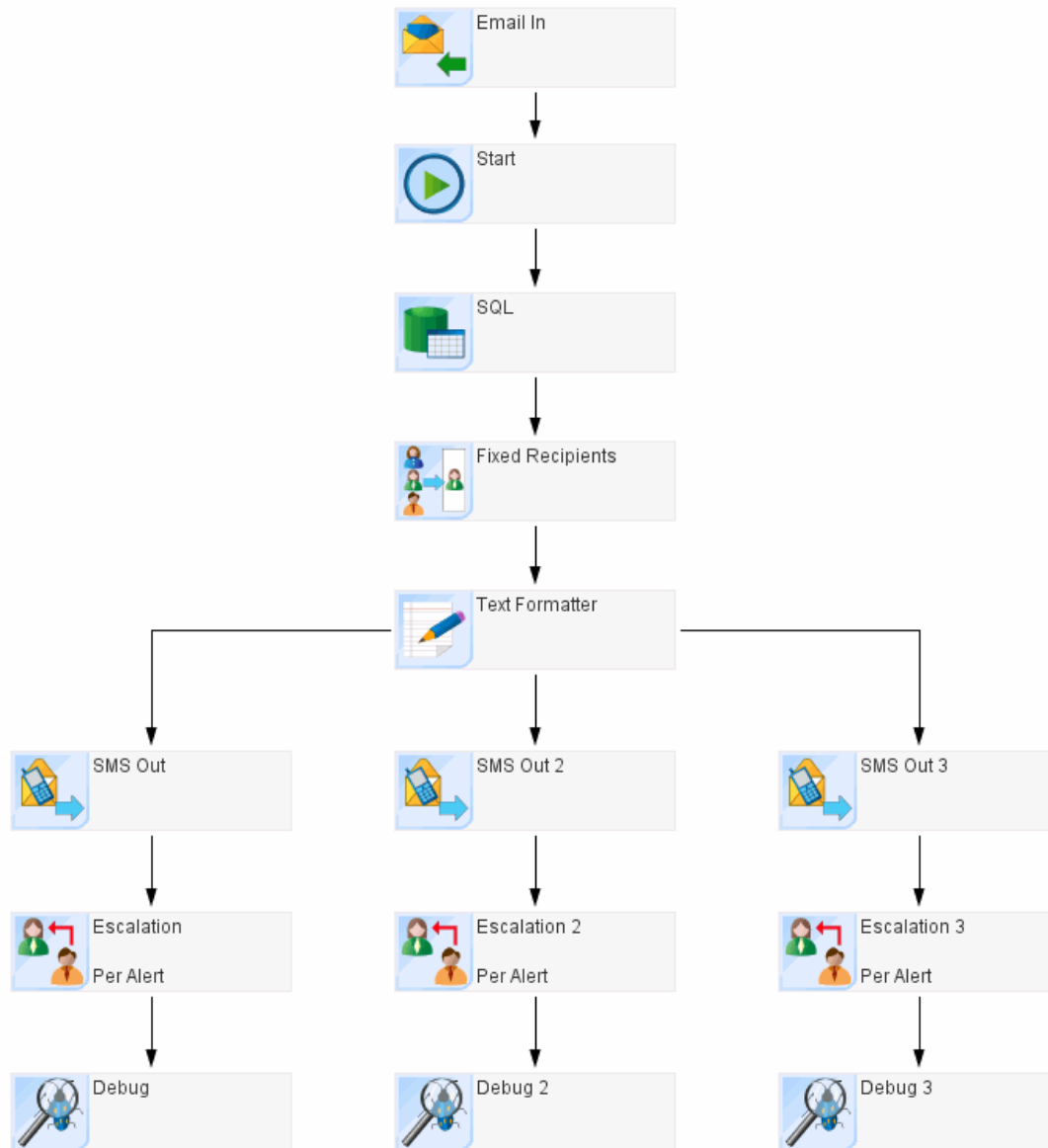
In this example, only one recipient for SMS Output must reply to the SMS message, for Escalation to be cancelled; only one recipient for SMS Output (1) must reply to the SMS message, for Escalation (1) to be cancelled; only one recipient for SMS Output (2) must reply to the SMS message, for Escalation (2) to be cancelled.



[How to use an Escalation Module](#)

Example of Per EMF Process Cancellation

In this example, only one recipient for SMS Output, SMS Output (1) or SMS Output (2) must reply to the SMS message for Escalation, Escalation (1) and Escalation (2) to be cancelled.



[How to use an Escalation Module](#)

How the Reply Module Works

You place an output module in an EMF Process, including in the message section the ESCALCODE dynamic function. Later in the EMF Process you place a Reply module and specify the timeout period in its Properties page.

Before running the EMF Process, you configure an Escalated EMF Processes Retriever to monitor for the termination of the timeout period. And you also configure a device listener to monitor for incoming responses from recipients.

When the EMF Process runs, the output module sends a message to the recipient. When a response is received, the device listener passes it to the [System EMF Process](#) (configured for that device listener), which typically includes a device router. The device router uses the escalation code to find the Reply module in the appropriate EMF Process, passing it the relevant data and message section. The processing of the EMF Process continues. If a response is not received before the waiting period times out, the Escalated EMF Process Retriever triggers the EMF Process to continue processing. You may want to insert a Halt Condition module after the Reply module to halt the flow of the EMF Process unless a reply was received (by testing to see if the **REPLY_DATA** data section is populated). Or you may want to perform further processing in a customized system EMF Process, without returning to the original EMF Process.

Click [here](#) to see a diagram illustrating the processing of an EMF Process with a Reply module and conditional processing after the Reply module. In this example, only one Conditional Halt module is used, although you could branch processing after the Reply module and insert a second Conditional Halt module.

[How to use the Reply module](#)

[Go to Start of Recipient Modules](#)

The REPLY_DATA Data Section

The router will pass all message and data sections to which it has access to the Reply module.

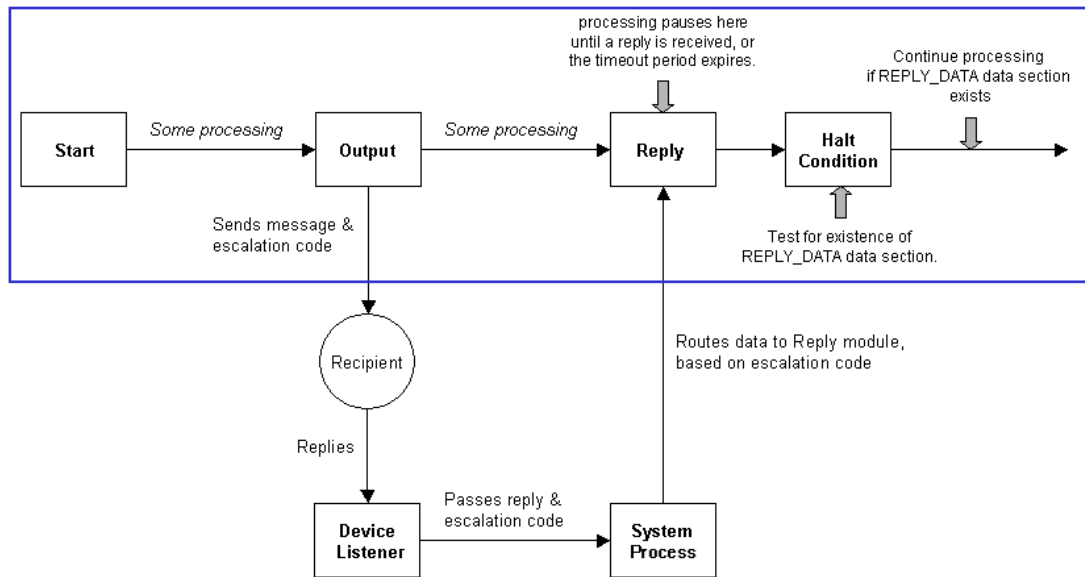
A router will have access to at least the **REPLY_DATA** data section (**note:** this section will not be present in the EMF Process if the reply has timed out). This data section contains the following two fields:

- **ListenerType** - the listener (POP3 or QUEUE) that received the reply
- **Timestamp** - time of the reply

Additionally, if the POP3 listener received the reply there will be a message section called **REPLY_MSG** that contains the received message. This text block is identical to that in the original message section for the incoming message, for example **POP3IN**.

[Reply Module](#)

Reply Module Diagram



Flow Modules

The **Flow** modules are used to influence the order and timing sequence of modules that are executed within an EMF Process. The following Flow modules are available:

- The [Conditional halt](#) module lets you define situations when an EMF Process should be stopped.
- The [Cascade](#) module allows EMF Processes to be chained together, transferring data from one EMF Process to the next in a master-slave relationship.
- The [Synchronization](#) module allows you to combine the results of two or more threads in an EMF Process.
- The [Delay](#) module allows you to pause the processing of a thread for a definite amount of time.
- The [Return](#) module is used in conjunction with the [Cascade](#) module so that a slave EMF Process can resume the master that initiated it and transfer data back to the master EMF Process.
- The [Loop](#) module defines the start/end point of a set of modules that should be looped/cycled/iterated over a number of times.
- The [Call](#) module allows a process to be selected to run as a sub-process.

In addition to the flow modules, you can insert [conditional links](#) to prevent or modify the flow of information between modules if certain conditions are met, and [remove redundant information](#) from EMF Process threads to improve efficiency.

Conditional Halt Module

The Halt Condition module allows you to stop the processing of an EMF process thread, depending on whether or not a specified section contains any information. This can be useful if you have two or more branches in an EMF process that have mutually exclusive purposes (or that are intended to perform identical purposes), or if you want to prevent an EMF process from running when a query returns no results.

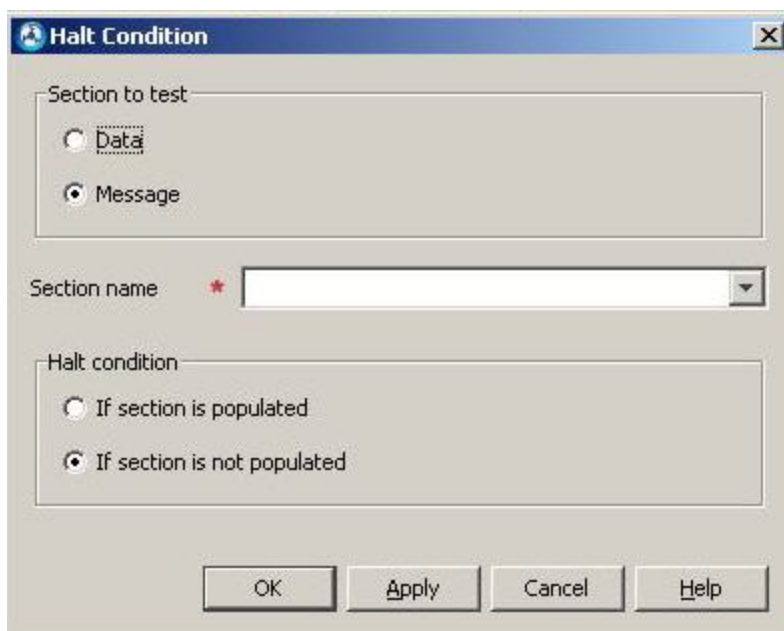
You can place any number of Halt Condition modules in an EMF process. If an EMF process consists of multiple branches, only the branch containing the Halt Condition will be stopped.

To use a Conditional Halt module:

1. Ensure that you have already created and configured all other modules within the EMF process.
2. Drag and drop a Halt Condition module into an EMF process after a data or message section, and create a link to it from another module.



3. Open the Halt Condition module to display its properties. By default, you have to double-click on the icon to open but you can configure mouse-click actions in the Options window. From the Tools menu, select **Options -> EMF -> Process Builder Behaviour** tab.



4. Select **Data** or **Message** as the **Section to test**, as required.

5. Select the required section from the **Section name** drop-down list, which will show all existing sections of the specified type.
6. Select the **Halt Condition**:
 - **If section is populated** - the EMF process will stop processing at this point if the section specified by **Section name** contains any data.
 - **If section is not populated** - the EMF process will stop processing at this point if the section specified by **Section name** is empty.

[Go to start of Flow Modules](#)

Cascade Module

You can use the **Cascade module** to chain EMF processes together. It allows one EMF process to cause another to run. It is important to understand the following terminology that is used by the cascade module:

- **Master** or **Parent** EMF process - the EMF process containing the Cascade module that fires another EMF process.
- **Slave** or **Child** EMF process - the EMF process fired by the Cascade module.

The child EMF process that is fired can either run independently or synchronously with the parent EMF process. If the child EMF process runs independently, then, as soon as the child EMF process has been queued for processing the parent EMF process will continue. If the child EMF process runs synchronously then the parent EMF process will wait for the child EMF process to reach a **Return module** before continuing (this is dependent on the settings set on the **Synchronization** tab).

Data can be passed from the parent to the child EMF process when running independently or synchronously. Data can be passed back from the child EMF process to the parent only when running synchronously.

The cascade module allows multiple instances of a child EMF process to be run dependent on the number of rows of data in a data section or the number of recipients. Each alert instance will be passed just a single row of data or a recipient. This is a powerful way of processing a block of data, giving similar functionality as iterating through a data/recipient set. It has large performance benefits over iteration, with all child EMF process instances being able to be processed simultaneously and distributed over multiple servers.

Important: The slave EMF process does not require an [Initiator Module](#) if it is only going to be fired by the master. If you want to be able to run the slave EMF process as a standalone EMF process you must include an initiator module as usual.

A Cascade module requires the existence of other EMF processes whose status is active.

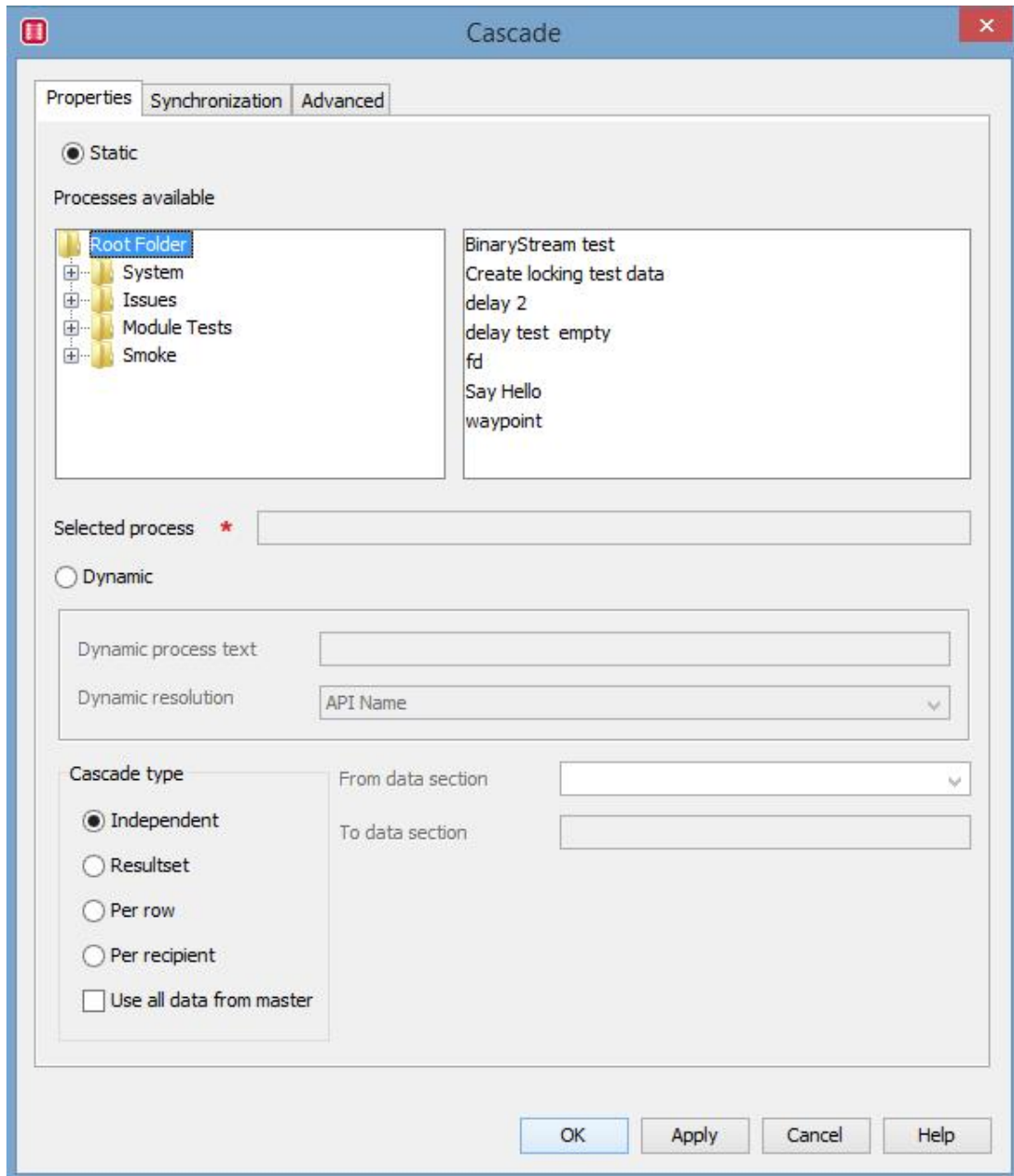
To use a Cascade module:

Note: The following details assume that you have already created and configured all other modules within the EMF process before adding the Cascade module. If you want to pass data to the slave EMF process, ensure that there are already Data modules in the EMF process thread before the Cascade module.

1. Drag and drop a **Cascade** module into an EMF process, and create a link to it from a module already in the EMF process. If a Master data section is to be passed to the slave, the Cascade module must be placed after a Data module.



2. Double-click on the Cascade module to display the **Cascade screen**.



The **EMF Processes available** panels show your EMF process folders (in the left-hand pane), and the EMF processes that they contain (in the right-hand pane).

3. Select **Static** if you want to specify the slave EMF Process, or **Dynamic** if you want to determine the slave EMF Process that should be run at runtime (using [dynamic functions](#)). Then:
 - **To specify a static EMF process**, navigate to the folder that contains the required EMF process in the left-hand **EMF Processes Available** pane, then select the required "slave" EMF process from the right-hand pane (the "slave" EMF process is the one that will be initiated by the cascade module). The full

path to the selected EMF process is shown in the **Selected process** field.

- **To determine the EMF process dynamically**, specify the slave EMF Process to run at runtime by either entering a literal string or a [dynamic function](#) (or a combination of both) in the **Dynamic EMF Process text** field, then selecting the required **Dynamic resolution** to specify whether the text in the Dynamic EMF Process field is an EMF Process ID, an EMF Process API name, or a **Relative path** to the EMF Process (i.e. specifying the location and name of the target EMF Process using a path from the source - e.g. `..\Voice\My EMF Process`).

Warning: If you have an HTTP in module and you want to make use of the **HTTPIN_PARAMS** data section, do not use the default index-based settings for the dynamic function. Ensure that you explicitly name the parameters. For example, if your parameter name is AlertID, instead of `$DATA('HTTPIN_PARAMS',0,0)$`, use `$DATA('HTTPIN_PARAMS','AlertID',0)$`

4. Select a **Cascade Type** in order to define the data that will be sent from the master to the slave. There are four options available:
 - **Independent** - the source EMF process will not pass any data sections to the slave EMF process and a single instance of the slave is fired. No further information is required.
 - **Resultset** - the source EMF process will pass the contents of either a single data section, or all data sections, to a data section in the slave EMF process and a single instance of the slave EMF process is fired. Select the required source data section from the **From data section** list, and enter a name for the destination section (in the slave EMF process) in the **To data section** field. Alternatively you can select **Use all data from master** to pass the contents of all the data modules.
 - **Per Row** - the source EMF process will pass individual rows in the specified **From data section** individually to the specified **To data section** in the slave EMF process, and fire a new instance of the EMF process for each populated row of data that is received.
 - **Per recipient** - all data sections are passed, and a new instance of the slave EMF process will be fired for each of the recipients in the source EMF process. The initial EMF process state of the slave EMF process will contain a single recipient section containing the appropriate recipient. If you select **Use all data from master**, all the data sections will be included as well.

Note: Selecting **Use all data from master** means that the slave EMF process will inherit *all* data (data sections, message sections, error sections, etc.) from the EMF process state of the parent just prior to the cascade.

5. Select the [Synchronization tab](#) to specify whether the child EMF process runs independently or synchronously with the parent EMF process, and if the parent runs synchronously - how data is merged from the child EMF process back into the parent EMF process.
6. Select the **Advanced** tab to set the advanced features in the Cascade module. All the Cascade modules will use the default queue to run the cascaded process. However, if

there are multiple queues set up to spread the workload, or a dedicated queue for the Cascade module, it can be selected in the **Queue to run the cascaded process(es)** on drop-down menu. The available queues are defined in the System Queues.

Note: The queue must have a [server manager](#) associated with it, else the process placed on the queue will not be run.

[Return Module](#)

[Go to start of Flow Modules](#)

[Example of Using the Cascade Module with HTTP](#)

Synchronization Module

Synchronization allows you to combine the outputs of two or more EMF Process modules into a single module for output, or for further processing. This allows multiple SQL queries to be carried out simultaneously. Synchronization can reduce the overall time taken for an EMF Process to run, especially if one or more of the modules require user input.

Important: Synchronization is a complex process that may produce undesirable results, if used in unstable or complex EMF processes, or if you set inappropriate values for certain options.

How Synchronization works

The Synchronization module takes EMF Processes and stores them in the repository, and when all EMF Process fragments have been received, they are synchronized. A fragment from a defined primary branch is used as the basis for the synchronization, and then sections from one or more secondary branches are synchronized with it. (You can choose whether to include or exclude data, message, recipient and error sections from the secondary branches). The synchronization creates a Data or Message section, and when the synchronization is complete (or when a specified timeout period expires), the EMF Process will continue - you can split the output from the synchronization if you wish.

Note: The resulting Data and Message sections will not be created if there is already an existing section of the same name (even if it is empty).

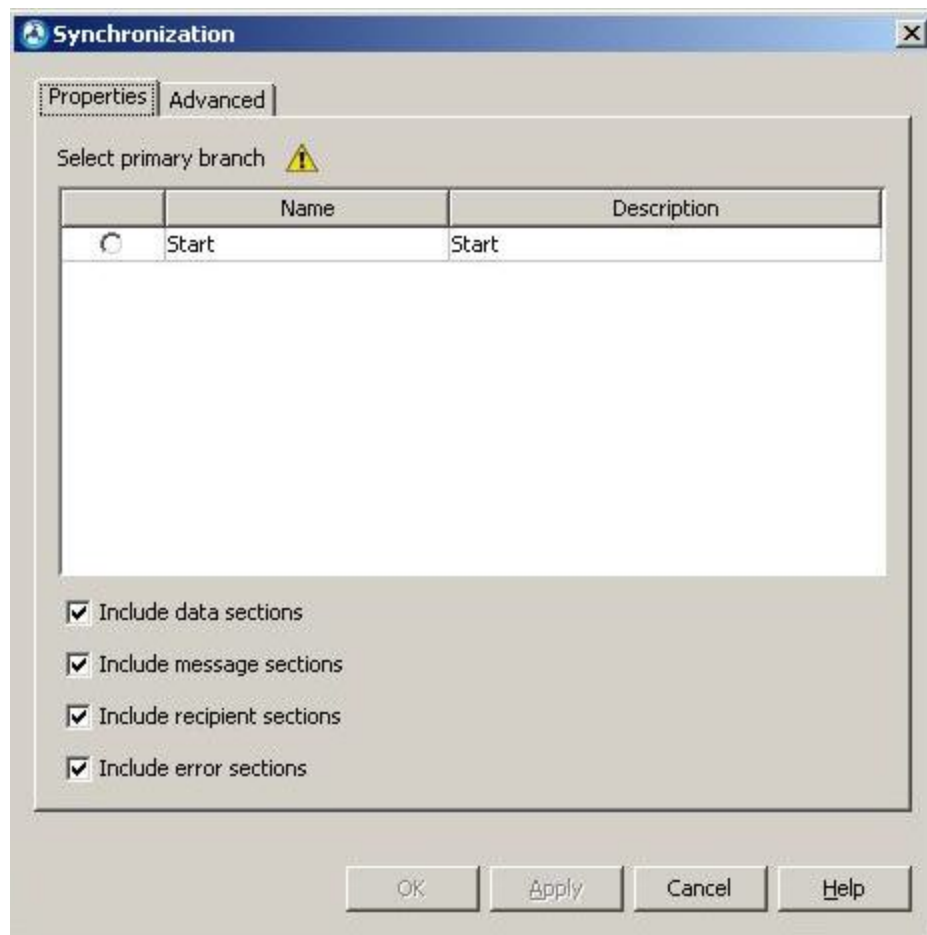
To use a Synchronization module:

1. Drag and drop a Synchronization module icon into an EMF Process and create a link to it from one or more modules that are already in the EMF Process instance.



You can place as many EMF Process synchronization modules as you wish in a single EMF Process instance. However, synchronization can be very processor- and database-intensive, and you should weigh up the benefits carefully. A complex synchronization, or one that has a high timeout period, might well be slower than running the individual modules consecutively.

2. Double-click the Synchronization icon to display the **Synchronization Properties screen**.



3. In the **Select** primary branch pane, select the required "primary" branch from the list of all the modules that are included in the synchronization. All the data, message, recipient and error sections for this branch will automatically be used, and the other modules will then be treated as modifiers of that primary branch.

Data and Message sections from the other branches are then added to the primary branch. If the existing section in the primary branch is empty it will be populated by data from the other branches.

Important: If a section in a secondary branch has the same name as one in the primary branch it will not be used. You should bear this in mind when naming sections.

- Although all the sections in the primary branch are always used in the synchronization, you can choose whether or not to use the equivalent sections from the other branches. Normally all sections are included, but you may be able to speed up the synchronization by excluding one or more. To exclude a particular section, select its tab and then click to deselect the **Include (other) sections:** check box.

Note: You can further refine the sections and branches that will be included in the synchronization by using [conditional links](#).

- Select the [Advanced](#) tab to set timeout and other options.

[Troubleshooting Synchronization](#)

Delay Module

You can use the EMF Delay module to pause the processing of an EMF Process between two consecutive (or close) modules for a given amount of time before continuing.

For example, it could be placed between two output modules in order to allow the recipients to respond to messages, or allow them to be delivered successfully - e.g. if an SMS Out module sends messages to 20 recipients but network problems prevent some of them from being delivered, a delay of ten minutes would allow that information to be fed back into the EMF Process. A second output module (e.g. email), placed after the Delay module, would then be able to send a different type of message to only those that could not receive the original one.

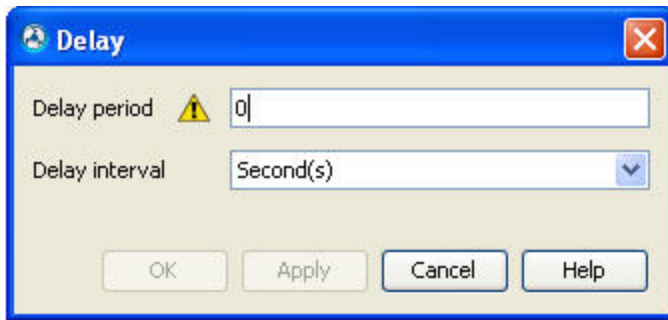
Important: In order to use a Delay module you must ensure that the [Escalated EMF Processes Retriever](#) is running.

To use a Delay module:

- Drag and drop a Delay module into the EMF Process at the appropriate place.



2. Double-click the icon to display the **Delay Properties** screen.



3. Set the required delay using the controls: **Delay period** is the number of time units that you want to delay the EMF Process by, and **Delay interval** defines the units - i.e. seconds, minutes, hours, days, weeks or months. So, for example, to delay the module by half an hour you would select **30** as the **Delay period** and **Minutes** as the **Delay interval**.

Note: **Delay Period** field accepts Dynamic Function expressions so the delay period can be set dynamically at runtime.

[Go to Start of Flow Modules](#)

Return Module

The **Return module** is used in EMF Processes that have been cascaded to from a parent EMF Process by the **Cascade module**. When the **Return module** is processed it will flag the **Cascade module** in the parent EMF Process that the child EMF Process has been processed. Depending on the options selected in the Cascade module, this may cause the parent EMF Process to resume, or wait for other child EMF Processes to also be processed. The processing of the current EMF Process will continue, irrespective of the options selected in the parent EMF Process.

If the running EMF Process contains a **Return module** and it was not initiated by a cascade from another EMF Process then the module has no effect. An EMF process can contain multiple **Return modules** - the first return module processed will flag any waiting parent EMF process that it has been processed, all other **Return modules** for that EMF Process instance will have no effect.

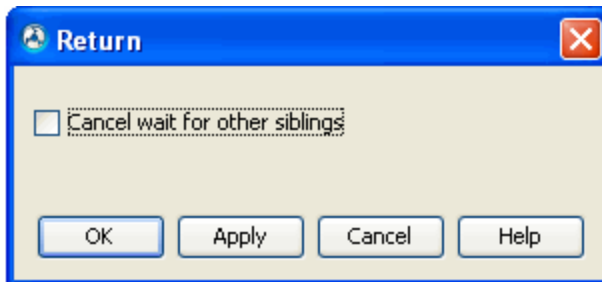
Typically a **Return module** appears as the last module in an EMF process. This however isn't a requirement. The **Return module** can appear anywhere in the EMF Process. Once processed, it will flag any waiting parent EMF Process which may then continue, depending on the options selected. The child EMF process with the **Return module** will continue its processing moving to the next module in the process branch.

To use a Return module:

1. Drag and drop a Return module icon into an EMF process. Create a link to it from a module already in the EMF process instance.



2. Double-click the Return icon to display the **Return Properties** screen.



3. Select **Cancel wait for other siblings**, if, once this module is processed and the parent cascade caused multiple child EMF Processes to start, then the parent EMF Process should continue and not wait for any of the other child EMF Processes. The **Return modules** will be ignored when reached in the other child EMF Processes.
4. The parent EMF Process is notified that it is being resumed by cancelling a child process, and this information can be used to take some alternative processing action. To detect this in the parent alert, check the value of the field "ReturnCancelled" in the **Returned Data Section** of the cascade module.

Tip: **Cancel wait for other siblings** would typically be used when a special condition has occurred in the processing of the EMF Process and some other action is needed, for example, an order has been cancelled. If the parent EMF Process should **always** continue after the first **Return module** is processed then select the option **Wait for first cascaded alert before continuing** in the **Cascade module** synchronization options.

[Go to Start of Flow Modules](#)

[Cascade Module](#)

Loop Module

The Loop module is used to perform iterative tasks in EMF processes. It defines the start/end point of a set of modules that should be processed in a loop/cycle/iteration.

This enables a set of modules that follow the Loop module to be processed a pre-determined number of times. For example, it is possible to loop through each row in a data section - the modules following the loop module could create and send a message related to the current row of that loop's iteration. This way a unique message is sent for each row in the data section.

It is an alternative to the cascade module which may set off many EMF processes to be processed in parallel. Instead the Loop module allows a data section or recipient section to be stepped through sequentially.

There are 5 different ways to set the criteria for the number of loop iterations:

- A **data section** - The loop module will iterate through each row of the data section. On each iteration around the loop, the current row of data is accessible to the EMF process via the \$LOOPDATA\$ dynamic function.
- The **recipients** - The loop module will iterate through each recipient in the current alert state. On each iteration around the loop, the current row recipient information is accessible to the EMF process via the \$LOOPDATA\$ dynamic function.
- A **predefined number** - The loop module will iterate the number of times specified (this number may be calculated at runtime using dynamic functions).
- **XPath** - The loop module will iterate over a section of an XML document identified by an **XPath expression**.
- **JSON** - The loop module will iterate over a JSON array identified by a JSON pointer expression.

A Loop module must have at least one link coming out, and a maximum of two out links. One link represents the path that will be followed when there are more items to iterate through. The second optional link is the path that is taken when there are no more items to iterate through.

There must be at least two other modules in a loop. Due to restrictions in the EMF processes builder, a loop module cannot be linked to another module and then immediately linked back to the loop module again. It must go to at least one other module first. This other module can be a 'dummy' module that performs no action if required (for example, a text formatter).

Breaking out of a loop - a loop can be used to search through a data section for a particular piece of data or to iterate through an operation a maximum number of times. However, at some point when the criteria are met, it may be necessary to break out of the loop. This can be achieved by adding a conditional link and selecting the **Loop is exiting if this branch is being taken** check box.

Data for a particular iteration of a loop can be accessed using the \$LOOPDATA\$ dynamic function. To find out what iteration the loop is currently on, use the \$LOOPITERATION\$ dynamic function.

Loops can be embedded. So, it is possible to have one loop inside another loop.

To use a Loop module:

1. Drag and drop a Loop module icon into an EMF process and create a link to it from a module that is already in the EMF process instance.



2. Open the Loop icon to display the Loop properties screen. By default, you have to double-click on the icon to open but you can configure mouse-click actions in the Options window. From the Tools menu, select **Options** -> **EMF** -> **Process Builder Behaviour** tab.

3. Enter the **Loop name**. This is the name used by LOOPDATA and LOOPITERATION dynamic functions to retrieve the correct data. Change this only when you are using embedded loops (one loop inside another loop). In this case the loop names must be different.
4. Select the source of data that is going to be iterated over. This is either:
 - **Datasection** - The loop module will iterate through each row of the selected data section. On each iteration around the loop, the current row of data is accessible to the EMF process via the \$LOOPDATA\$ dynamic function.
 - **Per Recipient** - The loop module will iterate through each recipient in the current alert state. On each iteration around the loop, the current row recipient information is accessible to the EMF process via the \$LOOPDATA\$ dynamic function.

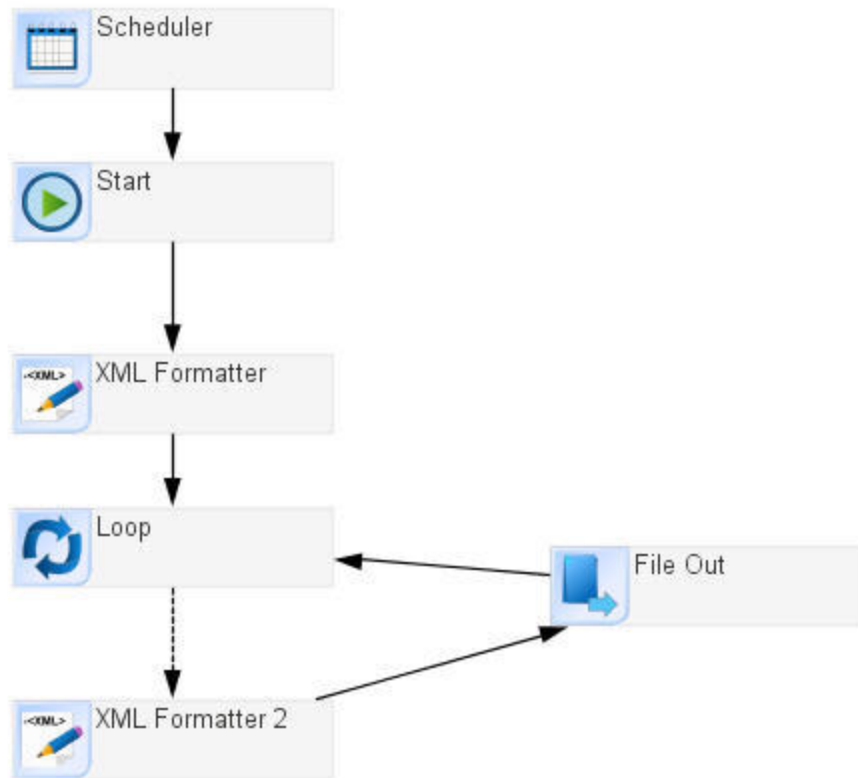
- **Count** - The loop module will iterate the number of times specified. A dynamic function may be entered as the expression.
- **XPath**- The loop module will iterate over a section of an XML document identified by an **XPath expression**. Select the **Message section** that contains valid XML. The **XPath expression** entered should evaluate against this document and may return multiple results. The loop module will loop once for each result returned. On each iteration, the \$LOOPDATA('Loop',0)\$ dynamic function can be used to return the result that the **XPath expression** has evaluated to on that iteration.

XML namespaces definitions can be entered for use by the **XPath expression** to reference uniquely named elements and attributes in the XML document. An XML instance may contain element or attribute names from more than one XML vocabulary.

- **JSON** (JavaScript Object Notation) - The loop module will iterate over a JSON document identified by a **JSON pointer expression**. Select the **Message section** that contains the JSON document. The **JSON pointer expression** entered must reference an array within this document. The loop module will loop once for each item in the JSON array. On each iteration, the \$LOOPDATA('Loop', "\$")\$ dynamic function can be used to return the resultant value for that item in the array or another JSON pointer expression can be entered in the \$LOOPDATA\$ dynamic function to reference child elements. See [LOOPDATA](#) for more information.
5. Select **Create equivalent data section with loop iteration information** to create a data section with the same name as the Loop name on each iteration of the loop. The data section will contain a single row of data containing the same information that can be retrieved using the LOOPDATA dynamic function, i.e. the data for the current loop iteration. This option is useful if you want to use other modules or dynamic functions that rely on the data being in a data section. For example, you could cascade to another process with the loop data, or use HTMLTABLE to output only the data of a particular iteration of a loop.
 6. Now other modules that are to be part of the loop should be added. The final module of the loop should link back to the Loop module.

Example of iterating over an XML document using XPath

1. Drag and drop modules as shown below:



2. Open the XML Formatter module and enter the following data:

```

<?xml version="1.0"?>
<Orders>
  <Order>
    <ID>1111</ID>
    <Title>Java</Title>
  </Order>
  <Order>
    <ID>2111</ID>
    <Title>Perl</Title>
  </Order>
  <Order>
    <ID>1141</ID>
    <Title>SQL</Title>
  </Order>
</Orders>

```

3. Click **OK** to save and close.
4. In the Loop module properties screen, leave the loop name as 'Loop' and select XPath as the **Loop type**.
5. On the XPath tab, select the **Message section** created by the first XML formatter and enter //Order as the **XPath expression**.
6. In the second XML formatter, enter the following dynamic function:
\$LOOPDATA('Loop',0)\$

7. In the File Out module, enter a path for the **Output file** and select Use incremental filenames. Select the message section created in the second XML formatter for output.
8. Run the process.

Three separate XML files are created. For each iteration, the Loop module generates an XML file containing the ID and Title details, as shown below:

Iteration 1

```
<?xml version="1.0" encoding="UTF-8" ?>
<Order>
<ID>1111</ID>
<Title>Java</Title>
</Order>
```

Iteration 2

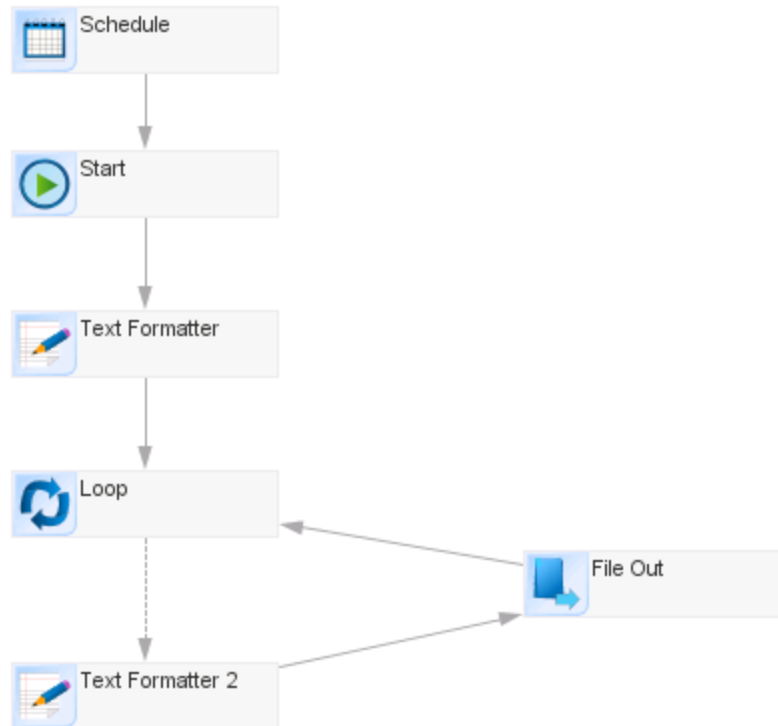
```
<?xml version="1.0" encoding="UTF-8" ?>
<Order>
<ID>2111</ID>
<Title>Perl</Title>
</Order>
```

Iteration 3

```
<?xml version="1.0" encoding="UTF-8" ?>
<Order>
<ID>1141</ID>
<Title>SQL</Title>
</Order>
```

Example 1 of iterating over a JSON document

1. Drag and drop modules as shown below:



- Open the Text Formatter module and enter the following data:

```
{
  "employees": [
    {
      "firstName": "John",
      "lastName": "Doe"
    },
    {
      "firstName": "Anna",
      "lastName": "Smith"
    },
    {
      "firstName": "Peter",
      "lastName": "Jones"
    }
  ]
}
```

- Click **OK** to save and close.
- In the Loop module properties screen, leave the loop name as 'Loop' and select JSON as the **Loop type**.
- On the JSON tab, select the **Message section** created by the first Text formatter and enter /employees as the **JSON pointer expression**.
- In the second Text formatter, enter the following dynamic function:
\$LOOPDATA('Loop','/firstName')\$
- In the File Out module, enter a path for the **Output file** and select Use incremental filenames. Select the message section created in the second Text formatter for output.
- Run the process.

The Loop module generates text files containing the firstName, as shown below:

Iteration 1

John

Iteration 2

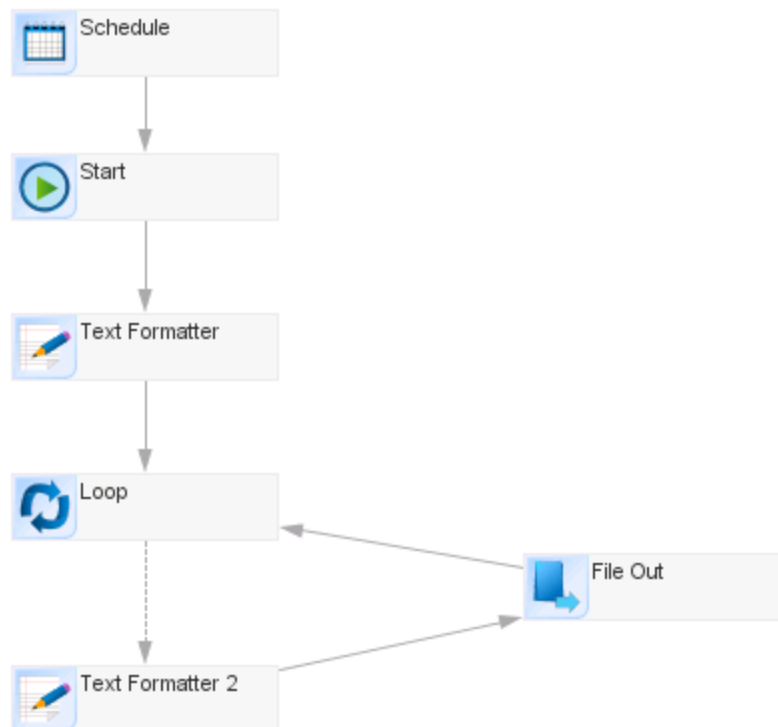
Anna

Iteration 3

Peter

Example 2 of iterating over a JSON document

1. Drag and drop modules as shown below:



2. Open the Text Formatter module and enter the following data:

[123,456,789]

3. Click **OK** to save and close.
4. In the Loop module properties screen, leave the loop name as 'Loop' and select JSON as the **Loop type**.
5. On the JSON tab, select the **Message section** created by the first Text formatter. The **JSON pointer expression** to enter in the loop module will be empty (referencing the root of the JSON document).
6. In the second Text formatter, enter the dynamic function `$LOOPDATA('Loop', '')$` to return the whole document.

7. In the File Out module, enter a path for the **Output file** and select Use incremental filenames. Select the message section created in the second Text formatter for output.
8. Run the process.

The Loop module generates text files as shown below:

```
Iteration 1
123

Iteration 2
456

Iteration 3
789
```

[Go to Start of Flow Modules](#)

Call Module

The Call module allows a process to be selected to run as a sub-process.

Data/message/recipient sections can be passed from the parent (calling) to the child (process interface/sub-process) process. The child process can only be run synchronously. This module works in a similar way to the Cascade module when using the sync/return options.

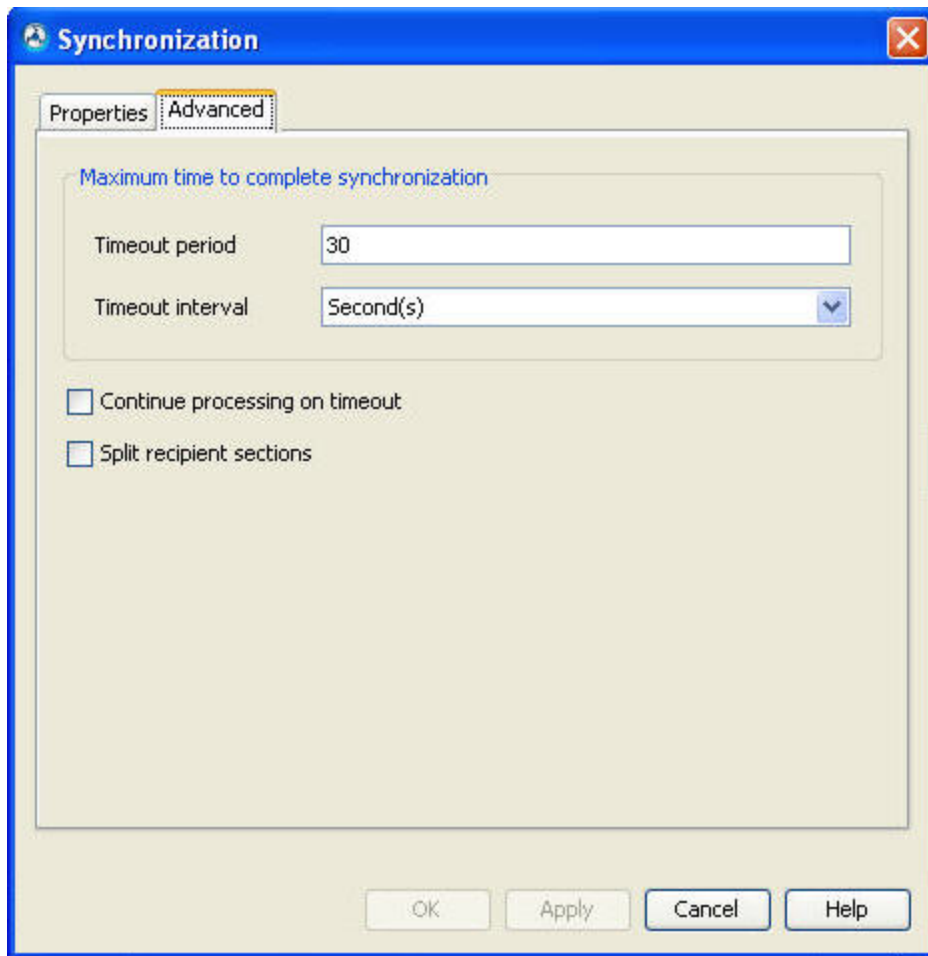


For more information about the Call module, see [Process re-use with Process Interface and Call module](#).

Advanced Synchronization Options

The **Advanced** Synchronization options allow you to specify how long the Synchronizer should wait for all the required modules to supply the required information before proceeding with the synchronization, and what to do if some modules do not respond within this period. You

can also specify whether or not to separate the individual recipient sections back into individual alerts after the synchronization.



- **Maximum time to complete synchronization:** allows you to specify how long the synchronization should wait for input from all the modules before proceeding. If, at the end of this period of time, one or more modules has not delivered the required information, the synchronization will cease (unless you enable the "Continue processing on timeout" option - see below). The timeout period starts when the first module reaches the synchronizer.
- **Continue processing on timeout:** forces the synchronization to continue after the timeout period, even if one or more modules has not delivered the required information.

Important: As a general rule, you should **not** enable this option, because it could result in misleading information as a result of a synchronization having been performed on an incomplete set of data. Also, if you enable this option and the EMF Process fails while running, an entry is added to the "SyncPending" table in the EMF repository. This table is polled by the Synchronized EMF Process Retriever, and if it has many entries system performance can be adversely affected. If necessary, you can clear the entries in the SyncPending table manually by using a database query tool to run the SQL query `DELETE FROM SyncPending` against the repository.

- **Split recipient sections:** allows you to specify whether or not the recipient sections should be split following synchronization in order to increase performance. Note that you cannot specify the size of the split; this is defined by the relevant recipient module. This option is normally not enabled.

Note: You may be able to reduce the system resource overhead by altering the frequency with which the synchronizer checks for completed modules. See [Setting the Synchronizer polling interval](#)

[The Synchronization Module](#)

[Troubleshooting Synchronization](#)

Process Re-use with Process Interface and Call module

Process Interfaces makes the building of reusable components easier. The Process Interface has two parts - the Process Interface process and the Calling process. A simple use of process interfaces would be to have some processing done (e.g. discounts calculated, total per item, reduce stock levels using quantity ordered) on order line items for an order - that is, each order line item could be looped over and the same processing done for each item within the loop, with the order line item, product, quantity, and amount, for example, changing per item. The repetitive part of the processing would be done by the process interface process, using the calling process to go through the master data.

Defining the Process Interface

The process interface is defined on the Start Module. It is accessed by editing the properties of the Start Module of the Process that will be used as a re-usable component. Since there is only one Start Module per process, there can only be one interface designed per process. The process containing the Process Interface can also be referred to as a sub-process.

Using the Process Interface

The Call Module uses the Process Interface defined on the selected process (within the Call Module), to pass any data/message/recipient sections and jdbc/sap connections/services to the sub-process and to store the parameter values returned from the sub-process.

Process Completion

The Process Interface requires a Return module to indicate the completion of the process, so that control can be returned back to the calling process, which can then continue with its processing. The Return module's **Cancel wait for other siblings** has no effect when used in this context (i.e. with the Call module).

Adding Process Interface parameters

If a message/data section/entity parameter is added to an existing Process Interface definition, then it is automatically added to the Call Module which uses the process of Process Interface, and will use the defaults as defined in the Process Interface parameter. For data sections, it will default to Merge data, and message sections will default to Text/Message depending on the type.

To define the Process Interface:

1. Edit the properties of the Start Module to bring up the Start Module **Process Interface Sections** screen.



Start

Process Interface Sections | Process Interface Entities | Advanced Run Control

Message sections passed in and out of this process

Name	Type	Description	Default value	Sample
message	IN			Show sample
message	IN/OUT			Show sample
ou	OUT			Show sample

Add
Remove
Remove all
Move up
Move down

Data sections passed in and out of this process

Name	Type	Description	Default value	Sample
d1	IN		SQL	Show sample
d2	IN		d2	Show sample
d3	IN		ds3	Show sample

Add
Remove
Remove all
Move up
Move down

Recipient section passed in and out of this process

☐ IN ☒ IN/OUT ☐ OUT ☐ Ignore

Description: email recipients

Icon/Label to assign to calling module

Default module name: email invoice

Import graphic to use as module icon Reset graphic

OK Apply Cancel Help

2. Define the parameters that will be used to pass the **Message sections** to be used by this process or return the message sections back to the calling process. The following columns are available within the **Message sections passed in and out of this process** section.
 - **Name** - Name of the message section that will be referenced within this process. If the message section is of OUT type, then it is the name of the message section that is used in the calling process.
 - **Type** - A drop down list that contains three possible choices:
 - **IN** - This message section will be passed into this process from the calling process.
 - **IN/OUT** - This message section will be passed into this process from the calling process. When this process completes, the message section is returned back to the calling process and the contents of this message section will replace the contents of the original message section in the calling process.

- **OUT** - A message section of this name will be created during the running of this process and it will be copied into the calling process when this process completes, overwriting any that may exist with the same name.
- **Description** - A textual description of the purpose of this message section, indicating what it should contain/returns.
- **Default value** - The default message section/text that will be used in the calling process, for the defined message section parameter, if the **Use Default** option is selected in the calling process.
- **Sample** - This column allows the user to define a sample message section. The sample is used with the test buttons on the Modules. Click the **Show sample** button to open the **Sample Preview** dialog box to allow larger text to be written for the sample.

Add/Remove/Remove All buttons allow you to add/remove the selected message section parameter.

Move up/Move down buttons allow you to move the selected message section parameter up or down in the list. Note that the parameters defined here will appear in the same order within the Call module of the calling process.

Note: After adding a new message/data section to a start module, to make the new sample sections available in the process builder to modules that call this process, the calling module must be opened and changes must be applied.

3. Define the parameters that will be used to pass in the **Data sections** to be used by this process or return the data sections back to the calling process. The following columns are available within the **Data sections passed in and out of this process** section.
 - **Name** - Name of the data section that will be referenced within this process. If the data section is of Out type, then it is the name of the data section that is used in the calling process.
 - **Type** - A drop-down list that contains three possible choices:
 - **IN** - This data section will be passed into this process from the calling process.
 - **IN/OUT** - This data section will be passed into this process from the calling process. When this process completes, the contents of this data section will replace/merge/merge without duplicates with the contents of the original data section in the calling process, depending on the option selected in the Call module.
 - **OUT** - A data section of this name will be created during the running of this process and it will be copied into the calling process when this process completes, replacing/merging/merging without duplicates with the contents of the original data section in the calling process, depending on the option selected in the Call module.
 - **Description** - A textual description of the purpose of this data section, indicating what it should contain/returns.

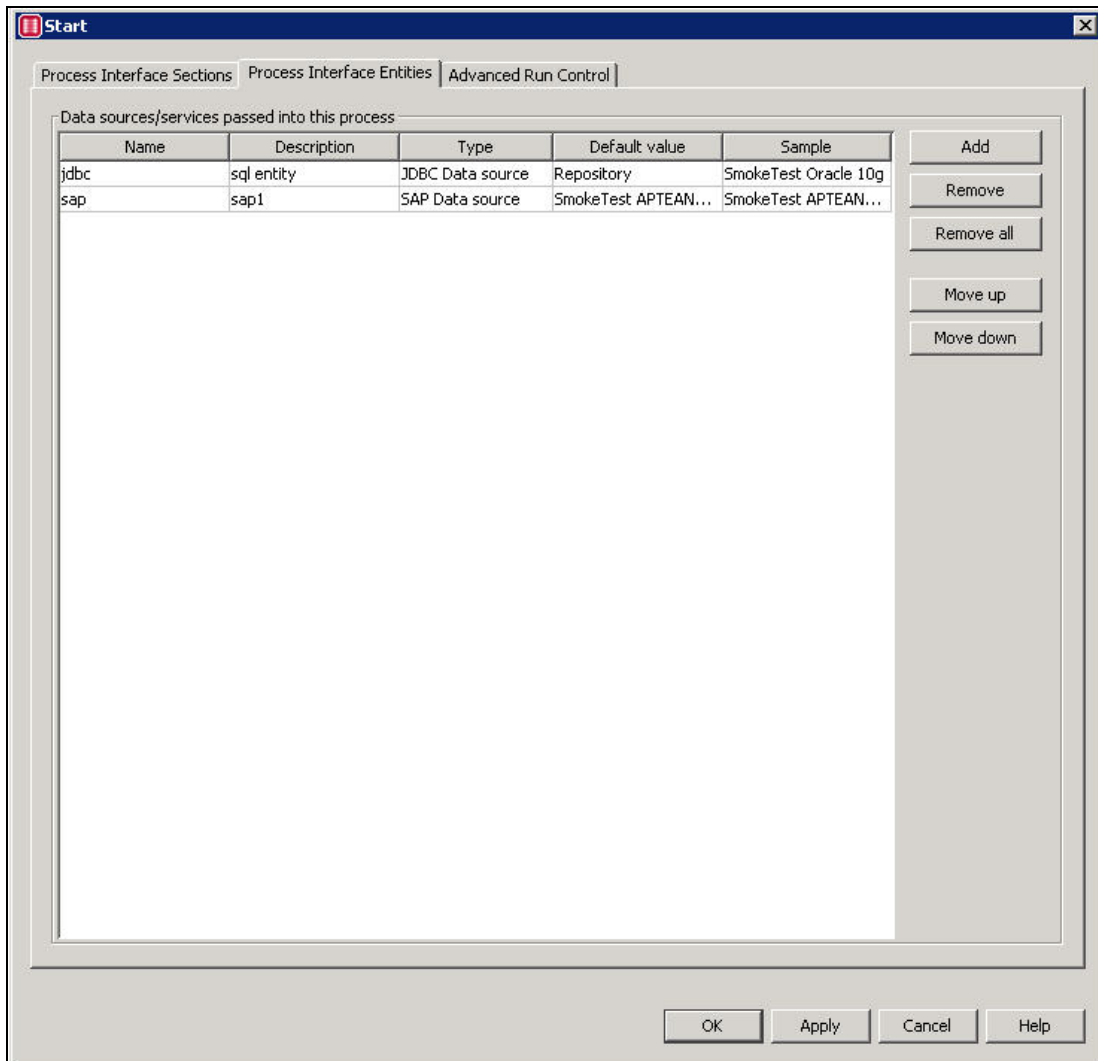
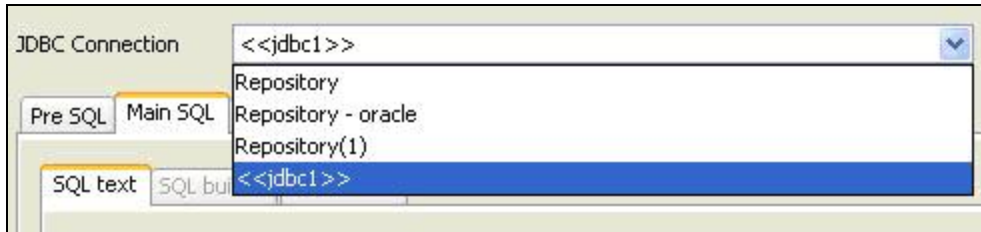
- **Default value** - The default data section that will be used in the calling process, for the defined data section parameter, if the Use Default option is selected in the calling process.
 - **Sample** - This column allows the user to define a sample data section. The sample is used with the test buttons on the Modules. Click the **Show sample** button to open the **Sample Preview** dialog box. Sample data copied from any of the modules capable of creating data sections can be pasted here.
4. The **Recipient section passed in and out of this process** section is used in a way similar to the data/message sections above. There is only one recipient section available. The following can be defined on it:
- **Description** - A textual description of the purpose of this recipient section.
 - **Type**: Options available are:
 - **IN** - Recipients in the calling process will be passed into this process.
 - **IN/OUT** - Recipients of the calling process will be passed into this process. When this process completes (see Process completion), the recipients will be copied back to the calling process. The exact way the recipients are copied back depends on the options selected in the Call module (replace/merge/merge no duplicates)
 - **OUT** - When this process completes (see Process completion), the recipients will be copied back to the calling process. The exact way the recipients are copied back depends on the options selected in the Call module (replace/merge/merge no duplicates).
 - **Ignore** - No copying of recipients happens between the calling process and the called process.
5. The **Icon/Label to assign to calling module** section is used to define how the module will look in the process builder when selected in the Call Module. It is possible to set:
- **Default icon** - this will change the default Call module icon in the process builder. Use **Import graphic to use as module icon** to select the graphic file to use. Two formats are supported, SVG and PNG. PNGs are not scalable, so it is recommended to use SVG files. Once selected, a preview of the graphic is shown. The **Reset graphic** will blank out the currently selected graphic.
 - **Default module name** - This is the default text the Call module will display in the process builder when this sub process is selected (as shown below) for a Call module.



The Process Interface Entities Tab

The **Process Interface Entities** tab is used to define parameters that will pass entities

from one process to another. These entities (also known as virtual entities) can be selected within the process wherever the entity type normally appears. E.g. JDBC Datasource defined here will be selectable in the SQL Module, SAP Databrowser batch insert, and SAP Authorization module. The entity will be shown as <<entity name>> (i.e. with << >> (angled brackets) enclosing the name) to differentiate it from the normally defined entity. This entity will not be visible in the services/jdbc/sap connections node in the tree view, as it is specific to the process within which it has been defined.



- **Name** - Name of the entity that will be referenced within this process.
- **Description** - This is a textual description of the purpose of this entities section, indicating what the entity is for.
- **Type** - This can be one of the following types: SAP Connections, JDBC Connections, FTP Services, SMS Services, File Services, Executable Services, SMTP Services, JNDI Services, HTTP Services and SNMP Services.
- **Default value** - This is the default entity that will be used in the calling process, for the defined entity parameter, if the **Use Default** option is selected in the calling process. The default value can only be selected from the drop down list of the available entities (as created in the EMF services/jdbc/sap connection node).
- **Sample** - This column allows the user to select a sample entity. The sample is used in this process when the **Test** button is pressed or the SQL/SAP connection is established. The sample entity can only be selected from the drop down list of the available entities.

The Call Module

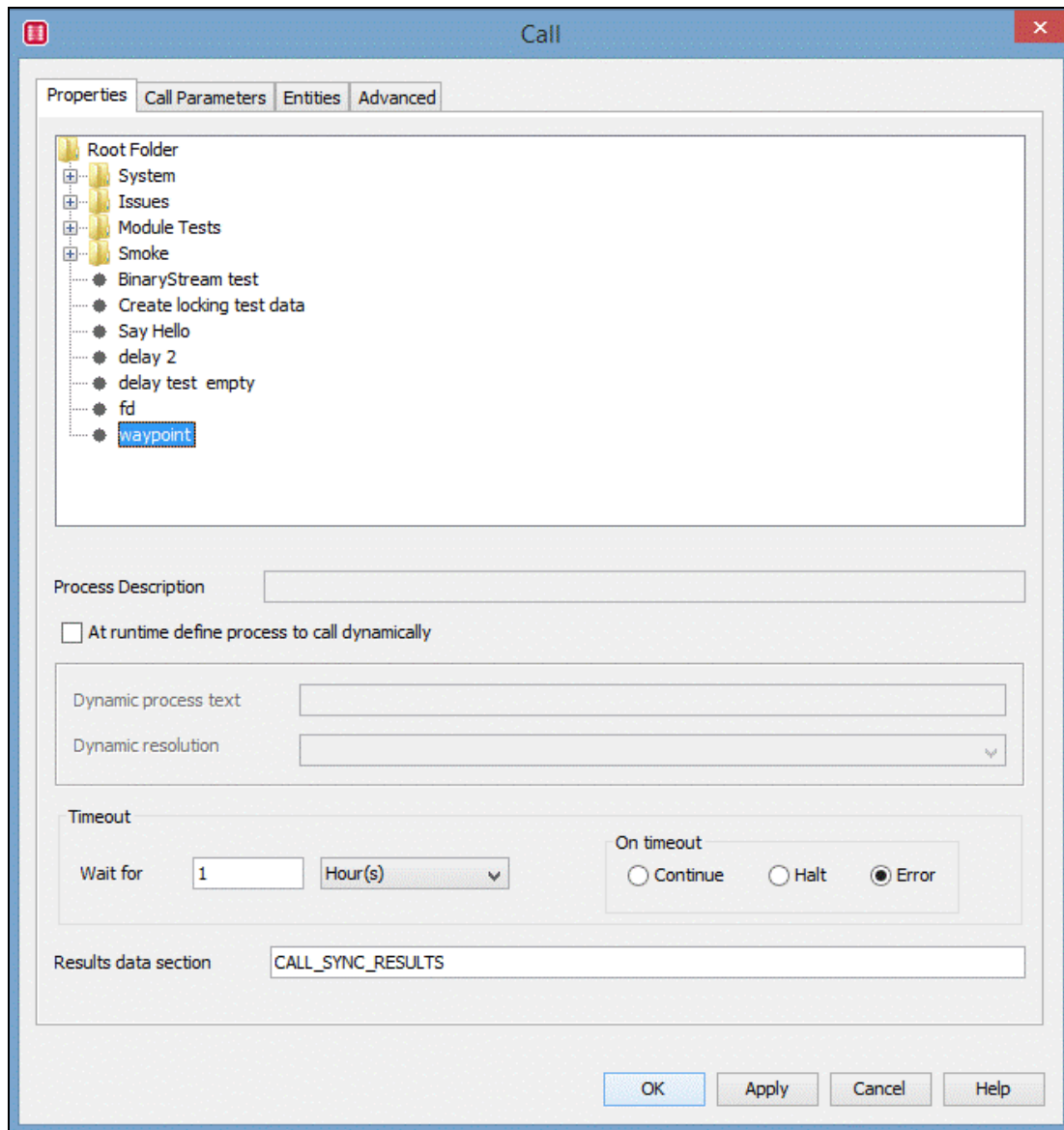
The [Call module](#) allows a process to be selected to run as a sub-process.

Data/message/recipient sections can be passed from the parent (calling) to the child (process interface/sub-process) process. The child process can only be run synchronously. This module works in a similar way to the **Cascade** module when using the sync/return options.

Important: The child process does not require an **Initiator** module if it is only going to be fired by the **Call** module. If you want to be able to run the child process as a standalone process, you must include an **Initiator** module as usual.

Note: The following details assume that you have already created and configured all other modules within the process before adding the **Call** module. If you want to pass data/messages/recipients to the child process, ensure that there are already Data/Message/Recipient modules in the process thread before the **Call** module, or that the data/message/recipient sections you are passing through to the child process are available at this point.

The Properties tab



1. Select the process that should be called. The top panel shows the available processes. Selecting a process will show its description in the Process Description box and the Call parameters and Entities tab will be populated with the selected Process's defined Process Interface.
2. Select **At runtime define process to call dynamically** if you want to determine the child Process that should be run at runtime (using dynamic functions).

To determine the EMF process dynamically, specify the child Process to run at runtime by either entering a literal string or a dynamic function (or a combination of both) in the Dynamic Process text field, then selecting the required Dynamic resolution to specify whether the text in the Dynamic EMF Process field is a Process ID, a Process API name, or a Relative path to the EMF Process (i.e. specifying the

location and name of the target EMF Process using a path from the source - e.g. `..\Voice\My Process`)

Note: The process to call must have the same interface as the selected process in the tree control above it (or an error will be generated at runtime).

3. The Timeout settings allow a time to be specified of how long the Call module should wait for the child Processes to return, and whether the parent EMF Process should then continue if it times out.

Note: A synchronized alert retriever (SyncAR) must be running for the timeout option to work.

IMPORTANT: If the sub-process does not contain a **Return** module and options have been selected on the **Call** module to wait for the sub-process before continuing, then the parent Process will never continue until the timeout period has passed.

4. The **Results data** section name is the name of the data section that will be created when the sub-process has returned back to the calling process. The data section contains the result of the synchronization as explained in the following table:

Column Name	Description
ChildrenExpected	This will always be 1.
ChildrenReturned	This will always be 1.
ReturnCancelled	Always false.
Timestamp	The time that this data section was created.
TimedOut	true if the Call module is continuing because it has waited longer than the Timeout period for the child Processes to return.

The Call Parameters tab

The **Call Parameters** tab allows the message/data/recipient sections that the sub process is expecting, to be mapped.

Call

Properties Call Parameters **Entities** Advanced

Message sections

Name	Description	Type	Pass By	Use default	Value
ms1		IN	Text	<input checked="" type="checkbox"/>	
ms2		IN/OUT	Message section	<input type="checkbox"/>	Text
ms3		OUT		<input type="checkbox"/>	

Data sections

Name	Description	Type	OUT Operation	Use default	Data section
ds1		IN		<input type="checkbox"/>	SAP Browser
ds2		IN/OUT	Merge	<input type="checkbox"/>	SQL
ds3		OUT	Merge	<input checked="" type="checkbox"/>	

Recipient section

Description

Type

On OUT operation

☐ Replace ☐ Merge ☒ Merge (removing duplicates)

OK Apply Cancel Help

The **Message sections** area maps the message sections defined in this process to those required by the sub process. The table contains an entry for each section defined in the sub process interface. The columns are:

- **Name** (read only) - the name of the message section as defined on the sub process interface.
- **Description** (read only) - the description of the message section as defined on the sub process interface.
- **Type** (read only) - IN, IN/OUT, OUT as defined on the sub process interface.
- **Pass By** - This option is only available on an IN operation (not IN/OUT or OUT). Two options are available in a drop-down list:

- **Message section** - If selected, then the **Value** column will display a list of available message sections in the process, so that you can select which message section will be passed to the sub process.
- **Text** - If selected, then any text can be entered in the **Value** column.

Note: For IN/OUT type, **Pass By** is set to **Message section**, and not editable. For OUT type, **Pass By** is blank and not editable.

- **Use Default** - Selecting this option will use the default message section as defined in the process interface, for the named message section. The **Value** column will, therefore, be blanked out and not editable.
- **Value** - If the **Pass By** value is:
 - **Messages section** - If selected, then a list of message sections is available.
 - **Text** - If selected, then text can be entered. The **Value** column supports dynamic functions. Clicking on the drop-down button opens up a large text area to allow larger text to be written, if required (e.g. HTML text).

Note: For OUT type, if the Use Default is selected, then the **Value** column is blank and not editable.

The **Data sections** area maps the data sections defined in this process to those required by the sub process. The table contains entries for the section(s) defined in the sub process interface. The columns are:

- **Name(read only)** - the name of the data section as defined on the sub process interface.
 - **Description(read only)** - the description of the data section as defined on the sub process interface.
 - **Type (read only)** - IN, IN/OUT, OUT as defined on the sub process interface.
 - **OUT operation** - This option is only available on an IN/OUT or OUT only type operation (not an IN type operation). The following options are available in a drop down list:
 - **Replace** - replace any existing data section of the same name when the sub process completes.
 - **Merge** - merge the data section into any existing data section of the same name when the sub process completes.
 - **Merge, no duplicates** - merge the data section into any existing data section of the same name when the sub process completes removing any duplicate rows.
 - **Use Default** - Selecting this option will use the default data section as defined in the process interface, for the named data section. The Data section column will, therefore, be blanked out and not editable.
 - **Data section** - On an IN and IN/OUT type operation a drop down list of data sections is available that allows a data section to be selected to map and pass to the sub process. No value is definable for

The **Recipients** section displays whether the sub process expects recipients (IN), or returns recipients (OUT) or both (IN/OUT). If the operation type is IN/OUT or OUT, then the three radio buttons will define how the recipients are combined back into the calling process:

- **Replace** - recipients from the sub process replace those in the calling process when it completes.
- **Merge** - recipients from the sub process are merged into those in the calling process when it completes.
- **Merge (removing duplicates)** - recipients from the sub process are merged into those in the calling process, removing any duplicates, when it completes.

Call entities

It is useful to be able to also pass in data source connections (so the same sub process could be run against different data sources). This allows for maximum ease of re-use and to avoid having to edit processes to set services after importing.

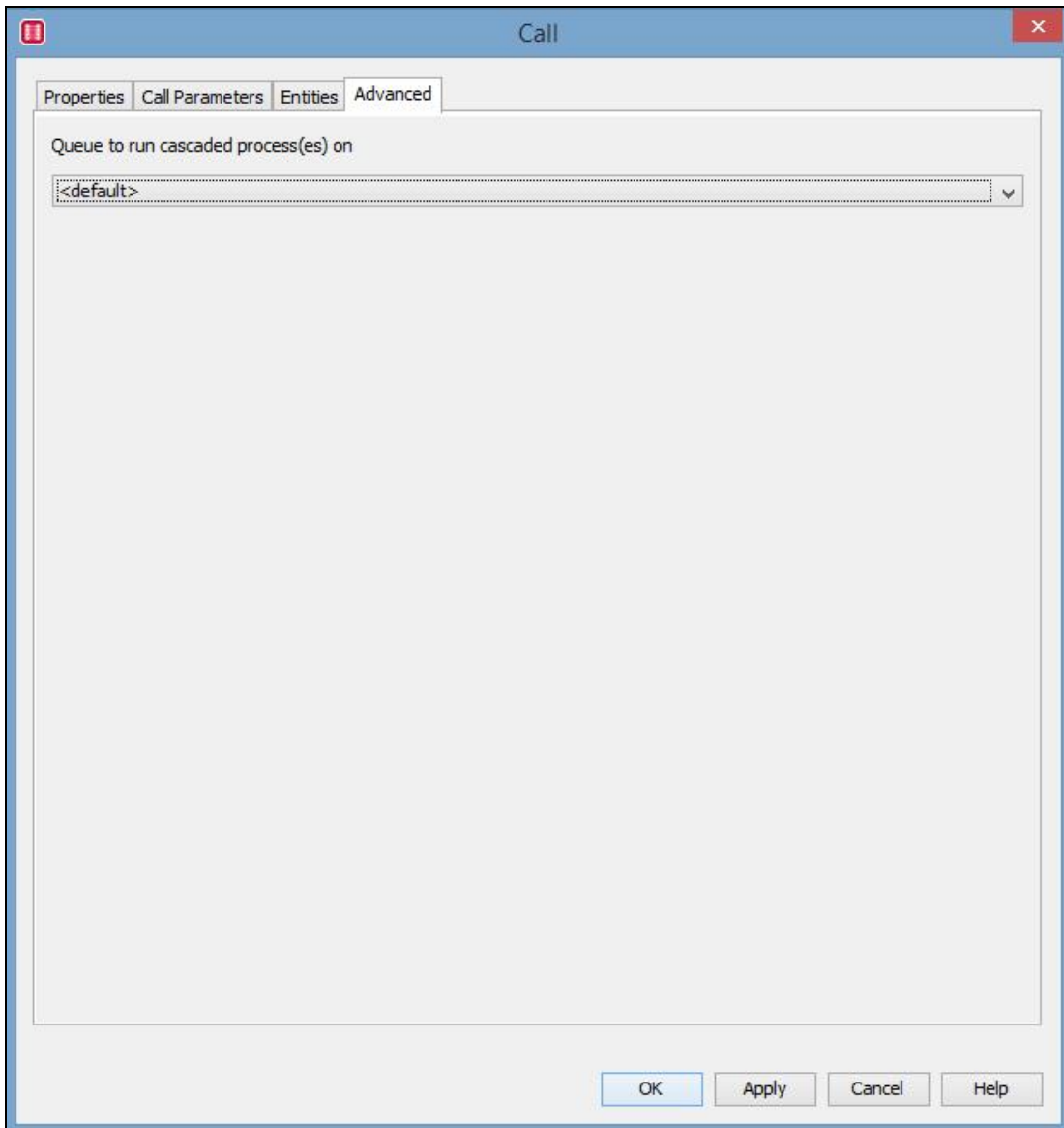
Name	Description	Type	Use d...	Entity
data source 1		JDBC Data source	<input checked="" type="checkbox"/>	
data source 2		JDBC Data source	<input type="checkbox"/>	Orade 10g
jndi		JNDI Service	<input checked="" type="checkbox"/>	
sap connection		SAP Data source	<input type="checkbox"/>	APTEAN_CPIC

- **Name (read only)** - The name of the message section as defined on the sub process interface.
- **Description (read only)** - The description of the message section as defined on the sub process interface.
- **Type (read only)** - As defined in the sub process.
- **Use Default** - Selecting this option will use the default entity as defined in the process interface, for the named entity. The **Entity** column will, therefore, not be editable.
- **Entity** - The entity can only be selected from the drop down list of the available entities. If an entity is not selected, then the process will error at runtime.

Advanced

The **Advanced** tab is used to set the advanced features in the [Call module](#). All the Call modules will use the default queue to run the cascaded process. However, if there are multiple queues set up to spread the workload, or a dedicated queue for the Call module, it can be selected in the **Queue to run the cascaded process(es) on** drop-down menu. The available queues are defined in the System Queues.

Note: The queue must have a [server manager](#) associated with it, else the process placed on the queue will not be run.



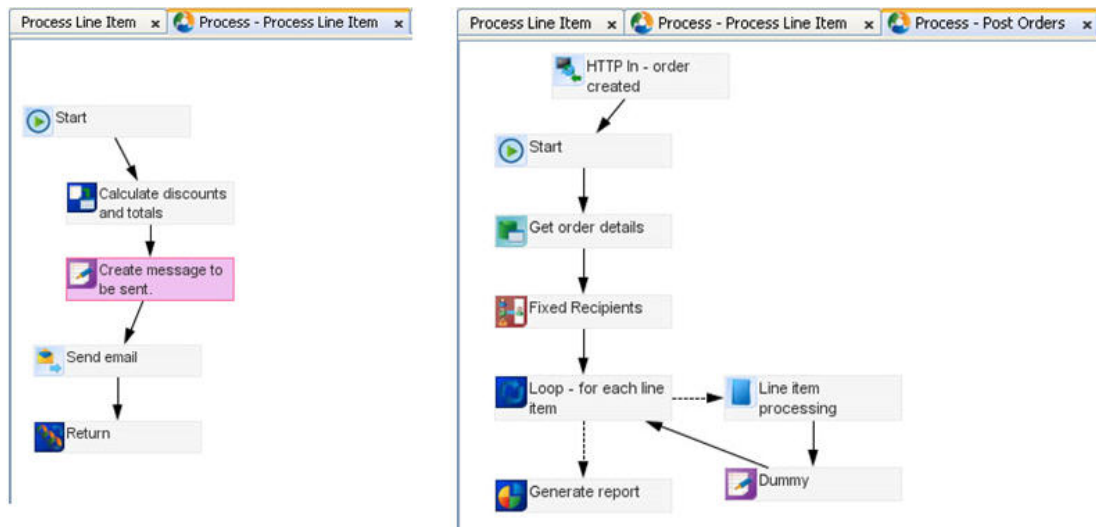
Drag and drop sub processes

To make the sub processes very simple to reuse, it is possible to drag a process icon in the left hand tree and drop it on the process builder (in another process). The point at which the process is dropped will turn into a **Call** module, automatically setting the process to call to be the dropped one. The name of the **Call** module automatically changes to the name of the dropped process, if no default name has been defined in the Process Interface; otherwise it uses the default name defined in the Process Interface of the dropped process. The module icon changes to the icon defined in the Process Interface of the dropped process.

Example of the usage of Process Interface

The following example shows a simple use of the way that Process Interfaces can be used. It uses an example of sales orders that gets posted in via http. The Process Line Item process defines the Process Interface to enable the order line items to be processed. It takes the order line item as an input data section, and returns the processed data back to the calling process (Post Orders). It also returns back the message that has been sent for each line item. The Call module in the calling process (Post Orders) passes through the Order line item, the recipients that will receive the message and the jdbc connection used for the orders database (there may be cases where another process gets it's orders from a different database – for a different company).

The left hand process is a Process Interface (sub process). The process on the right hand is the calling process (containing the Call module).



The Process Interface is defined as follows:

Start

Process Interface Sections | Process Interface Entities | Advanced Run Control

Message sections passed in and out of this process

Name	Type	Description	Default value	Sample
Posted Message	OUT	The message that has...		Show sample

Add
Remove
Remove all
Move up
Move down

Data sections passed in and out of this process

Name	Type	Description	Default value	Sample
Line Item	IN	Line item data		Show sample
Totals	OUT			Show sample


Add
Remove
Remove all
Move up
Move down

Recipient section passed in and out of this process

☐ IN ☒ IN/OUT ☐ OUT ☐ Ignore

Description: Recipients who will receive message

Icon/Label to assign to calling module

 Default module name: Line item processing

Import graphic to use as module icon Reset graphic

OK Apply Cancel Help

Start

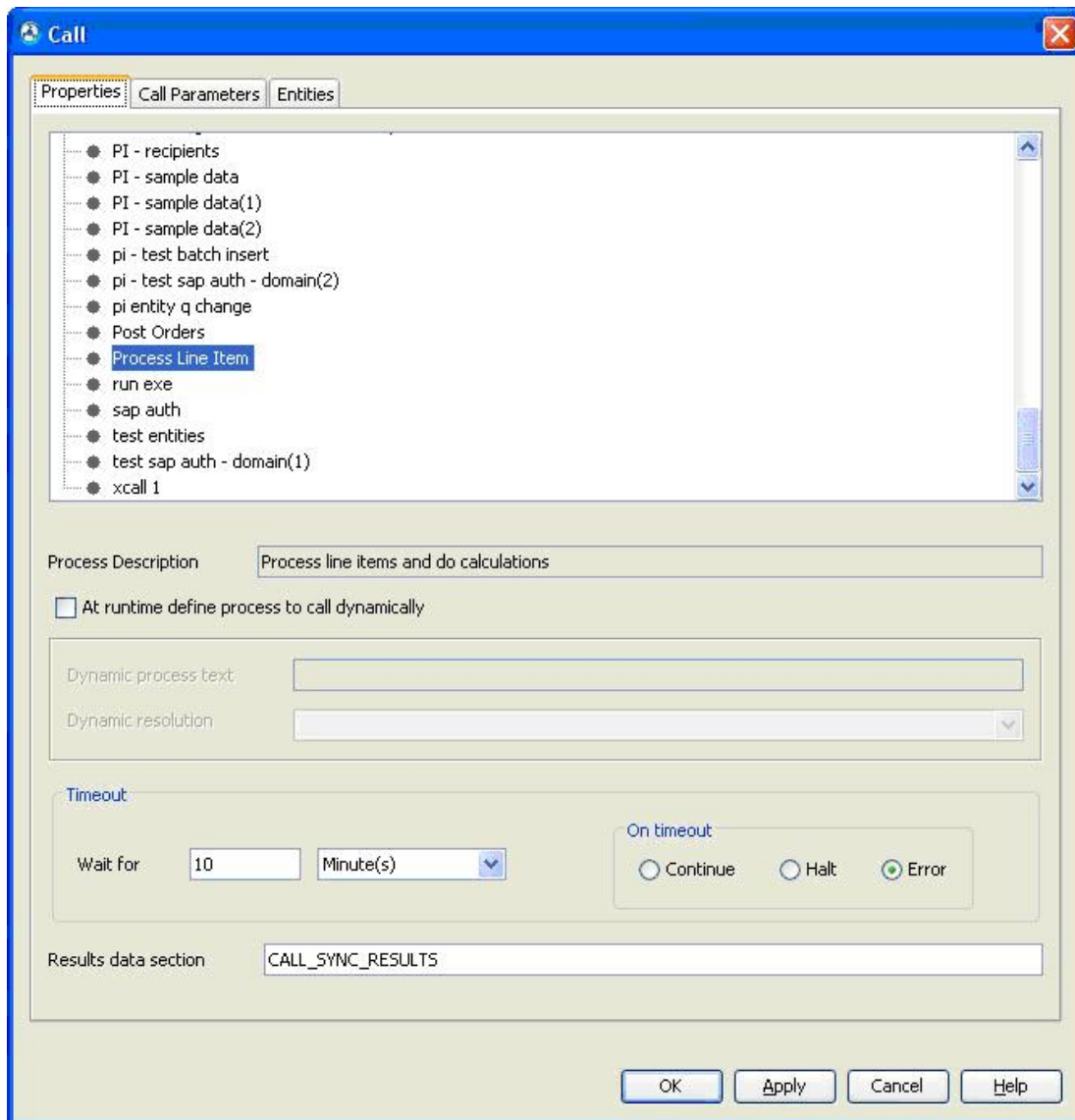
Process Interface Sections | Process Interface Entities | Advanced Run Control

Data sources/services passed into this process

Name	Description	Type	Default value	Sample
jdbc.connection	connection used for...	JDBC Data source	Repository	Repository

Add
Remove
Remove all

The **Call** module has the following properties set. As can be seen in the process Post Orders, the icon for the Call module has been used from the one specified in the Process Interface. The module name is also taken from the Default module name.



Call

Properties **Call Parameters** Entities

Message sections

Name	Description	Type	Pass By	Use d...	Value
Posted Message	The message that has ...	OUT		<input type="checkbox"/>	

Data sections

Name	Description	Type	OUT Operation	Use ...	Data section
Line Item	Line item data	IN		<input type="checkbox"/>	Orders
Totals		OUT	Merge	<input type="checkbox"/>	

Recipient section

Description: Recipients who will receive message

Type: IN/OUT

On OUT operation:

☐ Replace ☐ Merge ☒ Merge (removing duplicates)

OK Apply Cancel Help

Call

Properties Call Parameters **Entities**

Entities

Name	Description	Type	Use ...	Entity
jdbc connection	Connection used for the ...	JDBC Data source	<input checked="" type="checkbox"/>	Order processing connect...

Cascade Module Synchronization Options

You can use the Cascade module **Synchronization tab** to specify whether the child EMF Process runs independently or synchronously with the parent EMF Process, and if the parent runs synchronously - how data is merged from the child EMF Process back into the parent EMF

Process. The point that synchronization happens is when the child EMF Process processes a [Return Module](#).

The screenshot shows the 'Cascade' dialog box with the 'Synchronization' tab selected. The 'Synchronization method' section has three radio buttons: 'No wait/synchronization' (selected), 'Wait for first cascaded process before continuing', and 'Wait for all cascaded processes before continuing'. The 'Timeout' section has a 'Timeout period' of 1, a 'Timeout interval' of 'Hour(s)', and a 'Halt on timeout' checkbox. The 'Message section merge options' section has two radio buttons: 'Do not replace existing message sections if they already exist' (selected) and 'Overwrite message sections if they already exists'. The 'Data section merge options' section has two radio buttons: 'Do not alter existing data sections if they already exist' (selected) and 'Add child data into existing data sections'. The 'Recipient section merge options' section has three radio buttons: 'Ignore recipient sections in child processes' (selected), 'Add recipient sections from child processes', and 'Replace all recipient sections in parent with those from child processes'. The 'Results data section' is set to 'CASCADE_SYNC_RESULTS'. At the bottom are 'OK', 'Apply', 'Cancel', and 'Help' buttons.

IMPORTANT: If the cascaded EMF Process doesn't contain a [Return Module](#) and options have been selected on the cascade module to wait for the cascaded alert before continuing, then the parent EMF Process will never continue until the timeout period has passed.

- The **Results data section** name is the name of the data section that will be created after synchronization has been completed. The data section contains the results of the synchronization as explained in the following table:

ChildrenExpected The number of child EMF Processes that where

expected to be **returned**. e.g. if cascading per row then this will be the number of rows in the data section.

ChildrenReturned The number of child EMF Processes that actually **returned**. This may not match the expected value if a child EMF Process cancelled the wait in the **Return module** or if the cascade module has waited longer than the **Timeout period** for the child EMF Processes to **return**.

ReturnCancelled **true** if the **Return module** ran with the option **Cancel wait for other siblings**, **false** is otherwise returned.

Timestamp The time that this data section was created.

TimedOut **true** if the **Cascade module** is continuing because it has waited longer than the **Timeout period** for the child EMF Processes to **return**.

- The **Synchronization method** allows you to decide how the parent EMF Process and child EMF Process will interact, that is, whether they will run independently or synchronously
 - **No wait/synchronization** will cause the child EMF Processes to run independently. Once they have been queued the parent EMF Process will continue and no more interaction between the parent and child EMF Processes will occur.
 - **Wait for first cascaded alert before continuing** will mean that the parent alert will wait for the first child EMF Process to **return** before continuing. Any other instances of the child EMF Process **returning** will be ignored. An example use would be if the parent had cascaded per recipient, the child EMF Process sent out a message and waited for reply, and the parent should continue after the first recipient replied.
 - **Wait for all cascaded alerts before continuing** will mean that the parent alert will wait for all child EMF Processes to **return** before continuing.
- The **Timeout** settings allow a time to be specified of how long the **Cascade module** should wait for the child EMF Processes to **return**, and whether the parent EMF Process should then continue if it does timeout.

Note: a synchronized alert retriever (SyncAR) must be running for the timeout option to work.

- **Message section merge options** allows you to specify how message sections from the returning child EMF Processes are merged into the parent EMF Process.
- **Data section merge options** allows you to specify how data sections from the returning child EMF Processes are merged into the parent EMF Process. Example: If multiple child EMF Processes are initiated and they all create a data section called 'SQL' with a row of data in it, then if the **Add child data into existing data sections** option is selected and the synchronisation option **Wait for all cascaded alerts before continuing**, after synchronisation (i.e. all child alerts have returned) the parent EMF

Process will contain a new data section called 'SQL' with multiple data rows (one for each of the child EMF Processes).

- **Recipient section merge options** allows you to specify how recipient sections from the returning child EMF Processes are merged into the parent EMF Process.

Tip: if there is a data/message/recipient section that you don't want returned from the child EMF Process you can also use the **Remove** options on the **conditional links** dialog to remove any unwanted sections in the child EMF Process on the **link** before the **Return module**.

[Cascade Module](#)

[Return Module](#)

[Go to Start of Flow Modules](#)

[Example of Using the Cascade Module with HTTP](#)

Filter and Transformation Modules

The following Filter and Transformation modules are available:

- The [Parser](#) module is a filters and transformers module that converts source text in delimited or fixed width column format into tabular format (columns and rows), in order to populate an EMF process Data section.
- The [XML Parser](#) module extracts data from XML, breaks it down into the various component parts (content, form, function and so on) and stores it in a Data Section for use in another module.
- The [XSL Transformer](#) module is designed to facilitate the conversion of XML to another data representation. The destination type is normally a different form of XML but can effectively be almost any form that can be represented by text.
- The [Data filter](#) module allows you to remove information that has not changed since a previous instance of an EMF Process from a data or recipient section.
- The [Datasection Toolbox](#) module provides a useful set of functions for manipulating an EMF process data section.
- The [Duplicates](#) module allows the analysis of a data section for duplicate values (more than one occurrence of values).
- The [Gap Detection](#) module allows the analysis of a data section for gaps in sequences.
- The [White List Event](#) module is used to add (mark) items to a white list. It records those items that we don't want to be notified about again in future.
- The [White List Filter](#) module is used to filter out (or mark) rows in a data section that match previously white listed items. It will scan each row in a selected data section to check it against the data in the selected white list definition.
- The [Data Transformation](#) module allows transformations to be mapped between EMF data sections and XML documents. That is, map data from XML to XML, Data section to XML, XML to Data section and Data section to Data section.

Parser Module

The Parser module is a processing module that converts source text in delimited or fixed width column format into tabular format (columns and rows), in order to populate an EMF process Data section.

You can then use the resulting Data section in a different module in the EMF Process. Dynamic functions used to process the source text will be resolved at runtime.

The Parser module can be used for any input data, and is typically used to process EMF requests sent by field staff through [Email](#) or data files read using the [File In](#) module.

To use a Parser module in an EMF process:

1. From the toolbar on the right, drag the **Parser** icon onto an appropriate location on the EMF Process Design Graph.



2. Double-click the icon to open the **Parser** dialog box.

The **Parser** dialog box enables you to specify the source text criteria and preview the output data.

3. In the **Properties** tab, the **Destination data section** box displays the output data section name. You can either use the name of an existing section or type a new name to create a new data section.
4. Enter the text to be processed in the **Source text** field.
To insert a [dynamic function](#), right-click in the Source text area, and select from the available list, or use the dynamic function wizard to build an expression, for example, `$MESSAGE('MSG')$`.
5. Select **Trim spaces** to remove spaces from the beginning and end of each field within the source text.
6. In the **Text qualifier** box, enter the character to be used to differentiate your field data from your delimiter. The data for a single field will be enclosed within the text qualifiers. For example, if the data contains a comma, 10 Street1, City1 and you have selected to use a comma as a record delimiter, then the data should be enclosed in a text qualifier such as `'10 Street1, City1'`, and a single quote (') should be entered as

the text qualifier. This enables the parser module to distinguish between the commas in the data and the commas used to delimit fields during processing.

7. Select **Use column names from first row of source** to use the first row in the input text as output column headings.
8. Select the column type from the following:
 - **Delimited** - Select **Delimited** when the source text is delimited, for example, comma separated values.

Example:

First Name, Last Name, Age

John, Smith, 22

Doe, Jane, 25

Nathan, Bright, 27

- From the **Column Delimiter** drop-down list, select the character that separates columns in the source text, for example, comma (,).
- From the **Row Delimiter** drop-down list, select the character that separates rows in the source text, for example, CR.

- **Fixed width** - Select **Fixed width** if the source text is in fixed width column format.

Example:

First_Name Last_Name Age

John Smith 22

Doe Jane 25

Nathan Bright 27

9. Click **Auto define** to define the column output based on the data entered in the **Source text** field. Alternately, you can define columns directly using the **Add** or **Remove** buttons.

The **Column Definition** tab displays the field definitions (field names and datatypes). The field names are defined based on the data in the first row of the Source Text field, if the **Use column names from first row of source** check box is selected.

Parser

Properties | Preview

Destination data section: Parser

Source text:

```
First Name,Last Name, Age
John,Smith,22
Doe,Jane,25
Nathan,Bright,27
```

☒ Trim spaces Text qualifier:

☒ Use column names from first row of source

Column type:

☒ Delimited ☐ Fixed width

Delimiters:

Column delimiter: Comma Row delimiter: CRLF

Column Definition Column Output Order

Use	Name	Type	Format
<input checked="" type="checkbox"/>	First Name	Varchar	
<input checked="" type="checkbox"/>	Last Name	Varchar	
<input checked="" type="checkbox"/>	Age	Integer	

Auto define
Add
Remove
Remove all
Move up
Move down

OK Apply Test Cancel Help

- To show or hide a column in the output data section created, select or clear the corresponding check box to the left of the column.
- Double-click on a column name in the **Name** field, for example, First Name to rename it in the generated data section.
- Click on a column type in the **Type** field, for example, Varchar, and select a value from the drop-down list to change the column type.
- Use the **Format** field to specify the format of the input field to be parsed. Typically, the input field format is auto detected, but you may need to specify the format if, for example, your local Date format is different from the Date format of the input field to be parsed. In this case, if you know that the field format for the Date field in the input text is dd/MM/yy, then double-click on the field format in the **Format** field and enter dd/MM/yy.

The default time format is taken from the "Short date" of your locale, as defined initially in the Windows Region and Language settings. It ignores any changes to the "Short date" format set in the Windows Region and Language settings. So when you first select a country locale, the default format displayed in the Windows Region and Language settings dialog is the format that will be used. For example, the default format for the UK locale would be HH:mm. However,

you can change the format to any valid java time format (<http://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html>). If you had data of "08:30 PM", then you would specify "h:mm a" as the time format.

Note: If the locale is changed, the server must be restarted for the changes to take effect.

Note: The **Format** field is only applicable for the date and time fields.

- Click on a value in the **Width** field to change the width of a column.

Note: This is applicable only if the column type is **Fixed Width**.

- Additionally, use the following options to add, delete, or change the order of the columns:
 - **Add**
 - **Remove**
 - **Remove All**
 - **Move Up/Move down**

By default, the **Column Output Order** tab defines the order in which the columns are displayed in the source data.

Use this tab to configure the columns to be created and their order in the final data section.

10. Click **Test** to view the output data in the **Preview** tab.
11. Click **Save as sample** to save the data for testing purposes.

[Example of Parser Module](#)

[Back to Start of Filter and Transformation Modules](#)

Example of Parser Module

A user is querying a flight booking system for two forthcoming business trips and has been advised to send a message section containing city, destination and date of flight, with delimiters, so the following SMS message section is sent:

WASHINGTON # ATLANTA # 123001; LAS VEGAS# CALIFORNIA # 011502

Using Data Source "\$Message('MSG')\$", a column delimiter of "#", a row delimiter of ";", and destination data section called "SPLITDATA", the following will happen:

- The dynamic function "\$MESSAGE('MSG')\$" will resolve to WASHINGTON # ATLANTA # 123001; LAS VEGAS# CALIFORNIA # 011502
- This text will be split using " # " and ";". Since there is a " ; ", two rows will be returned.
- The Data Section "SPLITDATA" will be created and contain two rows with the following column data. Note that all data has had leading and trailing spaces removed.

	Col0	Col1	Col2
Row0	WASHINGTON	ATLANTA	123001
Row1	LAS VEGAS	CALIFORNIA	011502

This Data Section can then be used in later modules to resolve the flights available between the two locations, on the given day. The data may be sent to the user by appropriate means.

For example, `$DATA('SPLITDATA',0,0)$` will return "WASHINGTON".

[Parser Module](#)

[Back to Start of Filter and Transformation Modules](#)

The XML Parser Module

The **XML Parser** module extracts data from XML, breaks it down into the various component parts (content, form, function and so on) and stores it in a Data Section for use in another module.

To use the XML Parser module:

1. Drag the **XML Parser** icon onto the EMF Process Design graph at the appropriate place.



2. Double-click the XML Parser icon to display the XML Parser Properties screen:

3. The **Properties** tab defines the source text to be parsed and the data section that will receive the parsed information.
4. Enter the XML that is to be parsed in the **XML Source Text** field. This must be valid XML, and can contain any non-recipient based dynamic function that returns valid XML (this can either be entered directly or obtained via a [dynamic function](#)).
5. Enter the name of the Data section that is to be created and populated with the data extracted from the XML in the **Datasection Name to Create** field.
6. To define aspects of the way the different parts of your XML document are addressed, you can use the XPath feature.

XPath is a part of XSLT (eXtensible Stylesheet Language Transformation - a standard way to describe how to transform the structure of an XML document), and is a formal recommendation from the World Wide Web Consortium

(<http://www.w3.org/TR/xpath>). It traverses XML specified by a path to a destination node and allows information to be extracted from the XML.

Note: You can access XPath feature using the [XPATH Dynamic Function](#). This reads the specified language from the specified Message section (which must contain valid XML), and executes the supplied XPath returning a textual representation of the result.

7. **Parent XPath Expression** defines the XPath to the Parent element(s).
8. **Child XPath Expression** defines the XPath to the Child element(s).
9. The **Parent node behaviour** section determines whether the parent element(s), as defined by the Parent XPath, should be converted into the rows or columns in the DataSection. If you select **Parent node as column, child as row** you can also choose to **Correlate rows by name** (this is not required for the other option).
10. The **Get cell values from...** section specifies whether to extract the *names* or *values* of the nodes as the data (this is necessary because it is possible for the presence of the node, not its value, to be the significant factor). Select **Node text value** or **Node name** accordingly.

For example, in the element:

```
<ELEMENT ATTRIBUTENAME="attribute value">element value</ELEMENT>
```

The name of the element is **ELEMENT**, the text value is **element value**. Likewise the name of the attribute is **ATTRIBUTENAME** and its value is **attribute value**.

XML namespaces are used for providing uniquely named elements and attributes in an XML document. An example is given below:

```
<?xml version="1.0"?>
<x:DOCELEM xmlns:x="http://www.example.com/test">
  <AAA ddd="d" ccc="c">
    <CCC xmlns="http://www.example.com/test2">C1</CCC>
    <DDD>D</DDD>
    <CCC>C2</CCC>
  </AAA>
</x:DOCELEM>
```

XML namespaces are used for providing uniquely named elements and attributes in an XML document. An XML instance may contain element or attribute names from more than one XML vocabulary.

11. Use the **Add/Delete** buttons in the Namespace Prefixes section to add or remove namespace prefixes.
12. Click **Test** to review the parsed data section in the **Preview** tab.

[Examples of XML Parser settings](#)

[Back to Start of Filter and Transformation Modules](#)

The XPath tab

XPath is a part of XSLT (eXtensible Stylesheet Language Transformation - a standard way to describe how to transform the structure of an XML document into an XML document with a different structure), and is a formal recommendation from the World Wide Web Consortium (<http://www.w3.org/TR/xpath>). It traverses XML specified by a path to a destination node and allows information to be extracted from the XML.

Note: You can access XPath functionality using the [XPATH Dynamic Function](#). This reads the specified language from the specified Message section (which must contain valid XML), and executes the supplied XPath returning a textual representation of the result.

The **XPath** tab of the XML Parser allows you to define aspects of the way the different parts of your XML document are addressed.

- **Parent node as...** determines whether the parent element(s), as defined by the Parent XPath, should be converted into the rows or columns in the DataSection. If you select **column** you can also choose to **Correlate rows by name** (this is not required for the other option).
- **Parent XPath** defines the XPath to the Parent element(s).
- **Child XPath from parent** defines the XPath to the Child element(s).
- **Node text value** or **Node name** specifies whether to extract the *names* or *values* of the nodes as the data (this is necessary because it is possible for the presence of the node, not its value, to be the significant factor).

For example, in the element:

```
<ELEMENT ATTRIBUTENAME="attribute value">element value</ELEMENT>
```

The name of the element is **ELEMENT**, the text value is **element value**. Likewise the name of the attribute is **ATTRIBUTENAME** and its value is **attribute value**.

[Examples of XML Parser settings](#)

[Back to Start of Filter and Transformation Modules](#)

Examples of XML Parser settings

Following are some examples to show the results of different settings in the XML parser on the same source XML:

```
<?xml version="1.0"?>
<DOCELEM>
  <AAA ddd="d"  ccc="c">
    <CCC>C</CCC>
    <DDD>D</DDD>
    <CCC>C</CCC>
  </AAA>
  <AAA ccc="c" ddd="d">
    <CCC>C</CCC>
    <CCC>C</CCC>
    <DDD>D</DDD>
```



```

    </AAA>
    <BBB ccc="c" ddd="d" eee="e">
        <CCC>C</CCC>
        <DDD>D</DDD>
        <EEE>E</EEE>
    </BBB>
</DOCELEM>

```

If the settings are as follows:

Parent Node as: **Row**

Parent Xpath: **//AAA**

Child Xpath from Parent: *****

Node: **Text Value**

The output would be as follows:

CCC	DDD	CCC1
C	D	C
C	D	C

Note: When the column names match the names of the column nodes, it results in duplication and an incrementing number is appended to the name. If the Parent Node is set to **Column** instead, correlated by **Name**, the parent nodes become the columns and children become the rows.

If **Correlate Rows By Name** is set to **false**:

AAA	AAA1
C	C
D	C
C	D

Note: The ordering of the cells is based entirely on the ordering of the nodes in the XML.

If the **Correlate Rows By Name** is set to **true**:

AAA	AAA1
C	C
D	D
C	C

It is also possible to extract the node names by changing the **Value Type** parameter to **Node Name**:

AAA	AAA1
CCC	CCC
DDD	DDD
CCC	CCC

Extracting Attributes

To extract attributes, the Xpath must be changed (after setting the value type to the correct value). Also, for this example, all of the child nodes of DOCELEM must be classified as parent nodes. In this case, if the settings are as follows:

- Parent Node as: **Column**, Correlated by **Name**
- Parent Xpath: **/DOCELEM/***
- Child Xpath from Parent: **attribute::***
- Node: **Text Value**

The output would be as follows:

AAA	AAA1	BBB
C	C	C
D	D	D
		E

References

- XPath specification: <http://www.w3.org/TR/xpath>
- XPath tutorial: <http://www.zvon.org/xxl/XPathTutorial/General/examples.html>
- XPath demo: <http://www.vbxml.com/xpathvisualizer/default.asp>

[The XML Parser](#)

[The Filter and Transformation Modules](#)

The XSL Transformer Module

The **XSL Transformer** module is designed to facilitate the conversion of XML to another data representation. The destination type is normally a different form of XML but can effectively be almost any form that can be represented by text.

About XSL and XSLT

The XSL (eXtensible Style sheet Language) engine uses style sheets to transform XML

documents into other document types, and to format the output. It takes a source document (XML source tree) and outputs a new document (result tree) as specified by the definitions in the style sheet. XSLT is the language part of XSL that defines how to extract and transform the source tree information to the result tree.

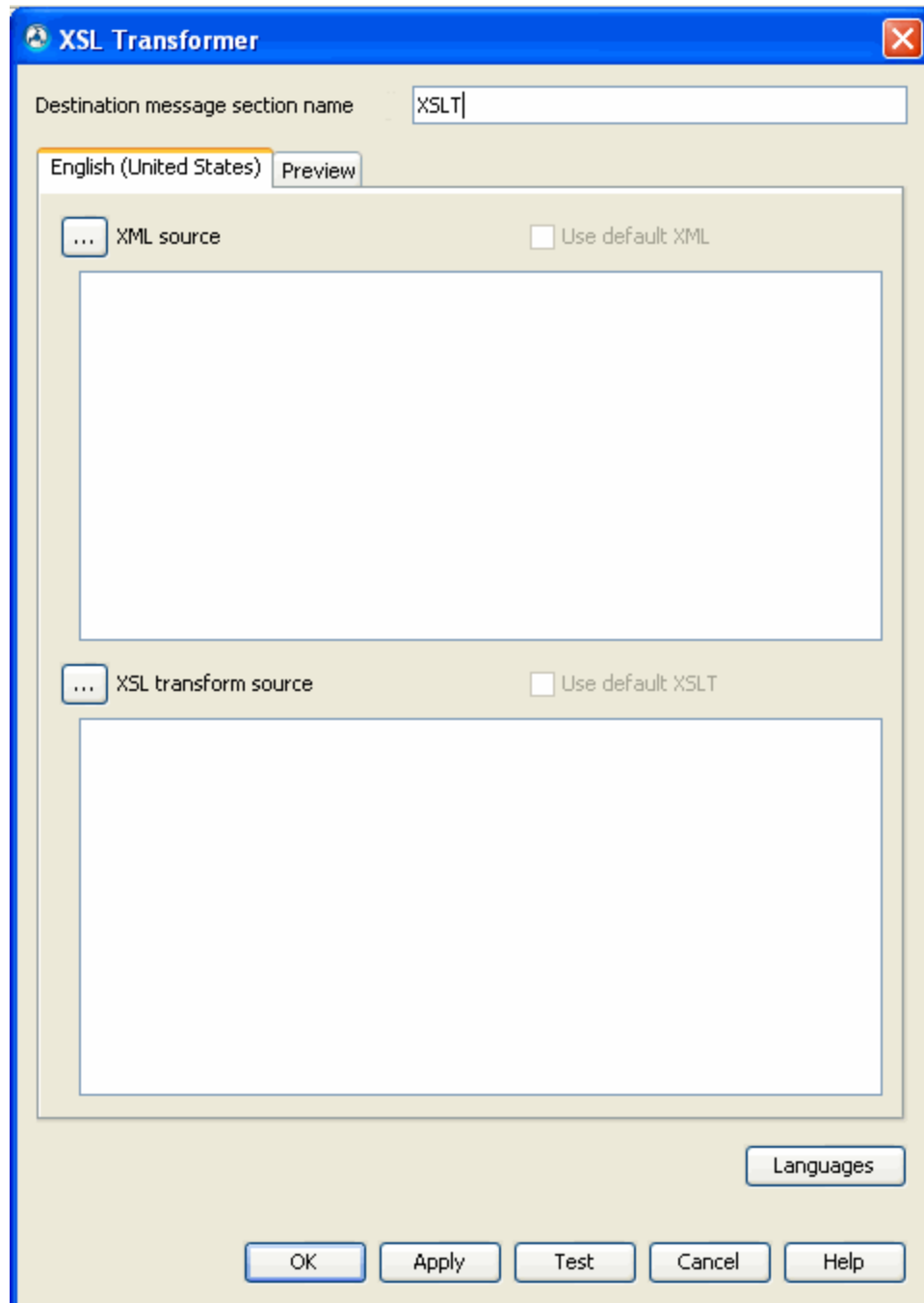
Note: EMF uses [Xalan](#) for the transformations. However, Apteian reserves the right to change this at a later date.

To use an XSL Transformer module:

1. Drag the **XSL Transformer** icon onto the EMF Process Design graph at the appropriate place.



2. Double-click the XSL Transformer icon to display the XSL Transformer Properties screen.



3. The XSL Transformer screen is used to define the source XML text to be transformed, the source of the XSL, and the data section that will receive the transformed information.
4. Enter the name of the message section that is to be created and populated with the data extracted from the XML in the **Destination message section name** field.
5. To select the languages that are to be processed as per formatter objects, click **Languages**.

6. Enter the XML that is to be parsed into the **XML source** field. It can be typed in directly, obtained from a file (by clicking on the **Browse [...]** button), or obtained from any non-recipient based dynamic function (this can be different for each language). If you are using a language other than the default you can select **Use Default XML** to use the XML that is defined in the default language XML.
7. Enter the XSL Transform to be used in the **XSL transform source** field. It can be typed in directly, obtained from a file (by clicking on the **Browse [...]** button), or obtained from any non-recipient based dynamic function, and can be different for each language. If you are using a language other than the default you can select **Use Default XSLT** to use the XSLT defined in the default language XSLT.
8. Click **Test** to test the connection and review the results in the Preview tab.

For further details on XSLT, consult the following references:

- XSLT specification: <http://www.w3.org/TR/xslt>
- XSLT tutorial: <http://www.zvon.org/xxl/XSLTutorial/Books/Book1/>
- Further information on XSLT: <http://www.javaworld.com/javaworld/jw-12-2001/jw-1221-xslt.html>
- Xalan: <http://xml.apache.org/xalan-j/index.html>

[XSL Transformer Examples](#)

[Back to Start of Filter and Transformation Modules](#)

XSL Transformer Examples

Example 1:

The first example shows how to make an EMF process with several SQL INSERT statements more efficient, by using XSLT to construct the SQL statement with all the INSERTs, rather than cascading per row for each INSERT statement.

Possible scenarios that could use this example are:

- A customer needs to transfer a large amount of data from their remote database to a local database. They know this can be achieved using a cascaded EMF process (see graphical representation) but this is inefficient.
- A customer's database contains sales figures of all products and customer accounts. They would like to produce a weekly spreadsheet showing actual product sales against accounts (i.e. the columns should show the customer account codes and the row should show the product code).

[Click here](#) to see the layout of an EMF Process, and the settings of the modules, to implement the following.

Given the XML:

```
<?xml version="1.0"?>
<order>
  <salesperson>John Doe</salesperson>
  <item>Production-Class Widget</item>
  <quantity>16</quantity>
```

```

<date>
  <month>1</month>
  <day>13</day>
  <year>2000</year>
</date>
<customer>Sally Finkelstein</customer>
</order>

```

and the style sheet:

```

<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
  <order>
    <date>
      <xsl:value-of select="/order/date/year"/><xsl:value-of
        select="/order/date/month"/><xsl:value-of select="/order/date/day"/>
    </date>
    <customer>Company A</customer>
    <item>
      <xsl:apply-templates select="/order/item"/>
      <quantity><xsl:value-of select="/order/quantity"/></quantity>
    </item>
  </order>
</xsl:template>
<xsl:template match="item">
  <part-number>
    <xsl:choose>
      <xsl:when test=". = 'Production-Class Widget'">E16-25A</xsl:when>
      <xsl:when test=". = 'Economy-Class Widget'">E16-25B</xsl:when>
      <!--other part-numbers would go here-->
      <xsl:otherwise>00</xsl:otherwise>
    </xsl:choose>
  </part-number>
  <description><xsl:value-of select="."/></description>
</xsl:template>
</xsl:stylesheet>

```

the output will be as follows:

```

<?xml version="1.0" encoding="utf-8"?>
<order>
  <date>2000/1/13</date>
  <customer>Company A</customer>
  <item>
    <part-number>E16-25A</part-number>
    <description>Production-Class Widget</description>
    <quantity>16</quantity>
  </item>
</order>

```

[Click here](#) to see the layout of an EMF Process, and the settings of the modules, to implement the above.

Example 2:

In the following example, the XSL Transformer is used to extract the "review" values from an XML file giving details of two books and create an HTML section containing the number of books that have a review value of 3.5 and 4.

Given the XML:

```
<?xml version="1.0"?>
<books>
  <book>
    <name>XML by Example</name>
    <author>John Smith</author>
    <listprice>24.99</listprice>
    <price>17.49</price>
    <review>4.5</review>
    <publish>QUE</publish>
  </book>
  <book>
    <name>Professional XML</name>
    <author>Mark Birkbeck, Michael Kay, Steve Livingstone</author>
    <listprice>49.99</listprice>
    <price>34.99</price>
    <review>4</review>
    <publish>Wrox</publish>
  </book>
</books>
```

and the XSL Transform:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="/">
  <html><body>
    <p>Review of 3.5 = <xsl:value-of select="count (books/book[review=3.5])" /></p>
    <p>Review of 4 = <xsl:value-of select="count (books/book[review=4])" /></p>
  </body></html>
</xsl:template>
</xsl:stylesheet>
```

the output will be as follows:

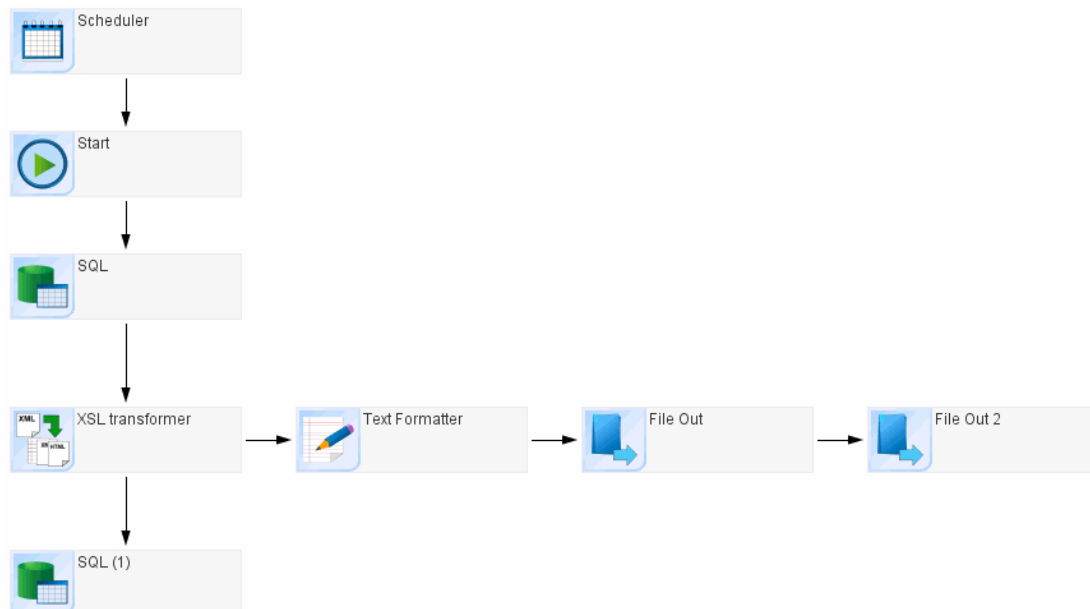
```
<html>
<body>
<p>Review of 3.5 = 0</p>
<p>Review of 4 = 1</p>
</body>
</html>
```

[The XSL Transformer](#)

[Back to Start of Filter and Transformation Modules](#)

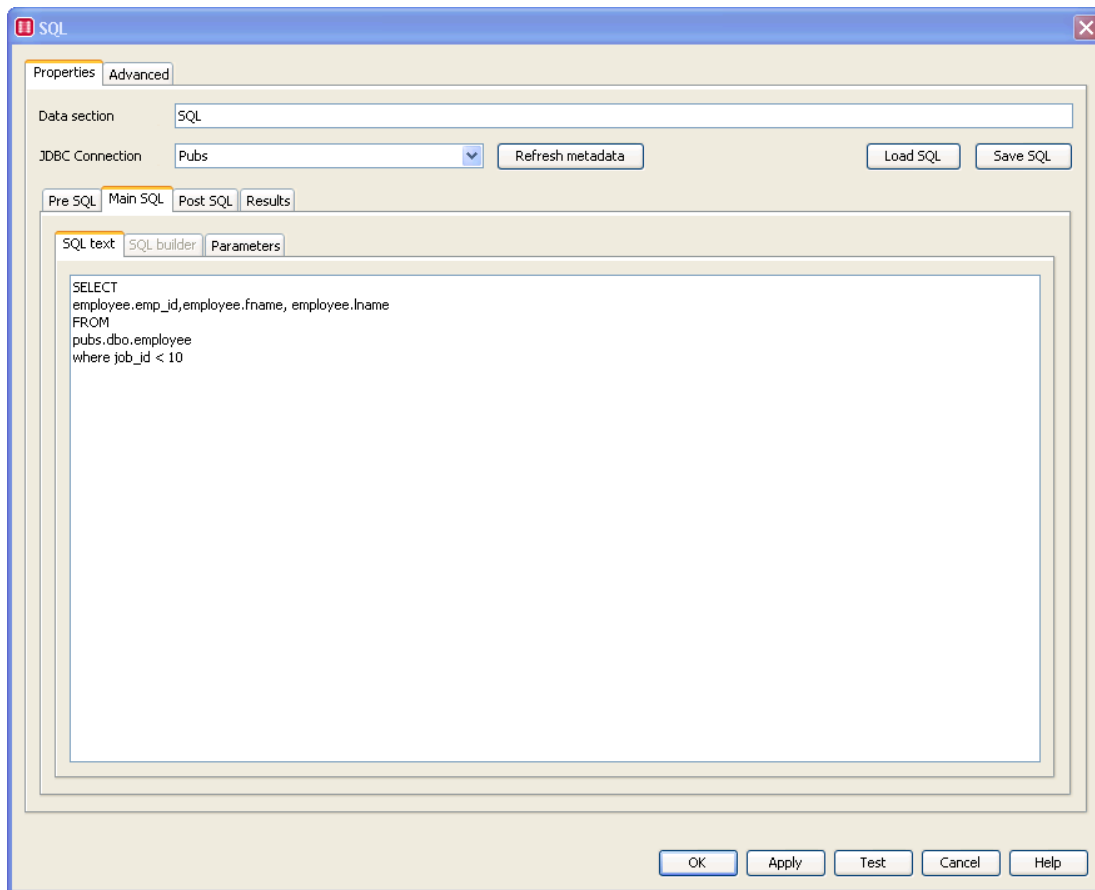
XSLT EMF Process Example

The EMF Process would look like this in the builder. See below for the settings of the various modules.



[Back to XSLT examples](#)

SQL module settings



[Back to XSLT examples](#)

SQL result set

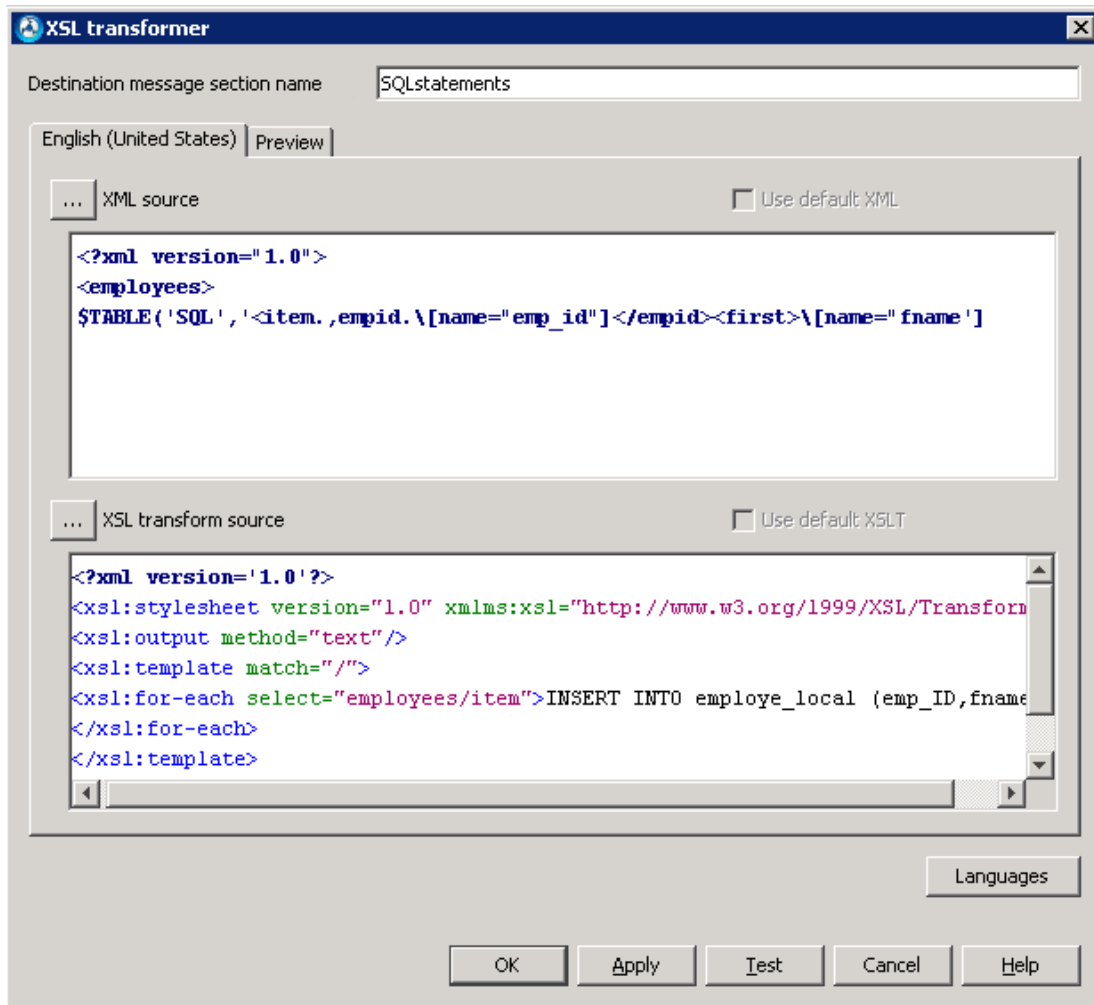
Results - 26 rows

emp_id	fname	lname
VPA30890F	Victoria	Ashworth
L-B31947F	Lesley	Brown
F-C16315M	Francisco	Chang
PTC11962M	Philip	Cramer
AMD15433F	Ann	Devon
ARD36773F	Anabela	Domingues
PXH22250M	Paul	Henriot
CFH28514M	Carlos	Hernandez
PD147470M	Palle	Ibsen
KJJ92907F	Karla	Jablonski
MGK44605M	Matti	Karttunen
JYL26161F	Janine	Labrune
M-L67958F	Maria	Larsson
LAL21447M	Laurence	Lebihan
RBM23061F	Rita	Muller
HAN90777M	Helvetius	Nagy
SKO22412M	Sven	Ottlieb
PSP68661F	Paula	Parente
M-P91209M	Manuel	Pereira
MJP25939M	Maria	Pontes
M-R38834F	Martine	Rance
DWR65030M	Diego	Roel
A-R89858F	Annette	Roulet
MMS49649F	Mary	Saveley
MAS70474F	Margaret	Smith
GHT50241M	Gary	Thomas

Close Help

[Back to XSLT examples](#)

XSLT module settings



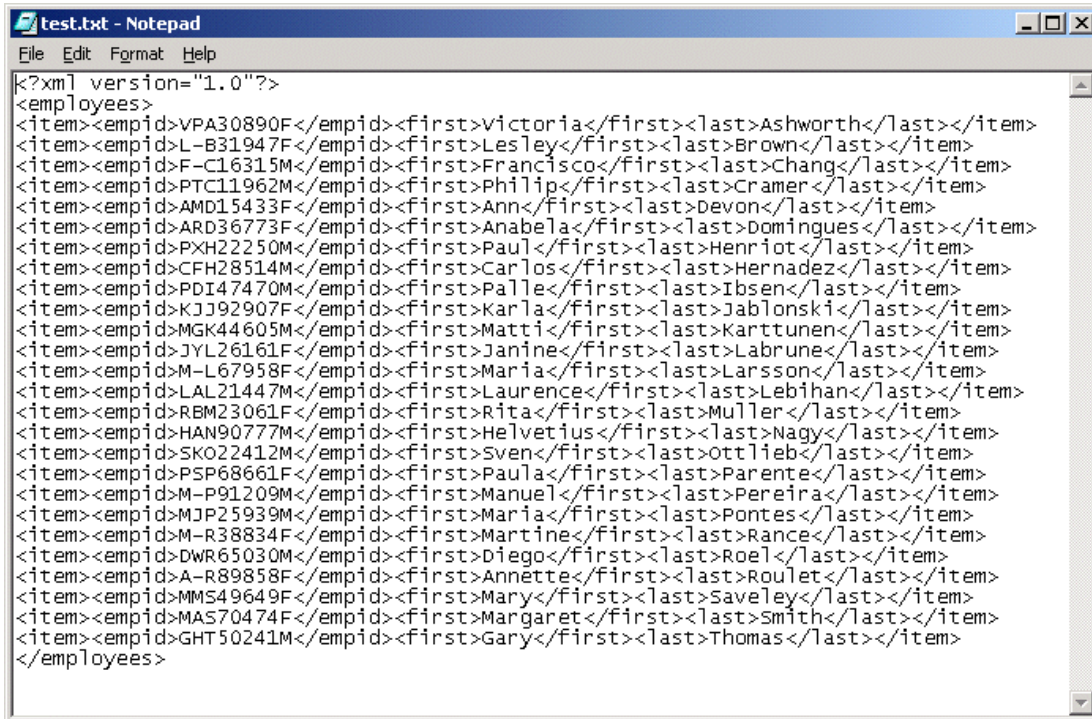
[Back to XSLT examples](#)

Source tree generation

```
<?xml version="1.0"?>
<employees>
  $TABLE('SQL', '<item><empid>\[name="emp_id"]</empid><first>\[name="fname"]</first><last>\[name="lname"]</last></item>\r\n',0)$
</employees>
```

[Back to XSLT examples](#)

Generated XML



```

<?xml version="1.0"?>
<employees>
<item><empid>VPA30890F</empid><first>Victoria</first><last>Ashworth</last></item>
<item><empid>L-B31947F</empid><first>Lesley</first><last>Brown</last></item>
<item><empid>F-C16315M</empid><first>Francisco</first><last>Chang</last></item>
<item><empid>PTC11962M</empid><first>Philip</first><last>Cramer</last></item>
<item><empid>AMD15433F</empid><first>Ann</first><last>Devon</last></item>
<item><empid>ARD36773F</empid><first>Anabela</first><last>Domingues</last></item>
<item><empid>PXH22250M</empid><first>Paul</first><last>Henriot</last></item>
<item><empid>CFH28514M</empid><first>Carlos</first><last>Hernandez</last></item>
<item><empid>PDI47470M</empid><first>Palle</first><last>Ibsen</last></item>
<item><empid>KJJ92907F</empid><first>Karla</first><last>Jablonski</last></item>
<item><empid>MGK44605M</empid><first>Matti</first><last>Karttunen</last></item>
<item><empid>JYL26161F</empid><first>Janine</first><last>Labrune</last></item>
<item><empid>M-L67958F</empid><first>Maria</first><last>Larsson</last></item>
<item><empid>LAL21447M</empid><first>Laurence</first><last>Lebihan</last></item>
<item><empid>RBM23061F</empid><first>Rita</first><last>Muller</last></item>
<item><empid>HAN90777M</empid><first>Helvetius</first><last>Nagy</last></item>
<item><empid>SKO22412M</empid><first>Sven</first><last>Ottlieb</last></item>
<item><empid>PSP68661F</empid><first>Paula</first><last>Parente</last></item>
<item><empid>M-P91209M</empid><first>Manuel</first><last>Pereira</last></item>
<item><empid>MJP25939M</empid><first>Maria</first><last>Pontes</last></item>
<item><empid>M-R38834F</empid><first>Martine</first><last>Rance</last></item>
<item><empid>DWR65030M</empid><first>Diego</first><last>Roel</last></item>
<item><empid>A-R89858F</empid><first>Annette</first><last>Roulet</last></item>
<item><empid>MMS49649F</empid><first>Mary</first><last>Saveley</last></item>
<item><empid>MAS70474F</empid><first>Margaret</first><last>Smith</last></item>
<item><empid>GHT50241M</empid><first>Gary</first><last>Thomas</last></item>
</employees>

```

[Back to XSLT examples](#)

XSLT Style Sheet

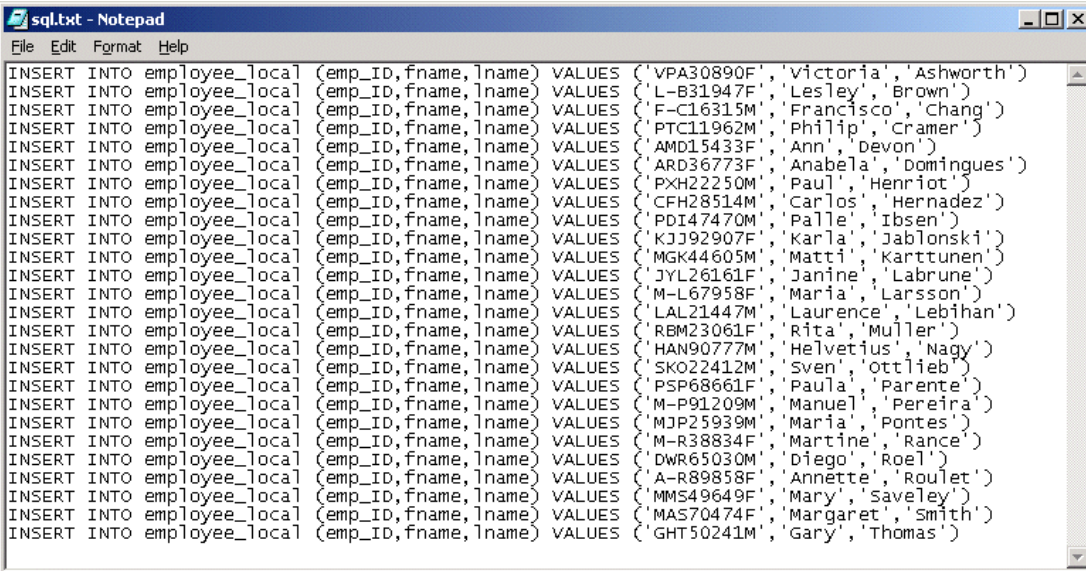
```

<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text"/>
<xsl:template match="/">
<xsl:for-each select="employees/item">INSERT INTO employee_local (emp_
ID,fname,lname) VALUES ('<xsl:value-of select="empid"/>','<xsl:value-of
select="first"/>','<xsl:value-of select="last"/>')
</xsl:for-each>
</xsl:template>
</xsl:stylesheet>

```

[Back to XSLT examples](#)

Generated SQL statement



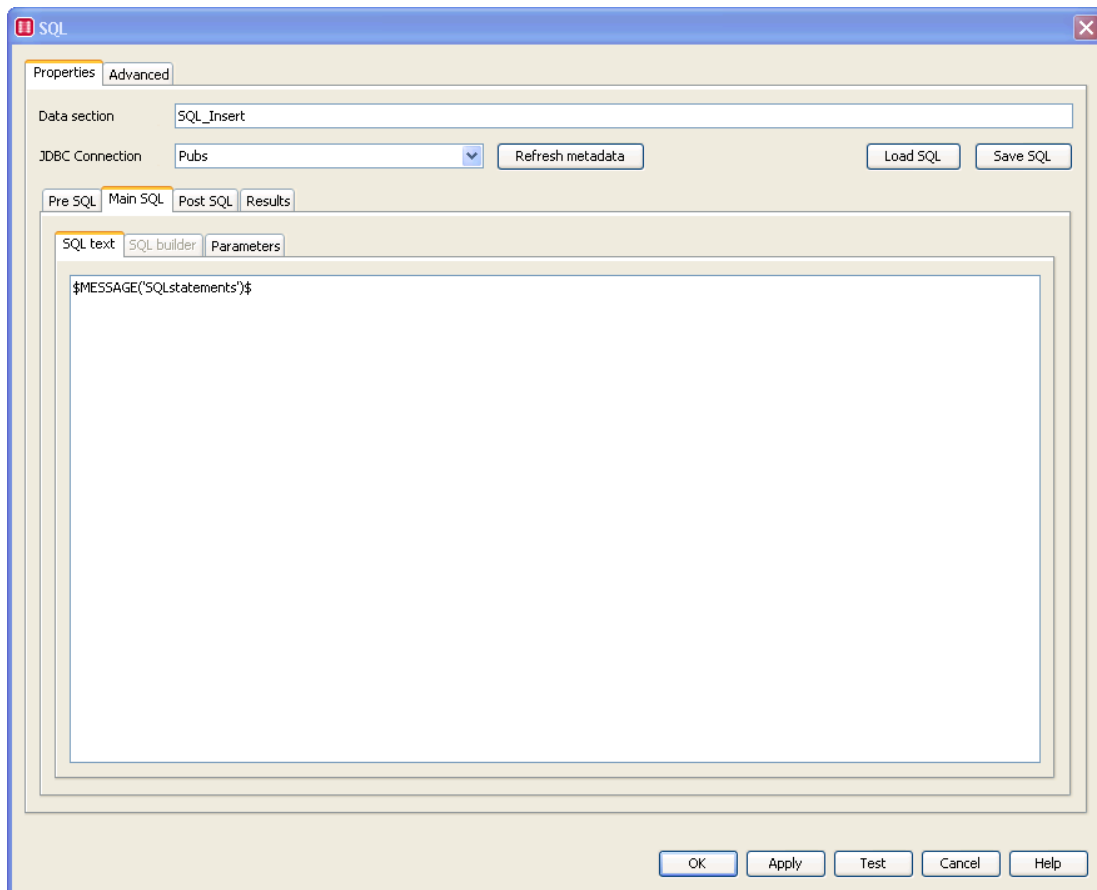
```

sql.txt - Notepad
File Edit Format Help
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('VPA30890F','Victoria','Ashworth')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('L-B31947F','Lesley','Brown')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('F-C16315M','Francisco','Chang')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('PTC11962M','Philip','Cramer')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('AMD15433F','Ann','Devon')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('ARD36773F','Anabela','Domingues')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('PXM22250M','Paul','Henriot')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('CFH28514M','Carlos','Hernandez')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('PDI47470M','Palle','Ibsen')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('KJJ92907F','Karla','Jablonski')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('MGK44605M','Matti','Karttunen')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('JYL26161F','Janine','Labruna')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('M-L67958F','Maria','Larsson')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('LAL21447M','Laurence','Lebihan')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('RBM23061F','Rita','Muller')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('HAN90777M','Helvetius','Nagy')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('SKO22412M','Sven','Ottlieb')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('PSP68661F','Paula','Parente')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('M-P91209M','Manuel','Pereira')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('MJP25939M','Maria','Pontes')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('M-R38834F','Martine','Rance')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('DWR65030M','Diego','Roel')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('A-R89858F','Annette','Roulet')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('MMS49649F','Mary','Saveley')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('MAS70474F','Margaret','Smith')
INSERT INTO employee_local (emp_ID,fname,lname) VALUES ('GHT50241M','Gary','Thomas')

```

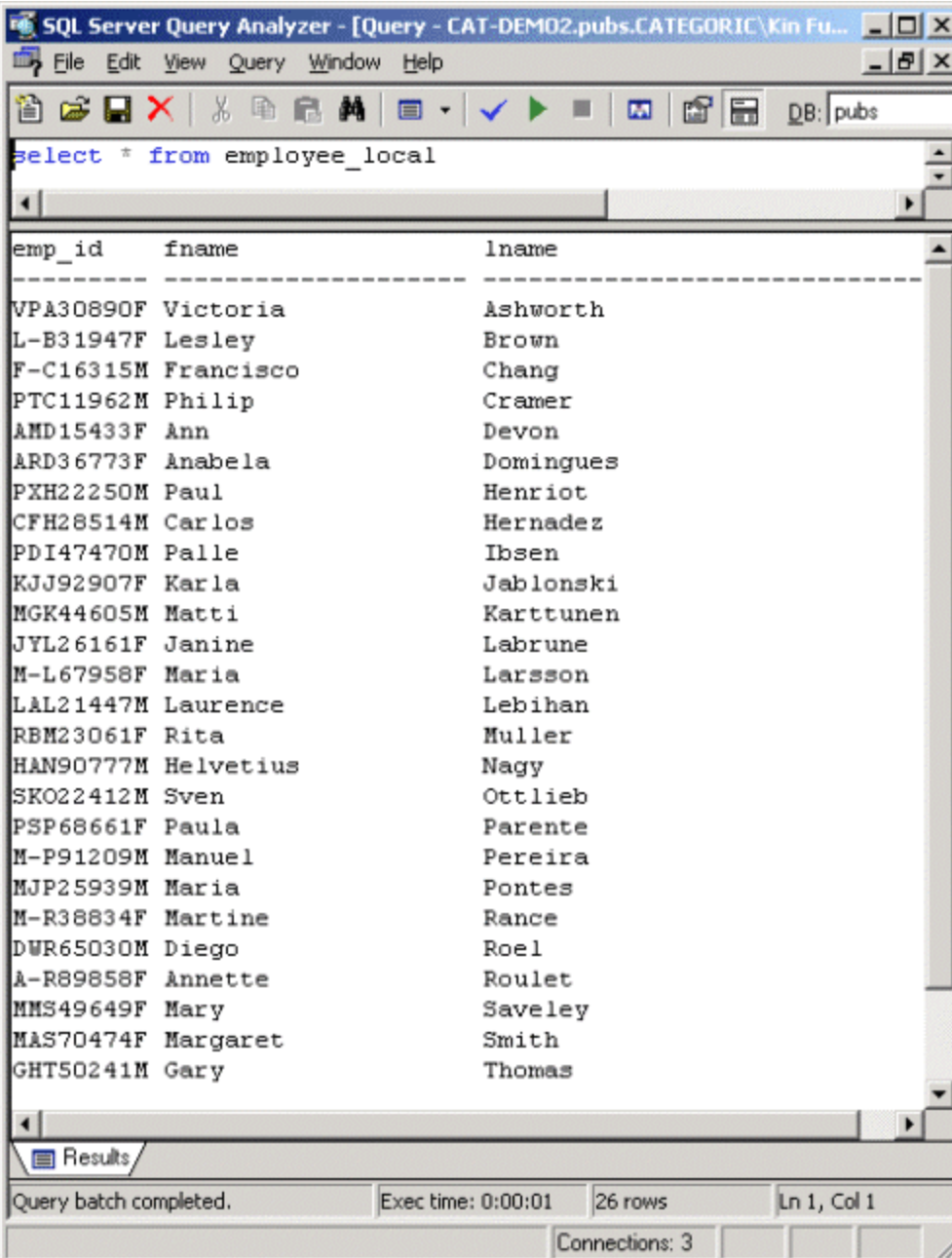
[Back to XSLT examples](#)

SQL_INSERT Properties



[Back to XSLT examples](#)

Content of the employee_local table



SQL Server Query Analyzer - [Query - CAT-DEM02.pubs.CATEGORIC\Kin Fu...]

File Edit View Query Window Help

DB: pubs

select * from employee_local

emp_id	fname	lname
VPA30890F	Victoria	Ashworth
L-B31947F	Lesley	Brown
F-C16315M	Francisco	Chang
PTC11962M	Philip	Cramer
AMD15433F	Ann	Devon
ARD36773F	Anabela	Domingues
PXH22250M	Paul	Henriot
CFH28514M	Carlos	Hernandez
PDI47470M	Palle	Ibsen
KJJ92907F	Karla	Jablonski
MGK44605M	Matti	Karttunen
JYL26161F	Janine	Labrune
M-L67958F	Maria	Larsson
LAL21447M	Laurence	Lebihan
RBM23061F	Rita	Muller
HAN90777M	Helvetius	Nagy
SKO22412M	Sven	Ottlieb
PSP68661F	Paula	Parente
M-P91209M	Manuel	Pereira
MJP25939M	Maria	Pontes
M-R38834F	Martine	Rance
DWR65030M	Diego	Roel
A-R89858F	Annette	Roulet
MMS49649F	Mary	Saveley
MAS70474F	Margaret	Smith
GHT50241M	Gary	Thomas

Results

Query batch completed. Exec time: 0:00:01 26 rows Ln 1, Col 1

Connections: 3

[Back to XSLT examples](#)

Data Filter Module

You can use the **Data filter** module to filter out either changed or unchanged data from a Data or Recipient section. It is similar to the Data filter option on the SQL module, except that it gives finer control over the type and conditions that cause the data to be filtered. It also allows data to be filtered from any data section, not just restricting the filtering to the SQL module.

Using the data filter module is an alternative to detecting changes in data which would otherwise only be detected by using database triggers or temporary tables - and both would require modifications to the database.

The data filter module can passively monitor a database table and detect changes without installing a trigger or changing the database. It can also report on what the change was, i.e. column X changed from value Y to value Z.

Example: A data section contains the ID, name, description and price of a number of items, that you would want to notify a purchaser when the price of an item changes. The first time when an EMF process is run, all data passes through the Data filter module and the purchaser receives all the pricing information (or the data filter module halts the EMF Process - see [Advanced](#) tab **On first run, build memory and halt** option).

If nothing has changed in the data section, the next time EMF process runs, all the data is filtered out and the resulting data section will be empty. For example, if the price of an item has changed, this item will not be removed and so will be included in the resulting data section.

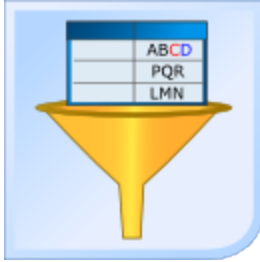
By default, all the columns in the data section are checked for changes and if the description or name of an item is changed, this would result in a change being registered, and the purchaser is then informed of a price change. In order to avoid this you can specify the columns you want to filter on (see [Specifying Columns to Filter on](#)). So in this example we will only check the price column. The price column, needs to be identified to a particular item - otherwise a price change on one item could incorrectly not get notified because the new price already existed on a different item. So a **Key column** also needs to be selected that is the items primary key in the columns to filter on.

Additionally, by default the section is only compared against its contents the previous time it was filtered. This means that if the price reverts to its original value, it is registered as a further change and another EMF process is sent out. If you wish, you can set the memory of the data filter to **return rows that are different/new when compared to ALL previous runs** and thereby compare the current value to all the previous values. By doing so, data only passes through the filter if it has never had its current value before.

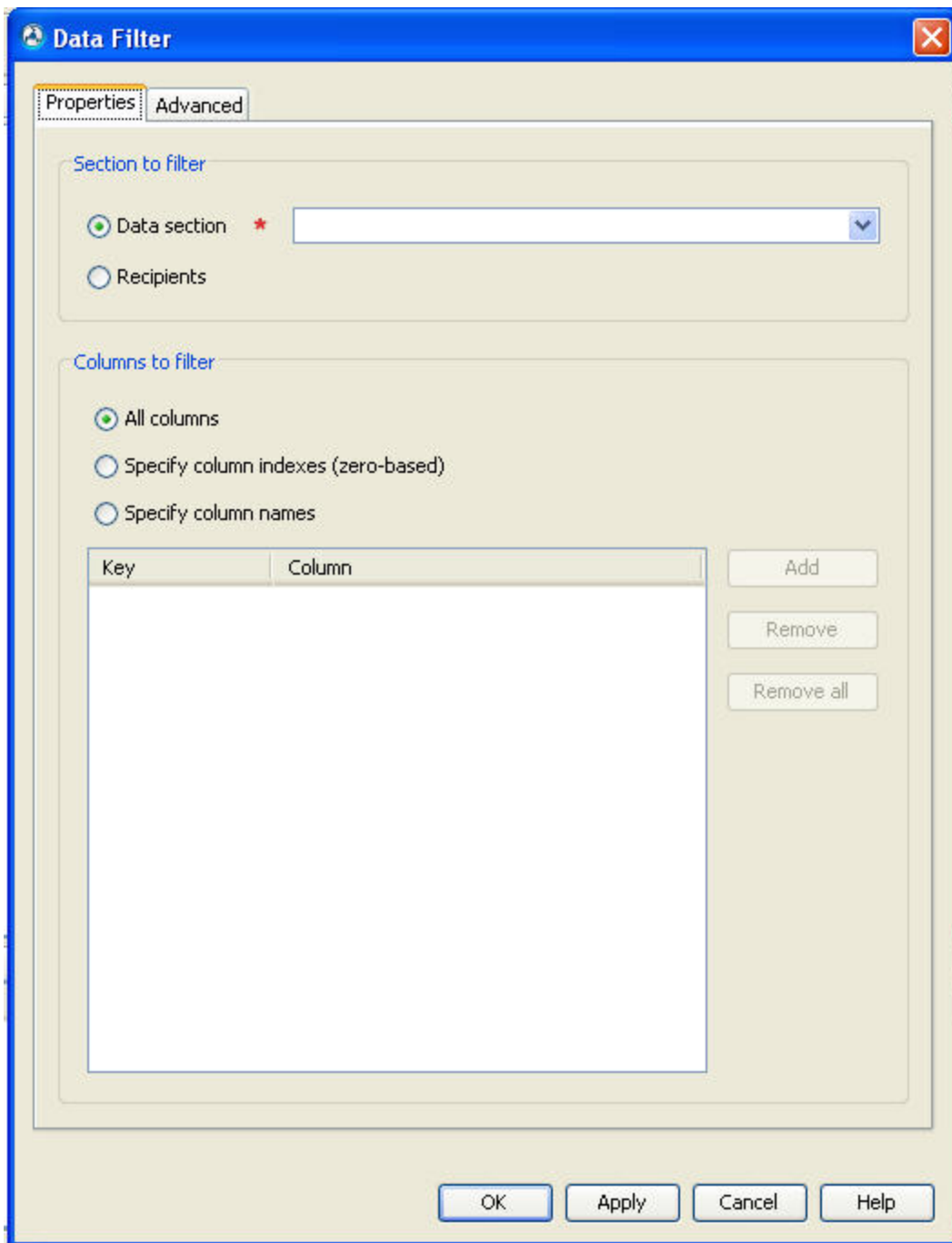
If the purchaser only wants to know about prices changes, and not when new items are added to the system then in the [Advanced](#) tab, de-select **Include new rows**.

To use a Data Filter module:

1. Drag and drop a Data filter module into the EMF Process at an appropriate place.



2. Double-click the icon to display the **Data Filter Properties** screen.



3. Select the type of filtering you want to perform. You can either filter individual recipients out of a **Recipients** section, or data from a **Data section**. If you choose Data section you must then select the required section from the drop-down list of those in the EMF Process that are available to the Data filter module (or enter its name if it does not yet exist - e.g. if it has been created by the API/script module).
4. Specify the columns that you want to include in the filtering, in the **Columns to filter** area.

5. Select the [Advanced](#) tab to specify the type of filtering to perform, halt conditions, the maximum amount of time to wait for a "lock", and, if necessary, to clear the current data store and reset the module.
6. Select **All columns** (the default option) to include all columns in the data filtering.
7. Select **Selected column indexes** if you want to define the column number(s) by entering them in the text box. **Note** that the column numbering starts from zero, so for example the third column would be number 2.
8. Select **Selected column names** if you want to define the column label(s) by entering their labels in the box.
 - If individual columns are selected to filter, then columns can also be marked as **Key**. When key columns are combined they should be able to uniquely identify a row of data. They would typically be the primary key columns of the data you are filtering. Adding Key columns allows the data filter to better identify the changes in the data. You are not monitoring for changes in the key columns, but the columns that aren't marked as key.
 - If no **key** columns are identified, then it is not possible to use the functionality of **Return previous data values** and of not **Including new rows** in the [Advanced Data Filter Options](#).

Important: if you change any of the settings on this page after the Data filter module has been used in an EMF Process, any existing non-replication data may be lost. This will cause the Data filter module to run as if for the first time.

[Filters and Transformation Modules](#)

[Advanced Data Filter Options](#)

Datasection Toolbox Module

The **Datasection Toolbox module** provides a useful set of functions for manipulating an EMF process data section. The **toolbox** provides the following functions:

- **Copy** - create a copy of a data section.
- **Rename columns** - rename the columns of a data section.
- **Rename section** - rename the data section.
- **Filter** - filters the data section based on an expression, removing the rows that don't match the criteria. Similar to a WHERE clause in SQL.
- **Join** - similar to database joins, data sections can be joined on selected columns. All the standard join types are supported (full, inner, left outer, right outer, cross) - so data from heterogeneous data sources can be joined.
- **Merge** - allows one data section to be merged with another. Generally the two data sections will contain the same columns.
- **Sort** - allows the user to sort data in a data section.

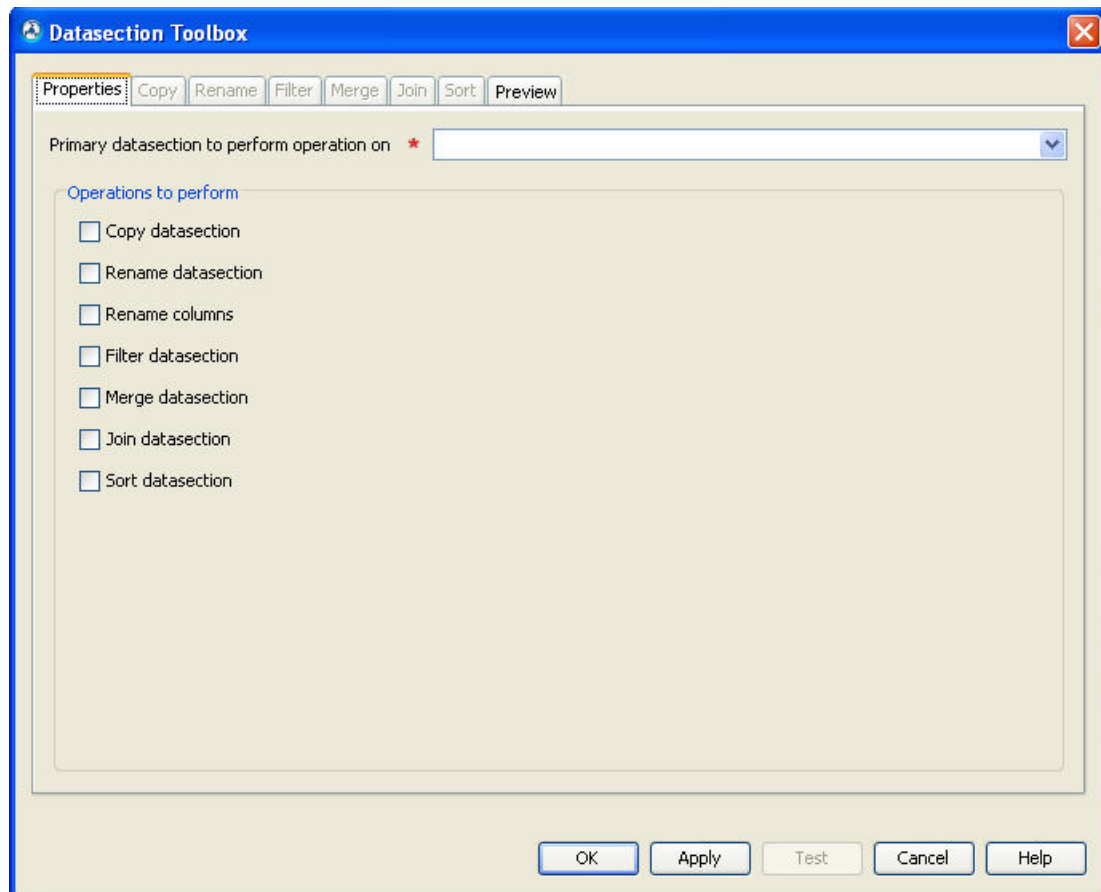
To use the Datasection Toolbox Module:

1. Drag the **Datasection Toolbox** icon onto the EMF process Design graph at the

appropriate place.



2. Double-click the Datasession Toolbox icon to display the Datasession Toolbox **Properties** screen.



3. In the Properties tab select the data section that you wish to perform the operation(s). Depending on the operation(s) selected this may or may not change this data section. For instance, selecting to *copy* a data section and to *filter* it, will cause the data section to be copied and the new, copied, data section to then be filtered. If the operation selected was just filter, then the selected primary data section would be changed by the filter.

Note: Operations are performed in the order they appear in the tabs, from left to right, so if copy is selected, then all operations following this occur on the copied data section.

4. Select the check boxes for the relevant **operations to perform** on the selected data section. As each checkbox is selected the appropriate tab becomes enabled. The tab should then be selected to fill in the required details about that operation.
 - [Copy datasection](#) - creates a copy of a data section.
 - [Rename datasection](#) - rename the data section.
 - [Rename columns](#) - rename the columns of a data section.
 - [Filter datasection](#) - filters the data section based on an expression, removing the rows that don't match the criteria. Similar to a WHERE clause in SQL.
 - [Merge datasections](#) - allows one data section to be merged with another. Generally the two data sections will contain the same columns.
 - [Join datasections](#) - similar to database joins, data sections can be joined on selected columns. All the standard join types are supported (full, inner, left outer, right outer, cross) - so data from heterogeneous data sources can be joined. A new data section is created from the joined data sections.
 - [Sort data sections](#) - You can sort the data sections.
5. Click **Test** to test the results on the Preview tab.

[Copy datasection](#)

[Rename datasection](#)

[Rename columns](#)

[Filter datasection](#)

[Merge datasections](#)

[Join datasections](#)

[Sort datasections](#)

[Filter and Transformation Modules](#)

Duplicates Module

This module allows the analysis of a data section for duplicate values (more than one occurrence of values). The module is typically used in highlighting suspicious data that should be investigated further. i.e. potentially fraudulent transactions. For example a typical use would be in the detecting of duplicate invoice numbers from suppliers to check that they are not "double invoicing". Far more complex rules can be built through, such as, detecting similar invoice amounts (say within 5% of each other) from suppliers that have the same postcode or the same company name where the invoice date is within 30 days of each other.

A duplicate value is where two or more rows in a data section contain a value that is the same (or meets the defined tolerance) in the same column.

Row ID	Invoice Number	First Name	Last Name
1	49643	John	Terry

2	63455	Terry	Waite
3	49643	Paul	Good

For example, in the above data set the invoice number for Row 1 and Row 3 is "Duplicated", and this would be returned in the results, if the "duplicate" comparison was being done on the Invoice number/span> column. Note that no names are duplicated, even though "Terry" appears in Row 1 and Row 3, as they are in different columns.

To analyze a set of data a data section must exist that contains one or more columns of data to be analyzed. On analyzing the data the module will then generate a new data section containing the desired information - this may be the rows with duplicate data, or the rows without duplicate data.

To use the Duplicates module:

1. Drag the **Duplicates** icon onto the EMF Process Design graph at the appropriate place.



2. Double-click the Duplicates icon to display the **Duplicates > Properties** screen:

Duplicates

Properties Preview

Data section to check *

Columns to match

☒ Specify by index (zero-based)
☐ Specify by name

Column	Match as	Case sensitive	Ignore whitespace	Ignore special	Tolerance	Tolerance unit	Match Nulls	Operator	Group

Add
Remove
Remove all
Move up
Move down

Results

Results data section name Duplicates

Records to return

☒ Duplicates (grouped)
☐ Duplicates (maintain original order)
☐ Those without duplicates (maintain original order)

Columns to return

☒ All columns from original data section
☐ Only those columns used in the test (unique rows only)

OK Apply Test Cancel Help

3. In the **Data section to check** select the data section that contains the data to be analyzed.
4. In the **Columns to match** select whether you are specifying columns by index (e.g. 0,1,2,3) or by the column name. Note that the column numbering starts from zero, so for example the third column would be number 2.
5. Press the **Add** button to add a new column to analyze. A blank row will appear in the table. In this row the column and type of analysis to perform is defined. Each of the columns is explained below:
 - **Column** - Contains either the column name or the column index that analysis is being performed on. This depends on what option was selected in **Columns to match**.
 - **Match as** - Select the type of data that the column contains, and how best to search that column for duplicates. Depending on the value selected, different options will be enabled/disabled.
 - **String** - This will match any text in the selected column against text in the same columns in other rows. Three additional match options are available if this option is selected:
 - **Case sensitive** - If selected the matching will be case sensitive. e.g. "London" would not match "london" if selected.
 - **Ignore whitespace** - If selected, any extra white space (tabs, extra spaces, line feeds) are ignored when matching. e.g. "10 The Road" would match "10 The Road" if selected.
 - **Ignore special** - If selected, only letters and numeric characters are used when matching. Any other special characters are ignored. For example "INV20080406/abc" would match "INV20080406abc" if selected.
 - The **Use absolute values, Tolerance** and **Tolerance unit** columns are not applicable for this option.
 - **Date** - This will match the selected column as a "Date". All rows must contain a valid "Date", a string that the system can treat as a date, or a database null. If not, the module will return an error that the column value could not be converted to a date. Additional options are available if this option is selected to enter a Tolerance that is applied during matching.
 - **Tolerance** - this is the amount of days/weeks/months/years within which a value in the same column in another row can be, for this to be treated as a match. For example, if 2 days was entered and the data set contained 3 dates, 5 May 2009, 10 May 2009, 12 May 2009 then 10 May 2009 and 12 May 2009 would match and be treated as a duplicate value. An example of where this could be useful. If you knew that your suppliers should only invoice you once a month then it would be possible to check for multiple invoices within the same month from the same supplier. If this value is left blank then the dates must match exactly to be treated as a duplicate.
 - **Tolerance Unit** - This is the unit that the tolerance amount above is defined in, Days/Weeks/Months/Years.

- The **Use absolute values, Case sensitive, Ignore whitespace** and **Ignore special** columns are not applicable for this option.
- **Numeric** - This will match the selected column as a "Numeric". All rows must contain a valid number, a string that the system can treat as a number, or a database null. If not, the module will return an error that the column value could not be converted to a number. Options are available to specify Absolute values and to enter a Tolerance that is applied during matching.
 - **Use absolute values** - If selected, only absolute values will be used when matching numeric comparisons. For example, "1234" would match "-1234", if selected.
 - **Tolerance** - this is either an amount added to/removed from the value that is being matched, or a percentage that a value in the same column in another row can be within, for this to be treated as a match. For example, if 10% was entered and the data set contained 3 numbers, 80, 90 and 100 then 90 and 100 would be treated as a duplicate value because 90 is within 10% of 100. Another example, if +/- 15 was selected on the same 3 numbers then all 3 would be returned as duplicates as 80 is within 15 of 90 and 100 is also within 15 of 90. If this value is left blank then the values must match exactly to be treated as a duplicate.
 - **Tolerance Unit** - this is whether the tolerance is a percentage % or must be within a certain absolute amount +/-.
 - The **Case sensitive, Ignore whitespace** and **Ignore special** columns are not applicable for this option.
- **Object** - This will match exactly the value in the column specified rather than potentially converting it to one of the other types specified above. This option is best selected if performing exact matching only, or the column type is not known. The **Use absolute values, Case sensitive, Ignore whitespace, Ignore special, Tolerance** and **Tolerance Unit** columns are not applicable for this option.
- **Match Nulls** - If the data contains any NULL values and this option is selected, then more than one NULL appearing in the specified column will be treated as a duplicate value. If it is not selected then NULL values will be ignored and not treated as duplicates.

For example if we match the following data set on the "Delivery Date" column:		Result of Match Nulls selected	
Row ID	Delivery Date	Row ID	Delivery Date
1	12/5/2008	2	1/6/2008
2	1/6/2008	3	<NULL>
3	<NULL>	4	1/6/2008
		5	<NULL>

		Result of Match Nulls not selected	
4	1/6/2008		
5	<NULL>	Row ID	Delivery Date
6	3/6/2008	2	1/6/2008
		4	1/6/2008

- **Operator** and **Group** - The operator and group columns define how multiple duplicate checks are combined. If only one check is being performed then these columns can be ignored. Multiple checks can be combined using the logical **AND** and **OR** operators. So for example, if a data set contains the columns "first name" and "last name" then a duplicate check could be set-up looking for people with the same name by adding two checks (lines) into the "columns to match" table, with the first set to check "first name" and the second "last name" and then AND the results together. This will then only return duplicates which have the same first name and last name. The operator acts on the line it is defined on with the line directly underneath it. Lines are evaluated top down. So the first line is evaluated first to find the duplicates, then the second line - and then the two lines are combined using the selected operator. Grouping lines together using the group column allows the order of evaluation to be changed. Putting two lines in the same group means these lines will be evaluated together before other lines. It is the equivalent of putting parenthesis around an expression. Any number can be entered into the group field to identify a group. Lines with the same identifier must be adjacent to each other.

So for example, to find duplicates where the first and second names are the same or the street address and postcode are the same could logically be written as:

("First name" AND "Second name") OR ("Street address" AND "Postcode")

To define this in the **columns to match** table, select the Ignore whitespace check box for all the entries and specify the Operators

Duplicates

Properties | Preview

Data section to check: QUEUEIN_DATA

Columns to match

☐ Specify by index (zero-based)

☒ Specify by name

Column	Match as	Case sensitive	Ignore whitespace	Ignore special	Use absolute values	Tolerance	Tolerance unit	Match Nulls	Operator	Group
Name	String	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>	AND	1
String	String	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>	AND	1
String	String	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>	AND	1

Buttons: Add, Remove, Remove all, Move up, Move down

Results

Results data section name: Duplicates

Records to return

☒ Duplicates (grouped)

☐ Duplicates (maintain original order)

☐ Those without duplicates (maintain original order)

Columns to return

☒ All columns from original data section

☐ Only those columns used in the test (unique rows only)

Buttons: OK, Apply, Test, Cancel, Help

- Enter the name of the data section in the **Results data section name** field that will contain the duplicate results when the module runs.
- Select the **Records to return**. There are 3 options, explained below using the following sample data (assuming matching on the "Invoice Number" column and the "All columns from original data section" option is selected):

Row ID	Invoice Number	Company Name
1	1245	Burton Bee Products
2	1267	Burton Bee Products
3	ABC74332	Fisher
4	1245	Burton Bee Products
5	ABC74333	Fisher
6	ABC74332	Fisher Ltd

- Duplicates (grouped)** - this will return only those records that have duplicates (more than one occurrence) as defined by the rules in the Columns to match table. The results will be logically grouped together using the column defined in the first row of the Columns to match. This method allows easy visual inspection of the results. For example, if invoice number 4356 appears 3 times in a set of data, those lines would be output together in the results rather than having to scan all the results for the corresponding duplicate records.

Row ID	Invoice Number	Company Name
1	1245	Burton Bee Products

4	1245	Burton Bee Products
3	ABC74332	Fisher
6	ABC74332	Fisher Ltd

- **Duplicates (maintain original order)** - this will return only those records that have duplicates (more than one occurrence) as defined by the rules in the Columns to match the table. The order of the results will be the same order as they appeared in the original data set.

Row ID	Invoice Number	Company Name
1	1245	Burton Bee Products
3	ABC74332	Fisher
4	1245	Burton Bee Products
6	ABC74332	Fisher Ltd

- **Those without duplicates (maintain original order)** - this will return only those records that have **no** duplicates as defined by the rules in the Columns to match table. The order of the results will be the same order as they appeared in the original data set.

Row ID	Invoice Number	Company Name
2	1267	Burton Bee Products
5	ABC74333	Fisher

8. Select **columns to return**. There are two possible options, explained below using the same example data above (assuming matching on the "Invoice Number" column and the "Duplicates (grouped)" option is selected):

- **All columns from original data section** - this will return all the columns in the original data set that have duplicates as defined by the rules in the Columns to match table.

Row ID	Invoice Number	Company Name
1	1245	Burton Bee Products
4	1245	Burton Bee Products
3	ABC74332	Fisher
6	ABC74332	Fisher Ltd

- **Only those columns used in the test (unique records only)** - this will return only the columns used for matching. The records will be unique, giving a summary view of the values that have duplicate values.

Invoice Number
1245
ABC74332

9. Click **Test** to review the results in the **Preview** tab.

[General Modules](#)

Gap Detection Module

This module allows the analysis of a data section for gaps in sequences. It can detect gaps in alpha numeric sequences and in date sequences. Typically used in highlighting missing invoice numbers, cheque numbers etc. Date sequences could be checked to find gaps in staff attendances.

A gap is where an expected value is missing. The values in the dataset do not need to be ordered, the module will order the values before looking for gaps.

Row ID	Invoice Number	First Name	Last Name
1	1003	John	Terry
2	1006	Terry	Waite
3	1005	Paul	Good
4	1007	Jack	Newman
5	1009	Julie	Good

For example, in the above data set the invoice numbers 1004 and 1008 are missing. Therefore the resultset would contain two rows, with the values 1004 and 1008. Those results could then be output to the relevant authority to investigate why they were missing.

To analyze a set of data a data section must exist that contains the column of data to be analyzed. On analyzing the data the module will generate a new data section containing the missing values.

If the data in the column being analyzed contains letters then the letters will be incremented through the alphabet unless excluded using the "mask". So in the following example the invoice numbers "100AY" and "100BA" would be detected as missing.

Row ID	Invoice Number
1	100AX
2	100AZ
3	100BB
4	100BC
5	100BD

Gaps in date sequences can be detected in similar ways. It is possible to specify increment sizes and types, so it is possible to detect, for example, if a weekly, monthly or yearly report is missing.

To use the Gap Detection module:

1. Drag the **Gap Detection** icon onto the EMF Process Design graph at the appropriate place.



2. Double-click the **Gap Detection** icon to display the Gap Detection **Properties** screen:

3. In the **Data section to check** select the data section that contains the data to be analyzed.
4. In the **Column to check** select whether you are specifying columns by index (e.g. 0,1,2,3) or by the column name. **Note** that the column numbering starts from zero, so for example the third column would be number 2. Enter the column name or number in the field provided. This field also accepts dynamic functions.
5. Select the type of data that the column being analyzed contains. This can either be "Date" if the data is a date, or "Alphanumeric" if the data contains numbers, letters or a combination of both. Letters are incremented through the Western alphabet, where "A" is the first character and "Z" the last character - all other characters are ignored.

Depending on the data type selected different options are available.

Alphanumeric options:

- **Apply mask** - if the checkbox is selected then a mask pattern can be entered to remove certain characters from the data being processed. For example, if all document numbers started with the characters INV for invoice or CRE for credit note (e.g. INV1001, INV1002, INV1003, CRE1004, INV1005) then the first 3 characters shouldn't be used to check the gaps in the sequence. Therefore it is possible to define a mask to say which characters to ignore. In this example the mask would look like *****9999** where the asterisk is used to mark the characters to ignore, any other character can be used to mark characters that should be processed but to keep it simple it is recommended to use the same character throughout, such as a '9'. The asterisk can appear anywhere in the sequence, so if the data looked like 10-INV-23, then a mask of **99*****99** may be appropriate.
- **Start Range** - two options are available:
 - "Smallest item" means that gaps will be checked from the smallest value in the data being processed. So if the data is 1001, 1002, 1004 then gaps are checked from value 1001 onwards.
 - "Use defined value" allows the value of the first item to be specified. For example, if a cheque book has been issued and the first cheque number is known then gap detection should start at this number rather than the "smallest item", as the first cheques could be missing from the system. So if a value of 1000 was specified and the data contained 1001 and 1002 then 1000 would be reported as a gap. Dynamic functions can be used to specify the start range - this could be useful to specify the start value to check data only from where the last check finished off.
- **End Range** - two options are available:
 - "Largest item" means that gaps will be checked up to the largest value in the data being processed. So if the data is 1001, 1002, 1004 then gaps are checked up to value 1004.
 - "Use defined value" allows the value of the last item to be specified. For example, if a cheque book has been issued and the last cheque number is known then gap detection should end at this number rather than the "largest item", as the last cheques could be missing from the system. So if a value of 1004 was specified and the data contained 1001 and 1002 then 1003 and 1004 would be reported as a gap. Dynamic functions can be used to specify the end range.
- **Increment Size** - specifies the step size that the data is expected to go up in. For example, numbers may be expected to increment by 10 in the data set 10010, 100020, 10030, 10050 (10040 would be reported as missing).

Date options:

- **Start Range** - two options are available:
 - "Smallest item" means that gaps will be checked from the earliest date in the data being processed.

- "Use defined value" allows a date to be specified that checking should start from.
- **End Range** - two options are available:
 - "Largest item" means that gaps will be checked up to the largest date in the data being processed.
 - "Use defined value" allows date to be specified that checking should end at.
- **Increment Size** - specifies the date unit type and increment step size. This setting has a big bearing on how the results are calculated.
 - "day(s)" - if selected then the gap detection will increment through the calendar in days (using the specified step size). Certain days of the week can be skipped (see *Days to exclude* tab below). The results will contain the dates of the missing days.
 - "week(s)" - if selected then the gap detection will increment through the calendar a week at a time. The start day of the week depends on the locale of the computer (USA it is Sunday, most of Western Europe it is Monday). For a gap to be reported then a date must not be present for the week being processed. The final results containing the gaps will report the dates of the first day of the weeks that had no entries. If a whole week is selected for exclusion in the *Days to exclude* tab then this week will not be reported as a gap. However, if a single day in that week is not excluded and that day did not appear in the data set then that week would be reported as a gap week.
 - "month(s)" - if selected then the gap detection will increment through a calendar month at a time. For a gap to be reported then a date must not be present for the month being processed. If at least one non excluded date is present for a month it will not be reported as a gap. The final results containing the gaps will report the dates of the first day of the month that had no entries. If a whole month is selected for exclusion in the *Days to exclude* tab then that month will not be reported as a gap. However, if a single day in that month is not excluded and that day did not appear in the data set then that month would be reported as a gap month.
 - "year(s)" - if selected then the gap detection will increment through a calendar a year at a time. For a gap to be reported then a date must not be present for the year being processed. If at least one non excluded date is present for a year it will not be reported as a gap. The final results containing the gaps will report the dates of the first day of the year that had no entries. If a whole year is selected for exclusion in the *Days to exclude* tab then that year will not be reported as a gap. However, if a single day in that year is not excluded and that day did not appear in the data set then that year would be reported as a gap year.

Enter in the **Results Datasection name** field the name of the data section that will contain all the gaps detected. This section contains a single column that will be named the same as the column containing the data being tested. It will contain either string values if alphanumeric column type was select, or date values if Date column type was selected. If no gaps were detected the result datasection will be empty (zero rows).

Days to exclude tab

The days to exclude tab is only enabled if Date field comparison is being used. The top section allows individual days of the week to be excluded. This is only enabled if the increment setting is set to "day(s)". For example if you knew that the data never included Saturdays or Sundays then you would not want these reported as gaps and you could exclude them all by selecting Saturday and Sunday.

The "Days to exclude" calendar at the bottom allows specific dates to be excluded, such as national holidays or weeks where a factory shuts down. This can be used to exclude dates that are known to be missing so that they will not be reported as gaps.

Date gap detection example

The following data contains monthly company report publishing dates. The settings of smallest value to largest value and an increment size of 1 month could be used to see if the company had failed to publish a report in a given month:

Report Date

11th March 2009
19th March 2009
7th April 2009
7th April 2009
17th May 2009
1st July 2009
5th January 2010

The results generated in the results data section would be as follows:

Report Date

1st June 2009
1st August 2009
1st September 2009
1st October 2009
1st November 2009
1st December 2009

6. Click **Test** to view the results based on the options you have selected in the Preview tab.

[Filter and Transformation Modules](#)

White List Event Module

[White lists overview](#)

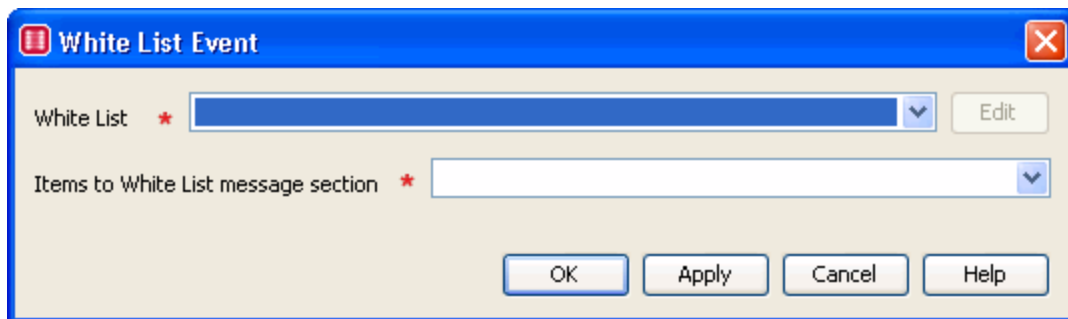
The white list event module is used to add (mark) items to a white list. It records those items that you do not want to be notified about again in future.

To add a White List Event module to an EMF Process:

1. In the EMF Process builder, drag the icon for the White List Event module to the appropriate place in your EMF process.



2. Open the White List Event module to display its properties. By default, you need to double-click the icon to open, but you can configure mouse-click actions on the Options window. From the Tools menu, select **Options -> EMF-> Process Builder Behaviour** tab.



3. Select the **White List** to which you want to add white listed items. You can click **Edit** to modify the properties of the selected White List.
4. **Items to White List message section:** From the drop-down list, select the message section that contains the XML message containing details of the items to white list. The format of this message section is explained below.

White List Event XML Format

The format of the XML must follow the structure defined as:

```
<?xml version="1.0" encoding="utf-8"?>
<whitelist version="1">

  <record user="" comment="" until="" for="">
    <key fieldname=""></key>
    <key fieldname=""></key>
    <key fieldname=""></key>
    . . .
  </record>
  <record user="" comment="" until="" for="">
    . . .
  </record>
  . . .
</whitelist>
```

Multiple `<record>` elements may appear in a single document.

The **user** attribute allows the name of the user who is white listing the item to be recorded. The **comment** attribute allows some text to be defined for why an item is being white listed. Both these items are optional and for reference only. They are displayed if the user selects **View list** in the white list definition and also can be returned as additional fields in the white list filter.

The **until** attribute defines how long an item is white listed for. When that time has passed, the item is automatically removed from the white list (when the white list event module runs next). The **until** field is formatted using the xsd dateTime type format. For example:

```
until="2013-05-22"
or
until="2013-05-22T21:12:00"
```

The lexical space of xsd: dateTime is the format defined by ISO 8601 of the form:

```
[ - ]CCYY-MM-DDThh:mm:ss[Z](+|-)hh:mm]
```

The **for** attribute is an alternative method of defining for how long to white list an item. It allows a period to be defined, specifying how long an item will be white listed for (this avoids the person creating the XML having to perform date calculations). It is defined using the xsd duration type. For example:

```
for="P5M" indicates 5 months, and
for="PT5M" indicates 5 minutes.
```

The lexical space of xsd:duration is also the format defined by ISO 8601 of the form:

```
PnYnMnDTnHnMnS
```

The **until** and **for** attributes are mutually exclusive. If neither is present, the item is white listed forever.

For each key column defined in the white list definition, there has to be a corresponding **key** tag with the **fieldname** attribute set to match the column name. The value of the key tags contain the values that identify the item to white list.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<whitelist version="1">

  <record for="PlM">
    <key fieldname="companyName">ABC Telecom</key>
    <key fieldname="address1">10 The Park</key>
    <key fieldname="address2">London</key>
  </record>
</whitelist>
```

When a white list item is added with the same key values as an existing item, then the existing item (expiry/user/comment) is overwritten.

When a white list event module runs, it also removes any expired items from the white list. If it is required to remove expired items, then schedule the white list event module to run against the required white list with an empty white list XML section (no records).

```
<?xml version="1.0" encoding="utf-8"?>
<whitelist version="1">
</whitelist>
```

This would be advisable if you had many items that had expired and wanted to free up storage and improve white list filtering performance.

As well as adding white list events, it is possible to “trick” the white list module to remove events. Simply pass in the XML in the same way you would for adding an event, but set the period to

for="-PT1S"

This will white list that item for 1 second in the past, overwriting any previous expiry period and effectively making that item “expired”. Using this functionality, it is possible to build “un-white list” functionality in a UI.

[White List Overview](#)

[White List Definition](#)

[White List Filter Module](#)

White List Filter Module

[White lists overview](#)

The white list filter module is used to filter out (or mark) rows in a datasection that match previously white listed items. It will scan each row in a selected datasection to check it against the data in the selected white list definition.

To add a White List Filter module to an EMF Process:

1. In the EMF Process builder, drag the icon for the White List Filter module to the appropriate place in your EMF process.



2. Open the White List Filter module to display its properties. By default, you need to double-click the icon to open, but you can configure mouse-click actions on the Options window. From the Tools menu, select **Options -> EMF-> Process Builder Behaviour** tab.

3. Select the **White list name** that will be used to filter the data section. You can click **Edit** to modify the properties of the White list name.
4. Select a data section to filter from the **Data section to filter** drop-down list.
5. There are two possible actions to take when a white listed entry is detected in the datasection:
 - **Filter row out of data section:** Any row of data that matches a white listed item will be removed from the datasection.
 - **Mark item as white listed in additional 'white listed' column:** Rather than filtering out the row, select this option to append extra columns to the datasection that will contain information about whether a row is white listed or not. There are up to 4 extra columns that can be added, and the name can be defined for each.
 - **White Listed:** This column is always added to the datasection if **Mark item as white listed in additional 'white listed' column** is selected. Each row in this column will contain either **TRUE** or **FALSE** to indicate whether the row is white listed or not.

- **Expiry:** If the **Include 'Expiry' column** is selected, then this additional column will be added. Each row in this column will be blank if the item is not white listed. If the item is white listed, and has no expired date, then it will also be blank. If the item is white listed and there is an expiry date of when the white listed item expires, then this column will contain that value.
- **User:** If the **Include 'User' column** is selected, then this additional column will be added. Each row in this column will be blank if the item is not white listed. If the item is white listed, then it will contain the user name of the person who white listed the item (if any). This is the value of the **user** attribute passed in the XML document to the white list event module.
- **Comment:** If the **Include 'Comment' column** is selected, then this additional column will be added. Each row in this column will be blank if the item is not white listed. If the item is white listed, then it will contain the comment set when the item was white listed (if any). This is the value of the **comment** attribute passed in the XML document to the white list event module.

6. Press the **Test** button to preview the datasection being filtered.

[White List Overview](#)

[White List Definition](#)

[White List Event Module](#)

Data Transformation Module

The **Data Transformation** module allows transformations to be mapped between EMF data sections, XML documents, and JSON documents. The following are supported mappings:

- XML to XML
- Data section to XML
- XML to Data section
- Data section to Data section
- JSON to Data section
- Data section to JSON

For example, with the following XML document:

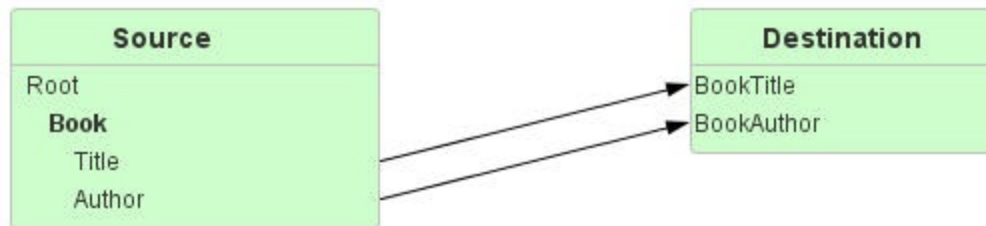
```
<?xml version="1.0" ?>
<Root>
  <Book>
    <Title>The Hobbit</Title>
    <Author>J R R Tolkien</Author>
  </Book>
  <Book>
    <Title>Harry Potter and the Philosopher's Stone</Title>
    <Author>J K Rowling</Author>
  </Book>
</Root>
```

```
</Book>
</Root>
```

If you want to convert and present the data in the following format:

BookTitle	AuthorName
The Hobbit	J R R Tolkien
Harry Potter and the Philosopher's Stone	J K Rowling

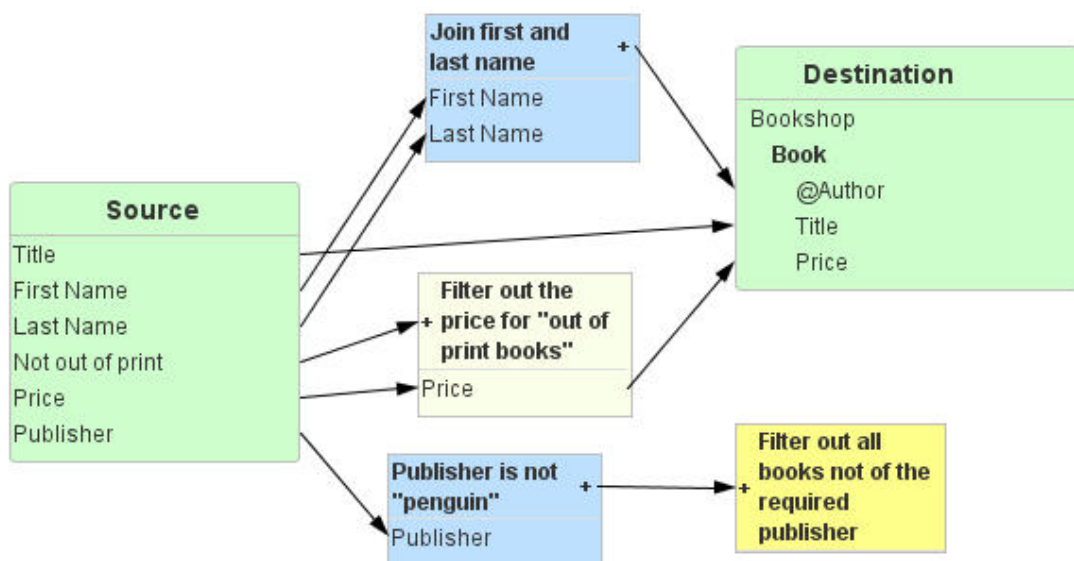
Then, the following data transformation can be defined:



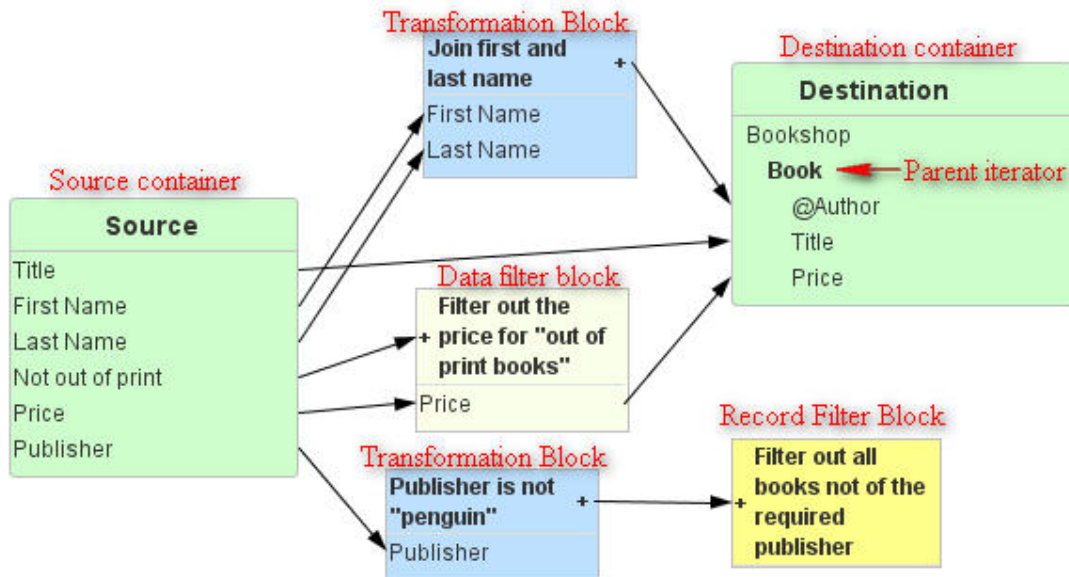
When this module runs through the server at runtime, it will convert the incoming XML into a data section of the appropriate format.

In addition, there are three different types of "blocks" that can be added to perform additional transformations and filtering. The following image shows an example of how those blocks are combined to build a process that converts a data section containing fields about "books" into an XML document needed by a bookshop. It performs the additional transformations and filtering:

1. Combines the "first name" and "last name" fields into a single attribute called "Author" in the XML.
2. If the book is out of print, then the "price" element is not generated in the final XML.
3. Only books by the publisher "penguin" are included in the final results, all others are filtered out.



To help understand the concepts described, the following image labels all the component parts of the data transformation module.



When a data transformation runs, the following points help you to understand how the server processes it:

- If the source is a data section, then it will run the whole transformation once per row of the data section.
- If the source is an XML document, then it will run the whole transformation once for each *parent iterator* element in the source XML. Hence, it is important to set the correct parent iterator to get the desired output.
- If the source is a JSON document, then it will run the whole transformation once for each item in the JSON array that is referenced by the *parent iterator* element in the source JSON. Hence, as with XML, it is important to set the correct parent iterator to get the desired output.

For the generated output:

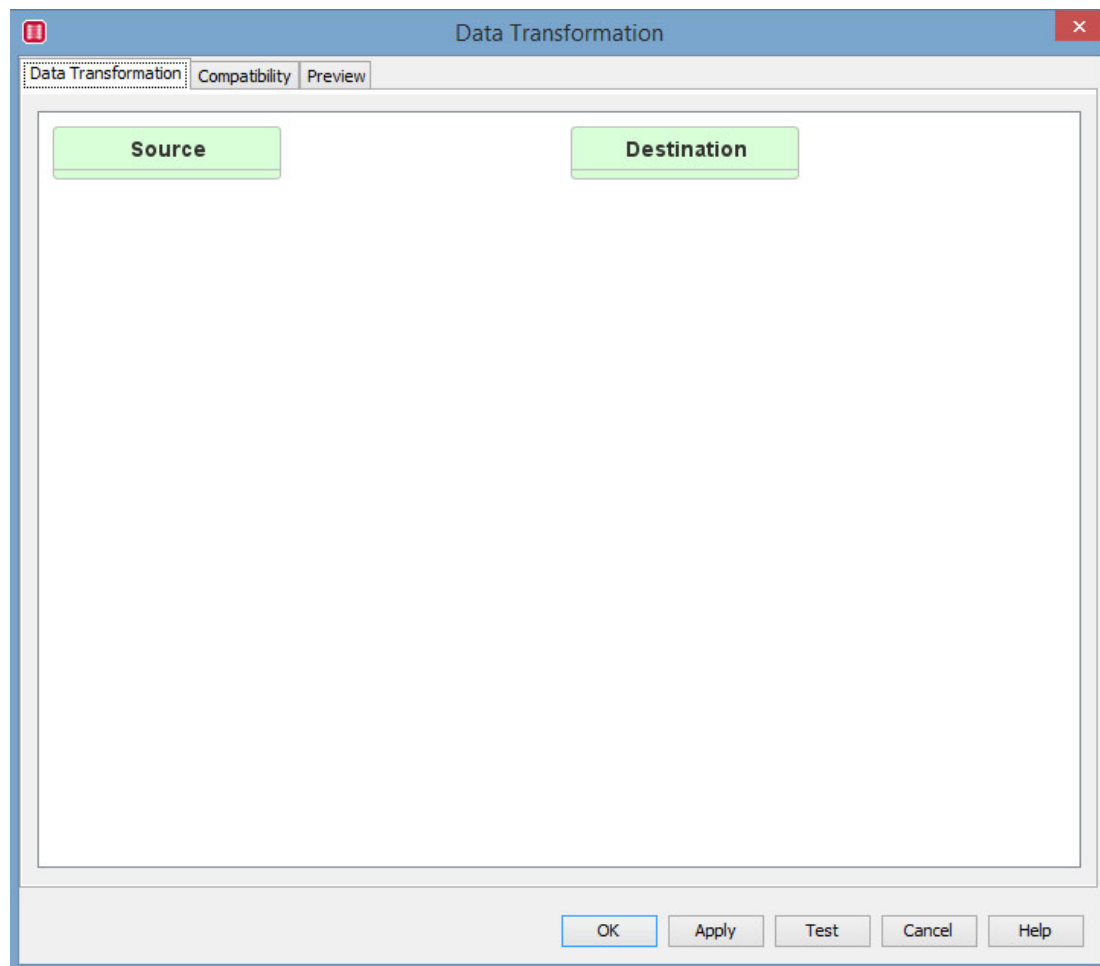
- If the destination is a data section, then there will be one row generated for each run of the transformation (assuming no results are filtered out).
- If the destination is an XML document, then there will be one parent iterator element and child elements for each run of the transformation (again, assuming no results are filtered out).
- If the destination is a JSON document, then there will be one item created in the JSON array referenced as parent iterator for each run of the transformation (again, assuming no results are filtered out).

To use a Data Transformation module

1. Drag and drop a **Data Transformation** module icon into an EMF process. Create a link to it from a module already in the EMF process instance.



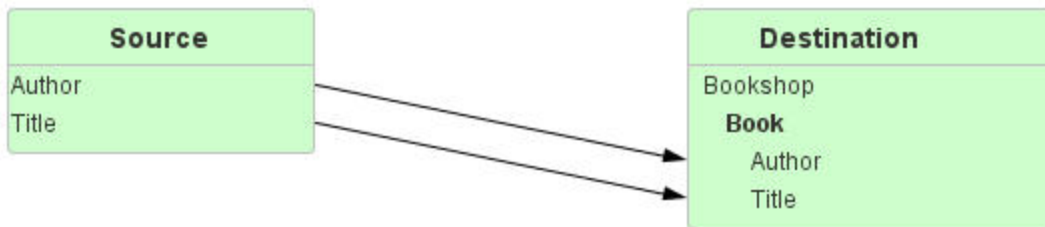
2. Open the **Data Transformation** module to display its properties. By default, you have to double-click on the icon to open. You can also configure mouse-click actions on the **Options** window. From the **Tools** menu, select **Options > EMF > Process Builder Behaviour** tab.



3. First the structure of the source data that will be transformed needs to be defined. This is done in the "[Source Container](#)".
4. Having defined the "Source Container", the desired output data structure needs to be defined. This is done in the "[Destination Container](#)".
5. Having defined the source and destination container, mapping is defined between the two containers to show which items in the source container should be mapped to which items in the destination container. To do this, right-click (and hold the mouse button

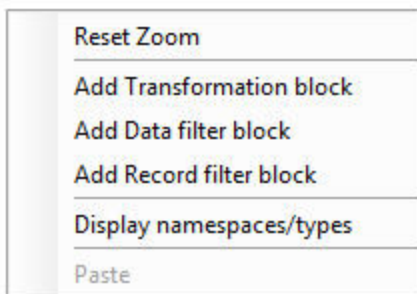
down) on an item in the source container and drag the mouse over to the item it should be mapped to in the destination container before releasing the mouse button. An arrow will then be drawn between the two items to show the mapping.

The following image shows an example mapping between a source data section, mapping onto a destination XML section.



Tip: You can directly add items from the source container to the destination container by **clicking** (and holding the button down) on an item (or a group of items) and dragging and dropping them in the destination container at the required position.

- Additional menu operations are available using the right-click option on the background and selecting the appropriate menu option.



- **Reset Zoom** - resets the zoom to the default value if the zoom level has been increased / decreased. The zoom is altered by holding the **CTRL** key and using the mouse wheel.
 - **Add Transformation block** - adds a new transformation block. These allow items of data to be manipulated / changed / hardcoded depending on conditions. For more information, see [Transformation Blocks](#).
 - **Add Data filter block** - adds a new data filter block. These allow items of data to be filtered out from the results depending on a particular condition. For more information, see [Data Filter Blocks](#).
 - **Add Record filter block** - adds a new record filter block. This allows whole records to be filtered out depending on a particular condition. For more information, see [Record Filter Blocks](#).
 - **Display namespaces/types** - if selected, then the source and destination containers will display the element / attribute namespace next to any element / attribute names and the field data type next to any field names.
- The **Compatibility** tab allows options to be selected to create output that would be the same as generated with previous versions of EMF. As improvements are made to

the mapping engine, the output generated may change with newer versions. If mapping XML to XML, then select the mapping engine required:

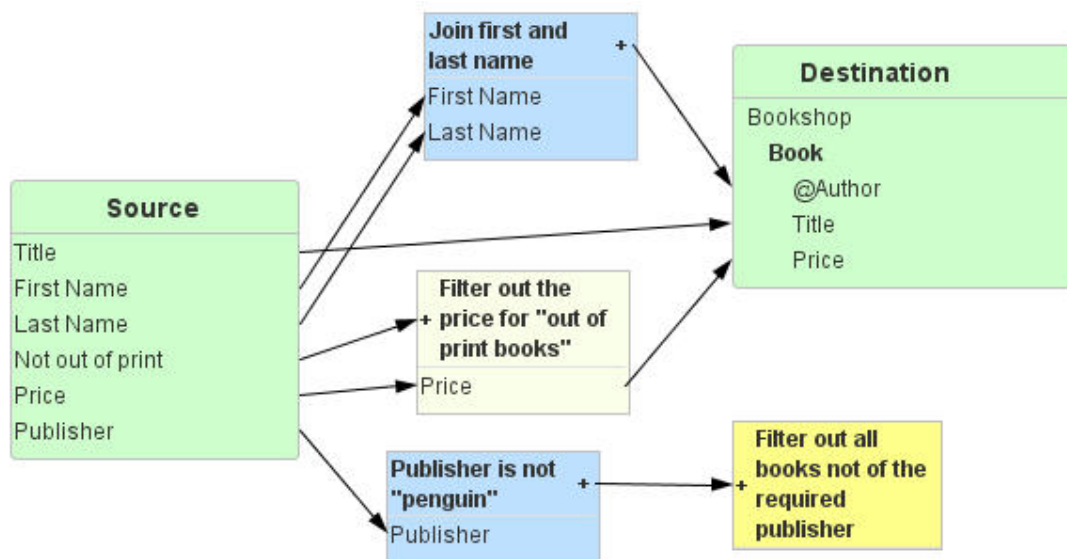
- Use pre EMF 7.2.1 mapping engine.
- Use latest mapping engine

The processing takes place based on the selection. The **latest mapping engine** creates more logical output when mapping multi level XML documents compared to versions of EMF pre 7.2.1. The **pre EMF 7.2.1 mapping engine** creates the output the same as versions of EMF previous to 7.2.1 would have and is there for backward compatibility.

8. The **Test** button can be used to test that the transformation output is generated as expected. You will need to make sure the source data has an appropriate sample section for the test results to give meaningful output.

Example Process

[Click this link](#) to save the process shown below locally. You can then import it to your local repository.



The sample data with this process represents the following data set:

Title	First Name	Last Name	Not out of print	Price	Publisher
A Tale of Two Cities	Charles	Dickens	false	11	Penguin
And Then There Were...	Agatha	Christie	true	10	Penguin
Dream of the Red Ch...	Cao	Xueqin	true	20	China Publishing Group
The Da Vinci Code	Dan	Brown	true	15	Penguin

And running through the transformation module, converts it to the following XML:

```

<?xml version="1.0" encoding="UTF-8"?>
<Bookshop>
  <Book Author="Charles Dickens">
    <Title>A Tale of Two Cities</Title>
  
```

```

</Book>
<Book Author="Agatha Christie">
<Title>And Then There Were None</Title>
<Price>10</Price>
</Book>
<Book Author="Dan Brown">
<Title>The Da Vinci Code</Title>
<Price>15</Price>
</Book>
</Bookshop>

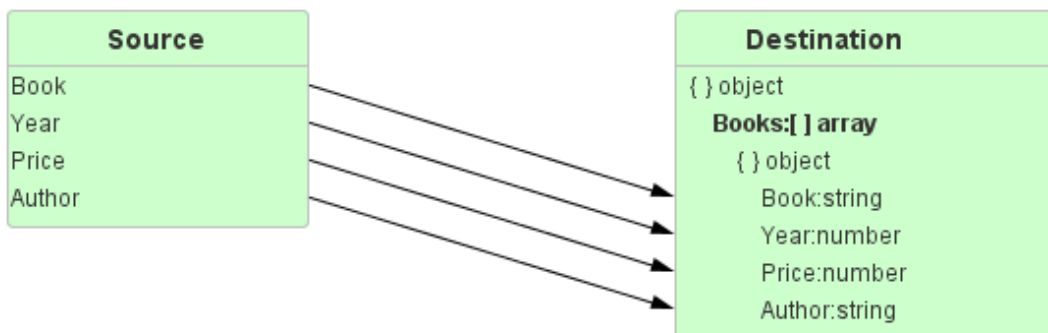
```

Example for JSON

For the following data section:

Book	Year	Price	Author
Ulysses	1922	20.50	James Joyce
To Kill a Mockingbird	1962	15	Harper Lee
Animal Farm	1945	18.75	George Orwell

The following Data Section to JSON Transformation can be defined:



Then it will generate a JSON document that looks similar to the following when the process is run:

```

{
  "Books"
  [
    {"Author" : "James Joyce", "Book" : "Ulysses", "Year" : 1922, "Price": 20.50
  },
    {"Author" : "Harper Lee", "Book" : "To Kill a Mockingbird", "Year" : 1962,
"Price": 15 },
    {"Author" : "George Orwell", "Book" : "Animal Farm", "Year" : 1945, "Price":
18.75 }
  ]
}

```

Source Container

The **Source Container** defines the data / XML / JSON structure of the data section / XML / JSON to be used for the transformation. Right-click the container labelled **Source** to perform the following actions:

- Source Properties
- Define Parent Iterator Element
- Add element
- Add attribute
- Add JSON value
- Add field
- Apply sample
- Edit item
- Delete item

Source Properties

The Source container properties (Double-click on the Source Container header or right-click the Source Container and select Source Properties to access) allows you to specify the data that needs to be transformed. This can be a Data section, XML document, or a JSON document contained in a message section.

• Source Section

Select:

- **Data section** - Select the data section from the drop-down list that will be used as the source data. The Source container will be populated with the columns from the sample data (if available) associated with the selected data section.
- **XML** - Select the message section from the drop-down list containing the XML source. This is the default selection for the Source Section. The Source container will be populated with the XML document structure.
- **JSON** - Select the message section from the drop-down list containing the JSON source. The Source container will be populated with the JSON document structure.

- **Record Filter skipped records** - If you are using a “record filter block”, then select **Create section containing records that have been skipped** to create a data section that stores the records that have been filtered out using any Record Filter blocks. The data section is available to use in modules that follow the Data Transformation module to get a list of filtered rows. This option is only available if the Source section is a “data section”.

Define Parent Iterator Element

For JSON and XML based sections, a parent element may be selected. For XML, this parent element is the XML element that is iterated over during processing, for JSON it is a JSON array that is iterated over. It is displayed in a bold font to indicate that it is the parent element.

For example, if the XML source document was structure like this:

```
<?xml version="1.0"?>
<Root>
  <Book>
    <Title>The Hobbit</Title>
    <Author>J R R Tolkien</Author>
  </Book>
  <Book>
    <Title>Harry Potter and the Philosopher's Stone</Title>
    <Author>J K Rowling</Author>
  </Book>
</Root>
```

Then **Book** would be the parent iterator element. The element that there are multiple items of, and that processing would be repeated once for each instance of a parent iterator element.

Note: Since EMF 7.2.1 it is not necessary to define a Parent Iterator when mapping XML to a data section and multi-level XML documents are processed more logically. By not defining a parent iterator, all the possible mapped combination of results are created in the output. For example, if there is an XML containing multiple products with multiple ingredients, then the final data section will have a row for each product ingredient.

A JSON example, if the source document was like this:

```
{ "Book" : [
  { "Title" : "The Hobbit",
    "Author" : "J R R Tolkien"},
  { "Title" : "Harry Potter and the Philosopher's Stone",
    "Author" : "J K Rowling"} ]
}
```

Then the JSON array referenced by **Book** would be the parent iterator element. Processing would be repeated once for each item in the JSON array.

Selecting **Define Parent Iterator Element** on an already defined parent iterator will undefine it.

Example results for XML to Data Section with and without a Parent Iterator defined

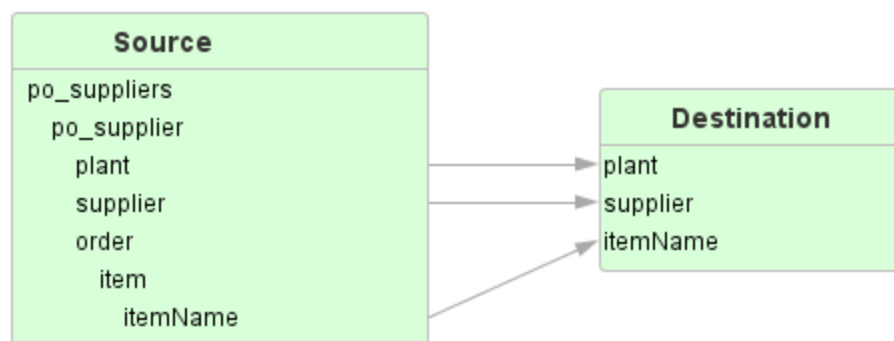
Given the following XML:


```

<?xml version="1.0" encoding="utf-8"?>
<po_suppliers>
<po_supplier>
<plant>001</plant>
<supplier>Supplier 1</supplier>
<order>
  <item>
    <itemName>Item A</itemName>
  </item>
  <item>
    <itemName>Item B</itemName>
  </item>
</order>
</po_supplier>
<po_supplier>
<plant>002</plant>
<supplier>Supplier 2</supplier>
<order>
  <item>
    <itemName>Item C</itemName>
  </item>
  <item>
    <itemName>Item D</itemName>
  </item>
  <item>
    <itemName>Item A</itemName>
  </item>
</order>
</po_supplier>
</po_suppliers>

```

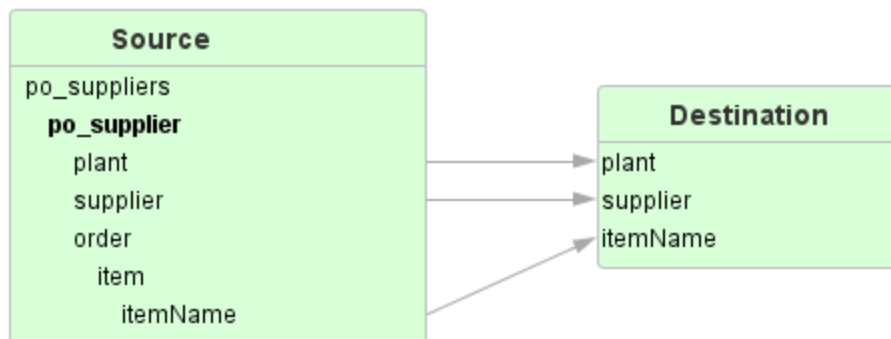
And define the following mapping with no parent iterator:



The following data section would be created:

Data section result			XML result	JSON	S
plant	supplier	itemName			
001	Supplier 1	Item A			
001	Supplier 1	Item B			
002	Supplier 2	Item C			
002	Supplier 2	Item D			
002	Supplier 2	Item A			

If po_supplier is defined as the parent iterator:



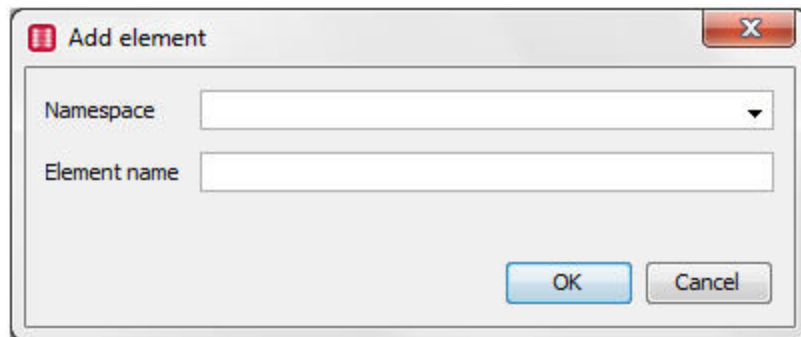
Then the following data section would be created:

Data section result			XML result	JS
plant	supplier	itemName		
001	Supplier 1	Item A		
002	Supplier 2	Item C		

Add element

Elements can be added into the XML that is displayed in the Source container. Adding elements does not change the original section selected in the Source properties. The added element is used within the Data Transformation module only.

Note: The elements can only be added to other elements, and not to attributes.



- **Namespace** - XML namespaces are used for providing uniquely named elements and attributes in an XML document. Type the namespace for the element that is being added into the XML source. An example XML document with namespaces may look like:

```
<?xml version="1.0"?>
<x:DOCELEM xmlns:x="http://www.example.com/test">
<AAA ddd="d" ccc="c">
<CCC xmlns="http://www.example.com/test2">C1</CCC>
<DDD>D</DDD>
</AAA>
</x:DOCELEM>
```

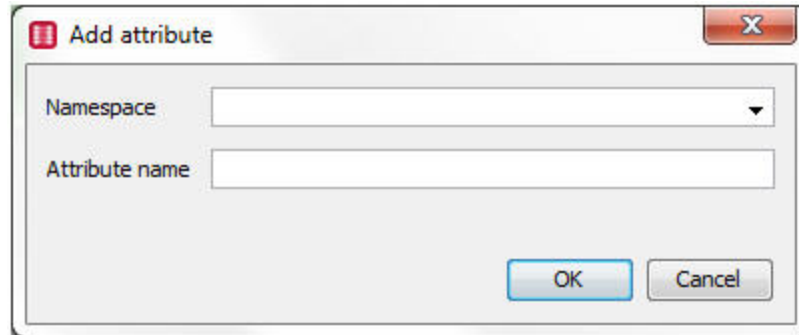
In the above example, element name "CCC" has a namespace "http://www.example.com/test2"

Leave the namespace blank, if the source XML element has no namespace defined.

- **Element name** - This is the name given to the element which will be added into the source container.

Add attribute

Attributes can be added to XML elements. This option is only available if an XML element is selected.



- **Namespace** - XML namespaces are used for providing uniquely named elements and attributes in an XML document. Type the namespace for the element that is being added into the XML source. An example xml document with namespaces may look like:

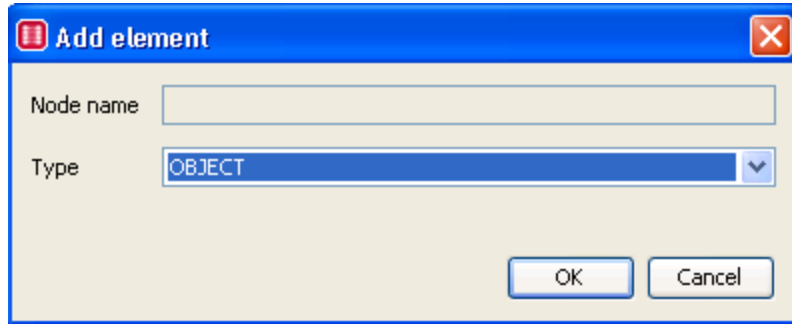
```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
  <Book xmlns:ns0="http://www.bookshop.com" ns0:Author="J R R
Tolkien">
    <Title>The Hobbit</Title>
  </Book>
</Root>
```

Attributes typically are in the default empty namespace.

- **Attribute name** - This is the name given to the attribute that will be added into the source container.

Add JSON value

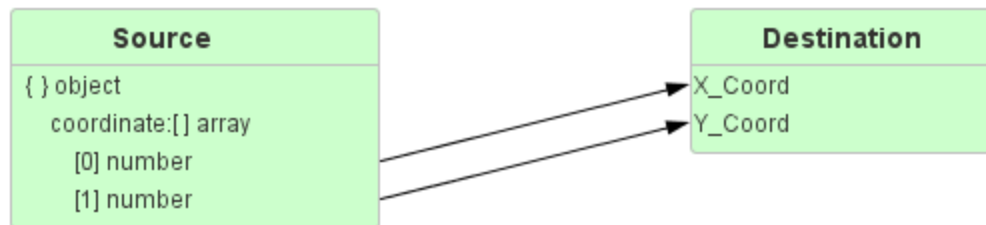
A new JSON value can be added to the specified location in the JSON document.



- **Node name** - Type the name of the node that will be added. Only nodes that are being added as children of a JSON object can have a name defined.
- **Type** - Select the type of value for the node from the drop-down list. The following options are available:
 - **Array** - Select to add array type of node. This node can have child elements. When adding an array, you have the option whether there should be a **Node for every element in the Array**. If the array is being used as a container to store a fixed number of values, and you would want to map those values individually to the destination container, then select to have a node for every item in the array. For example, an array may be used to store X/Y coordinates, such as:

[36.7, 86.3]

In this case, you would want to define each node in the array.



If you plan to use this array as a parent iterator, and the array contains multiple items of the same type, then you should not select this option. This option cannot be changed once the array has been added.

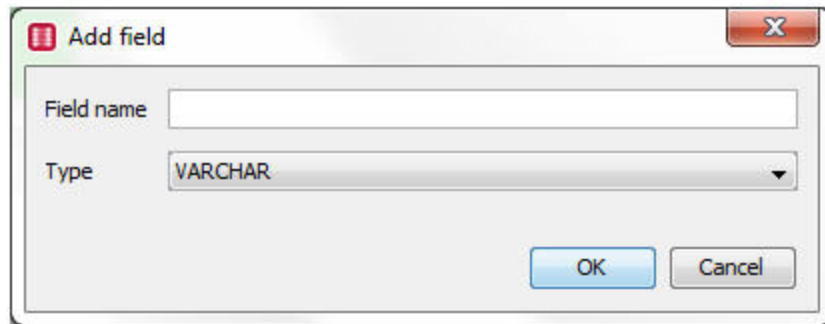
- **Object** - Select to add object type of node. This node can have child elements.
- **String** - Select to add string type of node. This node cannot have child elements.

- **Number** - Select to add number type of node. This node cannot have child elements.
- **Boolean** - Select to add boolean type of node. This type of node cannot have child elements.

Note: Nodes with the same name are not allowed at the same level in the tree.

Add field

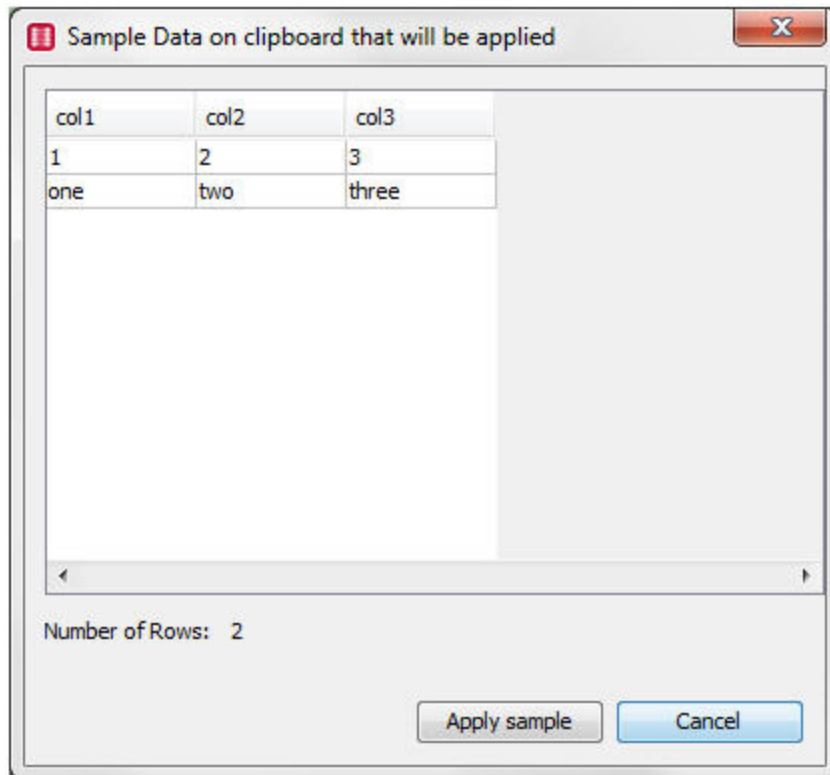
Fields are added to source containers that are configured as “Data sections”. They are added to define the structure of the data section that will be processed at run time.

A screenshot of a Windows-style dialog box titled "Add field". The dialog has a standard title bar with a red close button. Inside, there are two labels: "Field name" and "Type". The "Field name" label is followed by a text input field. The "Type" label is followed by a dropdown menu that currently displays "VARCHAR". At the bottom right of the dialog, there are two buttons: "OK" and "Cancel".

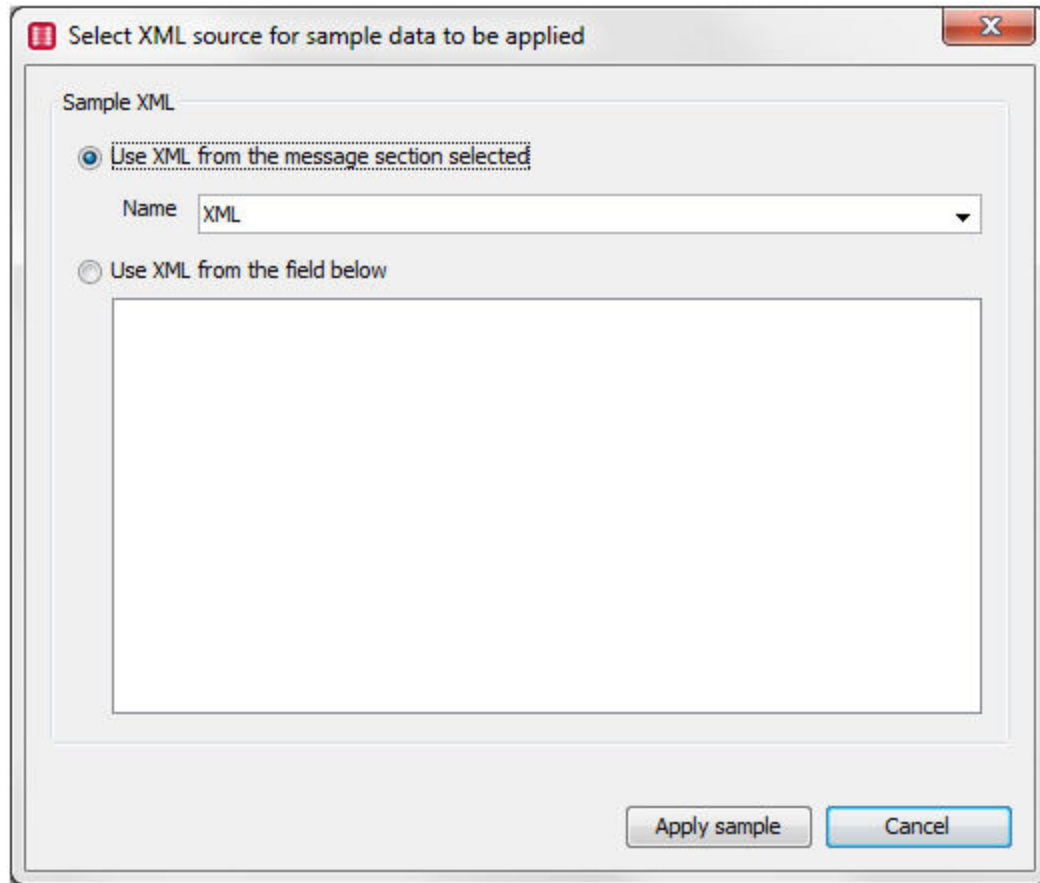
- **Field name** - Type the name of the field that will be added.
- **Type** - this is the data type for the field to be added. Select from the drop down list. Options include BIT, DATE, INTEGER, TIME, VARCHAR among others.

Apply sample

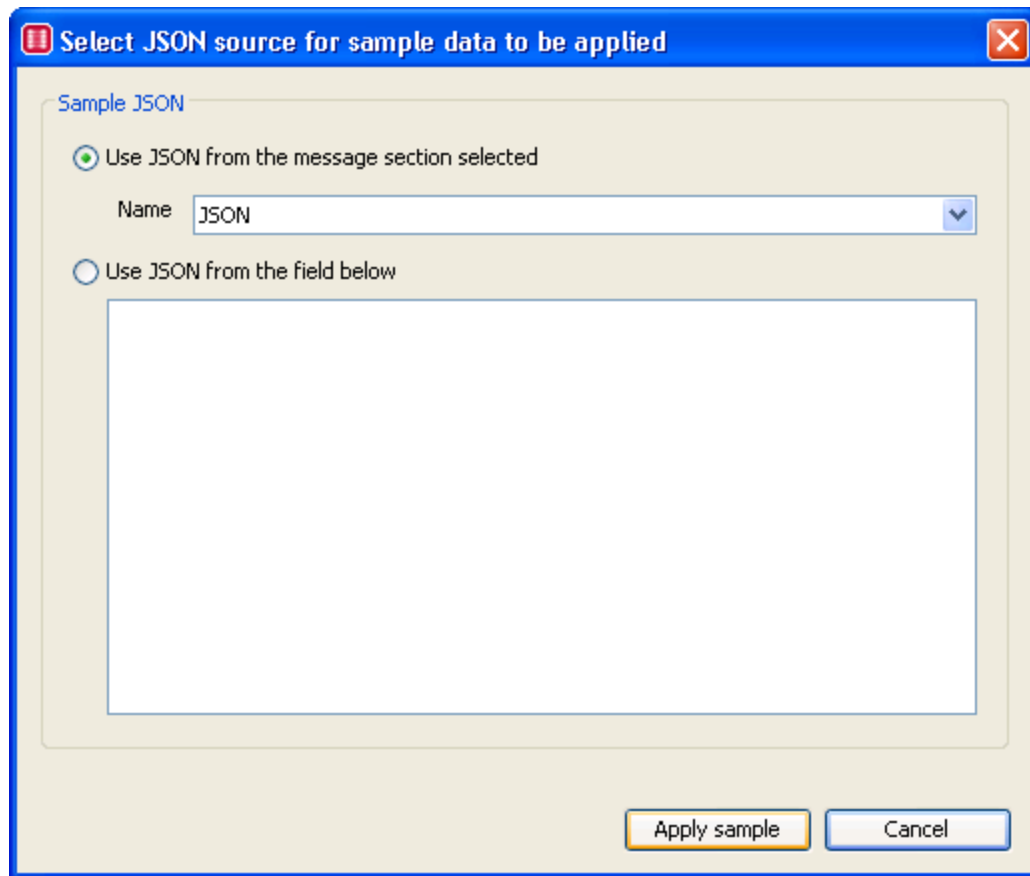
Depending on the Source data type selected (Data section, XML or JSON), this option can be used to apply a different data structure to the Source Container. It allows a new sample section to be applied. For data sections, the sample can be pasted from the clipboard.



For XML, text can be pasted into the text field or a new XML section can be selected.



For JSON, text can be pasted into the text field or a new message section containing JSON selected.



Editing / Deleting items

Right-click the items in the Source Container to edit or delete them using the appropriate menu option. They can be re-ordered by using drag and drop.

Destination Container

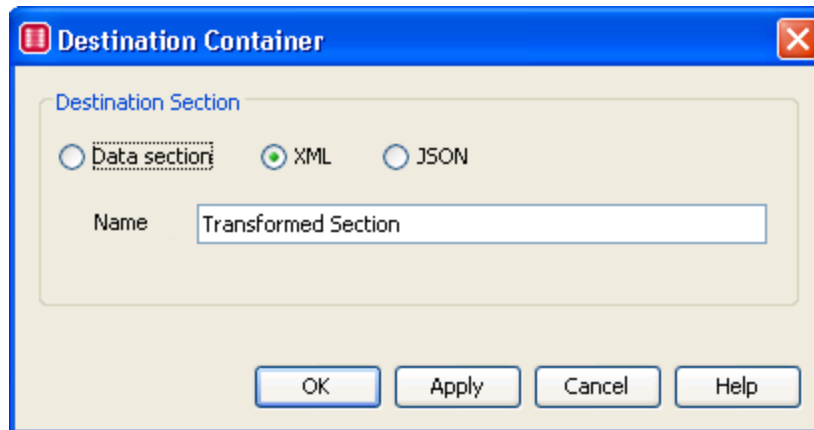
The **Destination Container** defines the structure of the data section / XML / JSON that will be created when the transformation module runs. Right-click the container labelled

Destination to perform the following actions:

- Destination Properties
- Define Parent Iterator Element
- Key field (*only for data section*)
- Group By field (*only for data section*)
- Add element
- Add attribute
- Add JSON value
- Add field
- Apply sample
- Edit item
- Delete item

Destination Properties

The Destination container properties (Double-click on the Destination Container header or right-click on the Destination Container and select Destination Properties to access) allows you to specify if the destination is a data section, XML or JSON. You must enter a name for the destination section that will be created at runtime. For JSON and XML, a message section will be created, instead of a data section.



- **Destination Section**

Select:

- **Data Section** - Type the name of the data section that is used to store the output data.
- **XML** - Type the name of the message section that is used to store the XML output. This is the default selection for the Destination Section.
- **JSON** - Type the name of the message section that is used to store the JSON output.

Define Parent Iterator Element

For XML based sections, a parent element needs to be selected. This parent element is the XML element that is the parent XML element that will be repeatedly added to the XML document for each iteration over the source content. It is displayed in a bold font to indicate that it is the parent element.

Note: To run the process, the parent iterator element must be defined for an XML section.

Key field

If the Destination container is a data section, you can define the container items as "key fields". This field provides a visual representation to the user of the key field (key fields are displayed in bold). When mapping XML to data sections, some extra processing is also applied to the result set:

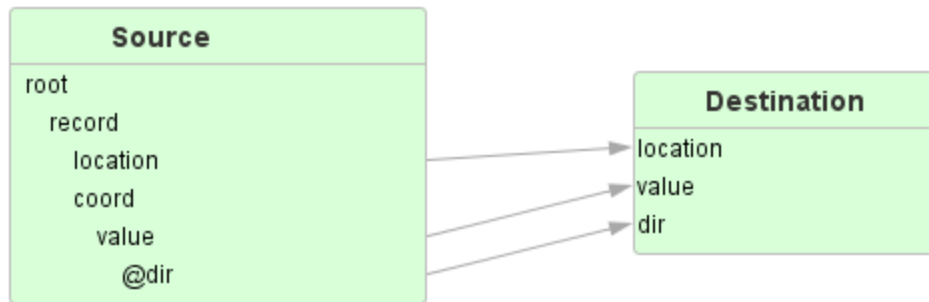
During generation of the final data section, often it is possible for multiple rows of similar data to be generated. By defining a key field it allows the result set to be condensed/merged into more logical grouping. If multiple rows contain the same keys, then the first row's values are taken as the base - if any of the values are null then they will be populated from other rows with the same key that have not got null values. Multiple fields can be defined as key fields to make a combined key.

Key Field Example for XML to Data section mapping

Given the following XML.

```
<?xml version="1.0" encoding="utf-8"?>
<root>
  <record>
    <location>London</location>
    <coord>
      <value dir="x">123</value>
      <value dir="y">532</value>
    </coord>
  </record>
  <record>
    <location>New York</location>
    <coord>
      <value dir="y">777</value>
      <value dir="x">888</value>
    </coord>
  </record>
</root>
```

A standard transformation such as:



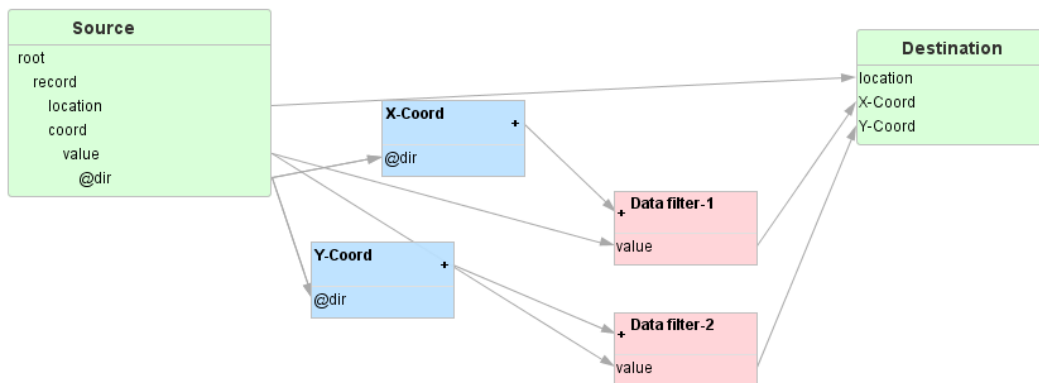
Would generate:

Data section result XML result

location	value	dir
London	123	x
London	532	y
New York	777	y
New York	888	x

A better structure for the result data section would be to have columns "location", "X-coord", and "Y-coord". This is possible with a bit more effort.

Using a mapping as follows:



Where the X-Coord transformation block expression is:

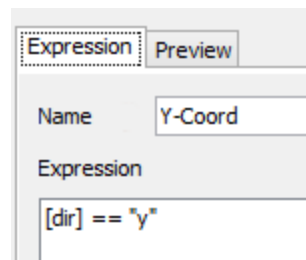
Expression Preview

Name X-Coord

Expression

`[dir] == "x"`

And the Y-Coord transformation block expression is:



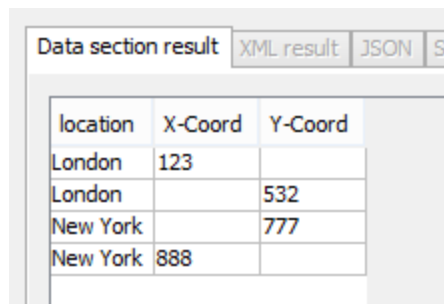
Expression Preview

Name Y-Coord

Expression

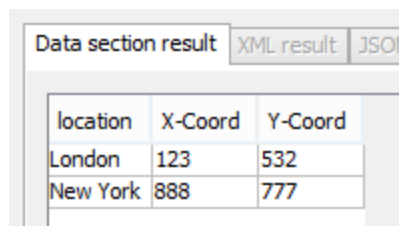
[dir] == "y"

Generates the following data section:



location	X-Coord	Y-Coord
London	123	
London		532
New York		777
New York	888	

Now, to merge the two London rows together and the New York rows together, define the "location" field as being a **Key** field. This will then merge similar rows, creating the following results:



location	X-Coord	Y-Coord
London	123	532
New York	888	777

Group By field

If the Destination container is a data section, you can define the container items as "Group by fields". This works in a similar way to the SQL "Group By" clause. During generation of the final data section, all rows with the same values in the group by column will be grouped together. Other fields that aren't set as grouped, need to have an aggregation function defined. See [Add field](#) for more details. Multiple fields can be defined as Group By fields. Group By fields are displayed in italics.

Add element

Elements can be added into the XML that is displayed in the Destination container.

Note: The elements can only be added to other elements, and not to attributes.

- **Namespace** - XML namespaces are used for providing uniquely named elements and attributes in an XML document. Type the namespace for the element that is being added into the XML.

An example XML document with namespaces may look like:

```
<?xml version="1.0"?>
<x:DOCELEM xmlns:x="http://www.example.com/test">
  <AAA ddd="d" ccc="c">
    <CCC xmlns="http://www.example.com/test2">C1</CCC>
    <DDD>D</DDD>
  </AAA>
</x:DOCELEM>
```

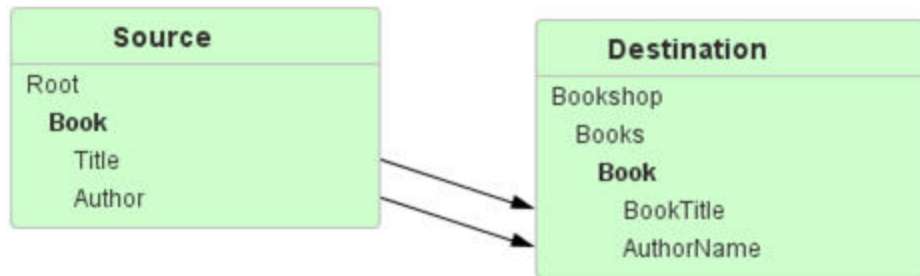
- **Element name** - This is the name given to the element which will be added into the destination container.
- **Output empty element if mapped value is empty and no child elements** - Selecting this check box causes the element to be output in the resulting XML if no value is mapped to it, or the mapped value is empty.

For example, if the source XML is:

```
<?xml version="1.0"?>
<Root>
  <Book>
    <Title>The Hobbit</Title>
    <Author>J R R Tolkien</Author>
  </Book>
  <Book>
    <Title>Harry Potter and the Philosopher's Stone</Title>
  </Book>
```

```
</Root>
```

And the following mapping is:



- If the check box for the "AuthorName" was unselected to **Output Empty Element**, then the output generated will be:

```
<?xml version="1.0" encoding="UTF-8"?>
<Bookshop>
  <Books>
    <Book>
      <BookTitle>The Hobbit</BookTitle>
      <AuthorName>J R R Tolkien</AuthorName>
    </Book>
    <Book>
      <BookTitle>Harry Potter and the Philosopher&#39;s
Stone</BookTitle>
    </Book>
  </Books>
</Bookshop>
```

- If the check box to **Output Empty Element** was selected for the "AuthorName", then the output will be:

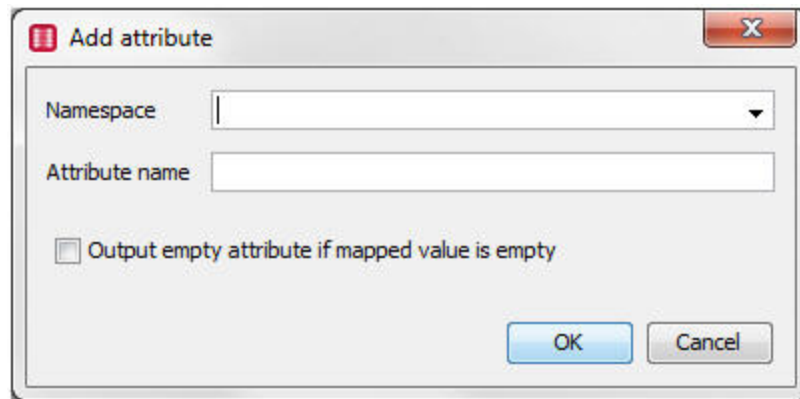
```
<?xml version="1.0" encoding="UTF-8"?>
<Bookshop>
  <Books>
    <Book>
      <BookTitle>The Hobbit</BookTitle>
      <AuthorName>J R R Tolkien</AuthorName>
    </Book>
    <Book>
      <BookTitle>Harry Potter and the Philosopher&#39;s
Stone</BookTitle>
      <AuthorName/>
    </Book>
  </Books>
</Bookshop>
```

- **Output any character as a CDATA section** - Selecting this check causes the output for the element to be output in a CDATA section, and any character data will not be XML escaped.

For example, if the check box was selected in the above example, then the output will be:

```
<?xml version="1.0" encoding="UTF-8"?>
<Bookshop>
  <Books>
    <Book>
      <BookTitle><![CDATA[The Hobbit]]></BookTitle>
      <AuthorName>J R R Tolkien</AuthorName>
    </Book>
    <Book>
      <BookTitle><![CDATA[Harry Potter and the Philosopher's
Stone]]></BookTitle>
      <AuthorName/>
    </Book>
  </Books>
</Bookshop>
```

Add attribute



- **Namespace** - XML namespaces are used for providing uniquely named elements and attributes in an XML document. Type the namespace for the element that is being added into the XML source. An example XML document with attribute namespaces may look like:

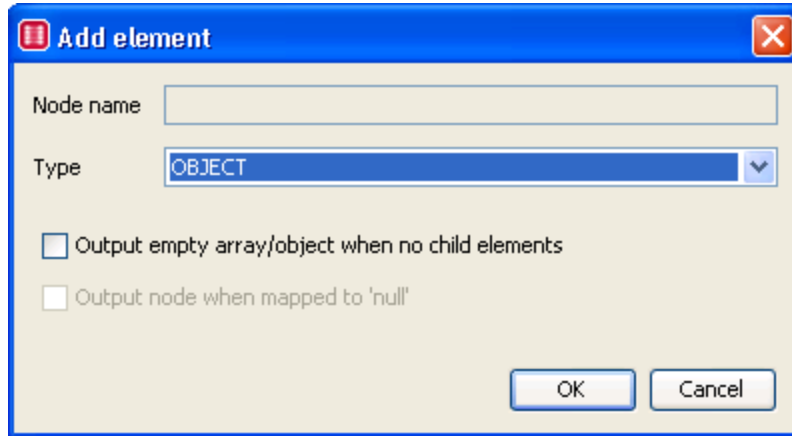
```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
  <Book xmlns:ns0="http://www.bookshop.com" ns0:Author="J R R Tolkien">
    <Title>The Hobbit</Title>
  </Book>
</Root>
```

Attributes typically are in the default empty namespace.

- **Attribute name** - This is the name given to the attribute that will be added into the destination container.
- **Output empty attribute** - Selecting this check box allows you to output the empty attribute if there is no value mapped for the attribute at runtime. For detailed example, see [Add Element](#).

Add JSON value

A new JSON value can be added to the specified location in the JSON document.



- **Node name** - Type the name of the node that will be added. Only nodes that are being added as children of a JSON object can have a name defined.
- **Type** - Select the type of value for the node from the drop-down list. The following options are available:
 - **Array** - Select to add array type of node. This node can have child elements. When adding an array, you have the option whether there should be a **Node for every element in the Array**. If the array is being used as a container to store a fixed number of values, and you would want to map those values individually from the source container, then select to have a node for every item in the array. In this case, you would want to define each node in the array.



If you plan to use this array as a parent iterator, and the array contains multiple items of the same type, then you should not select this option. This option cannot be changed once the array has been added.

- **Object** - Select to add object type of node. This node can have child elements.
- **String** - Select to add string type of node. This node cannot have child elements.

- **Number** - Select to add number type of node. This node cannot have child elements.
- **Boolean** - Select to add boolean type of node. This type of node cannot have child elements.
- **Output empty array/object when no child elements** - Select this option to output an empty array / object when the generated array / object node has no child nodes output at runtime (for instance, because they are filtered out using a data filter).
- **Output node when mapped to 'null'** - Select this option to output a JSON "null" value when the data section mapping value is null. If this option is not selected, then no node or label will be output when it is mapped to a null.

The following example shows the different output when the "Year" for the "Book" "Ulysses" had a "null" value. With the **Output node when mapped to 'null'** selected, the output would be:

```
{
  "Books"
  [
    {"Author" : "James Joyce", "Book" : "Ulysses", "Year" : null,
    "Price": 20.50 },
    {"Author" : "Harper Lee", "Book" : "To Kill a Mockingbird", "Year"
: 1962, "Price": 15 },
    {"Author" : "George Orwell", "Book" : "Animal Farm", "Year" : 1945,
    "Price": 18.75 }
  ]
}
```

If the option was not selected, it would output the following:

```
{
  "Books"
  [
    {"Author" : "James Joyce", "Book" : "Ulysses", "Price": 20.50 },
    {"Author" : "Harper Lee", "Book" : "To Kill a Mockingbird", "Year"
: 1962, "Price": 15 },
    {"Author" : "George Orwell", "Book" : "Animal Farm", "Year" : 1945,
    "Price": 18.75 }
  ]
}
```

Note: Nodes with the same name are not allowed at the same level in the tree.

Add field

Fields can be added to the destination container when it is defined as a data section.

- **Field name** - Type the name of the field that will be added.
- **Type** - Select the data type for the field to be added from the drop-down list. Options include BIT, DATE, INTEGER, TIME, VARCHAR, and so on. If the mapped value cannot be converted to the required data type, then the module will fail to process and an error will be thrown at runtime.
- **Key** - Select this option to indicate that the field is a key field. Fields with this option selected are displayed in bold in the Destination container. See section [Key field](#) for full definition of purpose.
- **Group By** - Select this option to use this field to group results. Fields with this option selected are displayed in italics in the Destination container.
- **Auto Increment** - Select this option to automatically increment the field value at run time. For example, if the results contained 10 rows, then auto incrementing starting from 1 would mean the first row would get the value 1, the second row 2 and so on. This is useful for creating an ID column. Fields with this option selected cannot be mapped to anything.
- **Starting value** - The start value used by the Auto Increment. Dynamic Functions can be used in this field.
- **Aggregate function** - Select the Aggregate function for the field from the drop-down list. The following options are available and will apply the aggregate function to the rows in each group (see **Group By**) or to the whole result set if no grouping is selected:
 - **Average** - The average value for this field in the grouped records. This is only valid for numeric fields.

- **Average Ignore Null** - The average value ignoring null values for this field in the grouped records. This is only valid for numeric fields.
The difference between **Average** and **Average Ignore Null** - if we have 4 results: 0, 2, NULL, and 6. The **Average** is $8/4=2$. **Average Ignore Null** is $8/3=2.67$.
- **Count** - The number of grouped records.
- **Maximum** - The largest value for this field in the grouped records. This is only valid for numeric fields.
- **Median** - The median value for this field in the grouped records. This is only valid for numeric fields.
- **Minimum** - The smallest value for this field in the grouped records. This is only valid for numeric fields.
- **Mode** - The mode (appears the most often) value in the field. If multiple values have the same most often frequency, then just one of the values is returned. This is only valid for numeric fields.
- **None** - Select if field does not require any aggregation.
- **Sum** - Calculates the total for this field in the grouped records. This is only valid for numeric fields.

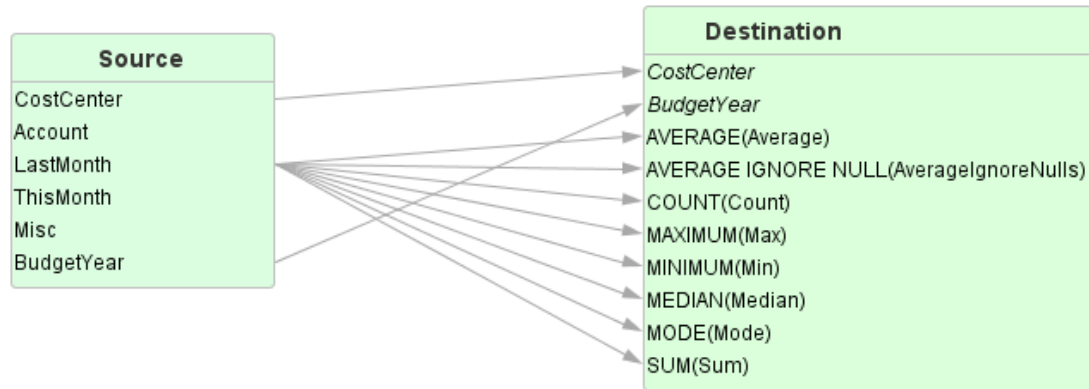
The selected Aggregate function is displayed in the Destination Container field item.
Eg. AVERAGE(field).

Example:

Using the sample data section.

CostCenter	Account	LastMonth	ThisMonth	Misc	BudgetYear
CC1	0	0.0	0.0	101.0	2008
LINEA	MATERIAL	0.0	0.0	100.0	2008
CC1	LABOR		0.0	0.0	2009
CC1	CONTRACTS	607.0	0.0	0.0	2009
CC1	MATERIAL	654.0	0.0	0.0	2009
LINEA	CONTRACTS	1383.0	0.0	0.0	2009
LINEA	LABOR	2.0	0.0	0.0	2009
LINEA	MATERIAL	1069.78	10763.22	0.0	2009
LINEB	MATERIAL	3.2	198.0	0.0	2009
LINEB	LABOR	0.0	1.5	0.0	2009
LINEC	CONTRACTS	638.0	0.0	0.0	2009
STORES	MATERIAL	0.0	0.0	0.0	2009

And having a transformation



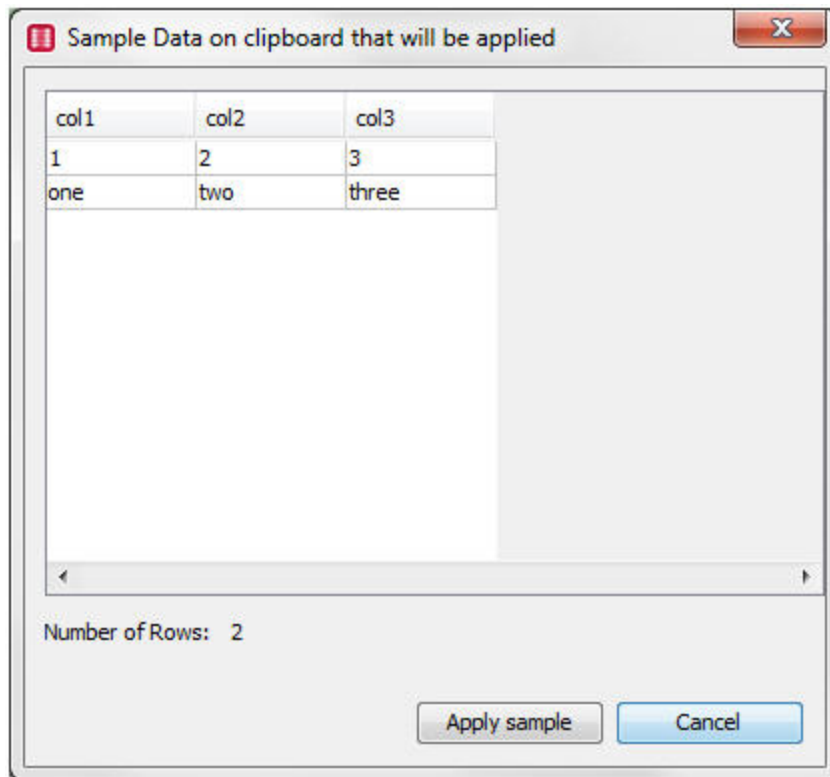
Grouping the results on CostCenter and BudgetYear would give the following for the field LastMonth:

CostCenter	BudgetYear	Average	AverageIgnoreNulls	Count	Max	Min	Median	Mode	Sum
CC1	2008	0.0	0.0	1	0.0	0.0	0.0	0.0	0.0
LINEA	2008	0.0	0.0	1	0.0	0.0	0.0	0.0	0.0
CC1	2009	420.333333333333	630.5	3	654.0	607.0	630.5	607.0	1261.0
LINEA	2009	818.259999999999	818.259999999999	3	1383.0	2.0	1069.78	2.0	2454.779999999997
LINEB	2009	1.6	1.6	2	3.2	0.0	1.6	0.0	3.2
LINEC	2009	638.0	638.0	1	638.0	638.0	638.0	638.0	638.0
STORES	2009	0.0	0.0	1	0.0	0.0	0.0	0.0	0.0

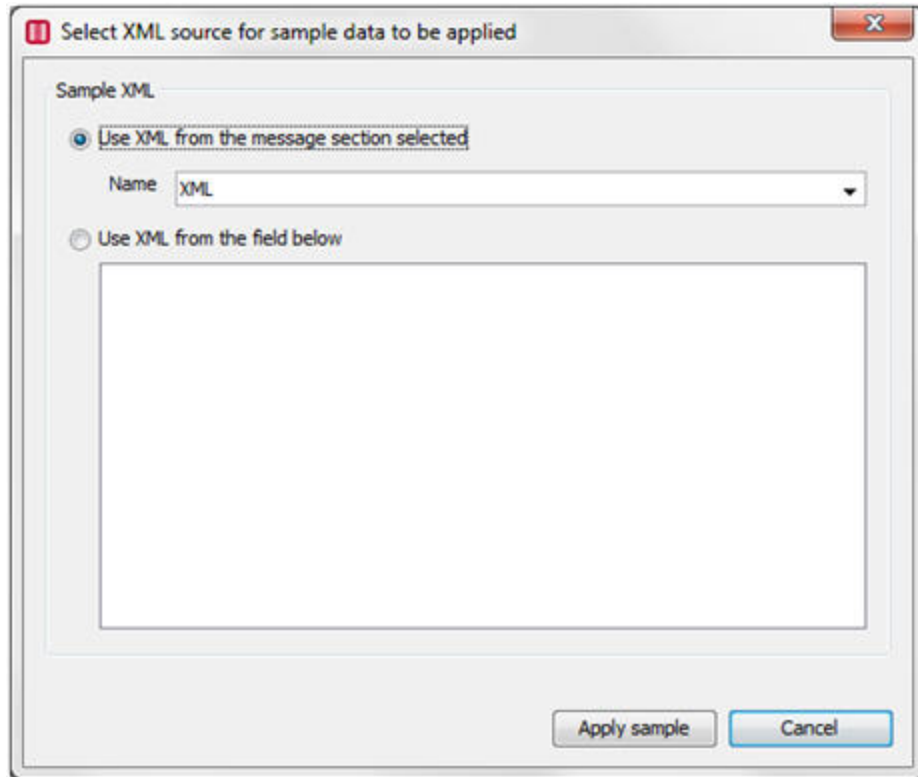
Note: If at least one field has been selected for **Group By**, then all fields that are not defined as **Group By** or **Auto Increment** must have an **Aggregate function** other than "None" defined.

Apply sample

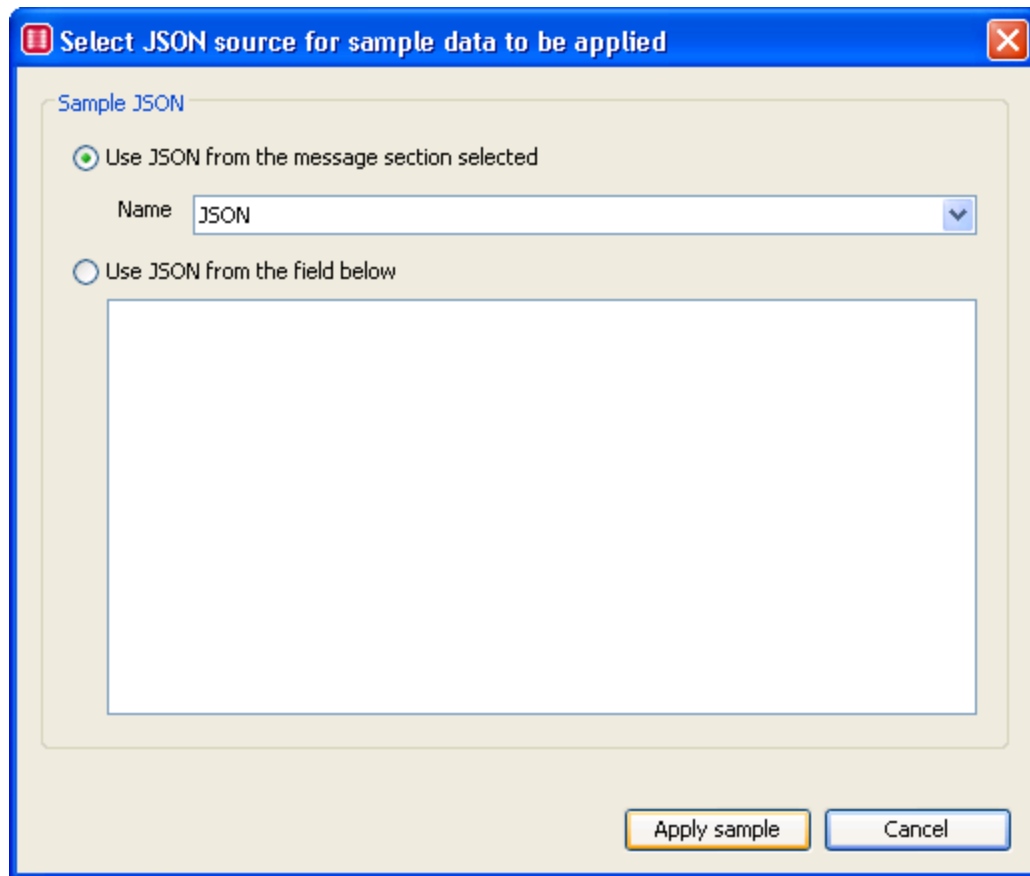
Depending on the Destination data type selected (Data section, XML or JSON), this option can be used to either apply the structure of the fields in the sample data section or the structure of the XML in the sample XML. For data sections, the sample can be pasted from the clipboard.



For XML, text can be pasted into the text field or a new XML section can be selected.



For JSON, text can be pasted into the text field or a new message section containing JSON selected.



Editing / Deleting items

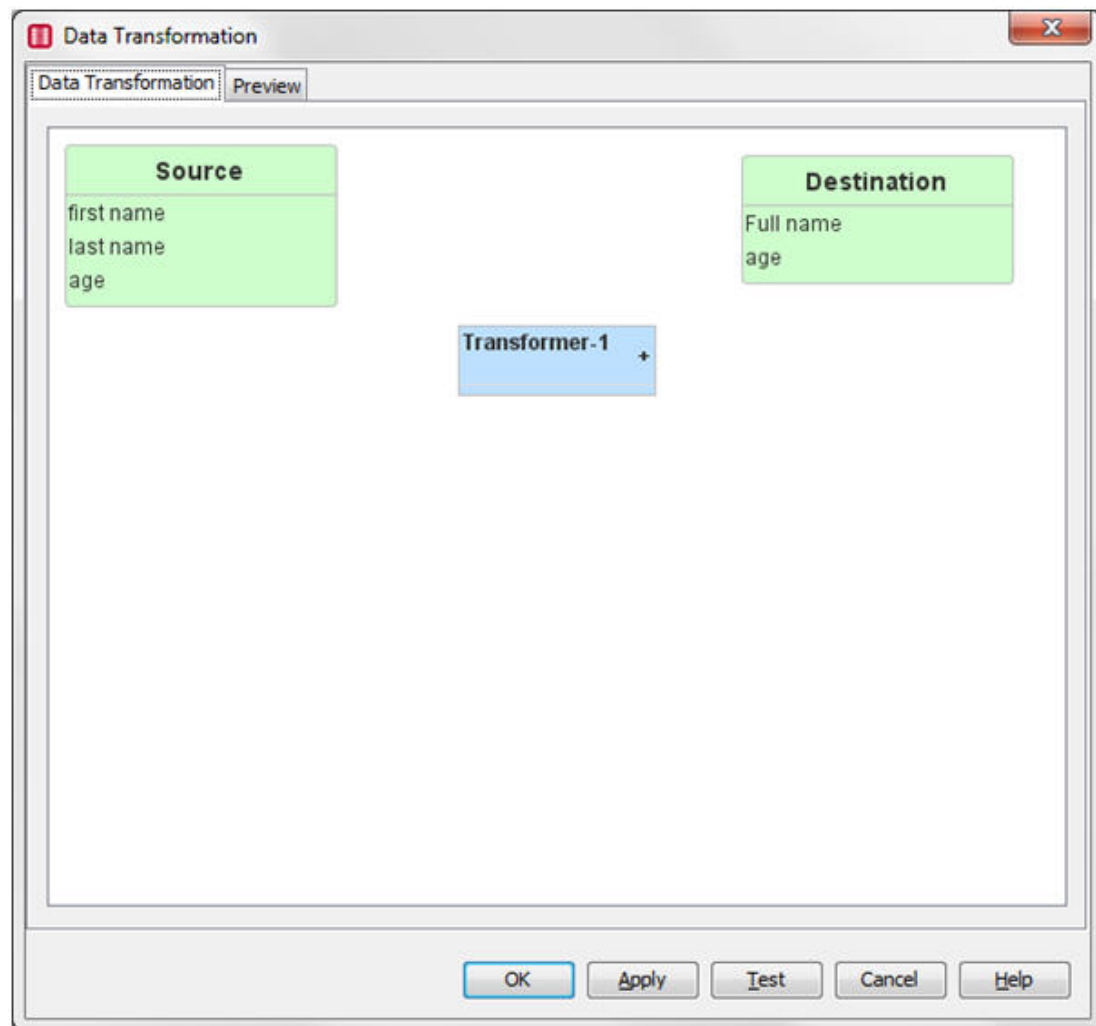
Right-click the items in the Destination Container to edit or delete them using the appropriate menu option. They can be re-ordered by using drag and drop.

Transformation Blocks

The **Transformation Block** allows you to define a transformation on the data (Example: Concatenation, conversion to upper case, Maths functions, and so on). You can enter Javascript expressions in the **Transformation Expression** dialog box to achieve this.

To add a transformation block

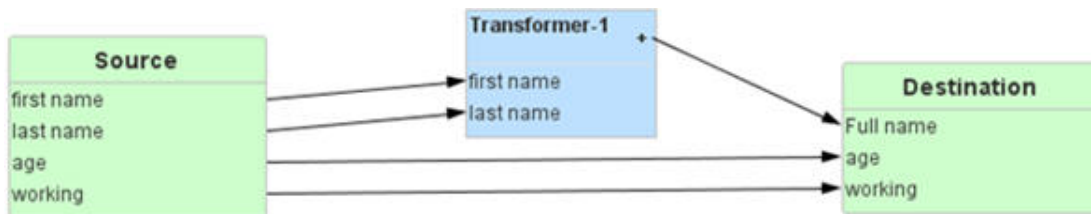
1. Right-click on the **Data Transformation** module background and select **Add Transformation block** from the pop-up menu. The transformation block is added as displayed.



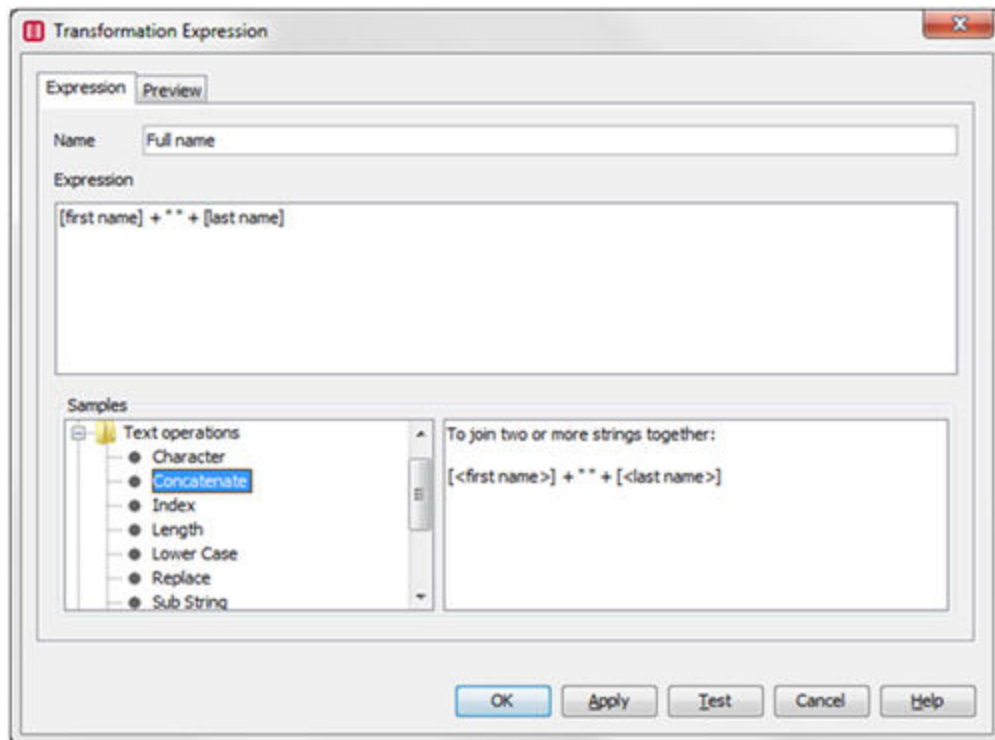
Each transformation can have zero or more inputs and one output. These inputs / outputs are defined as follows:

- **Transformation Input:** A data item mapped from the source container, or the output from another transformation block or data filter block. To add an input mapping, click the item that needs to be added and drag it over the transformation block before releasing the mouse button.
- **Transformation Output:** The result of the transformation that will map onto an item in the destination container, or as input to another transformation block or data filter block. An output link can be mapped multiple times, that is, it can have an arrow coming out of transformation block more than once – you can go to another transformation block, and you can map on to a Destination item.

For example, to concatenate "first name" and "last name" field and map it onto a "Full name" field, you have to create the following connections. To add an output mapping, click the "+" symbol in the data transformation block and drag it over the required container before releasing the mouse button.



2. Double-click the **Transformation** block to enter a transformation expression. The **Transformation Expression** dialog box is displayed.



3. In the **Name** field, type a simple textual description.
4. In the **Expression** field, type the JavaScript code that will perform the transformation. A right-click pop-up menu gives a shortcut to any of the transformer input items (as well as the standard **Dynamic Function** menus). Selecting the transformer input item (in the above case: "first name") inserts the item name surrounded by square brackets, into the **Expression** field.
5. In the **Samples** section, click an operation name to display an example in the right side of the pane. You can also double-click the selected operation to insert the sample JavaScript code into the **Expression** field.

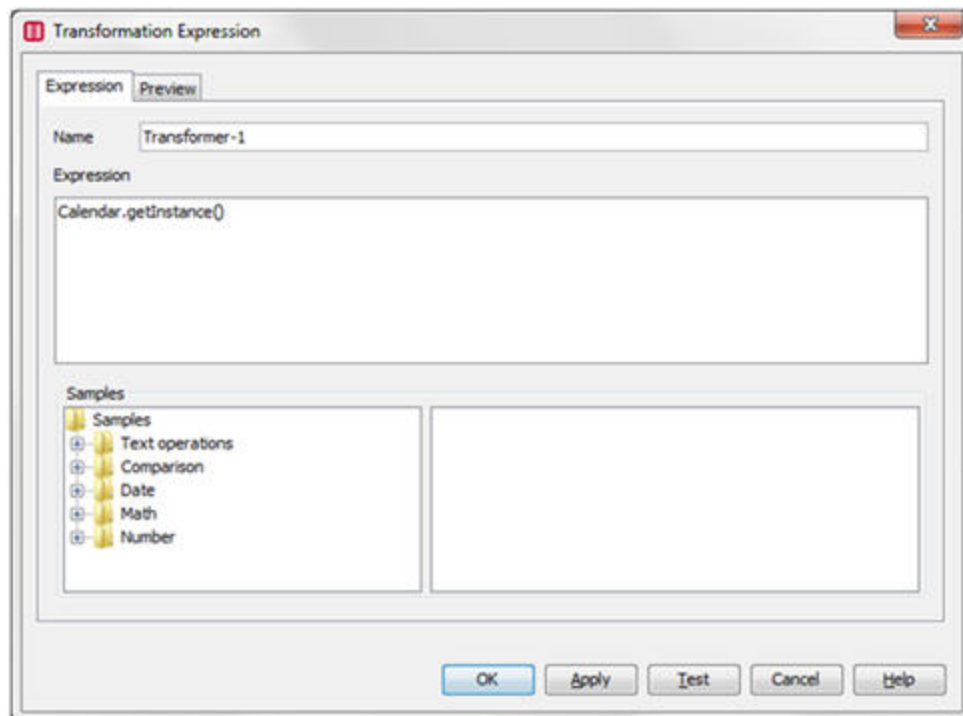
The **Samples** section provides a reference library on the JavaScript code to enter and perform common operations, such as

- concatenation
- converting to upper/lower case
- creating conditional statements
- maths functions

Note: The **Test** button can be used to test the expression entered. Note that valid sample data for the source container needs to be available for the **Test** button to work as expected.

A transformation block does not need to have any inputs. It can be used to output a constant value that is then mapped to the Destination Container.

For example, to set an import date on a record, a transformation such as the following can be used:



You can enter more complex expressions that are not a single expression. If a complex

expression is entered that contains more than a single statement, then the last calculated value will be returned.

For example, both the following transformation scripts will return the same results:

Script 1:

```
if ([first] == null || [first] == "") {
    result = [last];
} else {
    result = [first] + " " + [last];
}
```

Script 2:

```
if ([first] == null || [first] == "") {
    [last]
} else {
    [first] + " " + [last]
}
```

Accessing an Element's Position in XML

Sometimes in the XML structure being processed the element names don't provide enough information, and the order that the elements appear in the XML defines their meaning. For example, in the following XML the first "coord" element is meant to represent the X coordinate, and the second "coord" element the Y coordinate.

```
<?xml version="1.0" ?>
<Root>
  <position>
    <coord>34</coord>
    <coord>125</coord>
  </position>
</Root>
```

It is possible to access the elements position in a transformer block. For example, this could be used to filter out all elements that weren't the first.

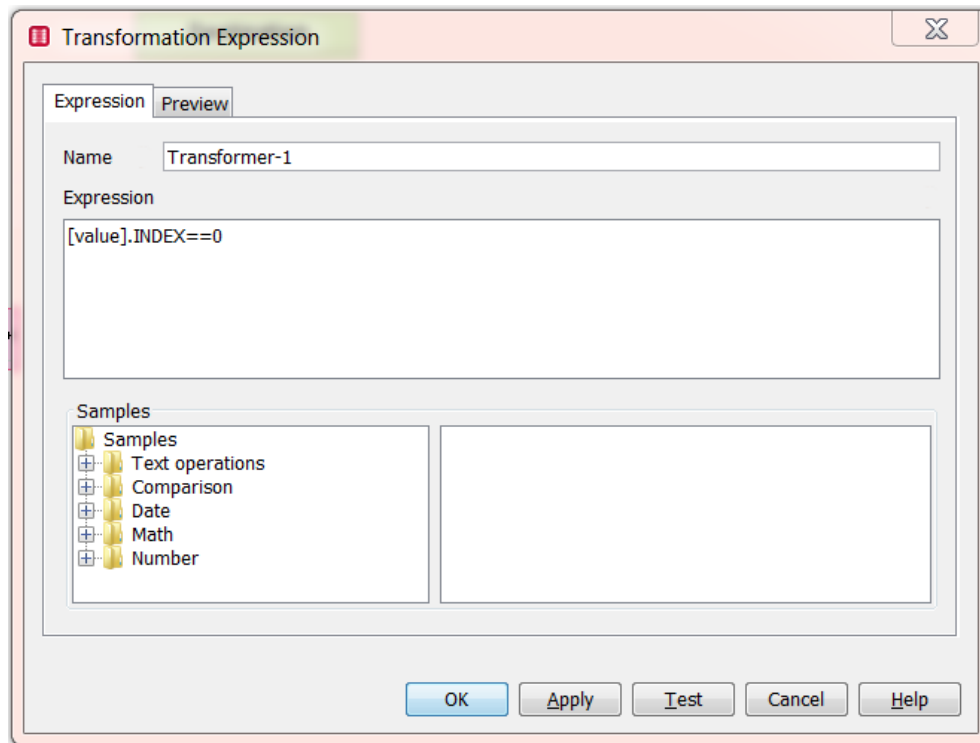
To access the index of individual elements in the Source container, the following format is used:

```
[item].INDEX
```

where "item" is the element name.

With this setting, it is possible to find the index or position of an element in the XML. For example, if there are 3 item elements under a parent, then the index of the first would be 0, the second 1, and the third 2.

So, for example, to create a transformation that would return "true" if the first element was being processed, enter the following (the result of this could then be fed into a data filter block to allow only the first elements through to the result set).



Dealing with Undefined Elements in XML

When dealing with source XML, it is often desirable to be able to detect when an element or attribute is missing.

For example, given the XML:

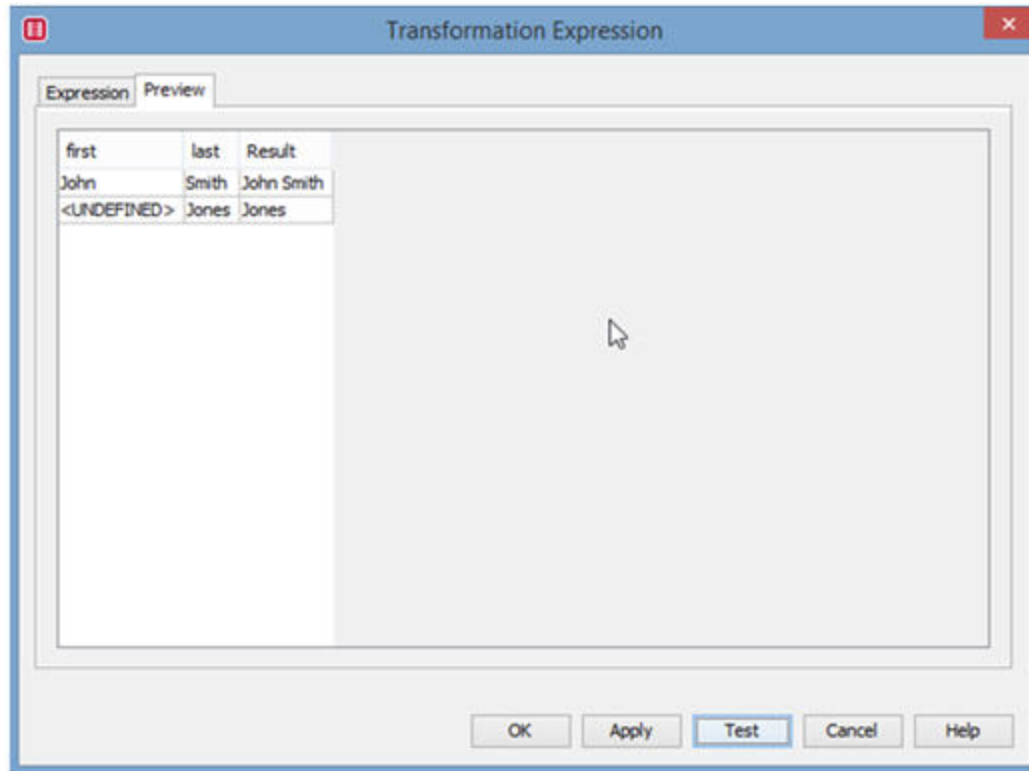
```
<?xml version="1.0" ?>
<Root>
  <person>
    <first>John</first>
    <last>Smith</last>
  </person>
  <person>
    <last>Jones</last>
  </person>
</Root>
```

The second person has no first name defined. It is desirable to detect this and to be able to take different actions on this. Any undefined element/attribute is assigned a special value called **UNDEFINED** at runtime that can be used in comparisons.

For example

```
if ([first] == UNDEFINED) {
  [last]
} else {
  [first] + " " + [last]
}
```

The following image shows how the results are displayed in the **Preview** tab of the Transformation block.



Record Filter Blocks

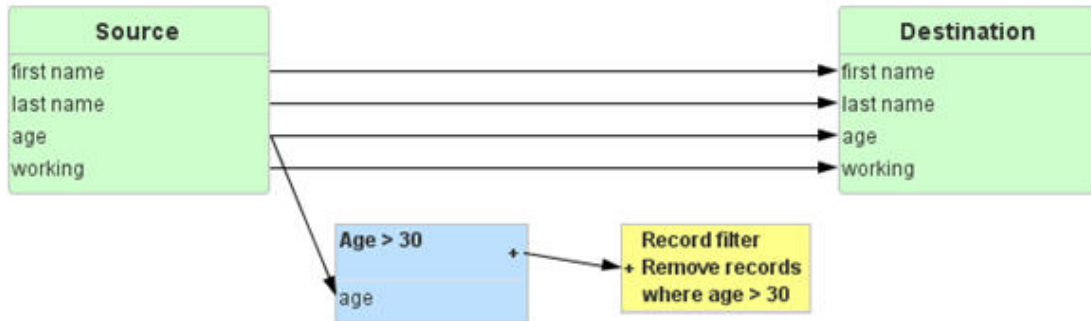
The **Record Filter Block** takes a single input mapping, and depending on the value of the input mapping may skip the whole processing of the current record. The value mapped must be a Boolean type or convertible to a Boolean. In effect, this means it must be a value of "true", "1" or "yes" to be treated as true. Records are filtered out and not added to the destination result section when the mapping evaluates to "true".

To add an input mapping to a record filter

- **Right-click** the item that needs to be linked to the record filter and drag it over the "+" symbol on the record filter before releasing the mouse button.

Multiple record filters can be added.

The following diagram shows how a Record Filter block is used to filter out all records where the person's age was greater than 30. A Transformation block checks the person's age and feeds the result into the Record Filter.



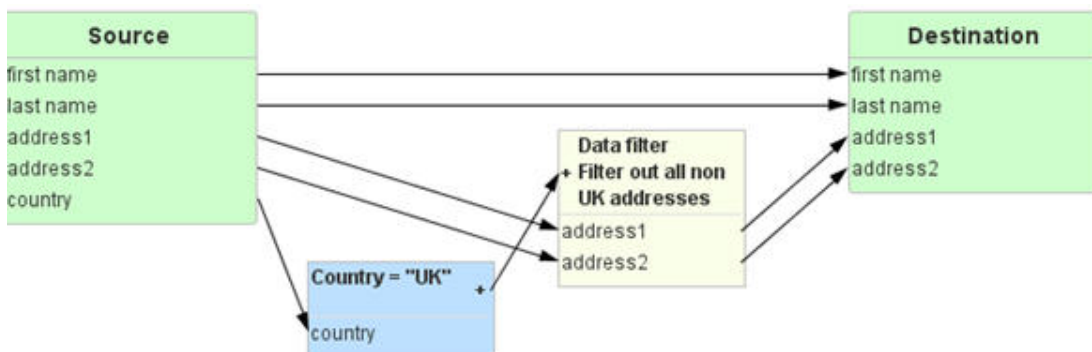
If the source data is a data section (not XML), then it is possible to return a separate data section containing all the records that were skipped. This option is configured in the source container properties (see **Record Filter skipped records** in [Source Properties](#)).

Data Filter Blocks

The **Data Filter Block** takes multiple input mappings. One of the inputs is the condition that decides whether data should be passed through, and the others are the data mappings that are passed through if the condition is "true". In effect, this means it must be a value of "true", "1" or "yes" to be treated as true. Any number of data mappings may be added to pass through the filter. For every data mapping coming in, there will be a corresponding link going out.

- To add a field to be filtered to the data filter, **click** on the item that needs to be filtered and drag it over the data filter block before releasing the mouse button.
- To add a link to the data filter condition, **right-click** on the item that contains the condition value, and then drag it over the "+" symbol of the data filter block before releasing the mouse button.
- To create out link mappings, either **click** on a filtered item and drag the mouse to connect it up to an existing item, or **right-click** to add the filtered item as a new item in the destination container / transformation block.

For example, the following diagram demonstrates how data can be filtered so that only "address1" and "address2" are mapped if the "country" is "UK".

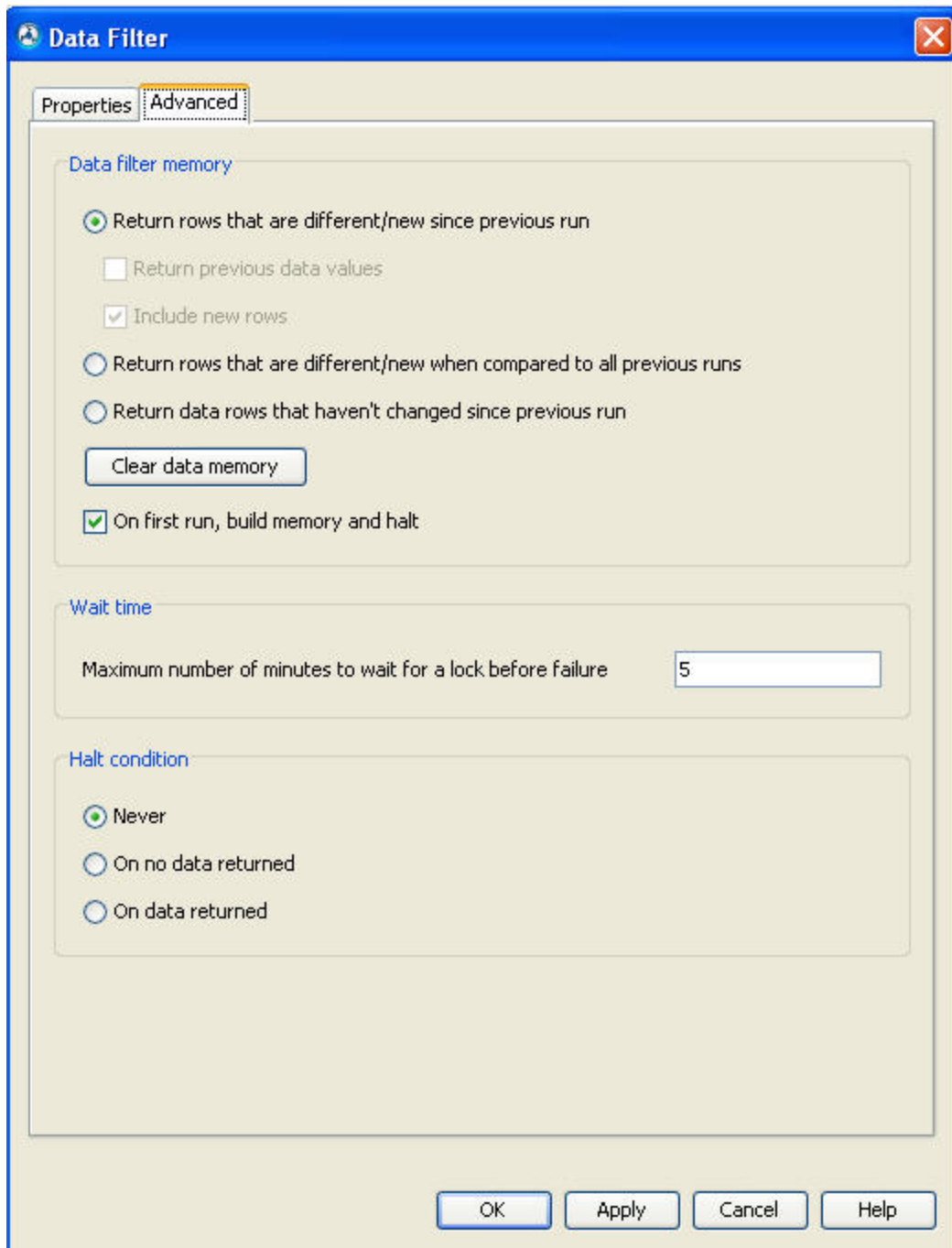


- If the destination section is a data section, and the data is filtered out, then a null value is written out to the mapped field. If the filtered values are referenced in a transformation block then they would have the special value UNDEFINED.

- If the destination section is XML, and the data is filtered out, then either an empty element / attribute is written or no element / attribute is written depending on the value of the **Output empty element if mapped value is empty and no child elements** check box.

Advanced Data Filter Options

You can use the **Advanced** tab of the [Data filter module](#)'s Properties pages to specify the rules that decide what data is filtered out and when processing should halt.



1. Select the required **Data filter memory**.

- If you select Return rows that are different/new since previous run, only those data rows that have changed/new since the time the EMF Process last ran will be returned. This is typically the mode that the data filter is most often used in - to detect changes in data in a database table and to someone of the change (e.g. the price of an item has changed).

Two additional options can also be specified in this mode:

- **Return previous data values** - adds extra columns on to the data section returned containing the values of the fields previously. This can be useful for reporting changed information (e.g. the price changed from €32.45 to €31.65). The extra columns added to the data section will have **OLD_** concatenated onto the field name, so if the column being monitored for changes was named Price, a new column would exist in the data section called OLD_Price containing the previous values.

This option is only available if at least one key column has been defined. See [Specifying Columns to Filter on](#).

- **Include new rows** - selects whether new rows of data should be returned. If this option is deselected then only changed data is returned, and no new rows. For example, if you wanted to inform your customer about a price change in the inventory table, then you would want to deselect this checkbox as you wouldn't want them to get a price change alert when a new item was added to the inventory table.

This option is on by default and can only be de-selected if at least one key column has been defined. [See Specifying Columns to Filter on](#).

- If you select **Return rows that are different/new when compared to all previous runs**, then only rows of data that have never been seen by the data filter before are returned. So, for example, if this option was selected, and the price changed from €32.45 to €31.65 then the data would pass through the filter. If the price then changed back from €31.65 to €32.45, then the data would be filtered out as the price €32.45 had been previously seen.
 - If you select Return data rows that haven't changed since previous run, then rather than passing through the changed data, this will pass through data that hasn't changed. So, for example, if you were monitoring a data section containing a list of tasks, this option could be used in an EMF Process to run weekly to notify of any tasks that hadn't changed over the previous week - so follow up action could be taken to see why.
2. If you want to clear all the existing non-replication data (and thereby cause the Data filter module to run as if for the first time), click **Clear data memory**.
 3. The On first run, build memory and halt option allows the data filter memory to be built the first time the EMF Processes runs, but will then halt that EMF Process branch. This can prevent end users being notified of events that have already happened in the past. e.g. if you have an alert to notify of a new inventory item being added, the first time the EMF Process runs it will notify of all existing inventory items in the system as they all appear to be new to the system. Selecting this option prevents this from happening.
 4. Specify whether the returned data section should be checked and, if so, whether the EMF Process processing should be halted, by selecting a Halt condition:
 - Select **Never** if you always want to pass the data section on, irrespective of whether it contains any data. This is the default option.
 - Select On no data returned if you want to stop the EMF Process if the selected data section is empty.
 - Select On data returned if you want to stop the EMF Process if the selected data section is not empty (i.e. when it contains any data).

5. The Data filter module incorporates a "lock" to guard against situations where the same EMF Process runs twice at the same time and to prevent concurrent instances of the Data filter module, which could cause the filter to run incorrectly. You can specify the Number of minutes to wait for a lock, after which (if it has not been able to run) it will fail and roll back the EMF Process.

Important: If you change any of the Data Filter Memory settings on this page after the Data filter module has been used in an EMF Process, any existing non-replication data may be lost. This will cause the Data filter module to run as if for the first time.

[The Data Filter Module](#)

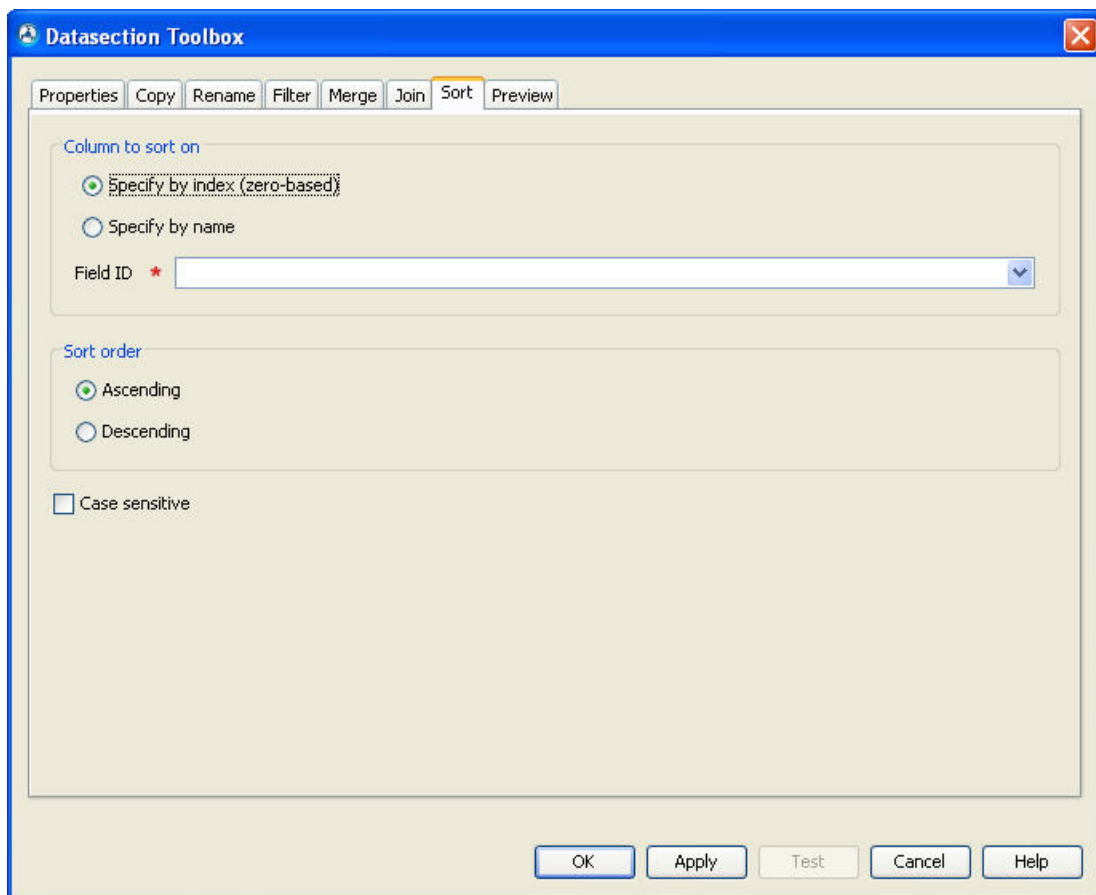
[Specifying Columns to Filter on](#)

[Filter and Transformation Modules](#)

Datasection Toolbox - Sort Datasection

The **Sort Datasection** operation allows you to sort the data in a data section.

To sort a data section:



1. Select the column to sort on from these two options:
 - Specify by index (zero-based)
 - Specify by name.
2. Select the field ID from the drop-down list.
3. Choose the order in which data is sorted. Select **Ascending** or **Descending** in the Sort order section.

[Rename datasection](#)

[Rename columns](#)

[Filter datasection](#)

[Merge datasections](#)

[Join datasections](#)

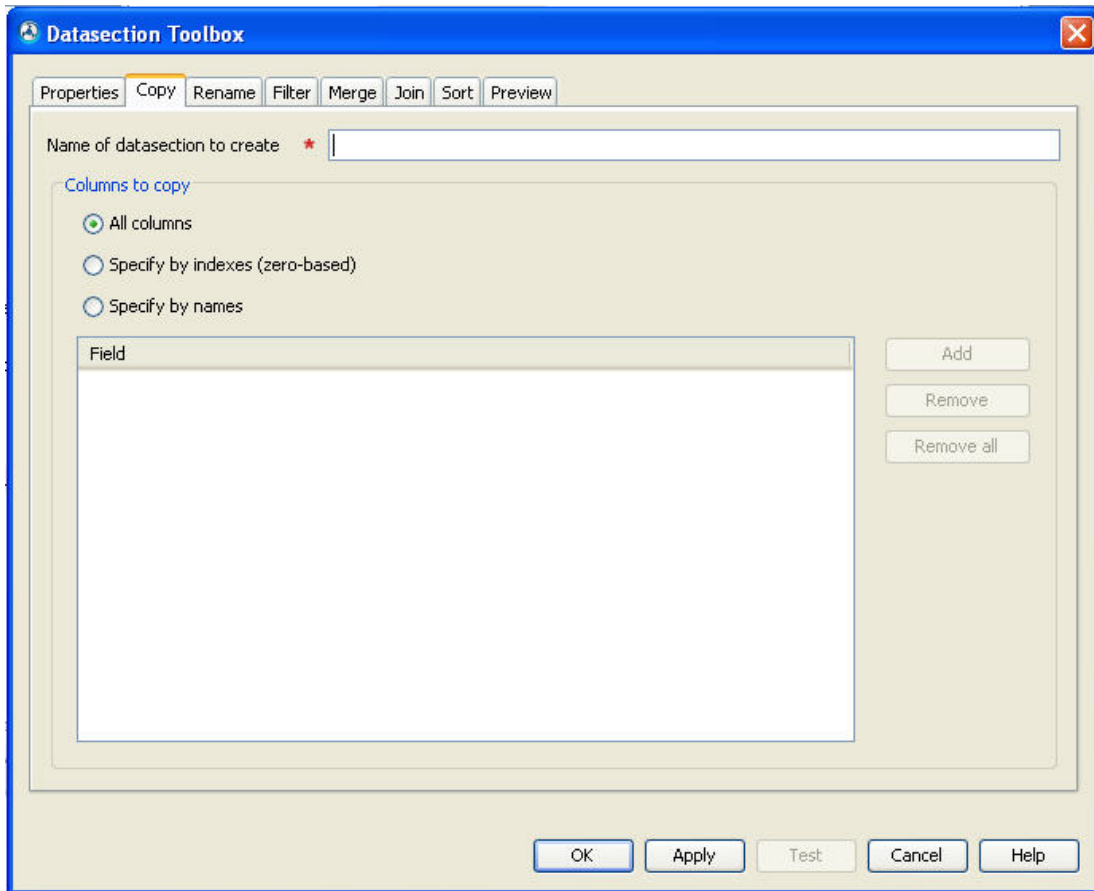
[Filter and Transformation Modules](#)

Datasection Toolbox - Copy Datasection

The **Copy Datasection** operation creates a copy of the datasection specified as the **Primary datasection** in the [Properties](#) tab. An example of where it is useful to copy a datasection is if you wish to filter a datasection but still retain the original datasection - you could create a copy and then filter the copied datasection. This saves having to requery the database, reducing database/network load, and ensures that the operations are performed on the same data (i.e. it hasn't changed between queries).

Note: Operations are performed in the order they appear in the tabs, from left to right, so if copy is selected, then all operations following this occur on the copied datasection.

To copy a datasection:



1. Enter the name of the new datasection that will be created. Dynamic functions may be used in the name (right click on the entry field to access the menus), so the name can be decided at runtime.
2. In **Columns to copy** select the columns that should be copied to the new datasection. By default this is all columns, but if only specific columns are required then these can be specified by column **name** or **index**. **Indexes** are zero based (i.e. the first column is column zero, the second is column 1 etc) - so to copy the 3rd column you would enter an index of 2.

[Datasection Toolbox Module](#)

[Rename datasection](#)

[Rename columns](#)

[Filter datasection](#)

[Merge datasections](#)

[Join datasections](#)

[Sort datasections](#)

[Filter and Transformation Modules](#)

Datasection Toolbox - Filter Datasection

The **Filter Datasection** operation filters out data rows from the **Primary datasection** selected in the [Properties](#) tab.

Note: If you are also performing a copy operation, then the data will be merged into the newly copied datasection, not the **Primary datasection**.

To decide what rows the datasection should contain you must write a filter expression. This is similar to writing a WHERE clause in SQL to filter the rows that should be returned in a query. Being able to filter a datasection allows you to filter data from sources where filtering cannot traditionally be achieved. It can also prevent requiring of the database as copies of the same datasection can be filtered in different ways, rather than re-running multiple database queries.

Each row in the primary datasection is evaluated against the expression, and if it is true then that row is not removed from the datasection.

The filter expression is written in JavaScript, but several automatic conversions have been added so the expressions are more "SQL" like.

Example Expressions

If the primary datasection is:

deliver_y_id	order_id	schedule_d_arrival	actual_arrival	ware-house	arrival_l_bay	arrival_status
DEL1	ORD10001	2006-10-26 15:04:53.0	2006-10-26 17:08:00.0	STAINES	A12	UNLOADED
DEL10	ORD10042	2006-10-26 15:04:53.0	2006-10-25 18:04:00.0	FELTHAM	C8	UNLOADED
DEL2	ORD10002	2006-10-26 15:04:53.0	2006-10-26 11:04:00.0	SLOUGH	A13	ARRIVED
DEL3	ORD10005	2006-10-26 15:04:53.0		STAINES	B5	CANCELLED
DEL4	ORD10015	2006-10-29 16:04:53.0		FELTHAM	C1	EXPECTING
DEL5	ORD10016	2006-10-29 16:04:53.0		FELTHAM	C7	EXPECTING

DEL6	ORD10021	2006-10-29 16:04:53.0		STAINES	A4	EXPECTING
DEL7	ORD10022	2006-10-29 16:04:53.0		FELTHAM	C2	CANCELLED
DEL8	ORD10036	2006-10-26 15:04:53.0	2006-10-27 17:07:00.0	SLOUGH	A10	UNLOADED
DEL9	ORD10039	2006-10-26 15:04:53.0	2006-10-26 16:04:00.0	FELTHAM	C4	ARRIVED

The following expressions will cause the dataset to be filtered as shown:

Show all delivery orders for 'STAINES'

warehouse = 'STAINES'

delivery_id	order_id	schedule_d_arrival	actual_arrival	warehouse	arrival_bay	arrival_status
DEL1	ORD10001	2006-10-26 15:04:53.0	2006-10-26 17:08:00.0	STAINES	A12	UNLOADED
DEL3	ORD10005	2006-10-26 15:04:53.0		STAINES	B5	CANCELLED
DEL6	ORD10021	2006-10-29 16:04:53.0		STAINES	A4	EXPECTING

Show all 'UNLOADED' delivery orders for 'STAINES'

warehouse = 'STAINES' AND arrival_status = 'UNLOADED'

delivery_id	order_id	schedule_d_arrival	actual_arrival	warehouse	arrival_bay	arrival_status
DEL1	ORD10001	2006-10-26 15:04:53.0	2006-10-26 17:08:00.0	STAINES	A12	UNLOADED

Show all orders that arrived after their scheduled time

(NOT (actual_arrival = null)) AND (scheduled_arrival < actual_arrival)

delivery_id	order_id	schedule_d_arrival	actual_arrival	warehouse	arrival_bay	arrival_status
-------------	----------	--------------------	----------------	-----------	-------------	----------------

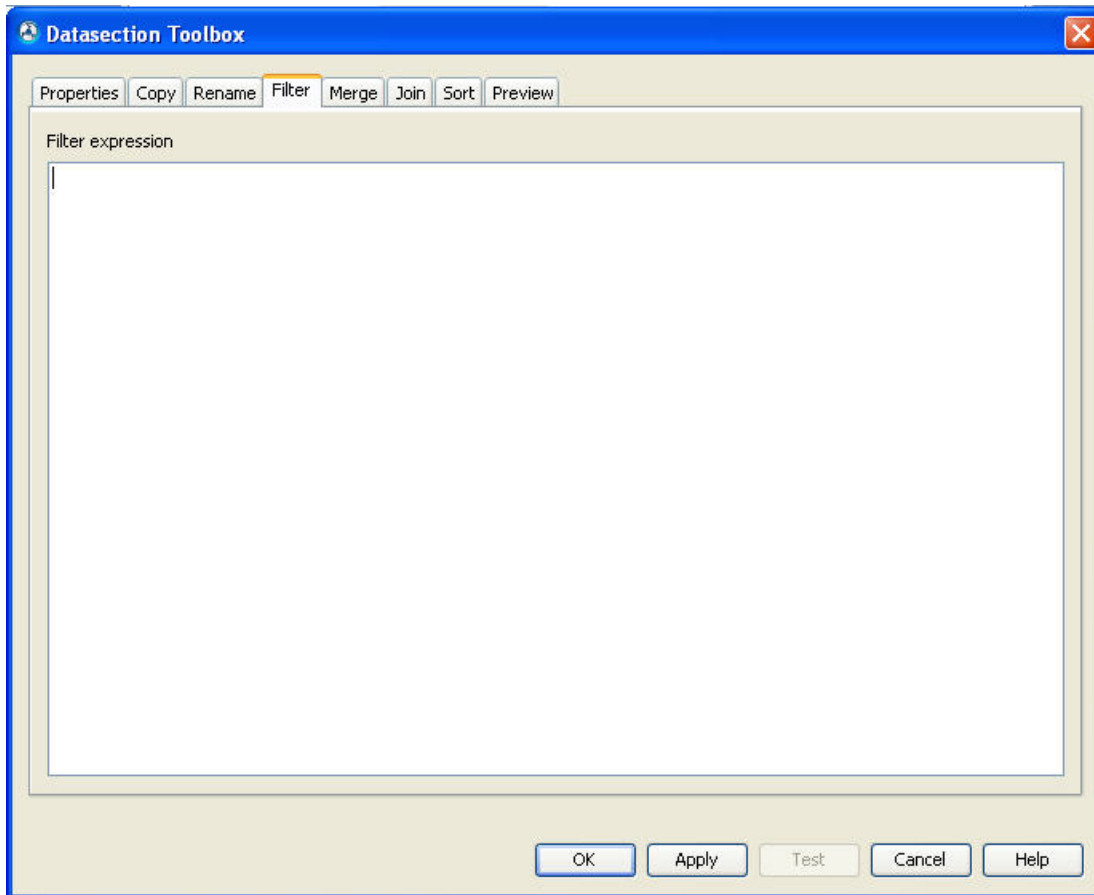
DEL1	ORD10001	2006-10-26 15:04:53.0	2006-10-26 17:08:00.0	STAINES	A12	UNLOADED
DEL8	ORD10036	2006-10-26 15:04:53.0	2006-10-27 17:07:00.0	SLOUGH	A10	UNLOADED
DEL9	ORD10039	2006-10-26 15:04:53.0	2006-10-26 16:04:00.0	FELTHAM	C4	ARRIVED

Show all overdue orders

(arrival_status = 'EXPECTING') AND (scheduled_arrival < NOW())

delivery_id	order_id	schedule_d_arrival	actual_arrival	warehouse	arrival_bay	arrival_status
DEL4	ORD10015	2006-10-29 16:04:53.0		FELTHAM	C1	EXPECTING
DEL5	ORD10016	2006-10-29 16:04:53.0		FELTHAM	C7	EXPECTING
DEL6	ORD10021	2006-10-29 16:04:53.0		STAINES	A4	EXPECTING

To filter a data section:



1. Enter the filter expression required in the **Filter expression** area, of the **Datasection Toolbox** dialog box.

The filter expression is written in JavaScript, but several automatic conversions have been added so the expressions are more "SQL" like.

The following automatic replacements occur to help this:

Text	Replaced with
OR	
AND	&&
NOT	!
=	==
NOW()	Current date time

Dynamic functions are supported. For Example:

Name = \$LOOPDATA('Outer Loop',0)\$"

Date/time comparison examples:

Description	Expression
Check scheduled arrival is before now - the current date/time	<code>scheduled_arrival < NOW()</code>
Check scheduled arrival is before a fixed day	<code>scheduled_arrival < new Date("July 3, 2008")</code>
Check scheduled arrival is before a fixed day/time	<code>scheduled_arrival < new Date("July 3, 2008 13:14:00")</code>
Check scheduled arrival is at least 2 years before now.	<code>scheduled_arrival < new Date("\$DATETIME('MMMM dd, yyyy', 'YYYY', -2, 'now')\$")</code>
Check scheduled arrival is at least 3 months after now.	<code>scheduled_arrival > new Date("\$DATETIME('MMMM dd, yyyy', 'M', 3, 'now')\$")</code>

Further examples:

Description	Expression
Returns all rows where the first letter of column NAME is an "F".	<code>NAME.substring(0,1) = 'F'</code>
Returns true, if the content of COLUMN1 is not a numeric value.	<code>isNaN(COLUMN1)</code>
Returns the integer value of string.	<code>parseInt(Column1,10)</code>

[Datasection Toolbox Module](#)

[Copy datasection](#)

[Rename datasection](#)

[Rename columns](#)

[Merge datasections](#)

[Join datasections](#)

[Sort datasections](#)

[Filter and Transformation Modules](#)

Datasection Toolbox - Join Datasection

The **Join Datasection** operation joins data from a specified datasection with the **Primary datasection** selected in the [Properties](#) tab creating a new third datasection. The original two datasections remain unchanged. The two data sections to be joined may have come from totally different sources e.g. a list of employers from Microsoft Active Directory using the JNDI module, being joined with department information pulled out of an Oracle Database using the SQL module.

Note: if you are also performing other operations, then these will happen first (i.e. data filtered, columns renamed).

The join types are very similar to the joins in SQL. Columns are specified to join on, and the type of join to perform. The following join examples assume the following two datasections exists and are being joined on *DepartmentID*:

Primary Datasection		Joining Datasection	
DepartmentID	LastName	DepartmentID	DepartmentName
3	Bloggs	3	Sales
3	Doe	4	Development
4	Potter	5	Marketing
2	Smith	6	Accounts

The following types of joins are supported:

- **Cross join** - the cartesian product of the rows from the joined datasection.

New joined datasection

DepartmentID	LastName	DepartmentName
3	Bloggs	Sales
3	Doe	Sales
4	Potter	Sales
2	Smith	Sales
3	Bloggs	Development
3	Bloggs	Marketing
3	Bloggs	Accounts
3	Doe	Development
3	Doe	Marketing
3	Doe	Accounts
4	Potter	Development
4	Potter	Marketing

4	Potter	Accounts
2	Smith	Development
2	Smith	Marketing
2	Smith	Accounts

- **Inner join** - the intersection between the two datasections.

New joined datasection

DepartmentID	LastName	DepartmentName
3	Bloggs	Sales
3	Doe	Sales
4	Potter	Development

- **Left Outer join** - all the rows from the primary datasection plus matched values from the joining datasection, or null if no match.

New joined datasection

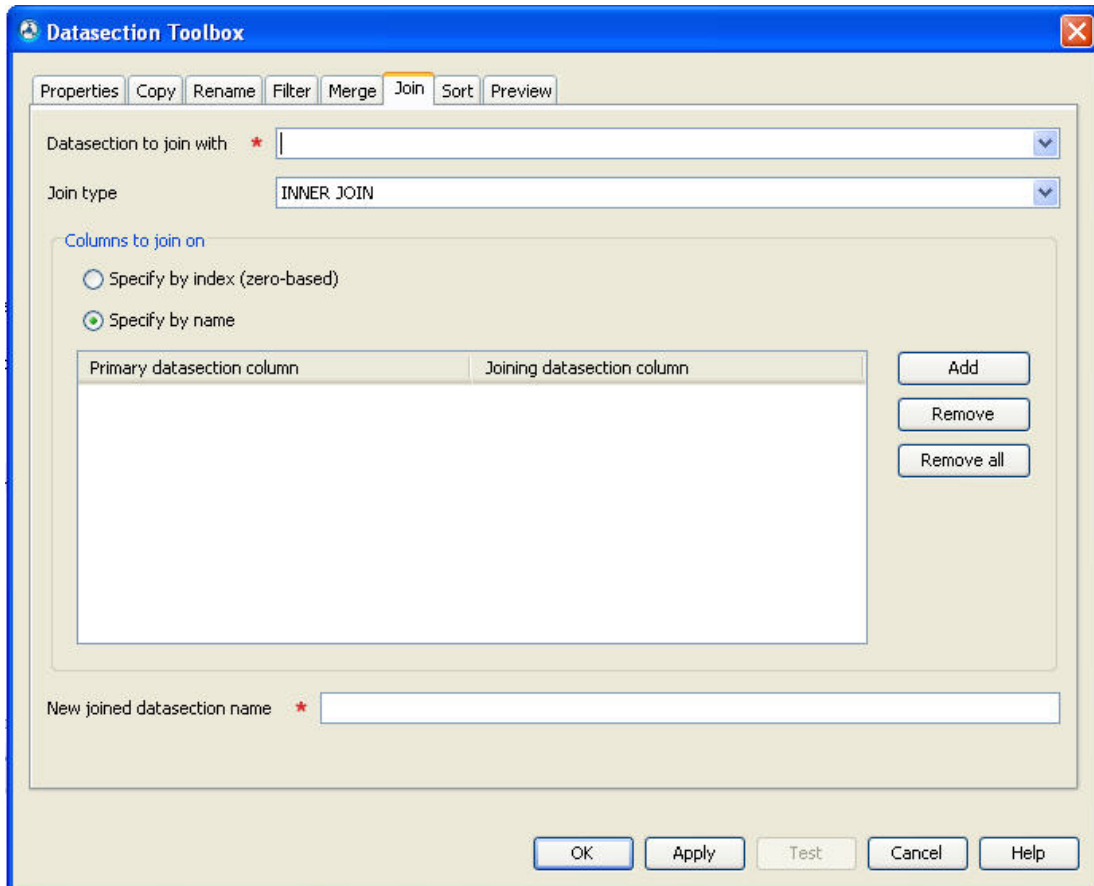
DepartmentID	LastName	DepartmentName
3	Bloggs	Sales
3	Doe	Sales
4	Potter	Development
2	Smith	

- **Right Outer join** - (the reverse of a left outer join) all the rows from the joining datasection plus matched values from the primary datasection, or null if no match.

New joined datasection

DepartmentID	DepartmentName	LastName
3	Sales	Bloggs
4	Development	Potter
5	Marketing	
6	Accounts	
3	Sales	Doe

To join two datasections:



1. Enter the name/select the datasection that is to be joined with the primary datasection.
2. Select the join type, these are explained above.
3. Select the columns to join on. These can be specified either by column name or index. If index is used, column indexes are zero based (i.e. the first column has an index of zero).
4. Enter the name of the datasection that will be created containing the joined datasections in the **New joined datasection name field**.

[Datasection Toolbox Module](#)

[Copy datasection](#)

[Rename datasection](#)

[Rename columns](#)

[Filter datasection](#)

[Merge datasections](#)

[Sort datasections](#)

[Filter and Transformation Modules](#)

Datasection Toolbox - Merge Datasection

The **Merge Datasection** operation merges data from a specified datasection into the **Primary datasection** selected in the [Properties](#) tab.

Note: If you are also performing a copy operation, then the data will be merged into the newly copied datasection, not the **Primary datasection**.

Note: If the **Primary datasection** does not exist then the newly created datasection will be a copy of the datasection to merge with - and given the name specified in **Primary datasection** field. This is useful if you are merging data in a loop, as it means the first time you call the merge, the datasection will be equal to the data you want to merge, and then future calls will merge datasections in to this section. This avoids having to use conditional logic and multiple paths to create the initial data section.

Merging is performed by adding all rows of data from the datasection to merge with into the **Primary datasection**. Only those columns that have the *same name* are copied over. For example, if the primary datasection is:

FirstName	LastName
Fred	Bloggs
Jane	Doe
Harry	Potter

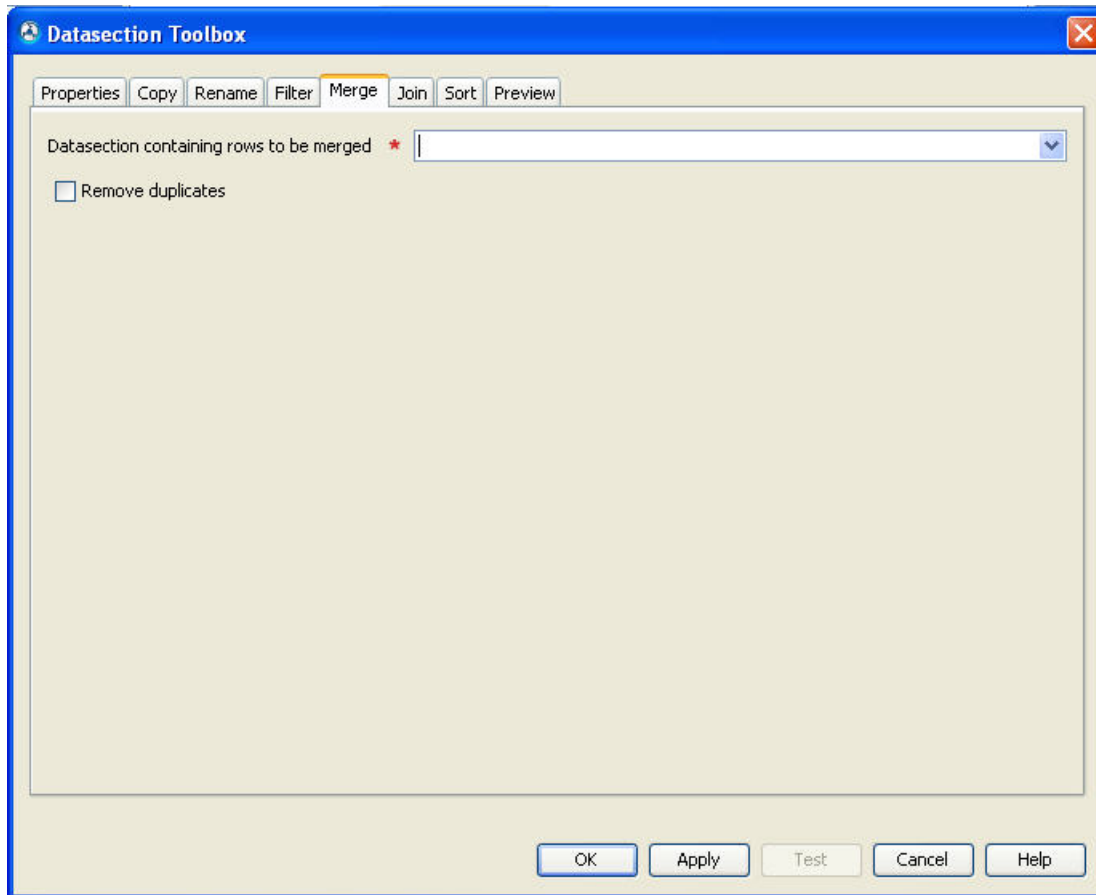
and the merging datasection is:

ID	FirstName	LastName	Sex
1044	Paula	Smith	Female
1045	Harry	Potter	Male

then the primary datasection after the merge would look like:

FirstName	LastName
Fred	Bloggs
Jane	Doe
Harry	Potter
Paula	Smith
Harry	Potter

To merge a datasection:



1. Enter the name/select the datasection that will have its data merged into the primary datasection.
2. Select the **Remove duplicates** check box to remove duplicate data that exists between the two datasections being merged.

[Datasection Toolbox Module](#)

[Copy datasection](#)

[Rename datasection](#)

[Rename columns](#)

[Filter datasection](#)

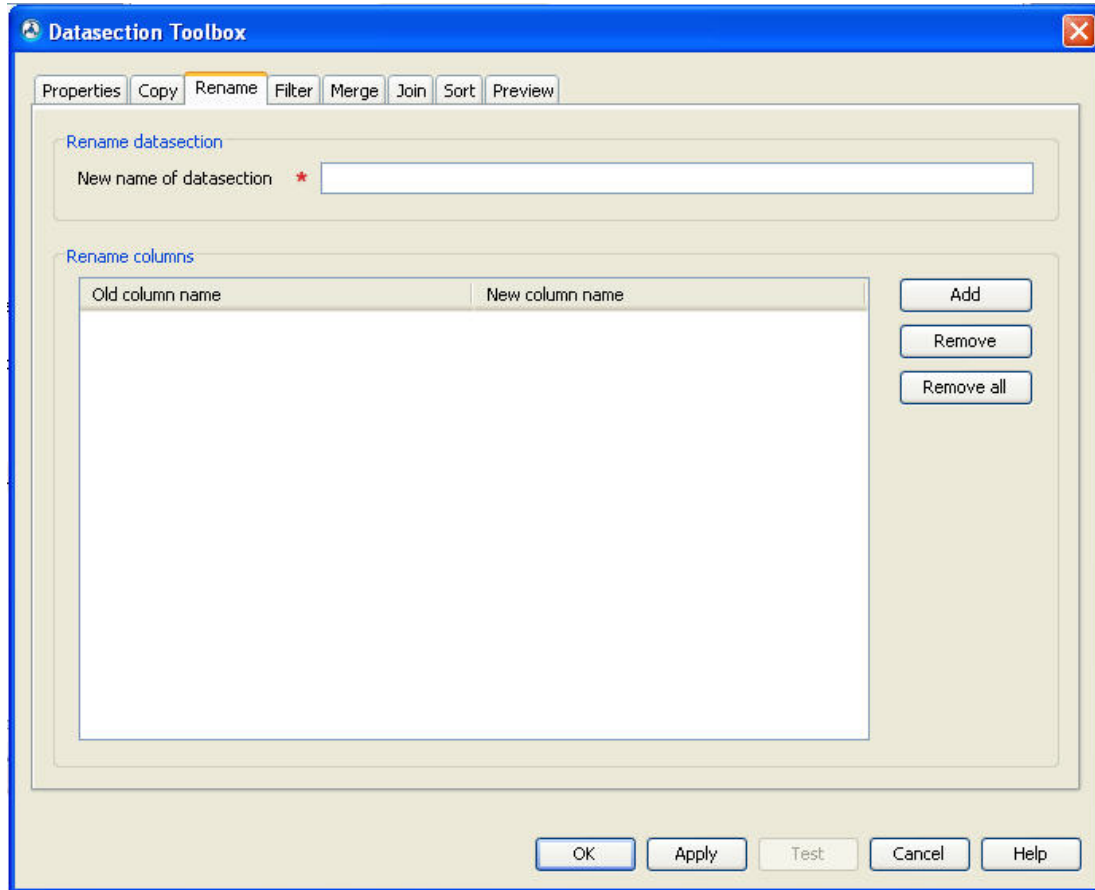
[Join datasections](#)

[Sort datasections](#)

[Filter and Transformation Modules](#)

Datasection Toolbox - Rename Datasection/Columns

The **Rename** tab allows properties to be entered for both the **rename datasection** and the **rename columns** operations.



Rename Datasection

This allows the **Primary datasection** to be renamed.

Note: if you are also performing a copy operation, then you are renaming the copied datasection, not the **Primary datasection**.

To rename the section simply enter the new name of the datasection. Dynamic functions may be used in the name (right click on the entry field to access the menus), so the name can be decided at runtime.

Rename Columns

This allows columns of **Primary datasection** to be renamed.

Note: If you are also performing a copy operation, then you are renaming the columns in the copied datasection, not the **Primary datasection**.

To rename a column, press the **Add** button and then enter the column you wish to rename in **Old column name** and the new name it should have in **New column name**.

[Datasection Toolbox Module](#)

[Copy datasection](#)

[Filter datasection](#)

[Merge datasections](#)

[Join datasections](#)

[Sort datasections](#)

[Filter and Transformation Modules](#)

White List Overview

A **white list** is a list of items that have been marked as being expected or allowable. Within EMF, there are 3 parts to a white list:

- The [White List Definition](#) defines the list and the fields that are used to identify whether an item is white listed or not.
- The [White List Event](#) module is used to mark an item as white listed, i.e., add it to the white list.
- The [White List Filter](#) module is used to filter a data-section for items that may have been white listed.

The white listing feature has been designed primarily for use with Aptean Respond to white list cases within Respond. However, it can be used with any system (or within just EMF) that can present the data in the required format.

When an item is white listed, there is the ability to say for how long it will be white listed. For example, 1 day, or 6 months.

Example

When EMF is used as a continuous auditing tool to monitor payment systems, a rule may be built that will detect payments that a company has made where there is no corresponding invoice. These payments would be flagged up for further investigation. Certain payments, such as those to utility companies (such as electricity and water) often do not have an invoice associated with them, and these regular payments could be white listed to avoid being flagged up repeated investigation.

To build this white listing feature in EMF, a White list definition is created first. This would identify the field names that uniquely identify the company, for example, company name, address and post code.

To white list an item, there must be some system to push the company information (company name, address, post code) into EMF so that the White List Event module can mark a company as white listed. In the case of the Apteian Respond Web UI, it makes an HTTP call to push an XML document to an HTTP Listener in EMF when the user selects to white list an item. A White List Event module then processes this XML document and white lists that company.

When EMF then next runs the rule to detect payments without invoices, the data-section created containing the payments without invoices, is passed through the White List Filter module to remove (or mark as white listed) any items that had been previously white listed. This data-section must contain the columns that were defined in the white list definition (company name, address and post code).

Assume that the following data was detected in a payment system as having no related invoice:

Payment Date	Amount	Company Name	Address	Postcode
2013-02-22	5000	Widget Inc	20 The Square	405305
2013-02-22	391	Cooks Cakes	12 Big Tower	953220
2013-02-23	990	ElecPower	Office 1	593330
2013-02-24	405	Paper Company	Office 1	593330

Each entry would be flagged up as needing investigation in a suitable application (for example, Apteian Respond). Because the amount to the ElecPower company is a regular payment and no invoice is expected, this item is marked by the auditor as white listed so that the next payments to the ElecPower company do not need investigating.

When the rule runs again the following week, the following items are detected:

Payment Date	Amount	Company Name	Address	Postcode
2013-02-28	25030	Nuts Ltd	1 The Road	405305
2013-03-02	990	ElecPower	Office 1	593330
2013-03-02	34	Paper Company	Office 1	593330

The item in red has been detected as white listed, and this result could have either been filtered out of the result set, or returned with the result set and an extra column to indicate that it was white listed.

General Modules

The following General modules are available:

- The [Log Message](#) module allows information to be easily logged to the EMF log. Messages written out by the Log Message module will be displayed in the console window and the EMF server log. This is useful for diagnosing issues, and recording additional information not automatically logged when a process runs.
- The [Scripting](#) module is used to carry out a sequence of commands without any user input.

- The [Debug](#) module is used to create a dump of the running process state to help diagnose any problems that occur when the EMF process runs.

Log Message Module

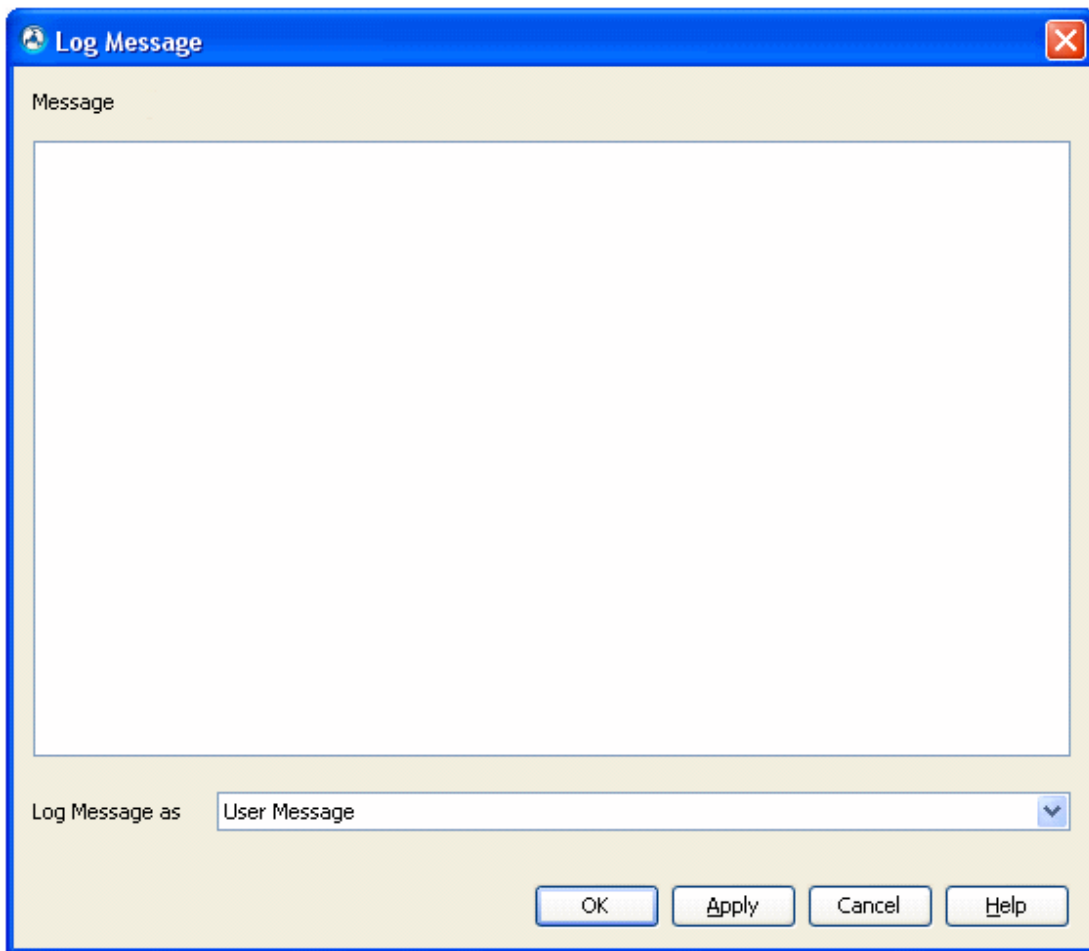
The Log Message module allows information to be easily logged to the EMF log. Messages written out by the Log Message module will be displayed in the console window and the EMF server log in the same standard way as other messages are written out by the EMF server. This is useful for diagnosing issues, and recording additional information not automatically logged when a process runs.

To use a Log Message module:

1. Drag and drop a Log Message module icon into an EMF process. Create a link to it from a module already in the EMF process instance.



2. Double-click the Log Message icon to display the **Log Message** properties screen.



3. Enter the message to be logged. Right-clicking in the message area allows non-personalized dynamic functions to be selected.
4. From the **Log Message as** drop-down list, select the log level that will be used to display the message. The log level for the process is defined in the [Process properties](#). For example, if the process **Debug log Level** is set to **Errors and warnings** and the **Log user messages** option is selected, and the Log Message module is set to log message as User Message, then the console window will display both the errors and warnings and any user messages defined in the Log Message module.

The following options are available for **Log Message as**:

- **Error** - displays the defined message as an error.
- **Warning** - displays the defined message as a warning.
- **Detailed** - displays the defined message as a detailed message.
- **Verbose** - displays the defined message as a verbose message.
- **User** - displays the defined message as a user message.

[Go to Start of General Modules](#)

Script Module

You can use scripting to carry out a sequence of commands without any user input. Instead of defining EMF process details using the icons within the EMF Process Builder, the information required by the EMF process is set using the JavaScript script engine.

For further information on how to use Scripting, please refer to the [Scripting API](#) Help. Also see the [Scripting examples](#).

Using dynamic functions in the scripting module

You can insert [dynamic functions](#) into a script so that calling a function that passes a single string argument containing the dynamic functions to be resolved will return a string containing the result of any functions. This allows for code, like the following, that iterates through a data section one row at a time:

```
for (i = 0; i < resolveDynamicFunctions("$ROWCOUNT('SQL')$"); i++) {
    print(resolveDynamicFunctions("$DATA('SQL', 'index', " + i + ")$"));
}
```

The function `resolveDynamicFunctions(String dfString)` resolves dynamic functions in the passed string `dfString`.

Note: You can also use `df` as a shorthand way of writing the method `resolveDynamicFunctions` - e.g. `df("$DATA('SQL', 'index', 1)$");`.

The following functions can help handle recipient data in the Scripting module:

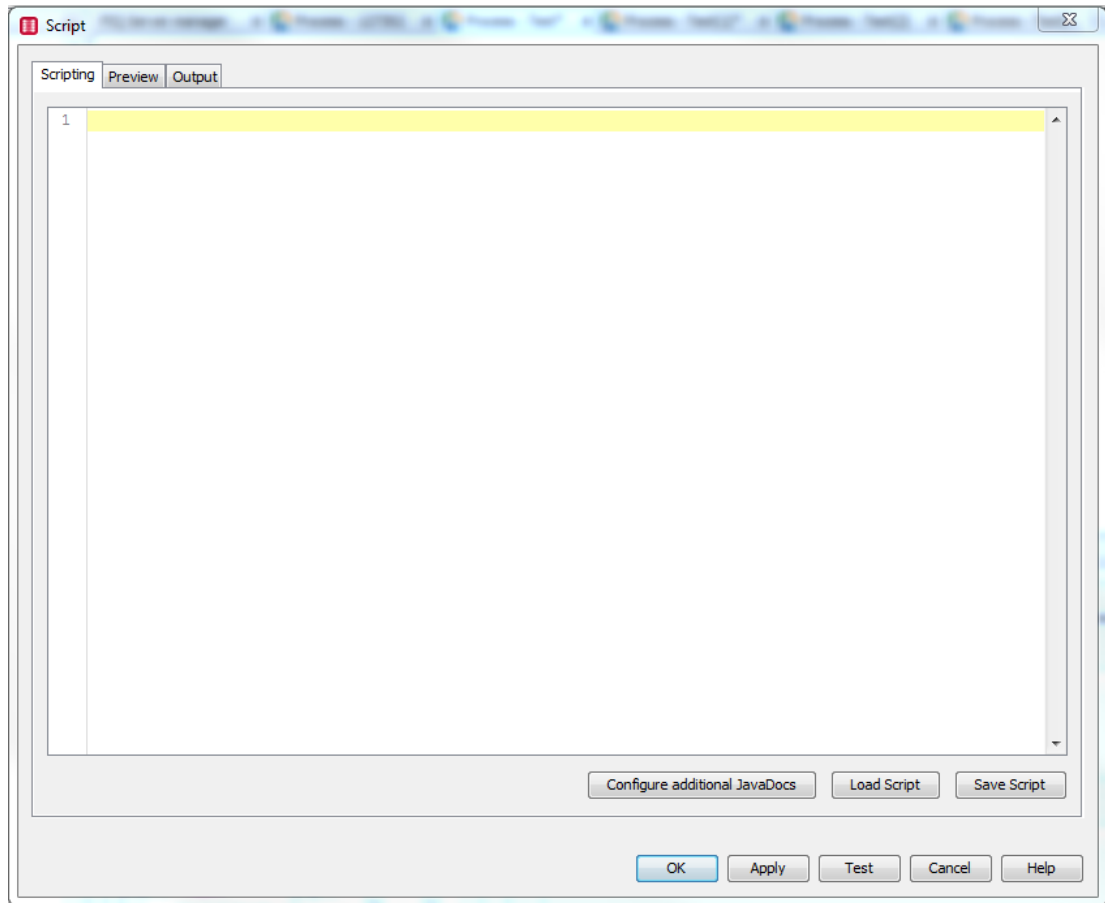
- [RECIPSECTIONCOUNT](#) allows you to get the number of recipient sections.
- [RECIPIENTCOUNT](#) allows you to get the number of recipients (in total or per recipient section).
- [RECIPIENTDATA](#) allows you to get the value of a cell within a recipient section.

To use the script module:

1. Drag a Script module icon into your EMF Process at the appropriate place.

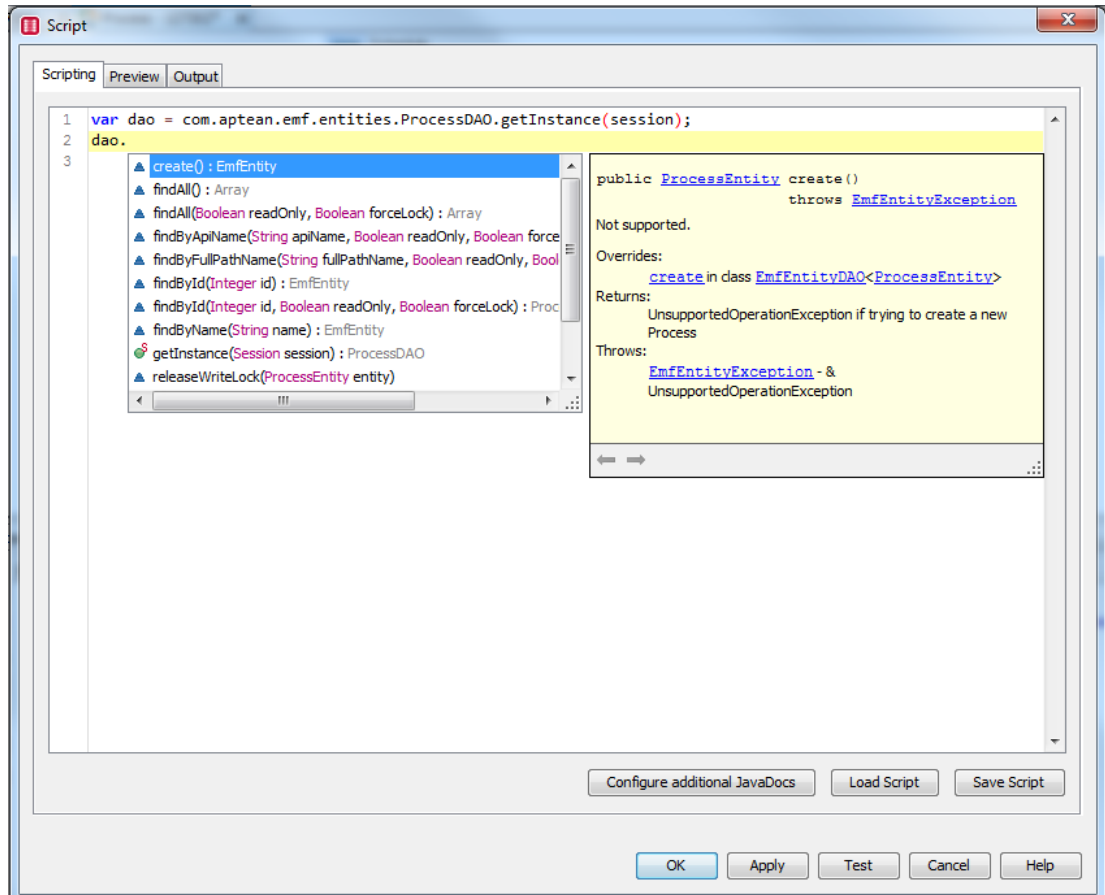


2. Open the Script module icon to display the Script screen. By default, you have to double-click on the icon to open but you can configure mouse-click actions in the Options window. From the Tools menu, select **Options** -> **EMF** -> **Process Builder Behaviour** tab.



3. Use **Configure additional JavaDocs** to add your own JavaDoc files that the Script module will be able to use for giving additional code completion hints. Click **Configure additional JavaDocs** and add the JavaDocs Zip/Jar files.
4. Enter the appropriate script commands into the main area of the Scripting screen. If you have an existing script that you want to use, click **Load Script** to select it.

Hint: Pressing **Ctrl+space** at any time in the script editor window will activate the code completion menu (if any hints are available). The code completion menu gives available options on what you may be entering, avoiding you having to manually type it. Examples of items that will appear on the menu are completing variable names, selecting methods on an object, inserting code samples.



Note: Although you can create scripts in many word processors and other programs, you must save the files as plain text in order to load them into the scripting module.

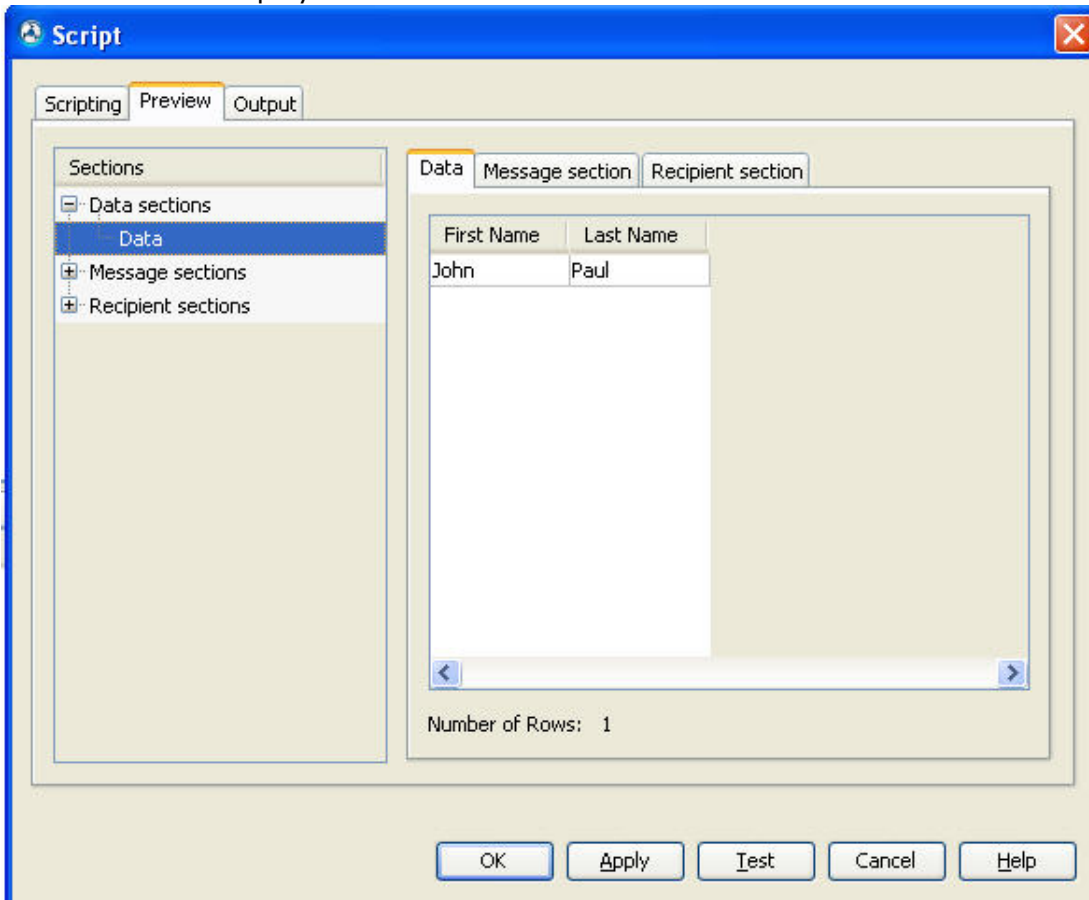
5. If you wish, click **Save Script** to export the text from the scripting module to a file.
6. Click **Test** to preview the results of running the script in the EMF UI before it is run through the server. The Preview tab displays the final contents of any message, data and recipient sections that are contained in the process state after the script has executed. The output tab displays the contents of any text outputted during the execution of the script.
7. Click **OK** to save the current script and close the Scripting module.

Example

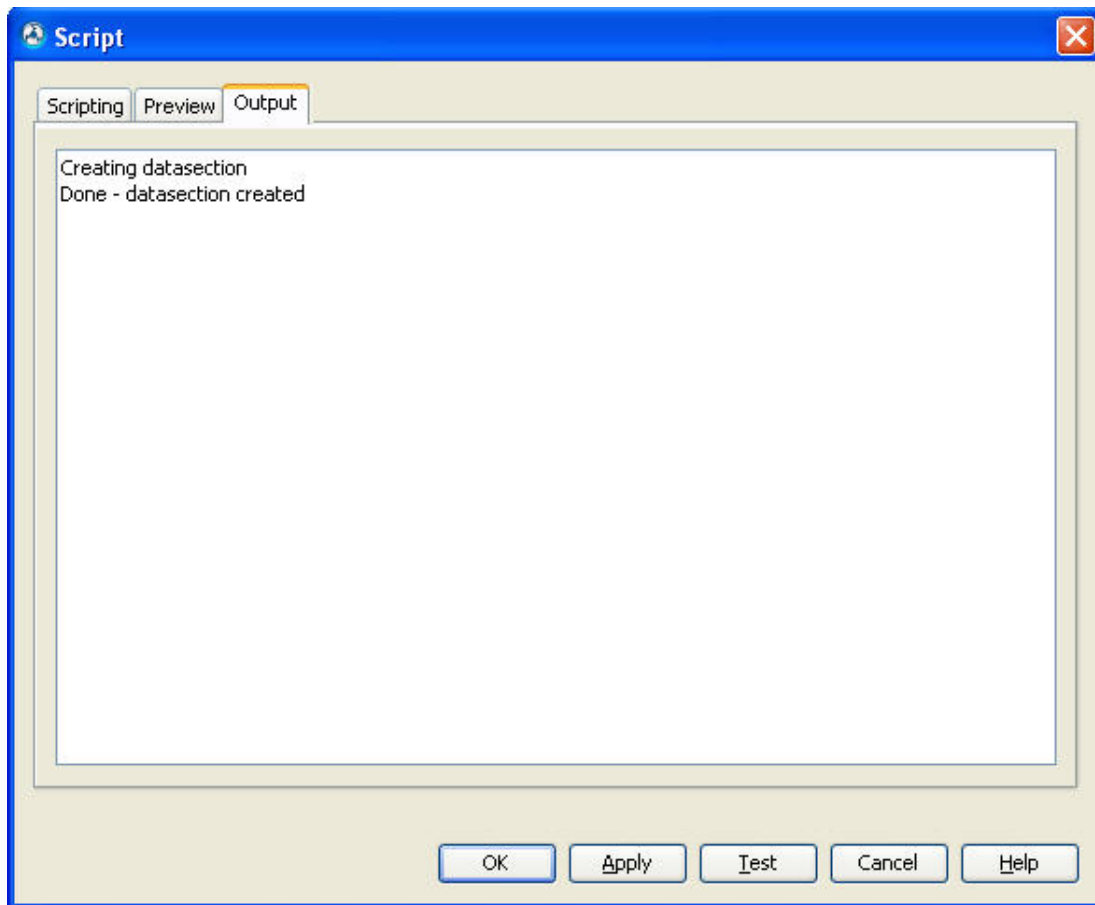
Enter the following script in the Scripting tab:

```
print("Creating datasection");
var columns = new Array();
columns[0] = new Packages.com.xalert.server.api.open.common.Column("First
Name", Packages.com.xalert.server.api.open.common.Column.TYPE_VARCHAR, false, false);
columns[1] = new Packages.com.xalert.server.api.open.common.Column("Last
Name", Packages.com.xalert.server.api.open.common.Column.TYPE_VARCHAR, true, false);
var newDataSection = alertState.addDataSection("Data", columns);
var values = new Array();
values[0] = "John";
values[1] = "Paul";
newDataSection.appendRecord(values);
print("Done - datasection created");
```

To test the script running in the EMF UI before running it through the server, click **Test**. A data section is created and listed in the Sections area of the Preview tab. Click on the data section name to display the data.



The Output tab displays the status messages generated when the script runs.



[Java API Guide](#)

Debug Module

You can use the **Debug** module to create a dump of the running process state to help diagnose any problems that occur when the EMF process runs.

To insert a Debug module:

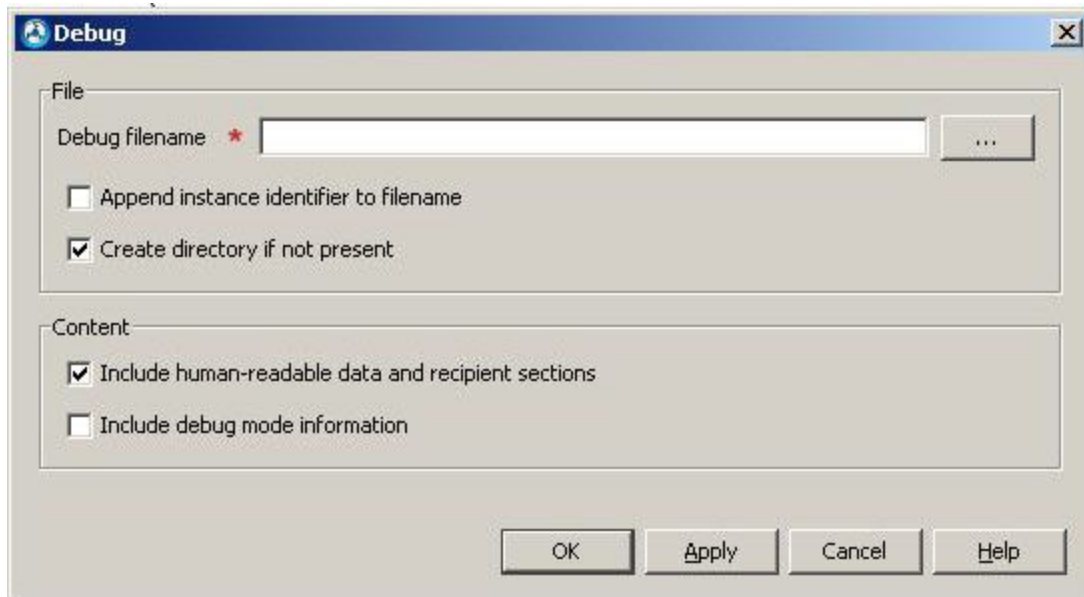
1. Drag the **Debug** icon onto the EMF Process Design Graph at an appropriate place.



You can place one or more Debug Out modules anywhere that you like within an EMF Process, but the output from each will only contain information from the modules that

precede it. To view all the debug information for an EMF Process, the debug module should be placed at the very end of the process.

2. Open the Debug module to display its properties. By default, you have to double-click on the icon to open but you can configure mouse-click actions on the Options window. From the Tools menu, select **Options -> EMF-> Process Builder Behaviour** tab. On the **Debug** screen, you can specify the name of the file that the module should write to, as well as the content of the file. The resulting file will be an XML file containing a complete description of the EMF process.



- **Debug filename** - Specify the path and file name to be used. Once defined, the file will be created when the EMF process reaches this module after it has been fired.

Notes:

- 1) You should not specify a mapped network drive, since the drive might not have the same mapping on the server. If you wish to output to a destination folder that is on a networked machine you should browse via the network node.
- 2) If you choose to type the path manually you should enter the path in a format that is appropriate for the network environment and platform of the EMF Server (e.g. using forward slashes or backslashes).
- 3) The path when the EMF Process is executed is relative to the EMF Server, not the EMF Administrator. For example, if you specify C:\Output.txt as the path in the EMF Administrator the output file Output.txt will be saved to the C: drive of the EMF Server.
- 4) You cannot use the **Browse** button [...] to access a directory on a Unix machine.
- 5) You may not be able to specify a Unix machine without extra third-party software to enable the connection.

- **Append instance identifier to filename** - When enabled, a unique reference to the EMF Process will be added to the Debug Filename so that multiple firing EMF Processes do not return the same Debug Filename.
- **Create directory if not present** (selected by default) - When enabled, this option automatically creates the directory in the specified debug path if it is not found when the file is generated. If the option is disabled, and the target directory does not exist, then the file will not be created and a warning will be written to the server log.
- **Include human-readable data and recipient sections** (selected by default) - When enabled, this option includes human-readable recipient and data sections in the XML. This can make the file substantially larger, so disable the option if you do not need to be able to read the XML.
- **Include debug mode information** (not selected by default) - When enabled, this option includes data and recipient section information in a form suitable for loading into debug mode.

3. Set a debug log level using the **Advanced** tab of the EMF process's [Properties](#) page.

Note: As well as logging errors that occur during the running of an EMF process, you can also [view a log of any errors that occur in the general EMF system](#).

[Go to start of Output Modules](#)

Formatting Modules

The **Formatting** modules are used to format data for Output modules. Data from the EMF Repository or from a Data Section generated by a data module earlier in the EMF process can be used. Each formatter module placed in an EMF Process generates a Message Section, which must have a unique name. You can then select the Message Section in an Output module.

Output formatting modules include [text](#), [HTML](#), [XML](#), and [DSV](#), and you can place dynamic functions into the **Text**, **HTML**, and **XML** Formatter modules in order to complete common tasks without complex coding.

The [Report Generator](#) module allows reports to be generated from a *Jasper Reports* report template.

You can use the [Excel Writer](#) module to output data, primarily from data sections, to create an Excel workbook.

Languages

If you want to support additional languages in the Text, HTML, and XML Formatter modules, you can click the **Languages** button at the bottom of any of the formatter screens to display the [Add/Remove Languages](#) screen. You can then specify which languages are supported by the formatter module.

Important: if you want to make changes to an output formatting module you must have read rights (see [security](#)) to the appropriate language component(s) as well as create, update or delete rights to the module itself. Even if you have full rights to an output formatting module you will not be able to modify it without read language rights.

[Data Modules](#)

[Dynamic Functions](#)

[Text Formatting](#)

[DSV Formatting](#)

[HTML Formatting](#)

[XML Formatting](#)

[Report Generator](#)

[Excel Writer](#)

[Using Different Languages](#)

Text Formatting

You can use the Text Formatting module to format a message for output as plain text. You can add content directly on the text formatter screen, or use [dynamic functions](#) to bring data into the message.

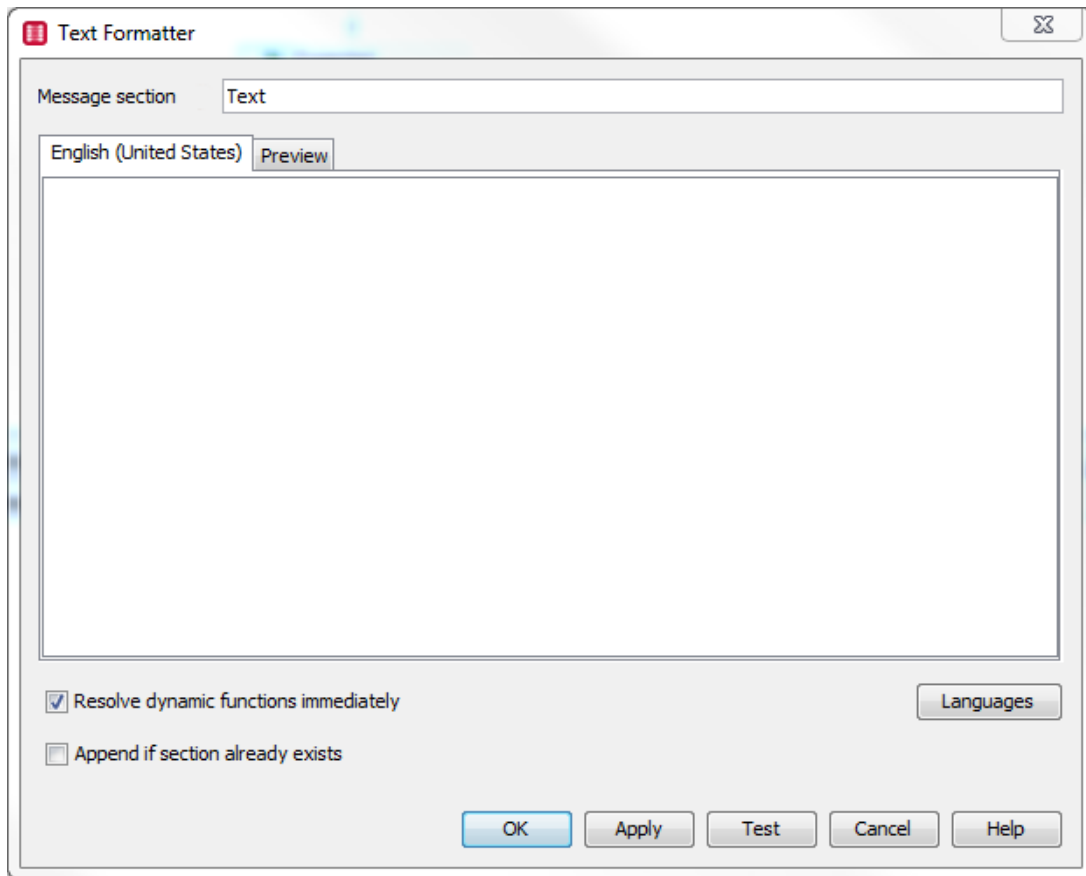
You can format data sections, created earlier in an EMF process by data modules or data from the EMF repository, into a plain text message section. The Message Section created can be sent to users via Output modules.

To add a text formatter module to an EMF Process:

1. In the EMF Process builder, drag the icon for the text formatter module to the appropriate place (generally before one or more Output modules) in your EMF process.



2. Open the Text Formatter module to display its properties. By default, you have to double-click on the icon to open but you can configure mouse-click actions on the Options window. From the Tools menu, select **Options -> EMF-> Process Builder Behaviour** tab.



3. Enter an appropriate name for the **Message section**. The name entered here is referenced by other modules in an EMF process.

Below the Message Section are a number of tabs, one for each of the languages specified using the **Languages** button (see below). Each language has its own copy of the text that is sent out; so that it can be sent in multiple languages. There will only be a single tab if you have not specified any other language apart from the default one.

4. Enter the text of the message in the **Message** pane.

Note: The default content for this module can be set from the **Text Formatter** tab in **Tools -> Options -> Module Defaults**.

5. To use [dynamic functions](#), right-click in the **Message** field and select the required function (**Note** that if you wish to use [Escalation](#) you must include the **ESCALCODE** function). Select **Resolve Dynamic Functions immediately** check box to view the resolved functions. If the **Resolve Dynamic Functions immediately** check box is selected on a formatter module, then the following dynamic functions are not resolved immediately:
 - "AUDITDATA" DF.
 - Any DFs that return binary data (for example, a gif image retrieved from a Web site using the HTTP module).
 - Any of the "per recipient" DFs.

6. Select **Append if section already exists** check box to append any text in the text formatter to the existing text in that message section if the message section of that name already exists. If the message section doesn't already exist, then it will be created with the contents of the text formatter. This option is useful, for example, to build up message section content in a loop with different text from each loop iteration.
7. To send text in more than one language, click **Languages** to display the [Add/Remove Languages Screen](#) where you can specify additional languages.
8. Click **Test** to review the output in the Preview tab.

[Output Formatting](#)

HTML Formatting

You can use an HTML Formatter module to format a message for output as HTML. You can add content directly on the HTML formatter screen, or use [dynamic functions](#) to bring data into the message.

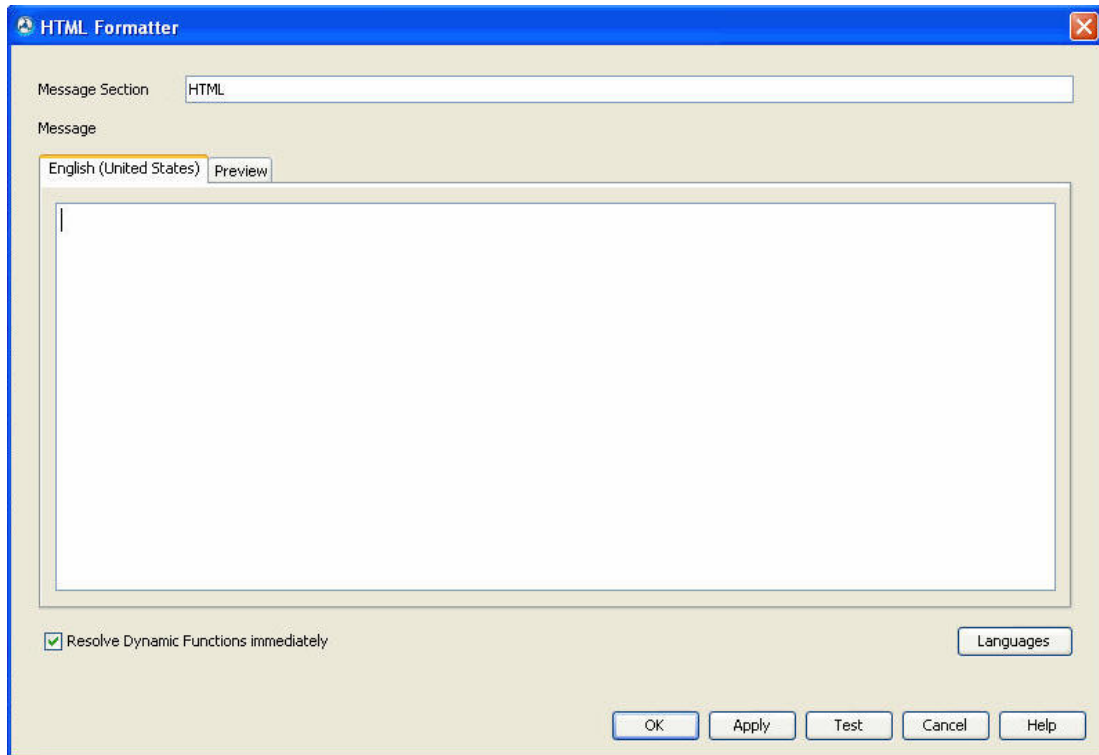
You can format data sections, created earlier in the EMF process by data modules or data from the EMF repository, into an HTML message section. The Message Section created can be sent to recipients via Output modules.

To add an HTML formatter module to an EMF Process:

1. In the EMF Process builder, drag the icon for the HTML formatter module to the appropriate place (generally before one or more Output modules) in your EMF process.



2. Open the HTML Formatter module to display its properties. By default, you have to double-click on the icon to open, but you can configure mouse-click actions on the Options window. From the Tools menu, select **Options -> EMF-> Process Builder Behaviour** tab.



3. Enter an appropriate name for the **Message Section**. The name entered here is referenced by other modules further on in the EMF process.

Note: The default content for this module can be set from the HTML Formatter tab in Tools -> Options -> Module Defaults.

Below the **Message Section** are a number of tabs, one for each of the languages specified using the **Languages** button (see below). Each language has its own copy of the text that is sent out; so that it can be sent in multiple languages. There will only be a single tab if you have not specified any other language apart from the default one.

4. Enter the text of the message in the **Message** pane.
5. To use [dynamic functions](#), right-click in the **Message** field and select the required function

Note: If you want to use [Escalation](#), you must include the **ESCALCODE** function).

6. To send text in more than one language, click **Languages** to display the [Add/Remove Languages Screen](#) wscreen where you can specify additional languages.
7. Select the **Resolve Dynamic Functions** check box, if you want EMF to automatically resolve dynamic functions. If the **Resolve Dynamic Functions immediately** check box is selected on a formatter module, then the following DFs are not resolved immediately:
 - "AUDITDATA" DF.
 - Any DFs that return binary data (for example, a gif image retrieved from a Web site using the HTTP module).
 - Any of the "per recipient" DFs.

8. Click **Test** to review the formatted HTML data in the **Preview** tab. The Preview tab has two panes:
 - The **Raw** text pane displays the HTML source code.
 - The **HTML** pane displays the formatted output as it may appear to recipients in a browser or an email window (output might slightly vary for different browsers/email clients).

[Output Formatting](#)

XML Formatting

You can use the XML Formatting module to format a message for output as XML. You can add content directly on the XML formatter screen, or use [dynamic functions](#) to bring data into the message.

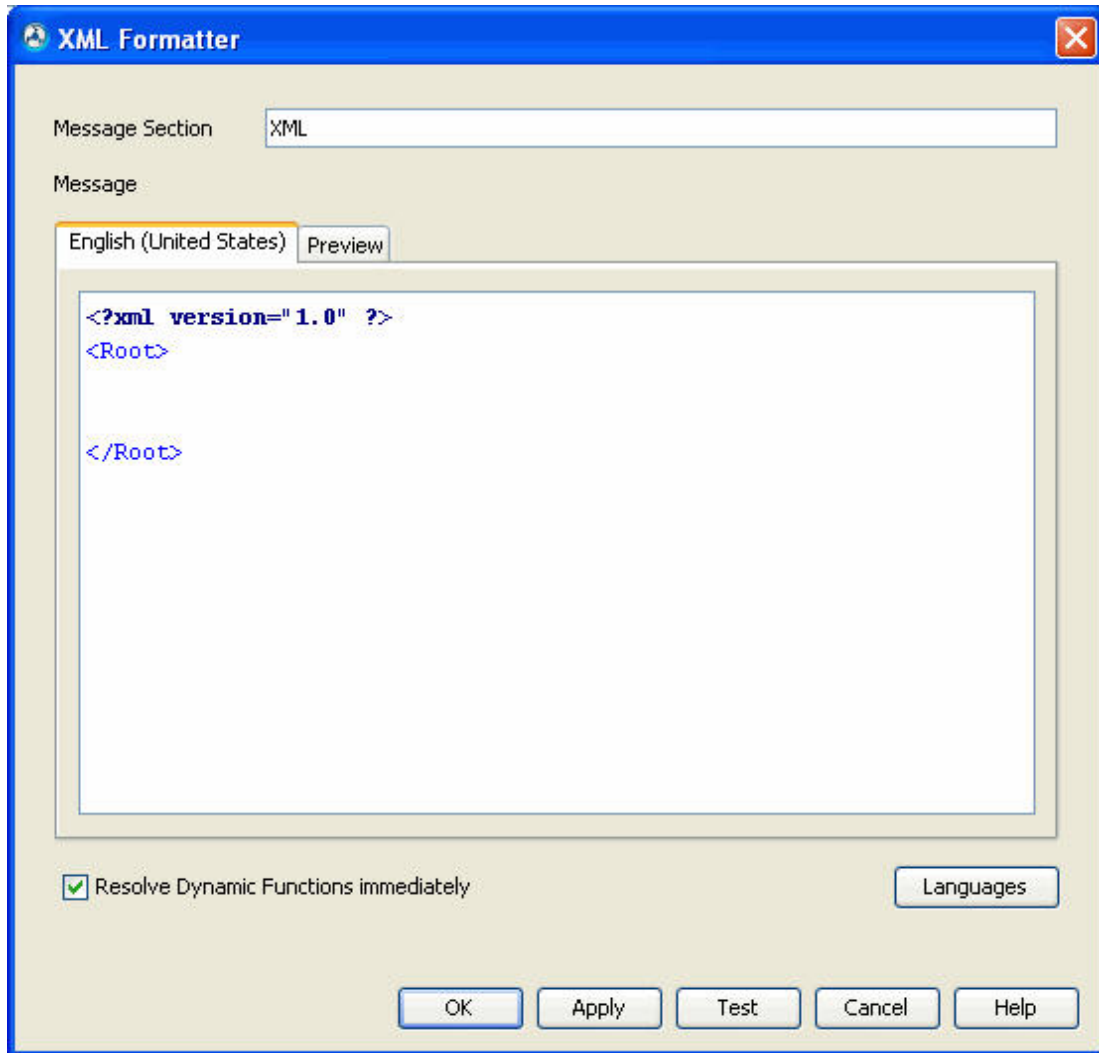
You can format data sections created earlier in an EMF process by data modules or data from the EMF repository, into an XML message section. The Message Section created can be sent to recipients via Output modules.

To add an XML formatter module to an EMF Process:

1. In the EMF Process builder, drag the icon for the XML formatter module to the appropriate place (generally before one or more Output modules) in your EMF process.



2. Open the XML Formatter module to display its properties. By default, you have to double-click the icon to open, but you can configure mouse-click actions on the Options window. From the Tools menu, select **Options -> EMF-> Process Builder Behaviour** tab.



3. Enter an appropriate name for the Message Section. The name entered here is referenced by other modules in an EMF process.

Note: The default content for this module can be set from the **XML Formatter** tab in **Tools -> Options -> Module Defaults**.

Below the **Message Section** are a number of tabs, one for each of the languages specified using the **Languages** button (see below). Each language has its own copy of the text that is sent out; so that it can be sent in multiple languages. There will only be a single tab if you have not specified any other language apart from the default one.

4. Enter the text of the message in the **Message** pane.
5. To use [dynamic functions](#), right-click in the **Message** field and select the required function (**Note** that if you want to use [Escalation](#), you must include the **ESCALCODE** function). Select the **Resolve Dynamic Functions immediately** check box to view the resolved dynamic functions. If the **Resolve Dynamic Functions immediately** check box is selected on a formatter module, then the following DFs are not resolved immediately:

- "AUDITDATA" DF.
 - Any DFs that return binary data (for example, a gif image retrieved from a Web site using the HTTP module).
 - Any of the "per recipient" DFs.
6. To send text in more than one language, click **Languages** to display the [Add/Remove Languages Screen](#) where you can specify additional languages.
 7. Click **Test** to review the formatted XML output in the **Preview** tab.

[Output Formatting](#)

DSV Formatting

You can use the DSV Formatter module to generate a delimited string of field values from a data section generated by a Data Module earlier in an EMF process and output as text. Typically, the DSV Formatter module is used to create a CSV file (that is readable by MS Excel). You can use the default separators (for example, a comma) or any other separator required.

To add a DSV formatter module to an EMF Process:

1. In the EMF Process builder, drag the icon for the DSV formatter module to the appropriate place (before one or more Output modules) in your EMF process.



2. Open the DSV Formatter module to display its properties. By default, you have to double-click on the icon to open, but you can configure mouse-click actions on the Options window. From the Tools menu, select Options -> EMF-> Process Builder Behaviour tab.

3. Enter an appropriate name for the **Message Section**. The name entered here is referenced by other modules in an EMF process.
4. Select the **Data section** that contains information that you wish to process from the list of those in the EMF process (Note: You cannot use a data section that is not in the same path in the process as the DSV formatter).
5. You can specify the text qualifier you want to use for formatting in the **Text qualifier** text box. In the message section created by the DSV Formatter, if the text in the data section contains a delimiter character then that text is wrapped within the text qualifier characters (one at the start of the text and another at the end). This ensures that the entire text is considered as one value.

Example:

Consider the value London,UK. If the comma is a delimiter then this would cause confusion and any program processing the output would treat London and UK as two separate values. If a text qualifier is specified (such as a quotation mark), then the

output becomes "London,UK" and, London,UK would be considered as a single value by any program processing it.

6. Select **Include Column Headers** if you want the first line of the output to be the column headers. The column headers will get values from the column headings of the data section.
7. Select the required **Field delimiter** character, which will be used to separate the fields in the output. You can either select one of the suggested default delimiters, or select **Other** and enter a character in the text box.
8. You can specify the text qualifier you want to use for formatting in the **Text Qualifier** text box. The text qualifier is a character that wraps the delimiter so that it can appear in the text.
9. Select the required **Row delimiter** character, which will be used to separate the rows in the output. You can either select one of the suggested default delimiters, or select **Other** and enter the required character in the supplied field.

Note: If you are using Windows Notepad as your external editor, you should not select **Carriage return** as the row delimiter. Notepad cannot display the CR character correctly. You should use **CR and LF** instead.

10. Select **Include headers** if you want the first line of the output to be the columns headers. The column headers will get their values from the column headings of the data section.
11. Click **Test** to review the delimited field values in the **Preview** tab.

[Output Formatting](#)

Defining Multiple Languages for EMF Processes

You can enter multiple versions of text in a Formatting module, each in a different language. Individual recipients will receive the appropriate version according to their [recipient profile](#). Each language version appears on its own tab in a Formatting module. You can add relevant text for each tab.

You can select the languages that you want to be available using the Add/Remove languages screen.

To Add/Remove languages screen:

- Click **Languages** at the bottom of any of the Formatting module screens.
 - **Languages Available** - This panel shows all the available languages (those that have been defined within the EMF Tree view Snap-in [Languages screen](#)).
 - **Languages Selected** - Each language displayed in this panel has its own copy of the text that is sent out so that it can be sent in multiple languages. If the recipient has expressed a language preference in their individual recipient profile, the message will be sent in that language as long as it has been defined in a Formatting module.

[Output Formatting](#)

Report Generator Module

The **Report Generator** module allows reports to be generated from a *Jasper Reports* report template. These report templates can be written by hand or built visually in the tools provided. The EMF installation comes with a report designer called Jaspersoft Studio, which can be installed from the support files folder of the installation image.

For information on building these templates, please refer to the Jaspersoft Studio help. For information on designing reports in Jaspersoft Studio using data from EMF data sections, see [Using EMF data in Jaspersoft Studio](#).

The **Report Generator** module generates the reports (populates them with data). It can generate reports in the following formats:

- Hypertext Markup Language (HTML)
- Adobe Acrobat (PDF)
- Delimiter Separated Variables (CSV)
- Rich Text Format (RTF)
- Microsoft Excel (XLS)

When the report generator module executes, it takes the report template, populates the template with data, and then generates message sections containing the desired output type. The output can then be outputted using the standard output modules, for example, write the report to disk using the flat file out module, or email the report as an attachment using the email out module.

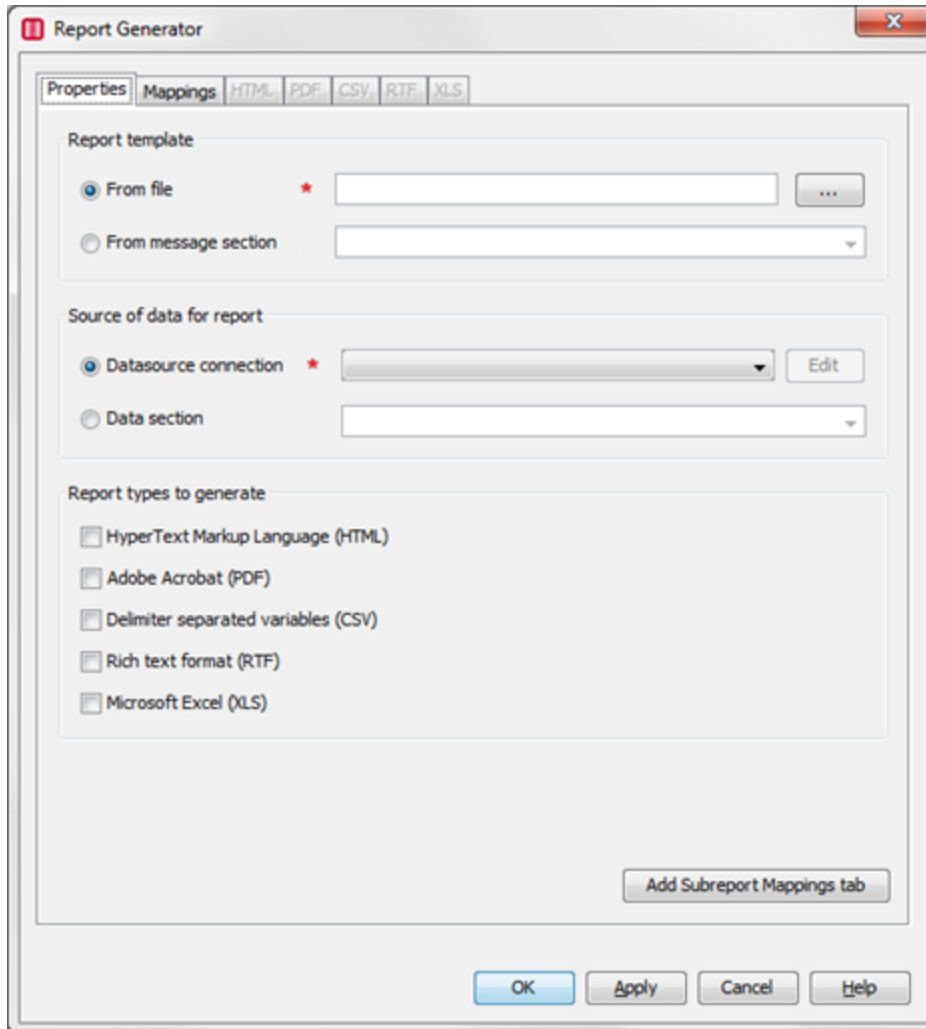
Many reports consist of data from more than one source, or multiple reports within a single report. A report that is contained within another report is known as a subreport. A subreport is a standard report, just that it is referenced from another report.

To use the Report Generator module:

1. Drag the **Report Generator** icon onto the EMF Process Design graph at the appropriate place.



2. Double-click the Report Generator icon to display the Report Generator Properties screen:



3. In the **Properties** tab, in the **Report Template** section, select the source of the report template. This can either be a file (the server must be able to see this file) or a message section containing the template. If a message section is selected, then reports will be generated for all languages. A report template exists in that message section, so that when the report is sent to the output device, it will be sent in the recipients preferred language, if available. Personalization of reports beyond this is currently not supported - so *recipient based* dynamic functions do not currently get resolved. For example, it is not currently possible to generate a report where a recipient's name is inserted into the report using the standard \$RECIPIENT(First name)\$ dynamic function.
4. In the **Source of data for report**, select the source from where the report will get its data. There are two different ways of passing data into a report with EMF. If the report that will be generated gets its data directly from a database and needs no preprocessing by EMF, then select **Datasource connection**. Also, select a datasource connection to the database that the data will be extracted from. You can click **Edit** to modify the properties of the selected Datasource connection. If, however, the data needs to be preprocessed (for example, old data that has been previously reported on being filtered out), or the data comes from a source that a

datasource connection cannot be created for - then it is possible to select an existing *datasection* as the source of the data to report on. The source of the data will have been decided when the report template was designed so the selection made here should match that. That is, if a datasection called "SQL" was used, then a datasection called SQL should be selected in the "Data section" drop down.

5. From the **Report types to generate** section, select the check boxes for the formats of report to be generated. When a check box is selected, the corresponding tab becomes enabled and it is then necessary to enter the format-specific information for the report on the appropriate tab, HTML, [PDF](#), [CSV](#), [RTF](#) and [XLS](#).
6. Select the [Mappings](#) tab to set the values of variables that the report may require.
7. To configure any Subreports that your main report may contain, click **Add Subreport Mappings tab**, which will add a tab in between the **Properties** and **Mappings** tab. The [Subreport Mappings](#) tab will be automatically named. When the **Report Generator** module is opened, any Subreport Mapping tabs will be automatically named and numbered in sequence. For Example, "Subreport 1", "Subreport 2". A subreport mappings tab must be configured for each subreport a main report contains.

Subreports

The subreport element in the Jasper Report allows the use of a report inside another report and it is very powerful because it allows the creation of complex layouts with different portions of a single document filled using different data sources and reports.

Using a subreport within a report is a two step process :

1. The main report must be configured to use the subreport within the report template. This is typically done using JasperStudio, see [Configuration of Report to use a Subreport in JasperStudio](#).
2. The subreport mappings must be configured in the subreport tab so that when the report is run in EMF the correct report template/data is reflected. For more information, see [Report Generator - Subreports](#).

Example Process

[Click this link](#) to save an example process that demonstrates the use of subreports within a main report. The report it produces is a report of most of the items in the EMF repository (e.g. Processes, data sources etc). Once the report is imported you will need to modify the [File out](#) module to point to a directory the server has write access to before running the report. The process produces a PDF file called *Repository.pdf* when run.

[Mappings tab](#)

[HTML tab](#)

[PDF tab](#)

[CSV tab](#)

[RTF tab](#)

[XLS tab](#)

[Using EMF data in Jaspersoft Studio](#)

[Configuration of Report to use a Subreport in JasperStudio](#)

[Formatting Modules](#)

Jaspersoft Studio Configuration for EMF

TIBCO Jaspersoft Studio is a 3rd party tool distributed freely that allows you to visually create report templates that can be used by EMF to generate reports. This help was written for version 6.0.1 of Jaspersoft Studio.

Reports typically take some data and output it. There are two different ways of getting hold of this data, these are explained in the following sections.

Querying the database directly

To get your data directly from the database Jaspersoft Studio needs to be configured with a JDBC connection to the database. The only extra configuration needed in Jaspersoft Studio is to add your JDBC driver to the Jaspersoft Studio classpath. This is done when setting up the Data Adapter.

Create a Data Adapter by selecting **Data Adapters > Create Data Adapter**. Select **Database JDBC Connection** and click **Next**. In the **Driver Classpath** tab, add the JDBC Driver.

Configure your driver on the **Database Location** and **Connection Properties** tabs. Details of driver settings should be provided with the JDBC. If, for example, you are using the jTDS driver to connection to a Microsoft SQL Server database, then the JDBC Driver name would be "net.sourceforge.jtds.jdbc.Driver" and the JDBC URL "jdbc:jtds:sqlserver://localhost/<My database>".

Now you can build a report template. In EMF, when selecting the report template to generate a report module for, you must also select a datasource connection that has been configured to access the same database as the one that was configured in Jaspersoft Studio.

Using a EMF datasection as the source of data

The second method of obtaining data for a report is to use data currently in an EMF datasection. As this data only exists while an EMF Process is running and is not stored anywhere (such as a database) it is necessary to capture a snap shot of the alert data to help with the design of the report template.

Before the snapshot is captured, some additional configuration to Jaspersoft Studio is needed.

1. In the **Project Explorer** tab, select the top node (typically **MyReports**).
2. Right-click and select **Build Path > Configure Build Path**.

3. Go to the **Libraries** tab for the **Java Build Path**. Add in the external jar files *xa_admin.jar*, *xa_server.jar* and *jakarta-regexp-1.1.jar* from the `EMF\server` and `EMF\server\lib` directories
4. Select the **Repository Explorer** View (**Window > Show View > Repository Explorer**).
5. Right-click the **Data Adapters** node and select **Create Data Adapter** to bring up the **Data Adapter Wizard**.
6. Select the **JasperReports DataSource Provider class** Data Adapter.
7. Click **Next** and name your connection. Enter **com.xalert.jasper.XalertsJRDataSourceProvider** as the **JasperReports DataSource Provider class**.

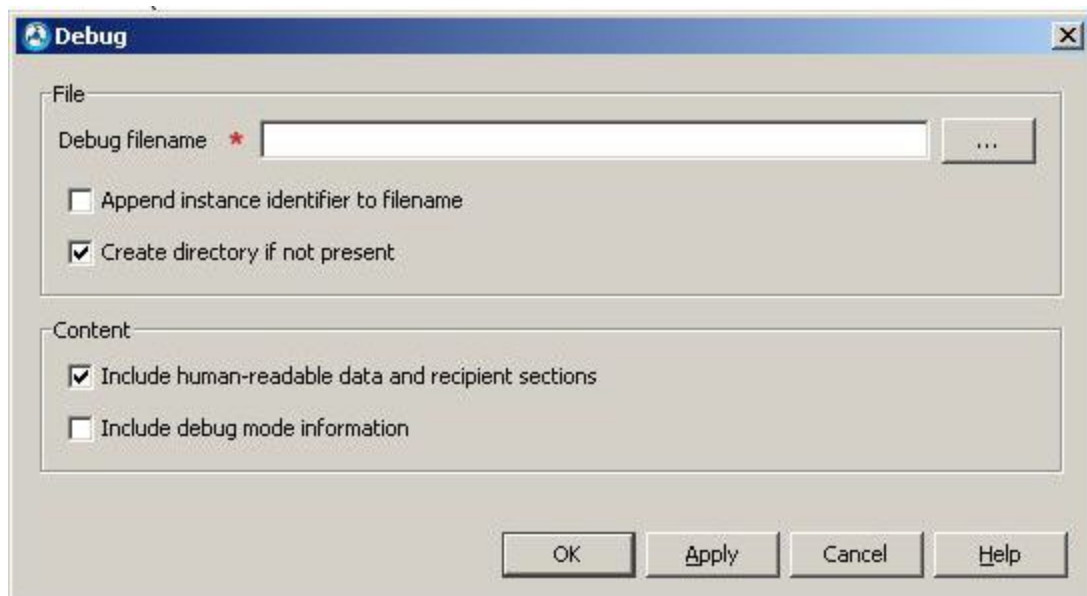
Note: Do not click **Test** at this stage as the properties required to test the custom DataSource Provider have not yet been setup.

Capturing sample data

To build a report template that gets data from an EMF data section, you need to capture some sample data.

To capture sample data:

1. Open an EMF Process that contains a data section with required data.
2. Drag and drop a **Debug** module icon into the process, and create a link to it from the existing data section.
3. Double-click on the **Debug** module to display the Debug screen.

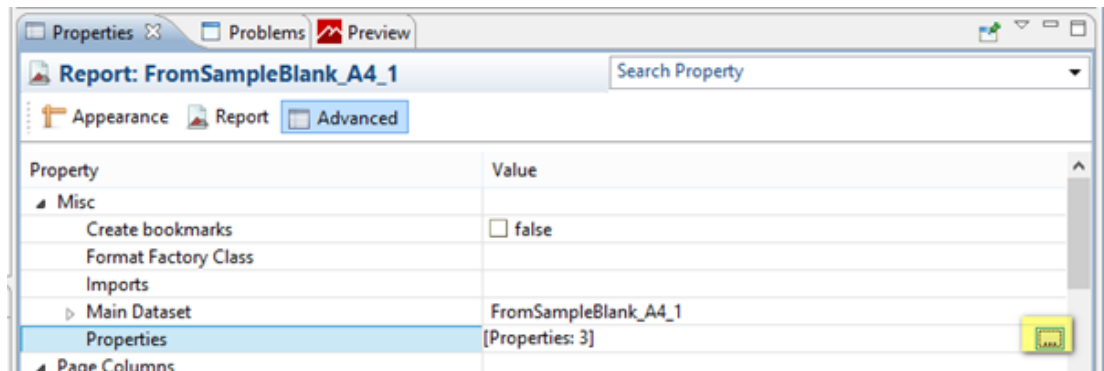


4. Select the **Include debug mode information** option, and enter the name of the file that you want to create in the **Debug filename** field.

5. Run the EMF Process. This captures the sample data and saves it in the specified file. You can now remove the debug module from the process.

In Jaspersoft Studio, to use the sample data to build a report template:

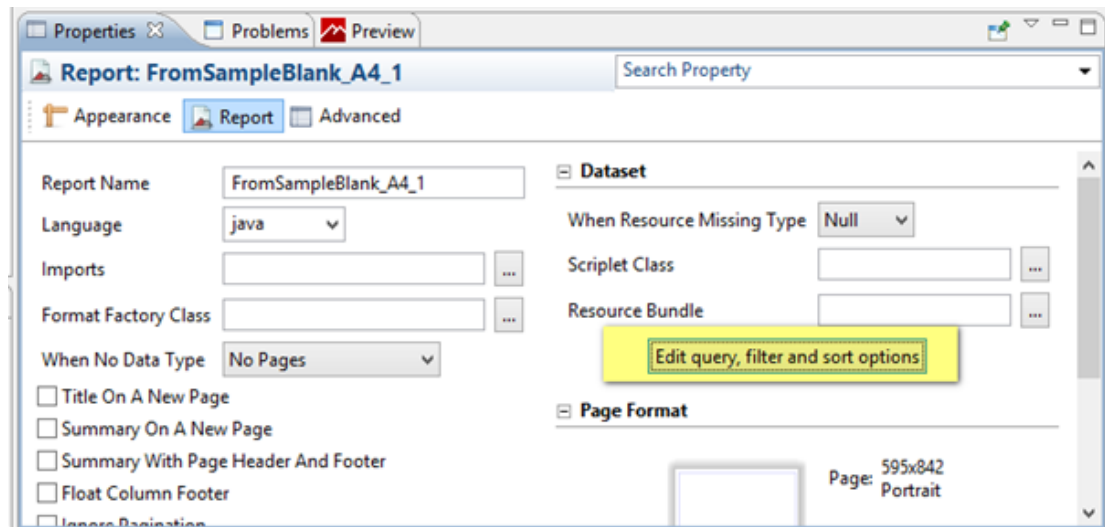
1. Open/create the report template you want to use.
2. Select the report element in the **Outline** view (**Window > Show View > Outline**). The properties for the whole report will now be displayed in the **Properties** view.
3. Select **Advanced** to display the report properties.
4. Select **Properties** in the **Misc** node.
5. Click the ellipse button in the **Value** field for **Properties** to bring up the **component properties** dialog box.



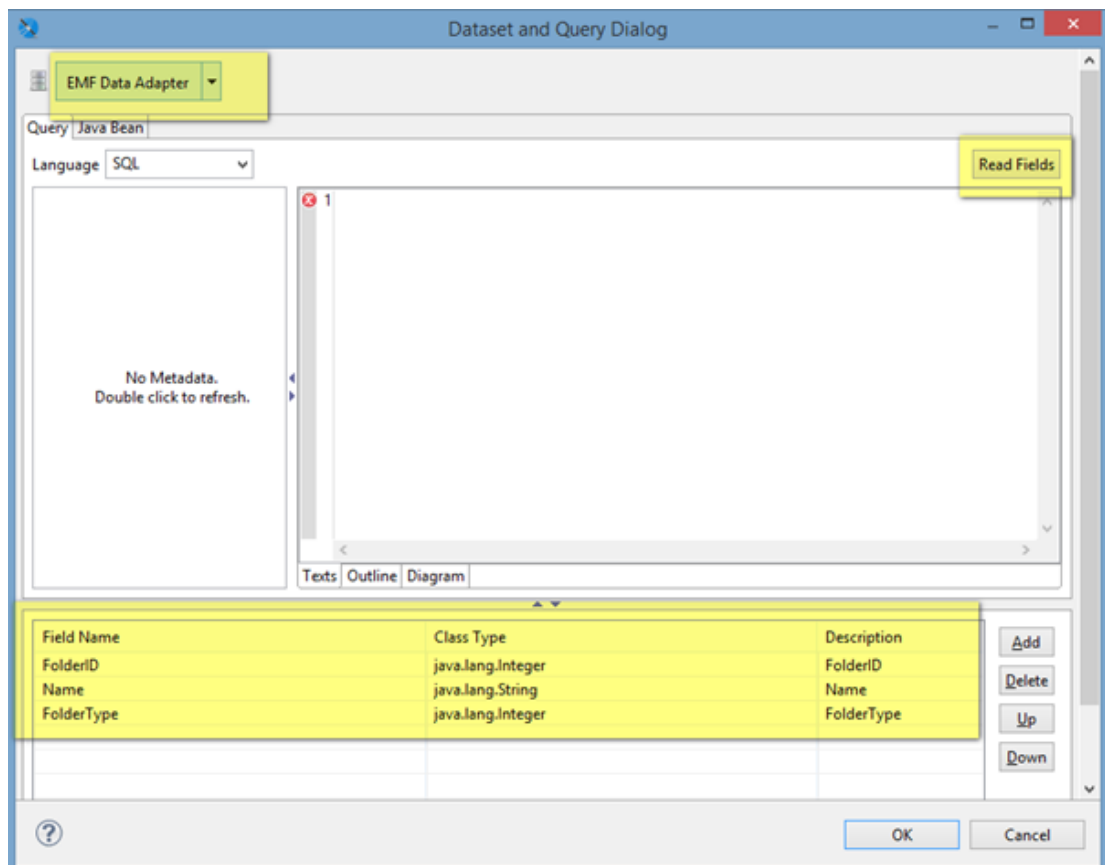
This should already have the data adapter selected above. Add two new properties as follows:

Property	Value
emf.exampleProcessStateFile	The name of debug file generated by EMF containing the sample data. When you enter the filename, you need to either use \\ for every single \, or use / instead of \. For example, enter c:\debug.xml as either c:\\debug.xml or c:/debug.xml.
emf.datasectionName	The datasection name containing the data of interest

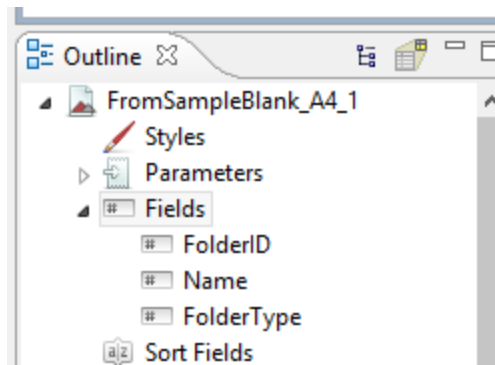
6. In the **Report properties** view, select **Report**.
7. Select **Edit query, filter** and **sort** options in the **Dataset** node.



8. Select the external datasource connection you configured earlier.
9. Click **Read Fields**. If everything is correctly configured, the fields in the sample data should be read and populated in the grid under **Fields** tab.



The fields must now be available in the **Fields** element in the **Outline** view and can be drag and dropped on to the report template.



[Properties tab](#)

[Mappings tab](#)

[PDF tab](#)

[CSV tab](#)

[RTF tab](#)

[XLS tab](#)

[Formatting Modules](#)

Report Generator - Subreports

The **Subreport** tab allows the configuration of the Subreport settings used by the main report to make sure the correct template/data/parameters are passed from the main report to the subreport at runtime.

The **Subreport** tab is divided into three sections.

Subreport template

This section is used to select the source of the subreport template.

- **Template parameter name** – This is the name of the subreport that the settings are mapped onto in the main report. This parameter must be defined in the main jrxml report file.
- **From file** – If this option is selected, the source of the subreport template will be a file. The server must be able to access this file.
- **From message section** – Select this option if a message section contains the subreport template. If this option is selected, then reports will be generated for all languages defined in the message section. A report template exists in that message section, so that when the report is sent to the output device, it will be sent in the recipients preferred language, if available. Personalization of reports beyond this is currently not supported - so *recipient based* dynamic functions do not currently get resolved. For example, it is not currently possible to generate a report where a

recipient's name is inserted into the report using the standard \$RECIPIENT(First name)\$ dynamic function.

Source of data for subreport

This section defines the source of data for the subreport. The source of the data will have been decided when the subreport template was designed.

- **Datasource parameter name** – This is the name of the Datasource in the subreport. This parameter must be defined in the *main jrxml* report file.
- **Datasource connection** – Select this option if the subreport will be generated with data obtained directly from a database. The Datasource connection must be selected from those available. The **Edit** button can be used to modify the properties of the selected Datasource connection.
- **Data section** – This option allows you to select a data section containing data that has been created previously in the EMF process. For example, from a data section that has been filtered, or the data comes from a source that a Datasource connection cannot be created for (such as a CSV file).

Subreport mappings

This is similar to the **Mappings** tab. It allows the subreport template defined parameters to be replaced with a value at report generation time. For instance, the title of the subreport could be dynamically set depending upon the company department it was being generated for.

- **Map parameter name** – This is the name of the mappings section defined in the main report. This parameter must be defined in the *main jrxml* report file if mappings are used. At runtime, each parameter defined in the map will be created and be available to the subreport.
- **Parameter mappings** – These can be added for the subreport by using the **Add** button and entering the **Parameter name** and the **value** it should have. The name/value may use dynamic functions which will get resolved at runtime.
- **Delete Subreport Mappings tab** – Use this button to remove the Subreport mappings tab from the report.

[Report Generator Module](#)

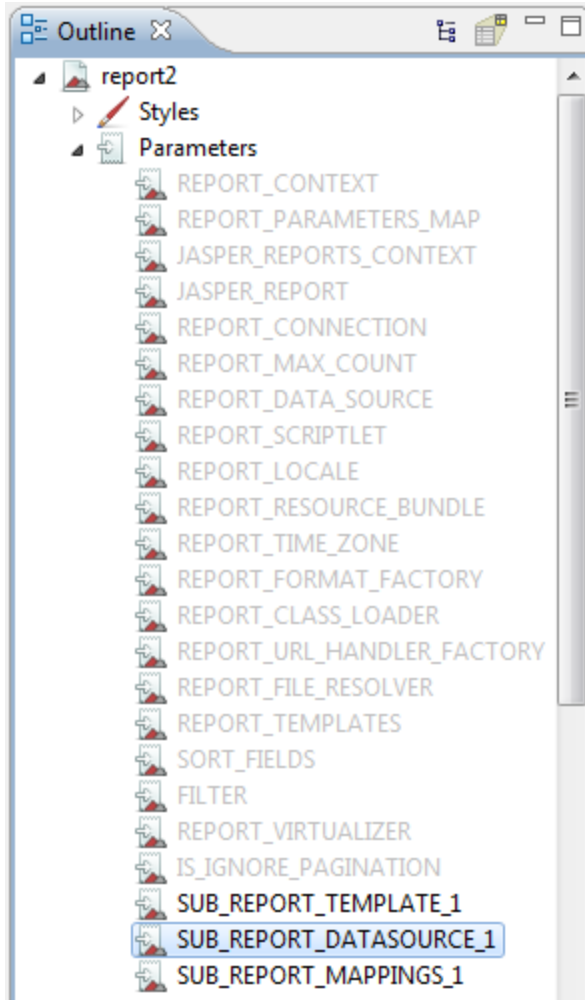
[Configuration of Report to use a Subreport in JasperStudio](#)

Configuration of Report to use a Subreport in JasperStudio

Each subreport that is added to the main report must have parameters defined to enable the report to identify the template/data/parameters that the subreport will use.

For each subreport a report includes the following set of parameters need to be defined so that EMF can populate these at runtime to generate the correct subreport:

1. The subreport report template (referred to as the "subreport template parameter").
2. The data that will be used by the subreport, either a datasource or a data section (referred to as the "subreport datasource parameter").
3. An optional map parameter that contains a map of additional values that the subreport uses (referred to as the "subreport map parameter").



The parameters are defined in the main report as follows:

1. Select the report element in the **Outline** view (**Window > Show View > Outline**). Expand the report element and right click on the Parameters node. Select **Create Parameter** from the popup menu, and enter a name in the **Name** field of the Properties view (for example, SUB_REPORT_TEMPLATE_1). This parameter will be the **Subreport template parameter**. Enter a name in the **Name** field. The parameter Class must be set as "net.sf.jasperreports.engine.JasperReport". Note that this name is used as the **Template parameter name** in the subreport tab of the **Report**

Generator module.

Parameter: ...TEMPLATE_1

Search Property

Object Advanced

Name SUB_REPORT_TEMPLATE_1

Class net.sf.jasperreports.engine.JasperReport

Description

Is For Prompting ☒ Is For Prompting

Default Value Expression

2. Repeat [step 1](#) to create another parameter. This parameter will be the **Datasource parameter**. Enter a name in the **Name** field (for example, SUB_REPORT_DATASOURCE_1). Note that this name is used as the **Datasource parameter name** in the subreport tab of the **Report Generator** module. If the source of data is a Datasource connection, the parameter Class must be set as "java.sql.Connection". If the source of data is a Data section, the parameter Class must be set as "net.sf.jasperreports.engine.JRResultSetDataSource".

Parameter: ...ATASOURCE_1

Search Property

Object Advanced

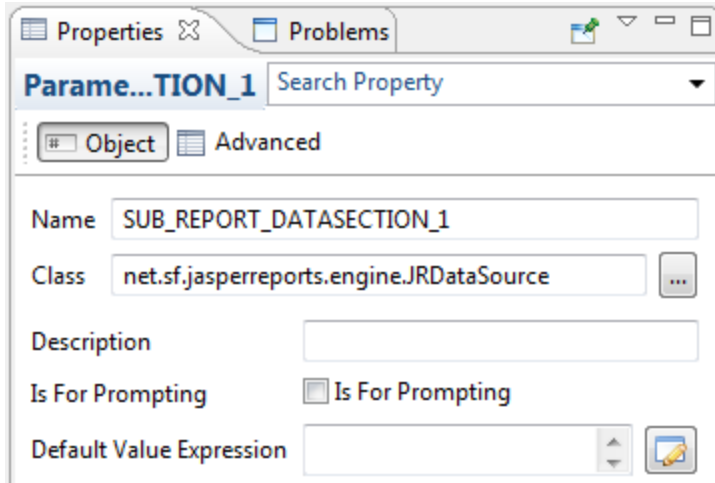
Name SUB_REPORT_DATASOURCE_1

Class java.sql.Connection

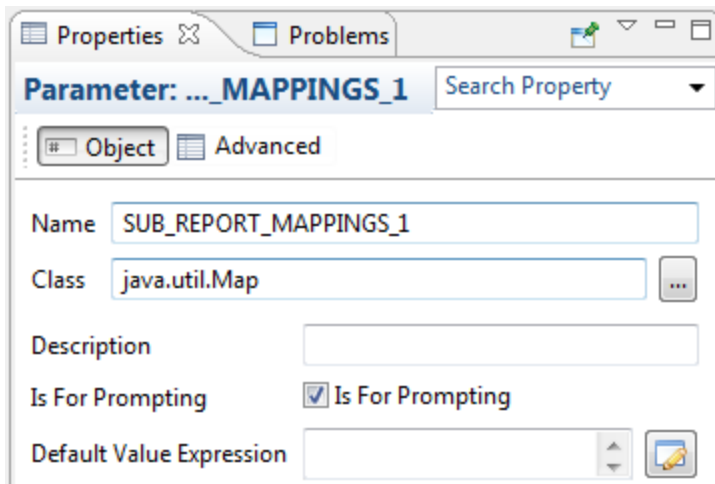
Description

Is For Prompting ☒ Is For Prompting

Default Value Expression

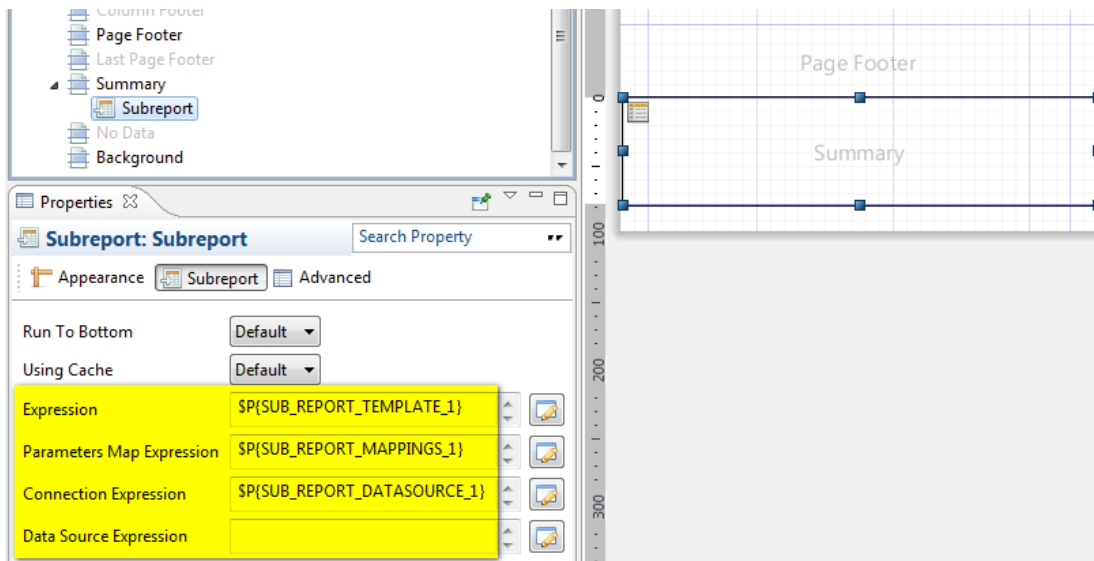



3. If a map of parameters is to be used in the subreport, repeat [step 1](#) to create another parameter. This will allow a map of parameters to be passed to the subreport. This parameter will be the **Map parameter**. Enter a name in the **Name** field (e.g. SUB_REPORT_MAPPINGS_1). Note that this name is used as the **Map parameter name** in the subreport tab of the **Report Generator** module. The parameter Class must be set as "java.util.Map".



Repeat [step 1](#) to [step 3](#) for each subreport. Once the parameters have been defined, they must be mapped onto the appropriate subreport element in the main report.

1. Select the Subreport element in the design view of the main report. In the properties view, select the **Subreport** tab.



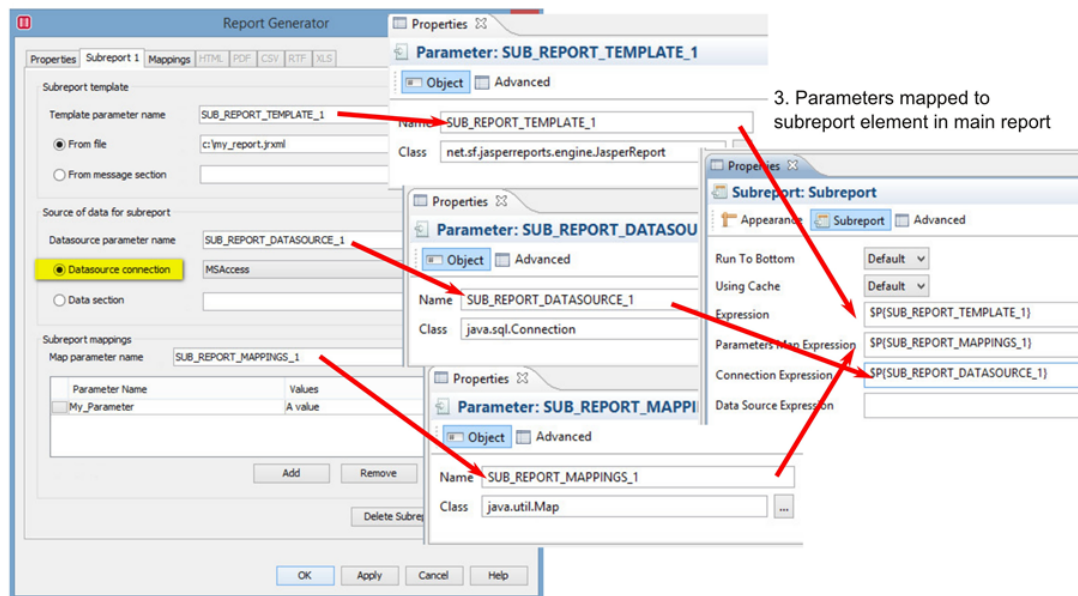
2. Enter the **Template parameter** name for the **Expression**. The name can be typed in directly using the parameters syntax `$P{}` or the parameter can be selected using the **Expressions Editor** button  (shown on the right hand side). So, if you named the **Template parameter** as `SUB_REPORT_TEMPLATE_1` then you would enter `$P{SUB_REPORT_TEMPLATE_1}`.
3. Depending on the source of data, enter either of the following:
 - The **Datasource parameter name** for the **Connection Expression** if the source of data is a *data source*. E.g. `$P{SUB_REPORT_DATASOURCE_1}`.
 - The **Datasource parameter name** for the **Data Source Expression** if the source of data is a *data section*. E.g. `$P{SUB_REPORT_DATASECTION_1}`.
4. Enter the **Map parameter name** for the **Parameters Map Expression** if a parameter map is being passed to the Subreport. E.g. `$P{SUB_REPORT_MAPPINGS_1}`.

Repeat these steps for each subreport included in the main report. The main report will then be configured to include the subreports in its output. These parameters must now be configured in the [Subreport](#) tab of the **Report Generator** module.

The following diagram summarises where the parameters need to be defined in EMF and JasperStudio when using a data source.

1. Parameters defined in EMF

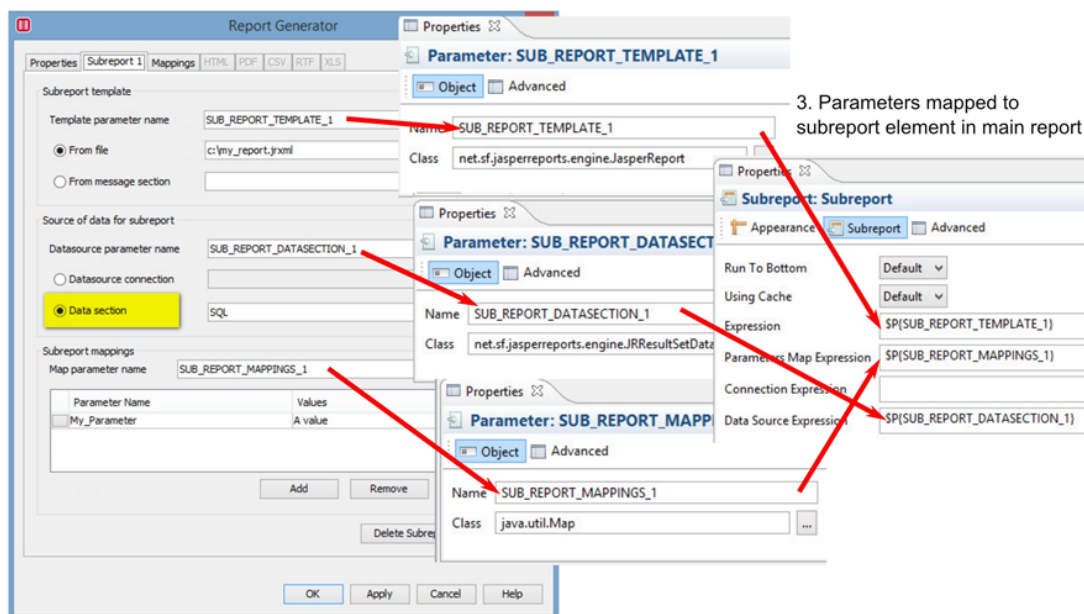
2. Parameters defined to main report



The following diagram summarises where the parameters need to be defined in EMF and JasperStudio when using a data section.

1. Parameters defined in EMF

2. Parameters defined to main report

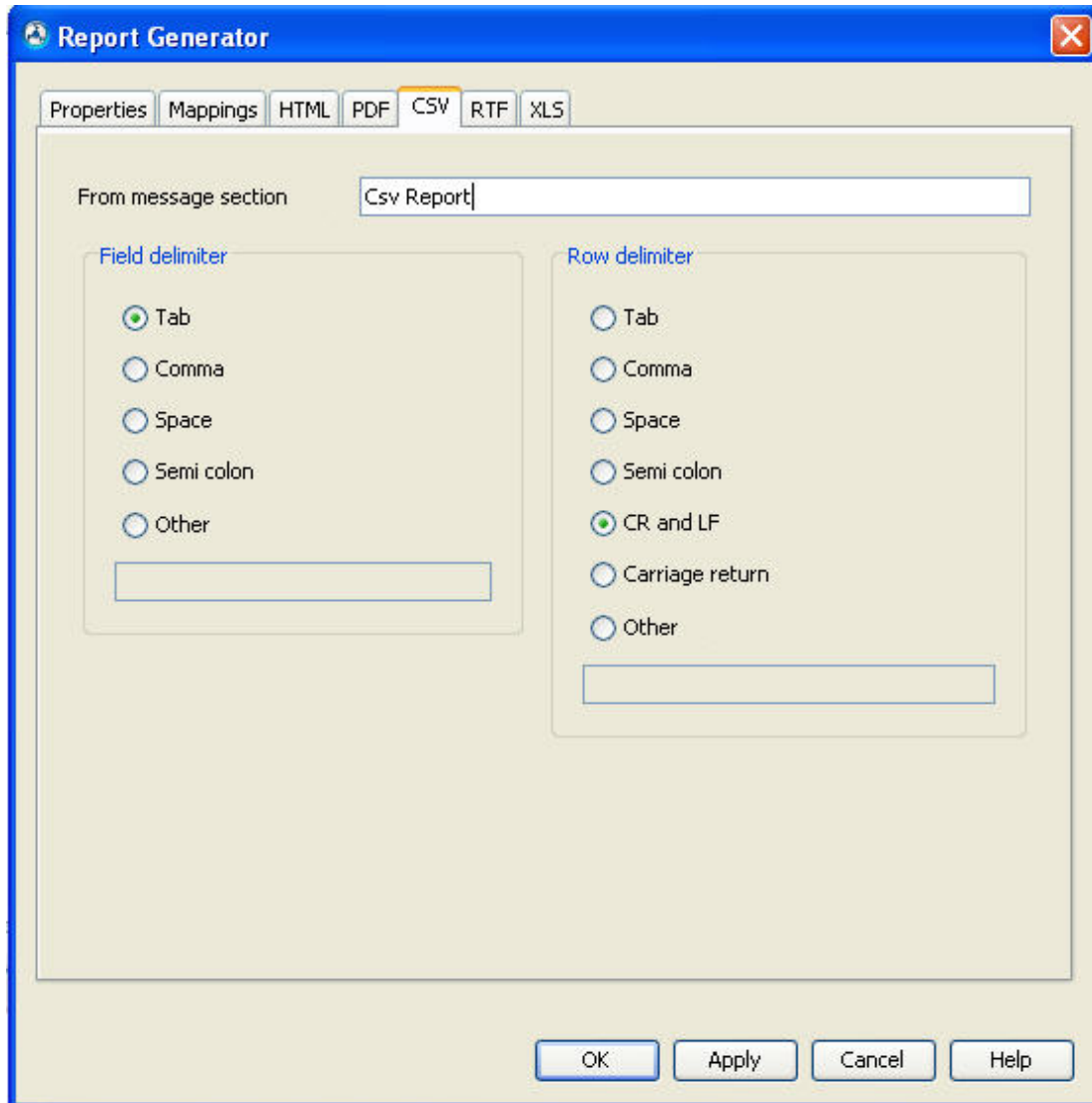


[Report Generator Module](#)

[Report Generator - Subreports](#)

Report Generator - CSV

The **CSV** tab configures items specific to an CSV (Delimiter Separated Variables) generated report.



1. In **Message Section** enter the name of the message section that will store the generated report.
2. Select the required **Field delimiter** character, which will be used to separate the fields in the output. You can either select one of the suggested default delimiters, or select Other and enter the required character in the supplied field.
3. Select the required **Row delimiter** character, which will be used to separate the rows in the output. You can either select one of the suggested default delimiters, or select Other and enter the required character in the supplied field.

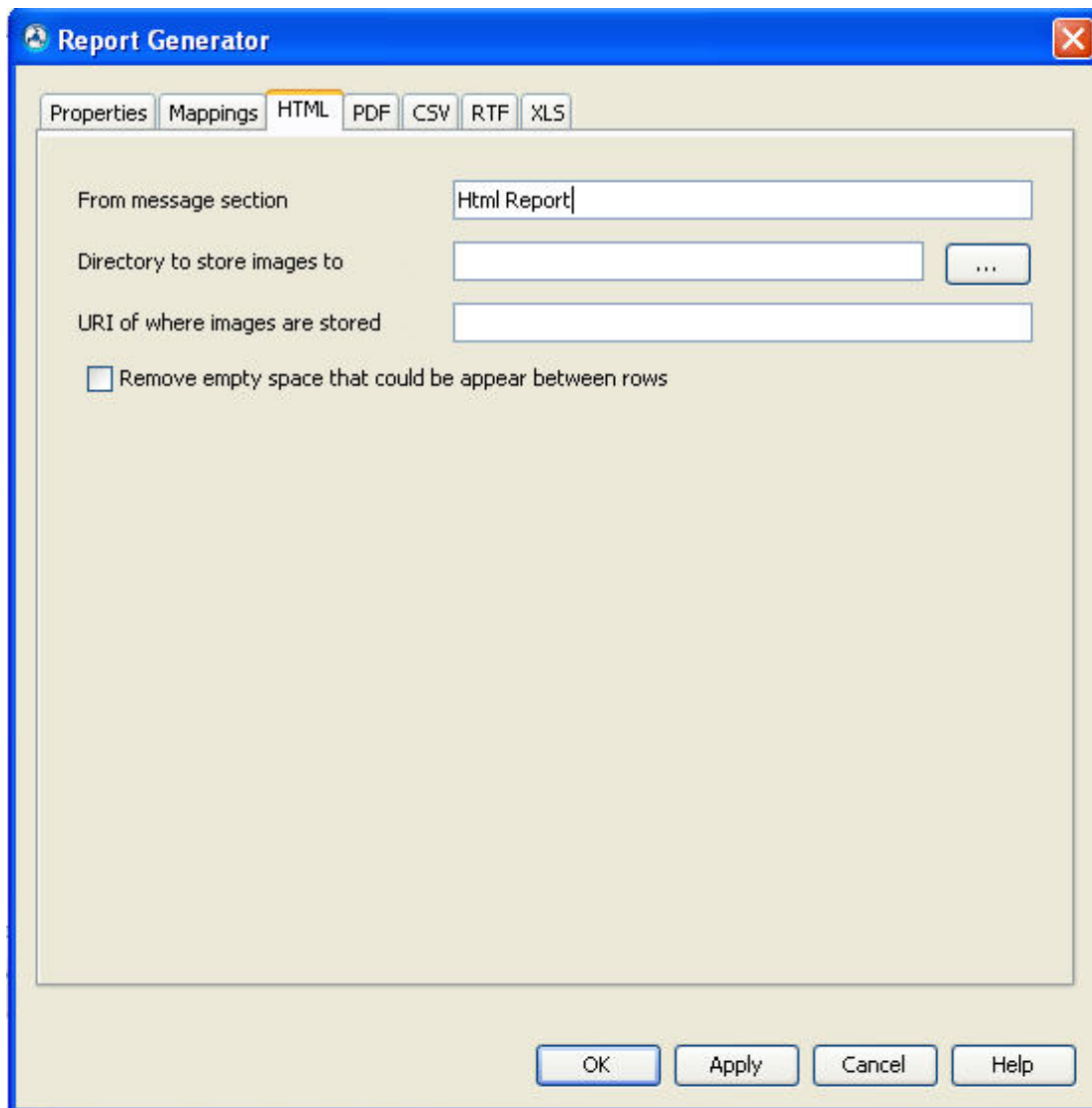
[Properties tab](#)

[Mappings tab](#)

[HTML tab](#)[PDF tab](#)[CSV tab](#)[RTF tab](#)[XLS tab](#)[Using EMF data in Jaspersoft Studio](#)[Formatting Modules](#)

Report Generator - HTML

The **HTML** tab configures items specific to an HTML generated report.



The screenshot shows the 'Report Generator' dialog box with the 'HTML' tab selected. The dialog has a blue title bar and a close button in the top right corner. Below the title bar are tabs for 'Properties', 'Mappings', 'HTML', 'PDF', 'CSV', 'RTF', and 'XLS'. The 'HTML' tab is active, showing the following fields and options:

- 'From message section' with a text box containing 'Html Report'.
- 'Directory to store images to' with a text box and a browse button ('...').
- 'URI of where images are stored' with a text box.
- An unchecked checkbox labeled 'Remove empty space that could be appear between rows'.

At the bottom of the dialog are four buttons: 'OK', 'Apply', 'Cancel', and 'Help'.

1. In **Message Section** enter the name of the message section that will store the generated report.
2. In **Directory to store images to** enter the name of directory where images contained in the report will be stored to. This directory must be accessible from the server. This may be the directory that your webserver stores its images in.

Note: Even if you don't think your report contains any images, blank images are often created so that the report items are spaced correctly.

3. In **URI of where images are stored** enter the URI that will be written out in the HTML report. If the images are to be served up from a webserver, then this will be the URI of the directory on the webserver, e.g.

`http://www.mycompany.com/images/`

If however, the report is to be sent out via an email and you wish to embed the images in the email, then the URI should be a local one pointing to where the files are to be written, e.g.

`file://c:/images/`

4. When html emails are sent via EMF, local images are automatically embedded in the message.
5. The **Remove empty spaces that could appear between rows** checkbox informs the report generator to remove any empty rows. This may lead to better final presentation of the report.

[Properties tab](#)

[Mappings tab](#)

[HTML tab](#)

[PDF tab](#)

[CSV tab](#)

[RTF tab](#)

[XLS tab](#)

[Using EMF data in Jaspersoft Studio](#)

[General Modules](#)

Report Generator - PDF

The **PDF** tab configures items specific to an PDF (Adobe Acrobat) generated report.

Currently, all documents are created with printing and copying of content allowed. If this is an issue please contact EMF support.

The screenshot shows the 'Report Generator' dialog box with the 'PDF' tab selected. The 'Message section' field contains 'Pdf Report'. There are checkboxes for 'Encrypt document' and 'Compress document', both of which are currently unchecked. A 'Password' text field is present. Below these is a 'Meta data' section with four text input fields for 'Title', 'Author', 'Subject', and 'Keywords'. At the bottom of the dialog are 'OK', 'Apply', 'Cancel', and 'Help' buttons.

1. In **Message Section** enter the name of the message section that will store the generated report.

Note: PDF message sections are stored as binary data, so if sent by email they must be sent as attachments and not as the main email message.

2. Select **Encrypt document** and enter a **password** if you want protect and encrypt the document. The end user will have to correctly enter the password everytime they open the document.
3. Select **Compress document** to reduce the size of the final document after generation.
4. The **Meta Data** section allows you to enter meta data (Author, Title, Subject and Keywords) about the document that will be created. This is used by search engines and humans to get a better understanding about the content of the document. Dynamic functions can be used to provide the metadata. Right clicking on any of the Author, Title, Subject or Keywords entry fields gives access to the dynamic functions menus.

Note: If the document is encrypted then the metadata is not visible to search engines. The password needs to be entered and the document loaded to view the meta data.

[Properties tab](#)

[Mappings tab](#)

[HTML tab](#)

[PDF tab](#)

[CSV tab](#)

[RTF tab](#)

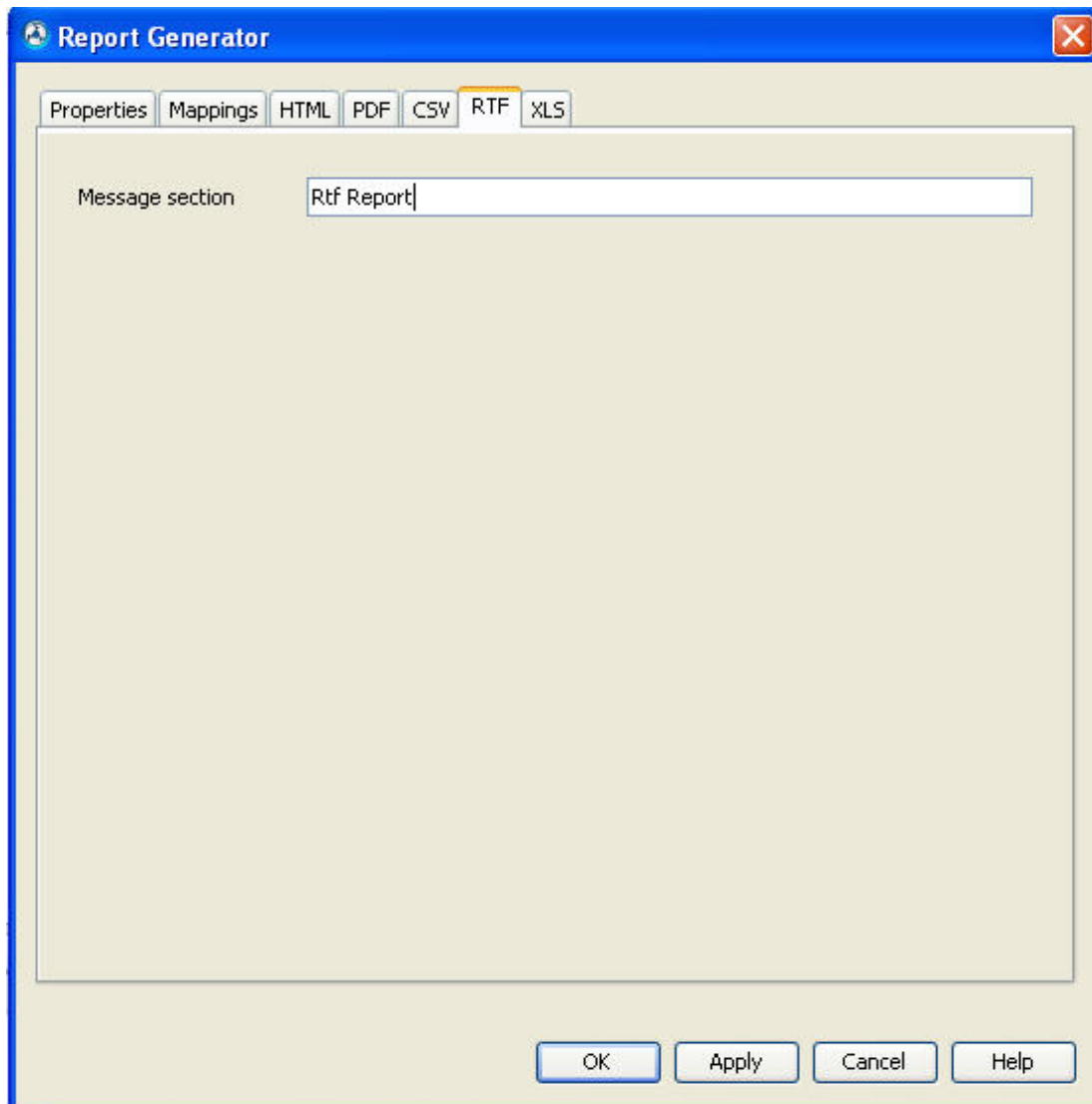
[XLS tab](#)

[Using EMF data in Jaspersoft Studio](#)

[Formatting Modules](#)

Report Generator - RTF

The **RTF** tab configures items specific to an RTF (rich text format) generated report.



1. In **Message Section** enter the name of the message section that will store the generated report.

Note: RTF message sections are stored as binary data, so if sent by email they must be sent as attachments and not as the main email message.

[Properties tab](#)

[Mappings tab](#)

[HTML tab](#)

[PDF tab](#)

[CSV tab](#)

[RTF tab](#)

[XLS tab](#)

[Using EMF data in Jaspersoft Studio](#)

Formatting Modules

Report Generator - XLS

The **XLS** tab configures items specific to an XLS (Excel) generated report.

The screenshot shows the 'Report Generator' dialog box with the 'XLS' tab selected. The 'Message section' field contains 'Xls Report'. There are three checkboxes: 'Write each page to a different sheet' (unchecked), 'Remove empty space that could be appear between rows' (unchecked), and 'Detect cell type and try to create cell types dependent on content of data' (unchecked). Below the first checkbox is a list box for 'Sheet name' which is currently empty. To the right of the list box are three buttons: 'Add', 'Remove', and 'Remove All'. At the bottom of the dialog are four buttons: 'OK', 'Apply', 'Cancel', and 'Help'.

1. In **Message Section** enter the name of the message section that will store the generated report.

Note: XLS message sections are stored as binary data, so if sent by email they must be sent as attachments and not as the main email message.

2. Select **Write each page to a different sheet** if you wish to have each page appear as a different sheet in the workbook in Excel. You can also name the sheets, by adding **Sheet names**. If no sheet names are present and each page is written to a separate

sheet then the pages will be called *Page 1*, *Page 2* etc. If the report is being written to a single sheet, then the sheet is called the same as the **Message Section** name.

3. Selecting **Remove empty spaces that could appear between rows** improves the presentation of the final document in some instances.
4. The **Detect cell type and try to create cell types dependent on content of data** will attempt to set the Excel field types to be the same as that specified in the report template.

[Properties tab](#)

[Mappings tab](#)

[HTML tab](#)

[PDF tab](#)

[CSV tab](#)

[RTF tab](#)

[XLS tab](#)

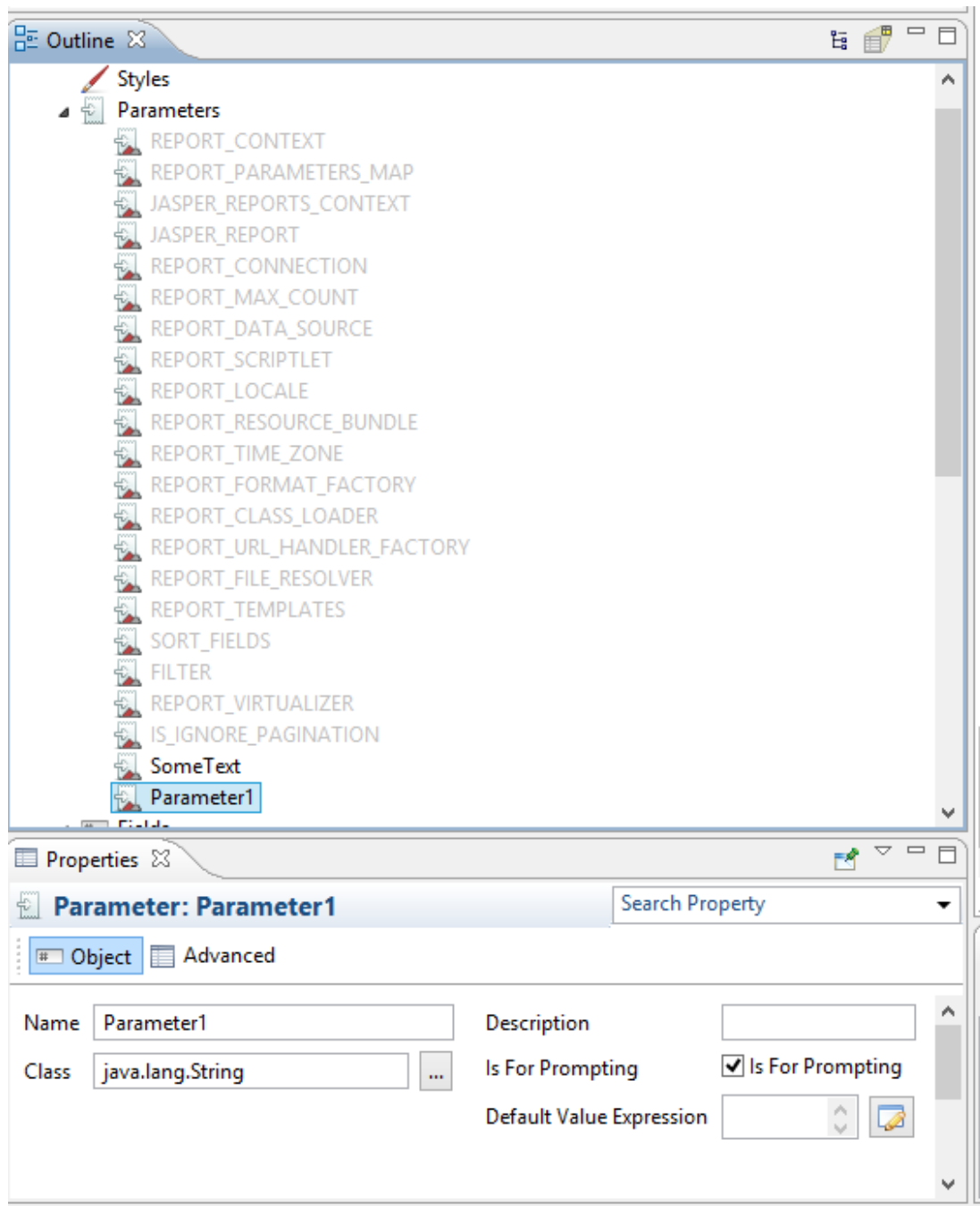
[Using EMF data in Jaspersoft Studio](#)

[Formatting Modules](#)

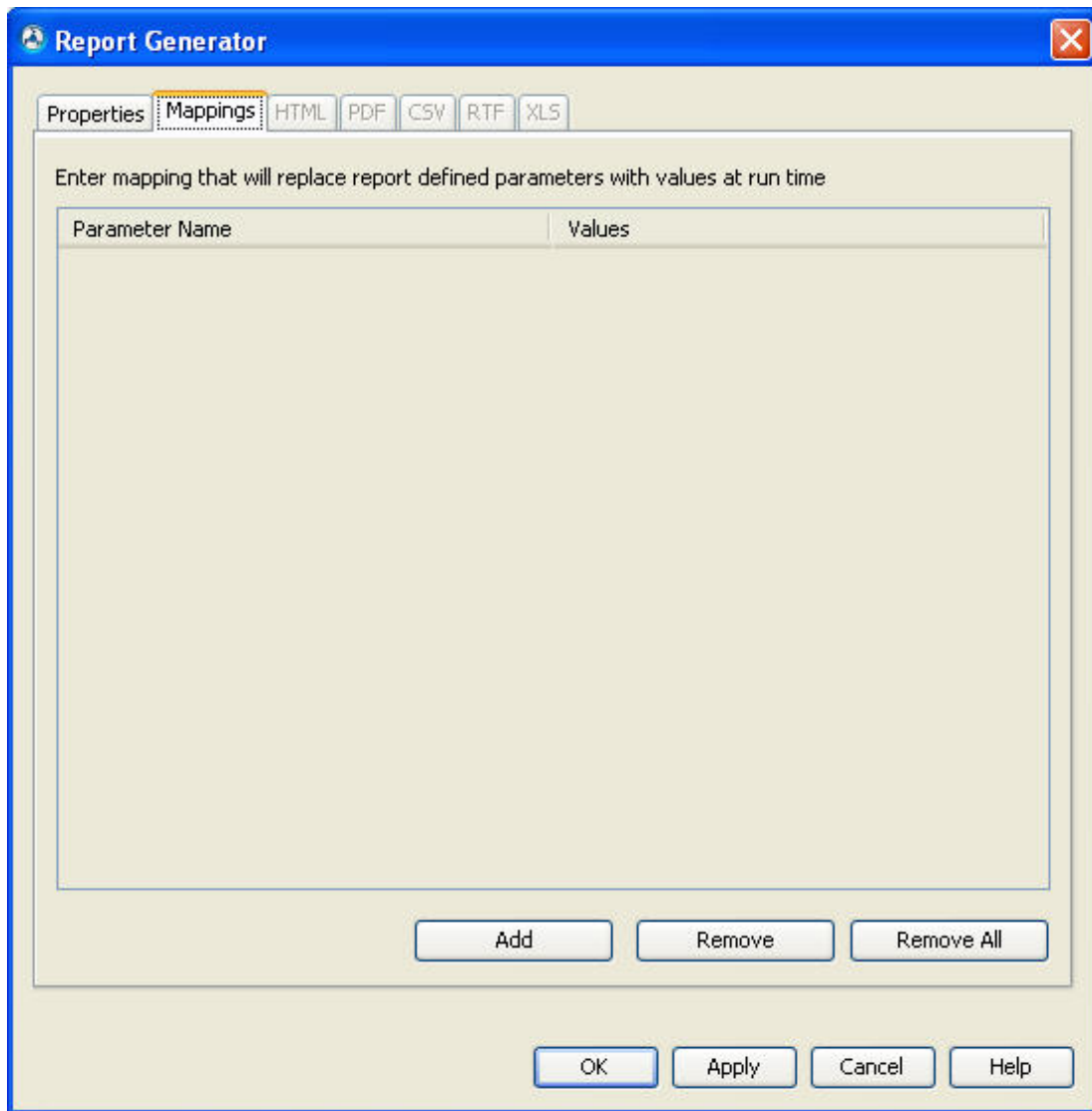
Report Generator - Mappings

Mappings allow report template defined parameters to be replaced with a value at report generation time. For instance, the title of the report could be dynamically set depending upon the company it was being generated for.

To define a new parameter to use in a report template in Jaspersoft Studio, select **Parameters** in the **Outline** window and right-click and select "Create Parameter" to create one. Enter a name, class and a default value. The default value is used when no value mapping exists for the parameter. This parameter can now be dragged on to a report template in the desired location.



To set the value of this parameter in EMF, on the **mapping** tab, press the **Add** button and enter the **Parameter name** and the **Value** it should have. The value may be a dynamic function which will get resolved at runtime.



[Properties tab](#)

[HTML tab](#)

[PDF tab](#)

[CSV tab](#)

[RTF tab](#)

[XLS tab](#)

[Using EMF data in Jaspersoft Studio](#)

[Formatting Modules](#)

Excel Writer Module

The Excel Writer module allows you to output data, primarily from data sections, to create an Excel workbook. The module has two modes of operating, either creating a new workbook, where basic output style information can also be defined, or using an existing Excel file as a template and inserting the data as specified locations. By using templates advanced charts and calculations can be set-up and then updated with the relevant data at the required time.

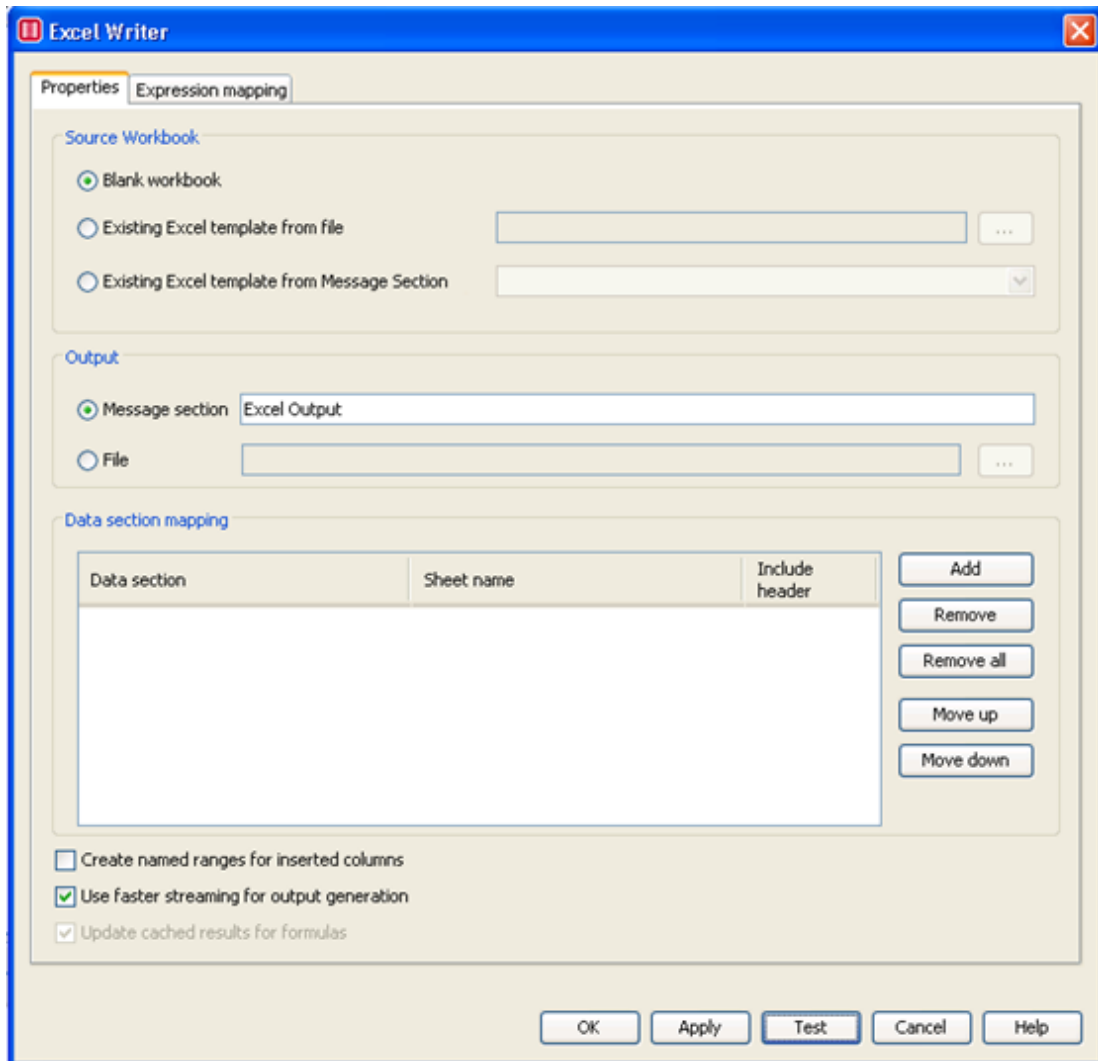
Note: Only XLSX/XLSM format files (Excel 2007 onwards) are supported.

To use the Excel Writer module:

1. Drag the **Excel Writer** icon to the EMF Process Design graph at the appropriate place.



2. Double-click the **Excel Writer** icon to display the Excel Writer Properties screen:

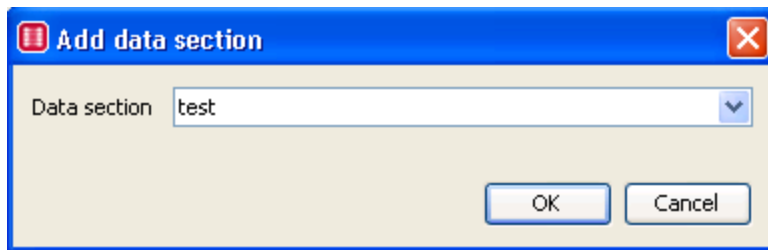


3. In the **Properties** tab, in the **Source Workbook** section, select the source Excel workbook that the output will be based on:
- **Blank workbook** - Selecting this option will create a new Excel workbook containing a sheet of data for every data section selected for output. The output formatting / style information for each column in a data section can be defined.
 - **Existing Excel template from file** - Selecting this option allows you to select an existing Excel workbook that will be used as the basis for the output. Data can then be inserted at specified cells / named ranges within that template. The template can have formula, charts and formatting options already defined in.
- Note:** At runtime the server must be able to access the Excel template.
- **Existing Excel template from Message section** - Similar to the above option, this option allows a template to be selected, except this time the

template is contained within a message section, rather than a file being read in. This has the advantage of allowing the complete process to be exported and re-imported without the need for transferring a separate "template" file. The message section name to use can be defined using dynamic functions.

Note: Excel templates are binary data, so if the message section is to contain the excel template directly, it is recommended to Base64 encode it (using a 3rd party tool or EMF), copy the Base64 encoded text into a message section and then use the BASE64DECODE dynamic function in another message section to return the correct binary contents.

4. In the **Output** section, select the output type. This can either be a **Message section** or a **File**.
 - **Message section** - Select to write data to a message section. The created message section will contain binary data. For more information, on working with Binary Data see [Binary Data](#). This message section then could, for example, be attached to an email message so as to send an Excel file as an attachment to an email.
 - **File** - Select to write data to a file. This will create an Excel formatted file.
5. In the **Data section mapping** section, select **Add** to select the data sections that you want to output. This will open the **Add data section** window where an existing data section can be selected or the name of a data section can be manually entered.



Click **OK** to insert the data section name into the **Data sections mapping** grid.

The columns available in the **Data sections mapping** grid are dependent upon the option chosen in step 3 when selecting the source workbook:

- For a **blank workbook** when a data section is added to the grid a new [Output style](#) tab is also created for that data section. At runtime a sheet will be created in the workbook for each data section added.

The order of data sections in the grid is the order in which the sheets are created in the workbook.

The following additional options are available for each data section:

- **Sheet name** - This is the name that the sheet will be called at runtime. If it is left blank then the sheet will be named after the data section. You can also use Dynamic functions in the **Sheet name** column.

Note: If the sheet name does not meet Excel sheet naming conventions then it will automatically be renamed at runtime, and a warning logged.

- **Include Header** - Select **Include Header** if you want the first line of the output to be the columns headers.
- For a workbook based on an existing template each data section added will be output into the template at the location specified. Note that the output style tabs are not created when selecting this option as the styles must be defined in the template.

The following additional options are available for each data section:

- **Selected columns** - Displays the names of the selected columns from the data section that will be output in the Excel workbook. Pressing the "Column selection" button allows these to be changed. By default, all columns are output.
- **Cell address/Named range** - Enter the target cell. This can either be a named range that must exist in the template or a cell address. The cell address must start with "=". Dynamic functions can be used. To specify a cell on a particular sheet, use the following format:

=<SheetName>!<Col><Row>

e.g.

=MySheet!C4

If using named ranges, and the named range does not exist at runtime in the template, the process will fail with an error.

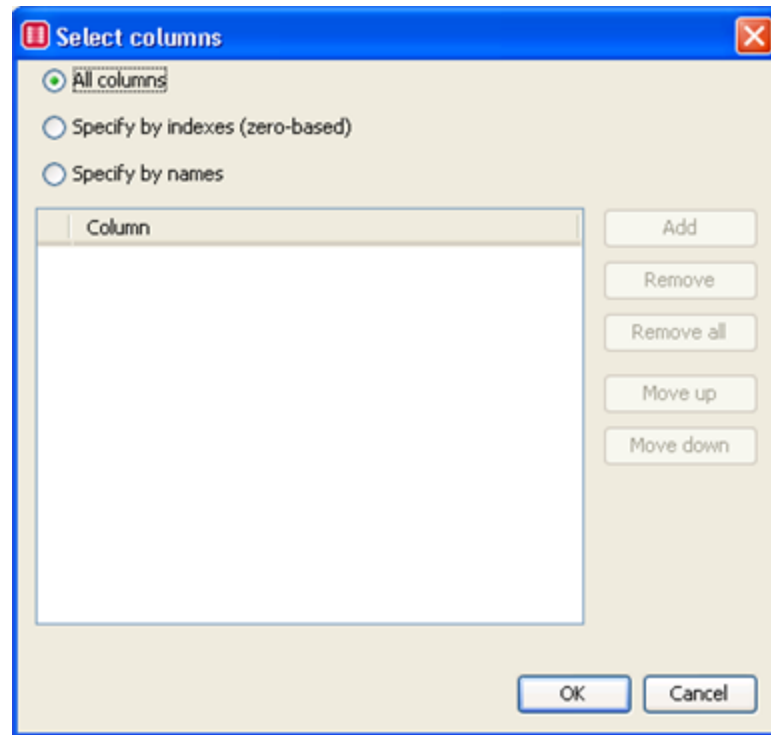
If using a cell address, and no sheet name is specified then a new sheet will be created named after the data section.

If using a cell address, and the sheet name specified does not exist in the template, then a sheet will be created with the specified name.

Note: If the sheet name does not meet Excel sheet naming conventions then it will automatically be renamed at runtime, and a warning logged.

- **Include Header** – Select **Include Header** if you want the first line of the output to be the columns headers.
- **Auto size** – Select to adjust the column width according to the maximum size of the cell contents in the column.
- **Columns selection** – Click the **Select** button to open the **Select Columns** window which allows you to select single or

multiple columns from a data section that will be output. The order the columns are output is also controlled by this list.



By default, the **All columns** option is selected. You can select the columns from the drop-down list or enter it manually. You can also use Dynamic Functions in the column.

You can use the following options to add, delete, or change the order of the columns:

- **Add**
- **Remove**
- **Remove all**
- **Move up**
- **Move down**

6. Select the following options as required:

- **Create named ranges for the inserted columns** - Selecting this option will cause Excel "named ranges" to be created for each data column added in the Excel workbook. The format of the name is <Datasection>_<Column name>. This can be useful if other cells contain calculations that need to reference the inserted column, and can be used in formula entered on the expression mapping tab (for example, to sum a column).
- **Use faster streaming for output generation** - Select to improve performance in the time taken to produce the output, and reduce memory

requirements.

Note: There are limitations as to when this option can be used. As the data is output as it is generated, certain options, such as “auto resize” will have limited effect. So if performance is an issue, test this option to see if it generates the desired output before deploying in a production environment.

- **Update cached results for formulas** - This option will cause any formula in the workbook generated from a template to be recalculated once the new data has been inserted.
7. Select the [Expression Mapping](#) tab to define any single values / formula that should be written to single cells in any workbook sheet.
 8. Click **Test** to create an output file or write the contents to a message section. For message section output, the contents cannot be previewed (as it is binary data), click **Save as sample** to save the sample data and click **OK**.

[Example Process](#)

[Expression Mapping tab](#)

[Output Style tab](#)

[Formatting Modules](#)

Excel Writer – Expression Mapping

Expression Mapping allows you to write values to single cells in the workbook. It also allows formula to be created in the workbook.

Use the following options to add, delete, or change the order of the columns:

- **Add**
- **Remove**
- **Remove all**
- **Move up**
- **Move down**

The expression mapping table contains the following columns:

- **Cell address/Named range** - Enter the target cell. This can either be a named range or a cell address. The cell address must start with “=”. Dynamic functions can be used.

To specify a cell on a particular sheet, use the following format:

=<SheetName>!<Col><Row>

e.g.

=MySheet!C4

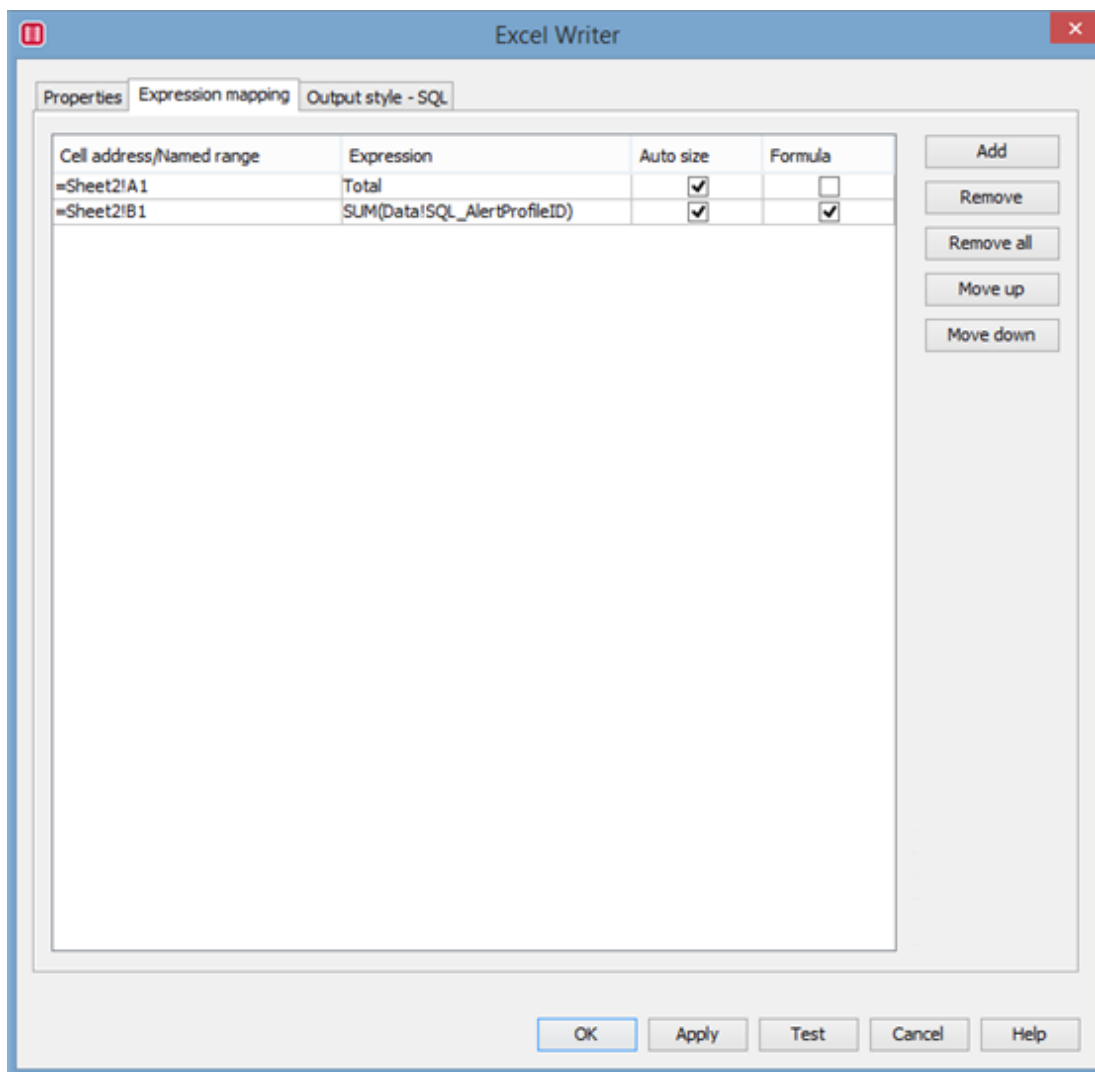
If using named ranges, and the named range does not exist at runtime in the template, the process will fail with an error.

If using a cell address, and no sheet name is specified then the expression is written to the first sheet in the workbook. If no sheet exists (because no data sections are being written) – then a new sheet is created.

If using a cell address, and the sheet name specified does not exist in the template, then a sheet will be created with the specified name.

Note: If the sheet name does not meet Excel sheet naming conventions then it will automatically be renamed at runtime, and a warning logged.

- **Expression** - Enter the text whose contents will be inserted into the workbook. Dynamic functions can be used.
- **Auto size** - Select to adjust the column width according to the maximum size of the cell contents in the column.
- **Formula** - If selected, then the expression entered is inserted as an Excel formula and its results will be calculated. For instance to sum a column enter an expression SUM (<Named_range_of_column>) and select the **Formula** check box.



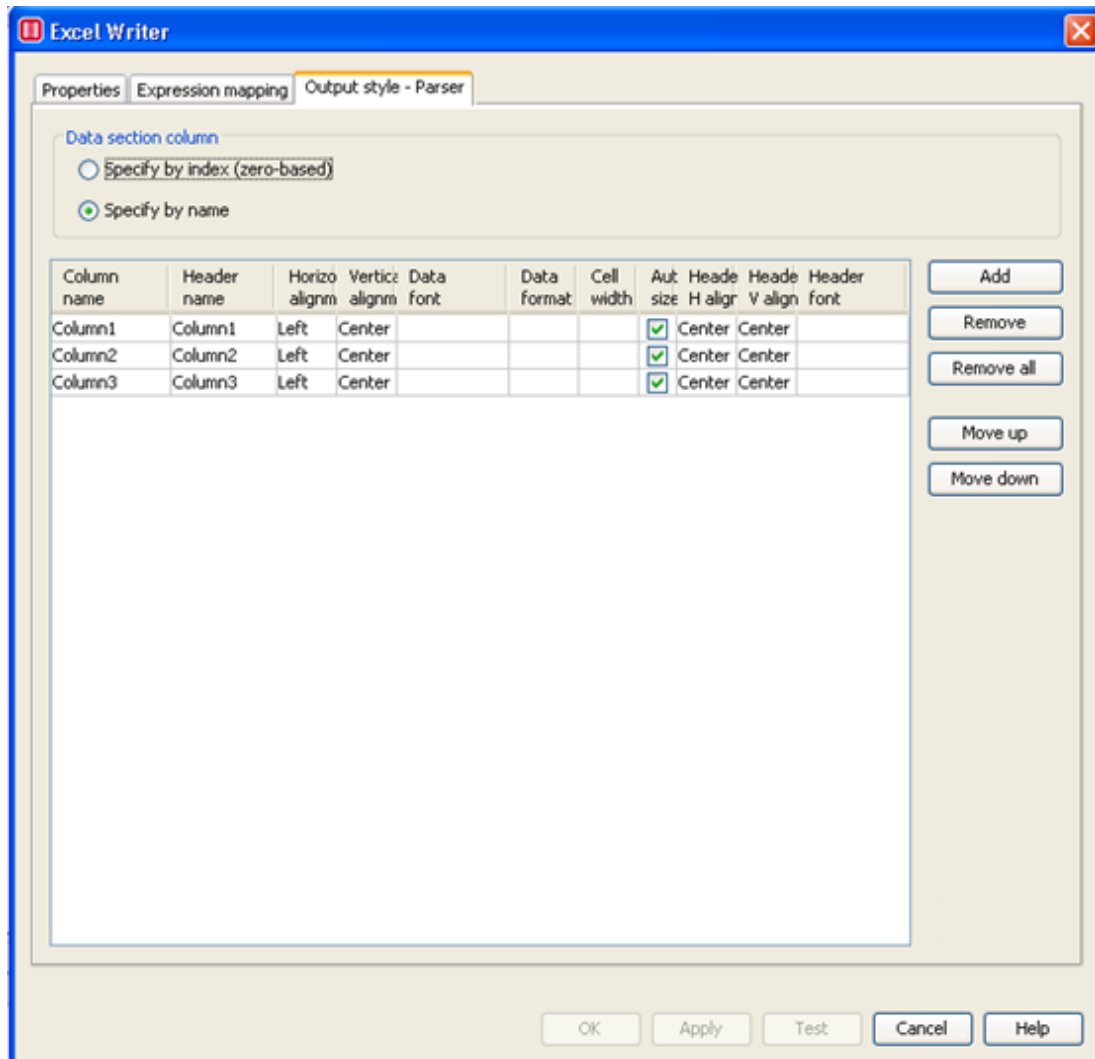
[Properties tab](#)

[Output Style tab](#)

[Formatting Modules](#)

Excel Writer – Output style

Output style tab allows you to define the format of a data section when using a blank workbook. Only the data section columns listed in this tab will be output for a selected data section. The order the columns are output is controlled by the order they are listed in the table.

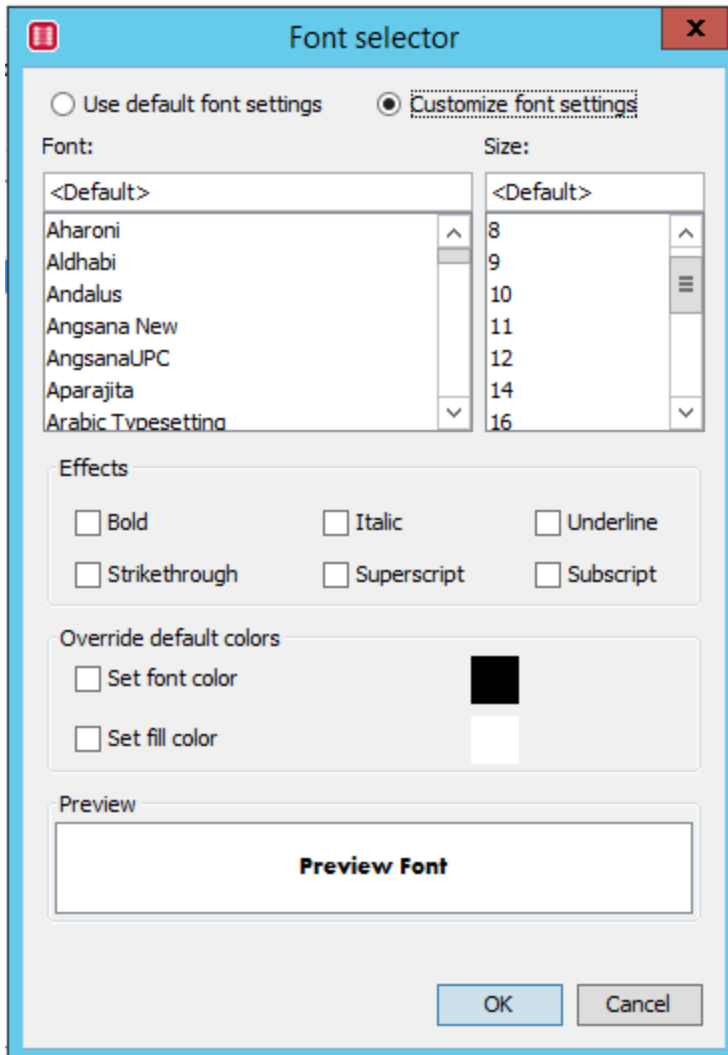


The dialog box titled "Excel Writer" has three tabs: "Properties", "Expression mapping", and "Output style - Parser". The "Output style - Parser" tab is active. It contains a section for "Data section column" with two radio buttons: "Specify by index (zero-based)" and "Specify by name". Below this is a table with columns: "Column name", "Header name", "Horizo align", "Vertic align", "Data font", "Data format", "Cell width", "Aut size", "Heade H align", "Heade V align", and "Header font". The table contains three rows: "Column1", "Column2", and "Column3". To the right of the table are buttons: "Add", "Remove", "Remove all", "Move up", and "Move down". At the bottom of the dialog are buttons: "OK", "Apply", "Test", "Cancel", and "Help".

Column name	Header name	Horizo align	Vertic align	Data font	Data format	Cell width	Aut size	Heade H align	Heade V align	Header font
Column1	Column1	Left	Center				<input checked="" type="checkbox"/>	Center	Center	
Column2	Column2	Left	Center				<input checked="" type="checkbox"/>	Center	Center	
Column3	Column3	Left	Center				<input checked="" type="checkbox"/>	Center	Center	

- **Column name** - Select the column name or index that must be output. If you specify by name the column will be a drop-down list with the column names. Columns are output in the order that they are defined in this table.
- **Header name** - Enter the header text that will be output if column headers have been selected for output. Dynamic functions can be entered.

- **Horizontal alignment** - Select the horizontal alignment of the data that is output: General, Left, Center, Right, Fill, Justify, Center-selection.
- **Vertical alignment** - Select the vertical alignment of the data that is output: General, Left, Center, Right, Fill, Justify, Center-selection.
- **Data font** - Double-click on the **Data font** column to open the **Font selector** window. Select the required font type, size and effects and click **OK**.



- **Data Format** - Enter the data format. For example, "\$##, ##0.00", "dd/mm/yy". These formats are the same formats that you would normally use to define a cells format within an Excel workbook.

Warning: If the data being output has been queried from a Microsoft SQL Server database and it contains date and/or time fields then use the MS JDBC driver to query the data rather than the jTDS JDBC driver due to known issues with the driver. If the field type is a datetime field then either driver is fine. Failure to do this will cause date/time formatting setting to be ignored when

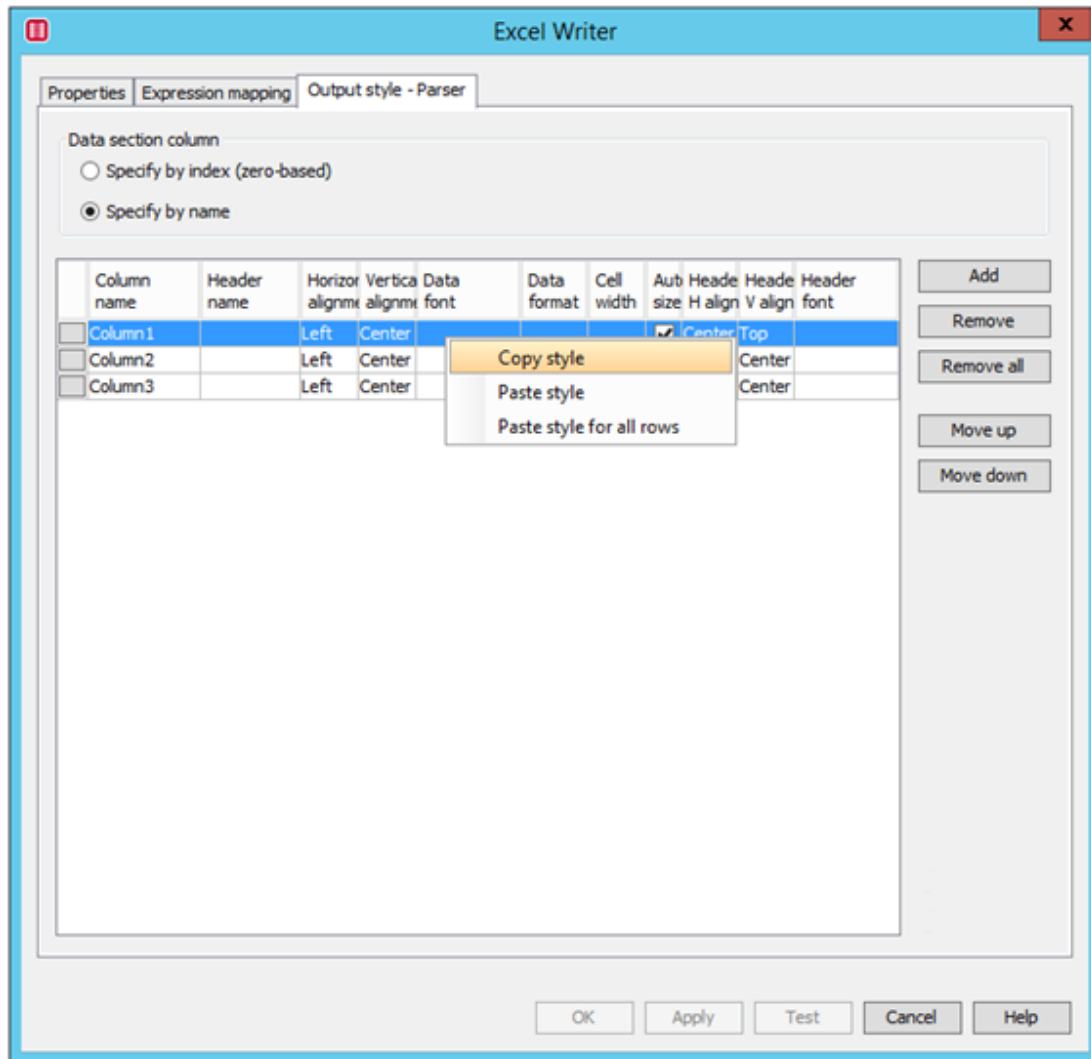
outputting those fields.

- **Cell width** - Enter the width of the cell. If auto-size is selected, then this size will be ignored.
- **Auto-size** - Select to adjust the column width according to the maximum size of the cell contents in the column.
- **Header H Alignment** - Select the horizontal alignment of the header column: General, Left, Center, Right, Fill, Justify, Center-selection.
- **Header V Alignment** - Select the vertical alignment of the header column: General, Left, Center, Right, Fill, Justify, Center-selection.
- **Header font** - Double-click on the **Header font** column to open the **Font selector** window. This is the same as the data font selector, except you are defining the font for the header row (if the header row has been selected to be output).

Output styles can be easily copied from one output column to others by selecting the appropriate row by using the right click copy and paste options.

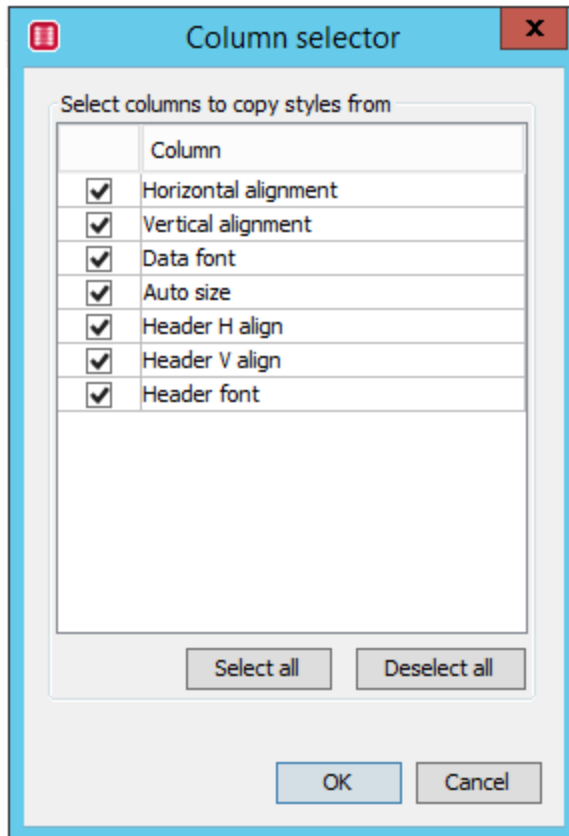
To copy the output styles:

1. Right-click the row from which you want to copy the style and select **Copy style**.



You can paste the styles to a single row or multiple rows.

2. Select the styles to copy and click **OK**.



[Properties tab](#)

[Expression Mapping tab](#)

[Formatting Modules](#)

Excel Example Processes

The following example processes are included in the help. Click the link for each of them and save the process shown locally. They can then be imported into your local repository and run. Note that each process needs some output file paths adjusting before they will run successfully (unless you have rights to create files in the root directory of your C drive).

[Example 1](#) - This demonstrates how to write the results of a database query to a new Excel workbook. It also creates a second sheet containing some formula to sum one of the columns output, referencing that column as a named range.

[Example 2](#) - Similar to example 1, except the summing of the data columns is placed at the end of the data, rather than on another sheet. Dynamic functions are used to calculate the data to sum, and the cell location to place the "Total" value in.

[Example 3](#) - Outputs two Excel workbooks, both based on an Excel template that is embedded in the process in a message section. The workbooks contain a chart. It demonstrates how to embed a template and use it within a process. The template itself uses Excel named ranges defined using the OFFSET and COUNTA functions (see ranges called RangeX and RangeY) so

that dynamic charts can be created that will automatically pick up a variable amount of data output and plot it on a chart (i.e. the number of rows output is different each time).

EMF Process Output Modules

The EMF Process **Output modules** are used in the EMF Process builder to send the information that has collected and collated by an EMF Process to subscribers of the EMF Process. There are several different output modules (see list below) and each behaves in a different way and produces a different result.

To add an output module to an EMF Process:

Drag the required module onto the [EMF Process Design Graph](#) and then double-click it to access the Properties pages. Each output module has a different set of definitions and options; click the **Help** button on the Properties tabs for further information on the specifics of that output module. In addition, all output modules share a common [Advanced Properties tab](#).

Note: Each Output module requires a corresponding **Service** to be available in order for EMF to send information. You must configure an appropriate service, if you have not already done so, in the [Services](#) section of the EMF tree view. (The Debug output module does not require a Service as it cannot send information to a recipient.)

For more information on an output module, choose from the list below:

- [SMS Output](#)
- [Email Output](#)
- [File Output](#)
- [FTP Output](#)
- [Queue Output](#)
- [Run Executables](#)
- [OPC DA Write](#)

Note: You can also use the [HTTP module](#) to send output from an EMF Process to a specific Internet address (URI), although the HTTP module is not classed as an output module because it cannot use recipients.

SMS Out Module

You can use an **SMS Output** module to send information to a mobile telephone via an SMS (Short Message Service) message. You can send SMS messages using the **SMS Service**.

The characters shown below cannot be sent to mobile/cell phones unless those phones support Unicode. The **SMS Service** will replace any of the following characters with a space.

Character	Description
[left square bracket

\	back slash
]	right square bracket
^	caret
`	grave accent
{	left curly brace
	pipe or vertical bar
}	right curly brace
~	tilde

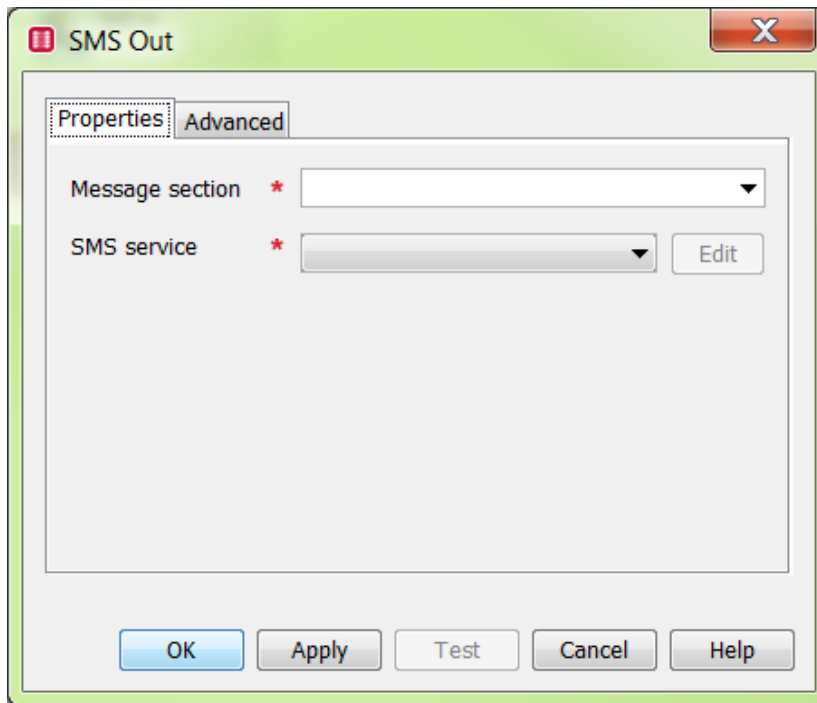
To use an SMS Out module:

1. Ensure that you have set up one or more SMS services (in the **Services** section of the EMF Processes tree view).
2. Drag an **SMS Out** module into your EMF Process at an appropriate place.



Important: Before you can use SMS output, you must ensure that your service provider is listed and that you have created an appropriate SMS service.

3. Open the SMS Out module to display its properties. By default, you have to double-click on the icon to open, but you can configure mouse-click actions on the **Options** window. From the **Tools** menu, select **Options** -> **EMF** -> **Process Builder Behaviour** tab. The actual recipients of the message are gathered from the various recipient modules that precede it in the EMF process flow.



- **Message section** - Select the message section that contains the message to be sent.

Note: If the message section is too long for a single text message, the information will be split after every 160 characters into separate text messages.

- **SMS service** - select the service that should be used for sending the message. You can click **Edit** to modify the properties of **SMS service**.
4. Select the **Advanced** tab to configure Message state logging (required for [Escalation](#)) and [Escalation](#) filtering.
 5. Click **Test** and enter the country code and mobile number of the recipient to send a sample mail.

[Go to start of Output Modules](#)

Email Out Module

You can use an **Email Out** module to send EMF Process information to recipients via an email message.

Important: Before you can use email output you must configure an [SMTP service](#).

Important: Before you can send an encrypted email, you need to install the **Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files** for your Java Runtime Environment (JRE). These cannot be shipped with EMF due to export regulations. The files for Java 8 can be downloaded from

<http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html>.

The included README.txt in the download explains how to install the files, but it should be just copying the included files to the Aptean\EMF\jreX.X.X.X\64\lib\security and the Aptean\EMF\jreX.X.X.X\86\lib\security directories, having backed up the existing files first.

Email Output supports the following features:

- **SMTP**
- **To:**, **Cc:**, and **Bcc:** (where the protocol supports them)
- Multiple or single attachments, and **UUencode** or **Base64** encoding
- **HTML** content for all email services
- **Encryption** of messages and attachments using either S/MIME or OpenPGP encryption

Note: If an HTML-formatted message section contains a reference to a local file that can be found (e.g. **C:\test.gif**), the image is encoded and included with the message and the tag is modified to reflect this. However:

- if a message contains an **http://** reference (e.g. **<IMG**

src="http://194.223.2.21/test.gif">) or a **file** reference (e.g. **<IMG**

src="file:///c:\test.gif">), the file is **not** included and the reference remains unchanged.

- if EMF cannot find the file when the message is sent, the original reference to the file will remain unchanged.

To use an Email Out module:

1. Ensure that you have set up one or more [SMTP services](#) (in the **Services** section of the EMF tree view).
2. Drag an **Email Out** module into your EMF Process at an appropriate place.



3. Open the Email Out module to display its properties. By default, you have to double-click on the icon to open but you can configure mouse-click actions on the Options window. From the Tools menu, select Options -> EMF-> process Builder Behaviour tab.
4. The **Email output** screen consists of three tabs that define sending details and content details (as well as the [Advanced](#) tab). The actual recipients for the pager message are gathered from the various recipient modules that precede it in the EMF process flow.

5. Select the **Message section** that contains the body of the message from the drop-down list of those in the EMF Process.
6. If the message that you are sending is an HTML message, you can also send an alternative plain text message section that will be rendered in the client's email client if it isn't capable of displaying HTML messages. To send an alternative plain text message, select the **Send plain text message for HTML formatted emails** checkbox and select the alternative message section. By selecting an alternative message section, only one email is still sent. The email message will contain both message sections, but the end user will see only one, depending on their email client.
7. Select the **Email service** that you want to use to deliver the email from the list of SMTP services that you have set up.
You can click **Edit** to modify the properties of the selected Email Service.

Note: Selecting an email service will automatically populate the **Reply name** and **Reply address** fields with default values derived from the "From" settings for that service. If you want replies to be sent to a different name or email address you should enter them in these fields and replace the defaults.

8. Enter the text that should appear in the "Subject" field of the email in the **Subject** field.

Note: You can use [dynamic functions](#) in the email subject by right-clicking and selecting the required function.

9. If the email service that you are using supports **Priority**, you can select **Urgent**, **Normal**, or **Low**.

Note: This has no effect on the delivery speed of the message, simply on how it appears when it arrives on the destination device.

10. If the email service you are using supports **Encoding**, you can select from **UUEncode** and **Base64**.

Note: UUEncoding is the most common method of encoding attachments to emails, and is supported by most gateways and email programs. However UUEncoded files can sometimes be corrupted by UNIX gateways and so you may prefer to use Base64 instead.

11. To encrypt the message, select the Encryption method of either S/MIME or OpenPGP. By encrypting a message it means that only the intended recipient of the message will be able to read the message even if it is intercepted. For a message to be encrypted, the recipients of the messages must have their public keys that will be used to perform the encryption associated with their recipient profile. This association is made on the [Digital ID](#) tab of a recipient's profile. A different type of public key is needed for each encryption technique.

Note: EMF currently only supports encrypting emails to fixed and aliased recipients. There is no way to associate a digital ID with a dynamic recipient.

If an encryption option is selected and one of the intended recipients doesn't have a digital ID for the selected technique, then a warning will be logged and the message will **not** be sent to that recipient. Deliver will still be attempted for the other recipients.

Note: The subject line or email messages are never encrypted, so do not put sensitive information here. The body of the message and all attachments are encrypted.

Important: before you can send an encrypted email you need to install the **Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files** for your Java Runtime Environment (JRE). These cannot be shipped with EMF due to export regulations. The files for Java 8 can be downloaded from <http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html>. The included README.txt in the download explains how to install the files, but it should be just copying the included files to the Aptean\EMF\jreX.X.X.X\x64\lib\security and the Aptean\EMF\jreX.X.X.X\x86\lib\security directories, having backed up the existing files first.

12. To include a file as an attachment, select the [File Attachment](#) tab.
13. To include information retrieved from a database by a Data module, select the [Message Attachment](#) tab.
14. To include files saved in the POP3IN_ATTACHMENTS data section of the EMF Process, select the [Data Attachment](#) tab.

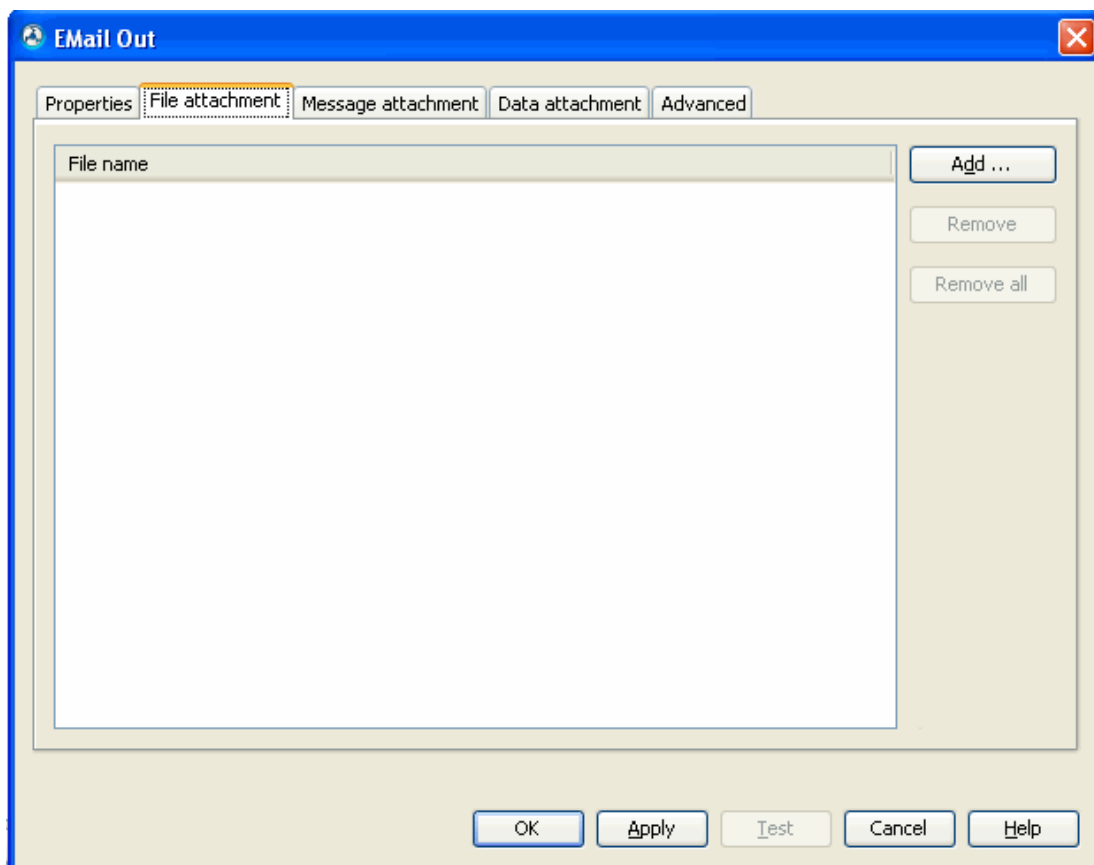
15. Select the [Advanced](#) tab to configure Message state logging (required for [Escalation](#)), and [Escalation](#) filtering.
16. Click **Test** and enter the Email address of the recipient to send a sample mail.

[Digital IDs](#)

[Go to start of Output Modules](#)

File Attachment Tab

You can send files with an e-mail message as attachments. The files can be either held on computers that the EMF machine has access to, or created by different EMF process modules.



The file attachment name supports dynamic functions, and also allows wildcards to be specified. The two wild card characters supported are "*" to mean any sequence of characters, and "?" to match a single character. So, for example, to attach all CSV files in a directory you can specify **c:\Data*.csv**. To include all files in a directory, use **c:\Data*** and to include all files for a particular year/month you could do this **c:\Data\data0902??**.csv assuming the files are called c:\Data\data090208.csv, c:\Data\data090210.csv, c:\Data\data090216.csv.

Click **Add** to browse for the required file(s) to attach (you can attach as many files as you wish). Click **Remove** to delete a file that you have previously attached.

Note: See [Attaching Files from a Data Section](#) for details on how to send email attachments stored in the POP3IN_ATTACHMENTS data section of the EMF Process.

[Email Output Screen](#)

[The Message Attachment Tab](#)

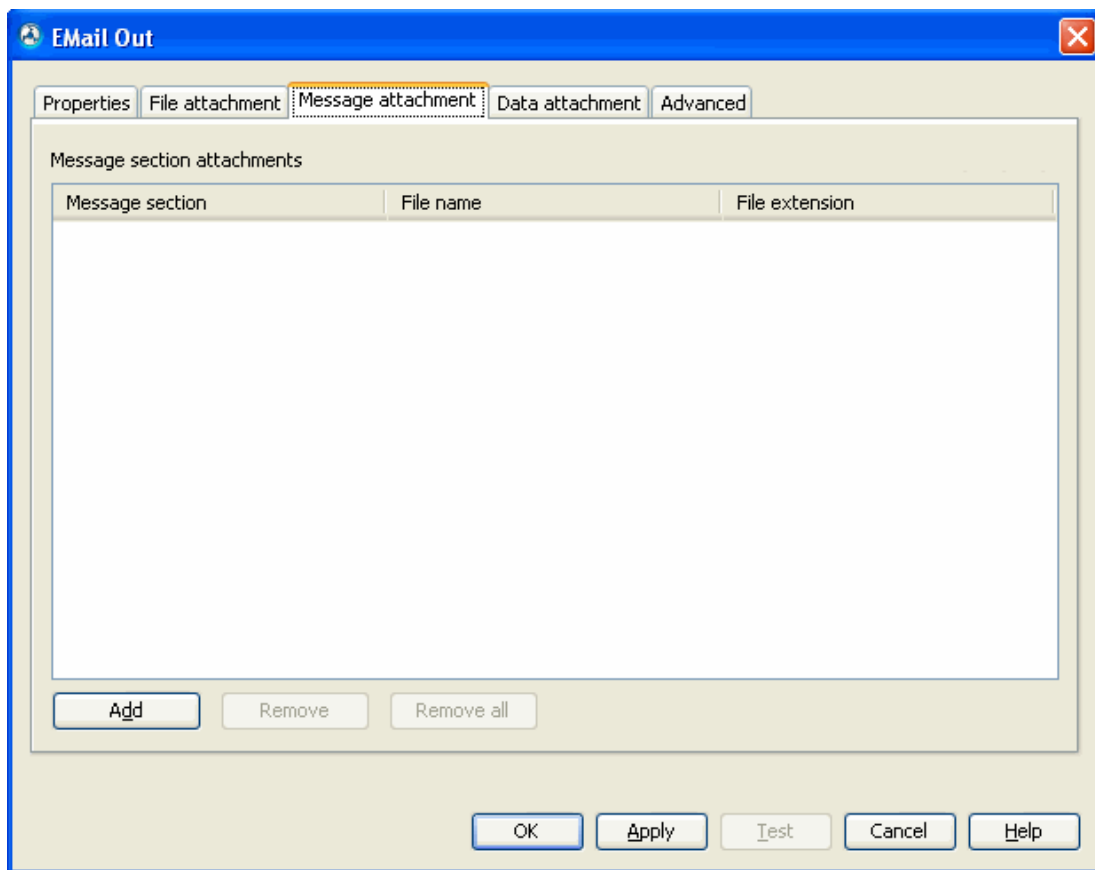
[The Data Attachment Tab](#)

[Advanced Output Module Options](#)

[Go to start of Output Modules](#)

Message Attachment Tab

Data modules collect information from databases and format the information into the required form. This can be done by defining a message section. The **Message attachment** tab is used to add this information to an email. When the message section has been decided, a file name and file extension for the information must be specified so the information can be added as an attachment.



- **Message Section** - Collated data created previously in the email.
- **File Name** - A user defined name for the attachment.

- **File Extension** - A user defined extension for the attachment.

[Email Output Screen](#)

[The File Attachment Tab](#)

[The Data Attachment Tab](#)

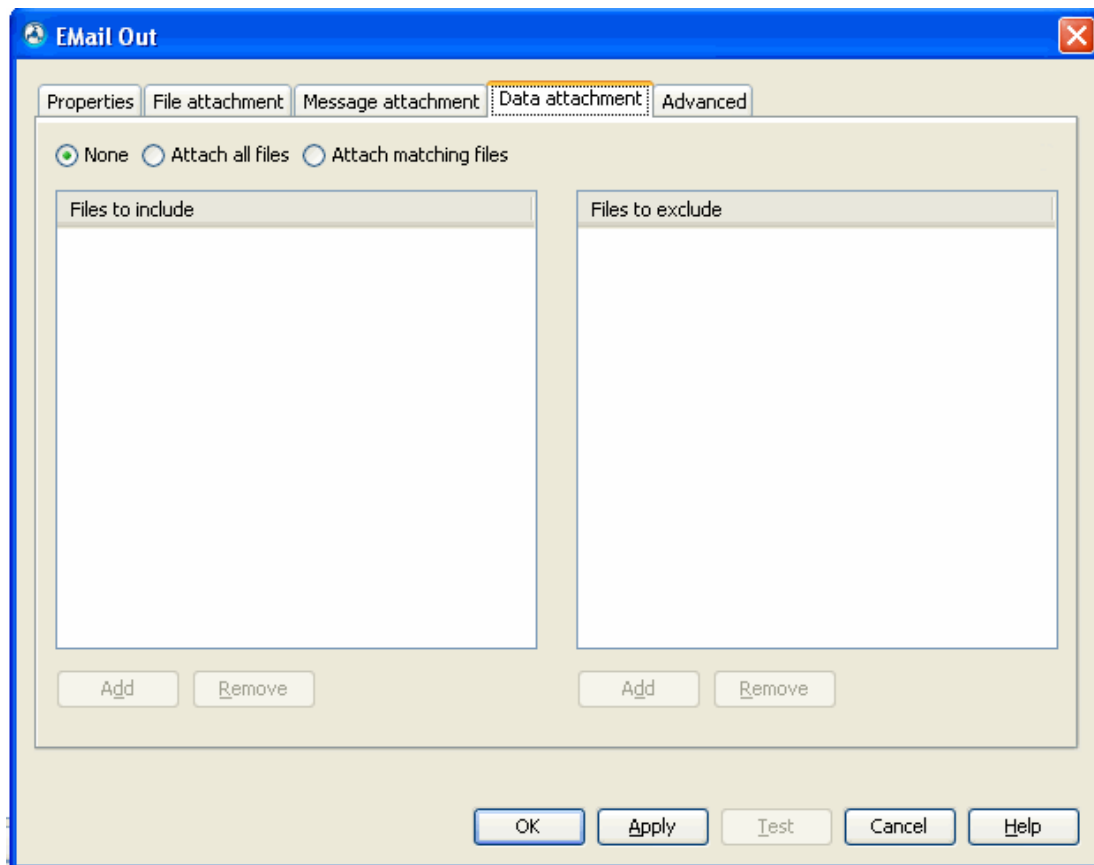
[Advanced Output Module Options](#)

[Go to start of Output Modules](#)

Attaching Files from a Data Section

You can send one or more files contained in the POP3IN_ATTACHMENTS data section of the EMF Process with an email as attachments.

Note: POP3IN_ATTACHMENTS is populated when the POP3 listener associated with the [POP3 initiator module](#) receives an email message with attachments. Attachments are only saved to the data section if you have selected the **Save attachments as data section** option in the [POP3 listener screen](#). For examples of manipulating the data section content in the [Script module](#), see **pop3attachments.js** and **base64decoding.js** in [Script Examples](#).



- Click **None** if you do not want to attach any files (this is the default).
- Click **Attach all files** to send all files in the POP3IN_ATTACHMENTS data section as attachments.
- To specify the files you want to send, click **Attach matching files** and then enter the names of the files you want EMF to attach in the **Files to include** area. If you want EMF to attach all files with names matching a certain pattern, you can use the "*" and "?" wildcards. "*" represents any number of characters, while "?" represents a single character.

To exclude files from the included fileset, type the filenames in the **Files to exclude** areas. You can also use the "*" and "?" wildcards.

The file match is case sensitive. For example, if you enter myfile*.doc, then myfile1.doc will be attached but Myfile2.doc will not be attached.

Using dynamic functions: You can use all types of [dynamic functions](#) in the filenames.

Examples

Include Files	Exclude Files	Result
*.doc		All files with a .doc extension
*.doc	myfile.doc	All files with a .doc extension except myfile.doc
.	*.htm	All files except those with a .htm extension
myfile?.*		All files with names matching myfile followed by a single character and a file extension. For example, myfile1.doc, myfilez.html
myfile*.*		All files starting with myfile. For example, myfile1.doc, myfile.my.htm, myfile_notyourfile.html, myfile 26
myfile*.* yourfile*.*		All files with names starting myfile and all files with names starting yourfile
\$RECIPIENT (User name) \$.*		All files with the same name as the email recipient and any file extension

[Email Output Screen](#)

[The File Attachment Tab](#)

[The Message Attachment Tab](#)

[Advanced Output Module Options](#)

[Go to start of Output Modules](#)

File Out Module

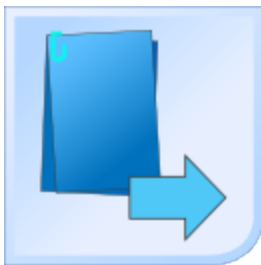
You can use the **File out** module to save information from an EMF process to a file in a location that you specify. You can specify an incremental naming system so that multiple files that are output to the same location do not overwrite each other.

EMF process output data is transferred to the file output service in the exact format that it is intended to be written to a file. It may then be further processed by services that are capable of processing files, such as FTP Out or Email Out (where the file is sent as an attachment).

Note: you can use [dynamic functions](#) (for example, \$Message\$, \$Date\$) in the **Message Section**, **Target Directory** and **Output File** fields. You can also use data within the message section to calculate the target destination.

To use a File out module:

1. Ensure that you have set up one or more [file services](#) (in the **Services** section of the EMF tree view).
2. Drag a **File Out** module into your EMF process at an appropriate place (that is after one or more recipient modules). Any formatting (for example, XML, HTML) must already have taken place earlier in the EMF process sequence.



3. Open the File Out module to display its properties. By default, you have to double-click on the icon to open but you can configure mouse-click actions on the Options window. From the Tools menu, select **Options** -> **EMF**-> **Process Builder Behaviour** tab.

The **Properties** tab defines the information that should be placed in the file, how the file should be created, and where it should be saved.

The screenshot shows the 'File Out' dialog box with the 'Advanced' tab selected. The 'Message section' is set to 'Text'. The 'File service' is 'Default File Service'. The 'Output file' is 'C:\Users\sshamsundar\Desktop\Test\example'. The 'Create directory if not present' checkbox is checked. The 'Attempt to auto detect character encoding' checkbox is checked. The 'Character encoding to use if auto detect fails or is not selected' section has 'Use platform default encoding' checked and 'Use specified encoding' set to 'US-ASCII'. The 'Append if file exists' checkbox is checked. The 'Use incremental filenames' checkbox is unchecked. The 'Sequence start number' is 0, 'Sequence end number' is 99, and 'Next sequence number' is 0. The 'Action at sequence end' section has 'Restart sequence' selected.

- Select a **Message section** from the list of those previously defined within the EMF Process.
- Select a **File service** from the list of those that you have set up in [Services](#). You can click **Edit** to modify the properties of the selected **File Service**.
- Specify the path and file name to be used in the **Output file** field. If EMF does not have write permission to the specified folder, the file will be created in the *Temp* directory of the local machine.

Notes:

1) You should not specify a mapped network drive, since the drive might not have the same mapping on the server. If you wish to output to a destination folder that is on a networked machine you should browse via the network node.

2) If you choose to type the path manually you should enter the path in a format that is appropriate for the network environment and platform of the EMF Server (for

example, using forward slashes or backslashes).

3) The path when the EMF Process is executed is relative to the EMF Server, not the EMF Administrator. For example, if you specify C:\Output.txt as the path in the EMF Administrator the output file Output.txt will be saved to the C: drive of the EMF Server.

4) You cannot use the **Browse** button [...] to access a directory on a Unix machine.

5) You may not be able to specify a Unix machine without extra third-party software to enable the connection.

4. Select the **Create directory if not present option** in order to automatically create the above directory if it is not found when the file is generated. If you do not select this option, and the target directory does not exist, the file will be saved into the machine's *temp* directory.

5. Select **Attempt to auto detect character** encoding to allow the EMF to scan the content and try to set the correct encoding. This will search the text for HTML, XHTML and XML encoding information and set the file encoding appropriately.

For example, if the text contained the following:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

then the file encoding would be set to ISO-8859-1.

If no encoding information is detected or the **Attempt to auto detect character** is not selected, then it will either:

- **Use platform default encoding**, if selected
 - Otherwise the encoding specified in the **Use specified encoding** drop-down list will be used. It is possible to type in the name of your own encoding to use in this list, as long as the Java platform supports that encoding.
6. If **Append if file exists** check box is selected then, if the output file already exists, the content will be added to the end of that file. If the file does not exist, the file will be created and the content written to it.
7. Select the **Advanced** tab to configure Message state logging (required for [Escalation](#)) and [Escalation](#) filtering.
-

Note: When you send EMF Process output to files, you should ensure you do not exceed any operating system limits on the number of files that can be written to a single directory.

Incremental File Names

To ensure that files that are output from an EMF process are not overwritten with each subsequent output, an incremental file naming system can be used. This appends a number to the end of the file name specified, and each time the EMF process is fired, the file created is given a higher reference number (in increments of 1). This continues until the sequence end number is reached, when the action to be undertaken is defined by the options (i.e. stop, re-use last number, etc.).

- **Use incremental filenames** - to ensure files are not overwritten on subsequent firings of an EMF Process, this checkbox should be selected and the other definition fields should be completed. If you do not select this option, each time the EMF Process runs it will create a new file that will overwrite the previous one of the same name.
- **Sequence start number** is the number that the incremental numbering will start from, or restart from after it reaches the Sequence end number, if you select **Restart sequence number** as the **Action at sequence end**. **Next sequence number** is the number that will be appended to the file name the next time that the EMF Process is run. Normally, the first time you run an EMF Process you would leave this number set to the same number as the Sequence Restart number. However if, for example, you wanted to keep versions 1-10 of a file you could set the **Next sequence number** to 11, and the numbering would start from 11 the next time the EMF Process runs.
- **Sequence end number** is the highest file number that EMF will create, and therefore can be used to limit the number of files that can be in the output folder. When the incremental file name reaches this number, the action will depend on the selected **Action at sequence end** option.
- **Reset Next sequence number** - Click this to reset the start number for the next sequence.
- **Action at sequence end** - use these options to specify what will happen when the file numbering reaches the **Sequence end number**:
 - **Restart sequence number** - the numbering will restart from the number specified in the **Sequence restart number** field.
 - **Use last sequence number** - all subsequent files will reuse the last number in the sequence (and overwrite the previous versions).
 - **Stop output** - no further files will be created when the last sequence number is reached.

[Go to start of Output Modules](#)

FTP Out Module

You can use the FTP out module to send information to a local or remote computer using the standard File Transfer Protocol (FTP). The information sent can be one or more files either created by the EMF process, or that already exist on the source computer. You can specify an incremental naming system so that multiple files that are output to the same location do not overwrite each other.

An FTP server must be running on the target machine for the file transfer, and you must specify the target directory and all parameters required to connect to the FTP server. This is done by configuring each FTP server as an [FTP service](#) in the Services section of the EMF tree view (you can define multiple FTP server configurations).

Note: The **FTP module** uses **passive** transfer mode and does not currently support the **active** mode so the FTP server must support passive mode.

Note: The security policy, as configured by the client, must allow access to the FTP server from the EMF server. Access can be through a firewall. The EMF FTP Output module does not break the security policy of the FTP account provider. FTP passwords are not made publicly available and FTP server settings can be hidden from the user building the EMF Process.

To use an FTP Out module:

1. Ensure that you have set up one or more [FTP services](#) (in the **Services** section of the EMF tree view).
2. Drag an **FTP Out** module into your EMF process at an appropriate place (that is after one or more recipient modules).



3. Open the FTP Out module to display its properties. By default, you have to double-click on the icon to open but you can configure mouse-click actions on the Options window. From the **Tools** menu, select **Options** -> **EMF**-> **Process Builder Behaviour** tab.

4. Select a **Message section** for output from the drop-down list of those previously defined within the EMF Process.
5. Select the desired **FTP service** from the list of those already defined in the [FTP Service screen](#).
You can click **Edit** to modify the properties of the selected FTP service. For more information, see [FTP Service](#).
6. Specify the **Target directory** on the FTP for the file to be sent to.

7. You can use dynamic functions (for example, \$MESSAGE\$, \$DATE\$) in the **Target directory** and **Output** file fields.
8. Enter a file name that the message section will be saved on the FTP server in the **Output file** field.
9. Select the required file format (ASCII or binary) from the **Send as** field.
10. Select **Attempt to auto detect character** encoding to allow the EMF to scan the content and try to set the correct encoding. This will search the text for HTML, XHTML and XML encoding information and set the file encoding appropriately. For example, if the text contained the following:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

then the file encoding would be set to ISO-8859-1.

If no encoding information is detected or the **Attempt to auto detect character** is not selected, then it will either:

- **Use platform default encoding**, if selected
 - Otherwise the encoding specified in the **Use specified encoding** drop-down list will be used. It is possible to type in the name of your own encoding to use in this list, as long as the Java platform supports that encoding.
11. Select the **Attachment** tab and choose the files that you want to send.
 12. If necessary, select the **Incremental filenames** check box and specify the required options there.
 13. Select the **Advanced** tab to configure Message state logging (required for [Escalation](#)) and [Escalation](#) filtering.
 14. Click **Test** to submit the file for delivery.

Incremental File Names

To ensure that files that are output from an EMF process are not overwritten with each subsequent output, an incremental file naming system can be used. This appends a number to the end of the file name specified, and each time the EMF process is fired, the file created is given a higher reference number (in increments of 1). This continues until the sequence end number is reached, when the action to be undertaken is defined by the options (i.e. stop, re-use last number, etc.).

- **Use incremental filenames** - to ensure files are not overwritten on subsequent firings of an EMF Process, this checkbox should be selected and the other definition fields should be completed. If you do not select this option, each time the EMF Process runs it will create a new file that will overwrite the previous one of the same name.
- **Sequence start number** is the number that the incremental numbering will start from, or restart from after it reaches the Sequence end number, if you select **Restart sequence number** as the **Action at sequence end**. **Next sequence number** is the number that will be appended to the file name the next time that the EMF Process is run. Normally, the first time you run an EMF Process you would leave this number set to the same number as the Sequence Restart number. However if, for example, you wanted to keep versions 1-10 of a file you could set the **Next**

sequence number to 11, and the numbering would start from 11 the next time the EMF Process runs.

- **Sequence end number** is the highest file number that EMF will create, and therefore can be used to limit the number of files that can be in the output folder. When the incremental file name reaches this number, the action will depend on the selected **Action at sequence end** option.
- **Reset Next sequence number** - Click this to reset the start number for the next sequence.
- **Action at sequence end** - use these options to specify what will happen when the file numbering reaches the **Sequence end number**:
 - **Restart sequence number** - the numbering will restart from the number specified in the **Sequence restart number** field.
 - **Use last sequence number** - all subsequent files will reuse the last number in the sequence (and overwrite the previous versions).
 - **Stop output** - no further files will be created when the last sequence number is reached.

[Attachment Tab](#)

[Go to start of Output Modules](#)

Attachment Tab

The **Attachment** screen is used to add files to be sent by the FTP Output module.

- **Add:** If any data collected by the EMF Process has been saved out in a file, it can be brought back into an Email as an attachment. (Any number of files can be attached to the email, whether it was created by EMF or not.)
- **Remove:** Information not required as an attachment can be removed using this button.

Notes:

- 1) You should not specify a mapped network drive, since the drive might not have the same mapping on the server. If you wish to output to a destination folder that is on a networked machine, browse via the network node.
 - 2) If you choose to type the path manually you should enter the path in a format that is appropriate for the network environment and platform of the EMF Server (for example, using forward slashes or backslashes).
 - 3) The path when the EMF process is executed is relative to the EMF Server, not the EMF Administrator. For example, if you specify C:\Output.txt as the path in the EMF Administrator the output file Output.txt will be saved to the C: drive of the EMF Server.
 - 4) You may not be able to specify a UNIX machine without extra third-party software to enable the connection.
-

[Explanation of the FTP Output screen](#)

[Explanation of FTP Output](#)

[Advanced Output Module Options](#)

[Go to start of Output Modules](#)

Queue Out Module

The **Queue Out** module allows you to send the EMF Process output to a JMS-compliant queue (for more information on JMS, see [Messaging and the Java Message Service](#)).

Queue output can be either **queue-based** or **recipient-based**. Queue-based output is sent to a selected external queue, while recipient-based output is sent to the external queue(s) configured for each recipient of the EMF Process output. Before you can use recipient-based output, you must ensure that you have set up external queues for the recipients (in the **Devices** tab of the [Recipient screen](#)).

To use a Queue Out module:

1. Ensure that you have set up one or more [External queue providers](#) and [External queues](#) (in the **Services** section of the EMF tree view). If required, you can use the same queue provider as the EMF system. However, you should not use the EMF system queues as these are reserved for internal use only.
2. Drag a **Queue Out** module into your EMF Process at an appropriate place (for example, after one or more recipient modules, if you want to use recipient-based output).



3. Open the Queue Out module to display its properties. By default, you have to double-click the icon to open, but you can configure mouse-click actions on the Options window. From the Tools menu, select **Options** -> **EMF**-> **Process Builder Behaviour** tab.

You must specify the message section that you want to use as the body of the message, and the type of queue (**recipient-based** or **queue-based**) that you want to use, on the **Properties** tab. You can specify additional optional properties in the **Reply** and [Message](#) tabs, if required.

4. Select a **Message section** from the list of those already defined within the EMF Process.
5. If you want to send the messages to the queue provider and queue defined in the recipient's settings, select **Use recipient based queue**. Messages are processed on a per-recipient basis, so more than one message may be sent.

Note: You cannot have a reply sent back if you select **Use recipient based queue**.

Alternatively, if you want to send messages to a queue that you specify yourself, select **Use queue settings specified below** and then select a **Queue provider** and **Queue** (the queue provider is the external queue manager used to specify where the external queue resides).

Important: When you configure external queues for use with the Queue output module, you should **not** use the EMF system queues (these are reserved for use by the EMF system). However you can use the same queue provider as the EMF system, as long as you configure additional queues for use with the Queue output module.

6. If you are using queue-based output and you want to have replies sent back to a queue or queue listener, you can specify the destination in the Reply settings section. Only queues and queue listeners using the same queue provider as you selected in the **Properties** tab will be displayed.
7. It is not possible to send a reply from a certain queue if you chose **Use recipient based queue** because the recipient may have several queue devices configured for different queue providers.
8. If you want replies to be sent from the queue, select the **Use queue settings specified below** check box and then select the **Queue provider** or **Queue** from the drop-down lists. The fields in the **Reply settings** section are optional. **Do not reply** is the default setting; it indicates that no replies need to be sent back from the queue. You can click **Edit** buttons available next to both the **Queue provider** and **Queue** drop-down fields, to modify their respective properties. For more information, see [External Queue provider](#) and [External Queue screen](#).

For more information on queue listeners, see [Queue Listener Server Class](#) and [Queue Listener \(EMF Process Builder\)](#)
9. If you need to set custom options for the messages (for example, priority, expiry, and so on.), select the [Message tab](#) and enter settings as required.
10. Select the **Advanced** tab to configure Message state logging (required for [Escalation](#)) and [Escalation](#) filtering.

[Back to Start of Output Modules](#)

Message Tab

You can use the Queue output **Message** tab to change message settings, such as priority and expiry period (if you do not make any changes the default settings are used).

The settings in the Message tab are used in the message header when the JMS message is constructed. The JMS message header fields contain identity and routing information. Refer to [Messaging and the Java Message Service](#) for more information.

- **Priority:** The external queue provider uses the priority selected here to prioritize the message being put on the queue. You can select a priority from Low (0) to Urgent (9). Choose a priority in the range 0-4 for 'normal' delivery, or a priority in the range 5-9 for 'expedited' delivery.

The JMS provider may not implement priority exactly, but it should attempt to deliver 'expedited' priority messages before 'normal' priority messages.

Note: The default Priority setting is 4, which is equivalent to "normal".

- **Correlation ID:** You can use this to link one JMS message with another, for example, to link a response message with its request message. Type the message ID of the message you want to link to in the Correlation ID field, or use a dynamic function to retrieve the message ID. Right-click in the Correlation ID field to select a dynamic function.
- **Persist when queued:** Select this checkbox if your JMS provider supports message persistence and you want to guarantee delivery of the message. A persisted message is stored in a database so it can be re-sent if it is lost before reaching the client.
- **Expiration period and interval:** allows you to specify the length of time that a message will remain in the queue. Enter the expiration period in the **Expiration period** textbox, and then select the units (i.e. seconds, minutes, hours, days, weeks or months) from the **Expiration interval** drop-down list.

Note: The default setting for the expiry period is **0**, which signifies that the message will never expire and will remain in the queue indefinitely.

[Queue Output Module](#)

[Back to Start of Output Modules](#)

Run Executable Module

Information gathered within an EMF process can be sent passively to a recipient's device (for example, a mobile phone or an email account) for them to read and act upon. It can also be used to launch programs (for example, to view output of a specific file type) or perform actions on the recipient's machine. You can do this using the Run Executable (RunExe) module. To do this, you must enter the IP address (or network name) of the recipient's computer in a recipient's profile, select that recipient in a recipient module, and then configure the RunExe Output module to send the appropriate data type. The Run Executable module now returns the output of a command back to EMF (example at the end) in a new data section.

Important: Before you can use the Run Executable output, you must ensure that both the EMF RunExe client and an appropriate client application are installed on the destination machine. The client application must be configured to handle the data that is sent, and can be anything appropriate. For example, if the data to be sent is plain text, any text editor or word processor could be configured to receive and process this.

Security

Communication between the EMF Server and RunExe client is by TCP/IP. By default, this communication uses secure sockets. Depending on the requirement, you can choose to run it securely or insecurely to transfer the data, and also configure it to use your own certificates for extra security. Perform the following configurations:

- **For insecure mode of data transfer:**

1. Open the **ExeClient.bat** file and set
`-Dcom.emf.runexe.ssl.useSSL=false`
2. Open the **EMF Server Service Configuration** and in the Java Options text editor enter
`-Dcom.emf.runexe.ssl.useSSL=false`
3. Open the **EMF RunExe Service Configuration** and in the Java Options text editor enter
`-Dcom.emf.runexe.ssl.useSSL=false`

- **For secure mode of data transfer:**

1. Open the **ExeClient.bat** file and set
`-Dcom.emf.runexe.ssl.useSSL=true`
2. Open the **EMF Server Service Configuration** and in the Java Options text editor enter
`-Dcom.emf.runexe.ssl.useSSL=true`
3. Open the **EMF RunExe Service Configuration** and in the Java Options text editor enter
`-Dcom.emf.runexe.ssl.useSSL=true`

The Run Exe client uses two certificates to secure the connection. The first server certificate is used by the Run Exe client to encrypt the data. The second certificate, which must be generated from the first certificate is a client authentication certificate. This is used by the EMF service to prove it is valid to connect to the Run Exe client. By default EMF uses two self-signed certificates included with the product. For extra security, properly signed certificates can be obtained from an authorized signing authority. To change the keystore that is used to use different certificates, make the following keystore changes:

- In the **ExeClient.bat** file, set

```
-Dcom.emf.runexe.ssl.keyStore=path to server keystore file
-Dcom.emf.runexe.ssl.keyStorePassword=password for server
keystore file
```

Open the **EMF RunExe Service Configuration** and in the Java Options text editor enter

```
-Dcom.emf.runexe.ssl.keyStore=path to server keystore file
-Dcom.emf.runexe.ssl.keyStorePassword=password for server
keystore file
```

- Open the **EMF Server Service Configuration** and in the Java Options text editor enter

```
-Dcom.emf.runexe.ssl.clientauthentication.keyStore=path to client
authentication keystore file
```

```
-
```

```
Dcom.emf.runexe.ssl.clientauthentication.keyStorePassword=passwor
d for client authentication keystore file
```

- To ignore the server certificate errors and warnings, in the **EMF Server Service Configuration**, set

```
-Dcom.emf.runexe.keyStore.ignoreInvalidCertificateWarnings=
```

[true|false]

Note: When using self-signed certificates, set the option as `true`.

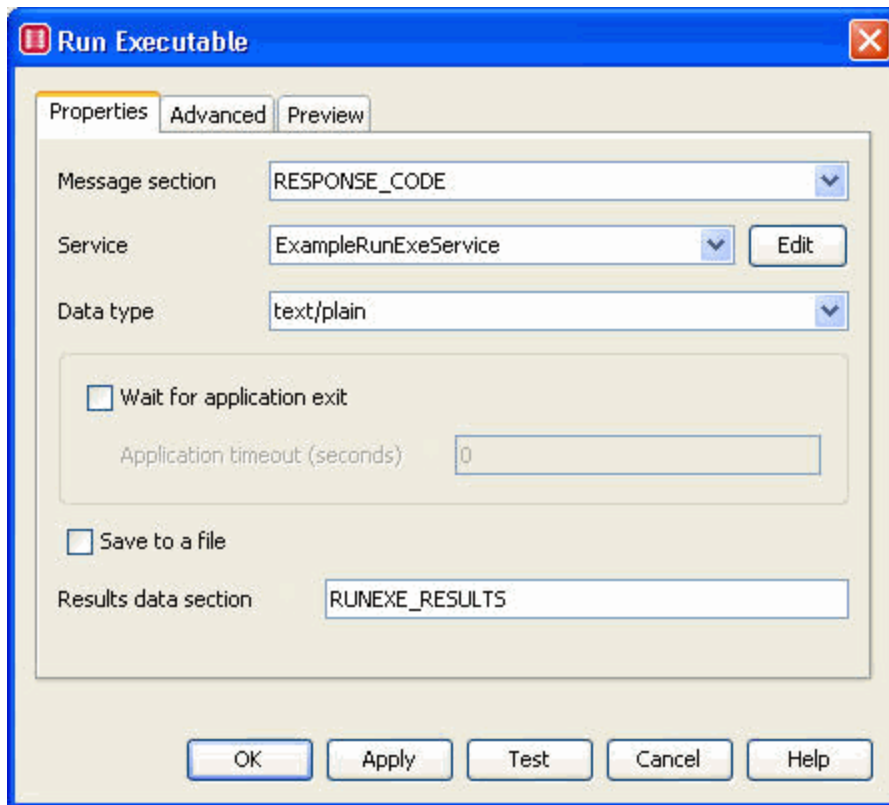
Note: Configuration of the remote application is done on the remote machine, not from within the EMF system.

To use a Run Executable module:

1. Ensure that you have set up one or more [Executable services](#) (in the **Services** section of the EMF tree view) and that the RunExe client is [installed and running](#) on the destination machine.
2. Drag a **Run Executable output** module into your EMF Process at an appropriate place after a recipient module. Any recipients that you want to send the output to must have an [IP address device](#) configured.



3. Open the **Run Executable** module to display its properties. By default, you have to double-click on the icon to open but you can configure mouse-click actions on the Options window. From the Tools menu, select **Options -> EMF-> Process Builder Behaviour** tab. The Run executable screen, which defines the information that should be sent to recipients via their computers.



4. Select the required **Message section** from the list of those already in the EMF Process.

Important: The selected message section must contain the path (on the remote machine) and filename of the document that you want to launch. For example, to launch the Excel file myxls.xls in the remote machine's c:\mydocs directory, type c:\mydocs\myxls.xls in the text formatter. This is used as the argument for the application being launched (in this case, Excel). Do not use command-line reserved characters such as "&" in the parameters.

5. Select the required Executable **Service** from the list of those that you have set up. You can click **Edit** to modify the properties of the selected Service. For more information, see [Executable Service](#).
6. Select the appropriate **Data Type**. Data types are mappings between the EMF Process output and the application that will handle the output on the client machine. Currently predefined choices are available: **text/plain**, **text/html** and **Shell**, but new data types can be added (if you do add a new data type, you must also edit **DataTypes.properties** on the client machine in order to define the data type and associate an application with it).

Note: If you want to run an executable file instead of a data file, select either **Shell** or the **Shell2** (recommended) data type.

- **Shell** - This will run the csShell executable - **csShell.exe** on Windows, **csShell.sh** on Solaris - on the remote machine. This executable in turn runs the commands you specify in the preceding Text formatter. For example, if you enter

"c:\\\\myfile.bat" **param1** in the Text formatter, csShell will run the **myfile.bat** batch file and pass it the parameter **param1**. Note that you can use the same command-line arguments with csShell.exe as you can with the standard Windows command prompt (CMD.EXE). If EMF is not installed on the remote machine, then you may need to [edit the location of csShell](#) in the Run Exe client.

If you are running the Run Exe Client as a service then csShell may not function correctly (depends on operator rights). It is recommended to use the Shell2 datatype.

Important: If you selected **Shell** as your Data Type (see above), you should not use the **Wait For Application Exit** option (because it would wait for the csShell executable to exit and not the batch file that csShell launches). CsShell always exits immediately after launching the batch file.

- **Shell2** - Launches the executable specified as a child process of the Run Exe Client. All console messages will appear in the Run Exe client and therefore no user interaction can take place with the console e.g. putting a **pause** in a batch file will cause the batch file to hang as the user will not be able to press a key to continue. If user interaction with the console is required, use the csShell datatype.
7. If you want the server to pause for a while before continuing with any other tasks, in order to allow the remote application to close, select the **Wait for application exit** option and then specify a period of time in the Application timeout field.

Note: If you select the **Wait for application exit** option you will receive a message in the **Messages** area of the Run Exe client. **Exit Code 0** means that everything has completed successfully, **Exit Code -1** means that an error has occurred and you should check what has happened. The location of the batch file is specified in the [General tab of the Run Exe client](#). If you do not select **Wait for application exit**, the screen will always show **000 OK** regardless of what actually happens

8. If you want to save a copy of the information in the message section to a location on the remote machine, select the **Save to a file** option. The default location for the file is **RunExeClient\Data**.
9. Click **Test** and enter the IP address to review the results.
10. Select the **Advanced** tab to configure Message state logging (required for [Escalation](#)), Target device, and [Escalation](#) filtering.

Example for a command being returned to EMF

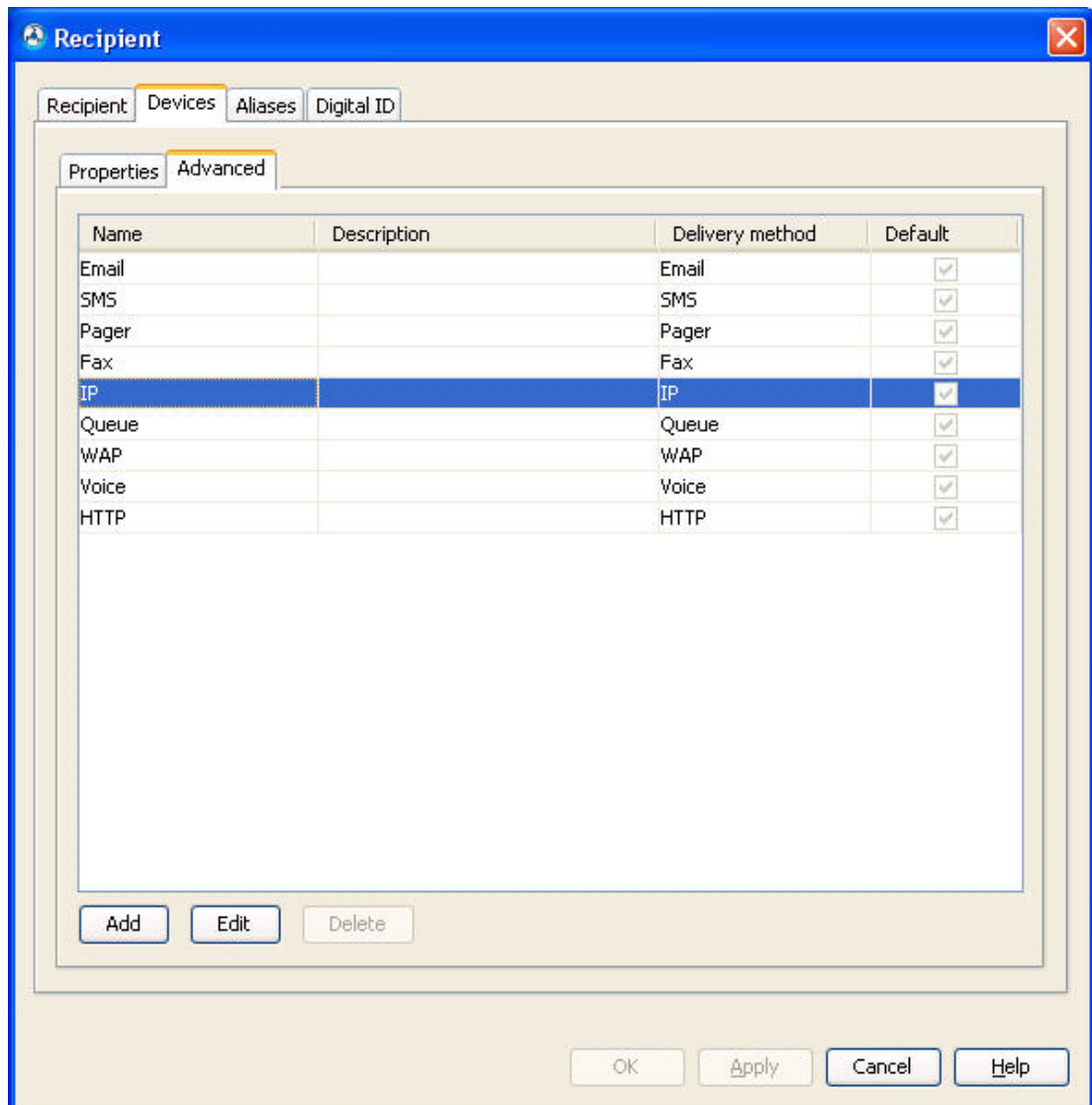
1. Select and drag modules as shown below:



2. Open the Text Formatter and enter the name of the file you want to run.
For example: c:\runexe.bat.

Note: The commands to be executed are in the .bat file.

3. In the Recipient module, add a recipient whose assigned IP address is of the machine running the RunExe client.
 - a. To define an IP address for a recipient, double-click the recipient name in the EMF tree view and select the Devices tab. Select Advanced.



- b. Select **IP** and click **Edit**.

The screenshot shows a Windows-style dialog box titled "Edit device". It has a blue title bar with a close button (X) in the top right corner. The dialog is divided into two main sections. The first section, labeled "Device", contains a "Name" text box with the value "IP", a "Description" text area, and a "Delivery Method" dropdown menu currently set to "IP". The second section, labeled "Device properties", contains an "IP address" text box. At the bottom of the dialog are three buttons: "OK", "Cancel", and "Help".

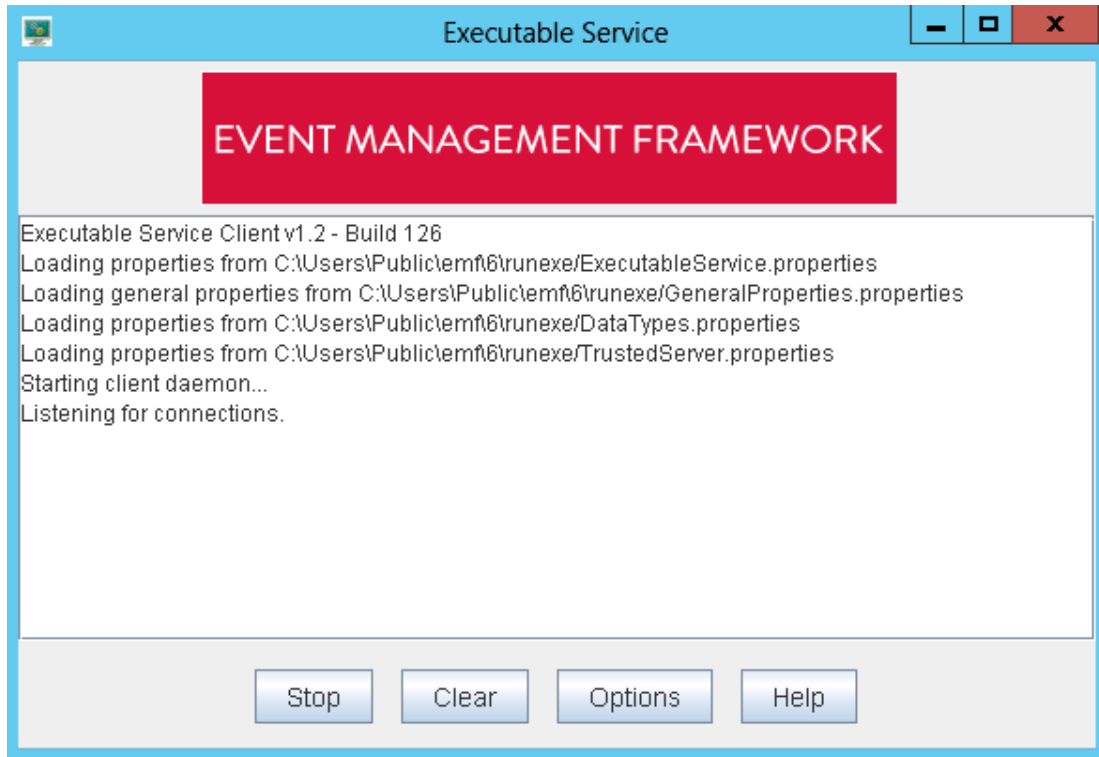
- c. Enter the **IP address** in the Device properties section and click **OK** to save and close.

4. Open the Run Executable module and enter the details as shown below:

The screenshot shows the 'Run Executable' dialog box with the following configuration:

- Message section:** Text
- Service:** ExampleRunExeService
- Data type:** Shell2
- ☒ Wait for application exit
- Application timeout (seconds): 30
- ☐ Save to a file
- Results data section: RUNEXE_RESULTS

5. Ensure that the ExeClient is running, either as a service (an UI is not present) or using ExeClient.bat.



6. Open the Run Executable module and click **Test**. Enter the **IP address** of the client machine and click **OK**.

The client machine executes the .bat file.

However, in a real-time scenario, the process runs through the EMF server. The server communicates to the RunExe client.

[Client Server Communication](#)

[Server Functionality](#)

[Client Functionality](#)

[Go to start of Output Modules](#)

Using the RunExe Client

The following topics contain information intended to assist you with installing and managing the Executable Client:

- [Installing and Running the Executable Service Client](#)
- [Defining Data Types](#)
- [Defining Trusted Servers](#)
- [Changing General Settings for the Executable Client](#)

Installing and Running the Executable Service Client

The **Executable Service** is a mechanism that EMF can use to run an application on a remote PC when an EMF Process is sent. A user can therefore subscribe to an executable EMF Process and receive its output on their client PC. It consists of two components - the Run Executable Output Module, which is configured using the EMF Process Builder in the EMF administrator, and the **Executable Client**. To run a program remotely, the **Executable Client** software must be installed and running on the recipient machine.

To install the Executable Service client on a machine:

1. Ensure that the recipient machine is Java 7 (or later) compliant, for example, if it has either the EMF administrator or the Java Runtime Environment (JRE) v7 installed on it, and a TCP/IP network connection.
2. Copy the files and folders as follows:
 - **On a Windows machine:** Copy the **EMF\RunExe** directory to a convenient location on the recipient machine. Note that some of the batch files (ExeClient.bat/ ServiceInstall.cmd) will need to be manually altered to point to a valid JRE on the recipient machine.

To start and configure the Executable Service client:

1. Start the client as follows:
 - **On a Windows PC:** Run the batch file **ExeClient.bat** from the **RunExe** directory on the recipient machine. Alternatively, if the remote machine has the EMF administrator installed on it, you can click the **Start** button and point to **EMF 7** and click **Executable service**.
2. The **Executable Service** screen appears (possibly minimized) and displays a log of all the activity that occurs. If this is the first time that you have started the client, or if you want to add more servers, click **Stop** to pause the client.
3. Click **Options** to open the configuration screen for the client.
4. Select the **Trusted Servers** tab and click the **Add** button on this tab to [add the host name and IP address](#) of the machine on which the EMF server is running.
5. Select the **Data Types** tab, and then click the **Add** button on this tab to [add a new data type](#).
6. Click **OK** to close the configuration screen and return to the main Executable Service screen.
7. Click **Start** to start the service.

To clear the activities log:

- Click **Clear** on the **Executable Service** screen.

To stop the Executable Service client:

- Click **Stop** on the **Executable Service** screen.

To install the EMF Run Exe Client as a Windows Service:

- Run the batch file ServiceInstall.cmd in the RunExe directory. If installing on a separate machine and the RunExe directory has been copied from an installation of EMF on a different machine, then the "ServiceInstall.cmd" will need modifying to point

to a compatible JRE. The ServiceConfigure.cmd can be run to configure the service.

[Configuring general settings for the Executable Service client](#)

[Adding the URLs of trusted servers](#)

[Defining Data types](#)

[Manually editing the Properties files](#)

Defining Data Types

You can use the **Data Types** tab to view, define and edit the information types that the Executable Client should accept from incoming EMF Processes and attempt to process.

Note: A **Data type** (as used by the Executable Service) is a mapping between data from an incoming EMF Process and the application that will be launched to handle it. For example, an EMF process whose data type is **text/plain** might be handled by Notepad under Windows, or by Emacs under Unix. The data type is usually entered as a MIME (Multipurpose Internet Mail Extensions) type, but it does not have to be, as long as it matches the Data Type name that is sent by the EMF server.

To view the Data Types tab:

- Click **Options** on the Executable Service client screen and then select the **Data Types** tab. The Data Types tab shows a list of all the types of data that the client is currently configured to deal with, the **application** that will open in order to deal with those data types, and the **prompt** (the length of time that the client will wait before opening each application).
 - **To add a data type:** Click **Add** to open the [Add/Edit Data Type screen](#) where you can specify a new data type, the application that should open to handle files of this type, and other options.
 - **To edit an existing data type:** Select the desired data type and click **Edit** to open the [Add/Edit Data Type screen](#), where you can modify details of a data type including the application that should open to handle files of this type, and other options.
 - **To remove a data type:** Select the desired data type and click **Delete**.

[Installing and running the Executable Service client](#)

[Configuring general settings for the Executable Service client](#)

[Adding the URLs of trusted servers](#)

[Manually editing the Properties files](#)

Defining Trusted Servers

You can use the **Trusted Servers** tab to view, add and edit the list of remote servers that the Executable Client should accept alerts from. Incoming data from a machine that is not identified on this screen, either by its IP address or DNS name, will be rejected by the Executable client.

To view the Trusted Servers tab:

- Click **Options** on the Executable Service client screen and then select the **Trusted Servers** tab. The Trusted Servers tab shows a list of all the servers that the client is currently configured to accept data from.
 - **To add a server:** Click **Add** to open the [Trusted Server screen](#), where you can specify a new server name and network address.
 - **To edit an existing server:** Select the required server and click **Edit** to open the [Trusted Server screen](#), where you can modify the server name and network address.
 - **To remove a server:** Select the required server and click **Delete**.

[Installing and running the Executable Service client](#)

[Configuring general settings for the Executable Service client](#)

[Defining Data types](#)

[Manually editing the Properties files](#)

Changing General Settings for the Executable Client

You can use the **General** tab of the Executable Service client's configuration dialog to change the following settings:

To specify the port that the Executable Service client should open:

- To listen for incoming EMF Processes, enter the required number in the **Listen on Port Number:** field. Any EMF servers that need to send messages to this client must use the same port number.

Note: The EMF server is configured to use a default port number (8100), although the port can also be configured per recipient.

To log all events to a file:

- Select the **Log to file** option and enter the location and name of the log file in the **Log File Name and Location** field.

Note: You can specify a relative or absolute path to the folder. If you specify a relative path, that path will be re-created relative to the working folder of the Executable Client service.

To write the log entries as Unicode (double-byte) characters:

- Select **Log as Unicode**.

To specify the folder where any temporary files should be saved:

- Enter the location in the **Directory to Store Temporary Files** field. If a Data type is configured by the EMF Server to save its data to a temporary file on the client, this is where the temporary file will be created.

To remove all old temporary files:

- The next time that the Executable Service client starts, select the **Delete Temporary Files on Startup** option. Note that only those files that have an extension specified in the [Data Types tab](#) will be deleted.

[Installing and running the Executable Service client](#)

[Adding the URLs of trusted servers](#)

[Defining Data types](#)

[Manually editing the Properties files](#)

Add/Edit Data Type screen

You can define the data types that you want the EMF Executable Service client to handle using the **Add/Edit Data Type** screen of the **Configuration** dialog.

To define or modify a data type:

1. Enter a name to identify the **Data Type** (for example, "application/msword"). This must match the name of the data type as configured in the EMF administrator.
2. Enter the name of the application to launch when data of this type is received in the **Application name** field. This can be anything you like, for example, "Microsoft Word".
3. Enter the exact path and filename of the application to be launched in the **File or process to launch** field, for example, "C:\Program Files\Office\msword.exe" (on a Windows machine) or /opt/office52/program/soffice/writer (on a Solaris machine - note that if the application is already in the Solaris path you need only enter the executable file name, for example, dtpad).

Important: If your target application cannot be launched directly from Java - for example, if the EMF Run Executable Output module is calling a batch file with a set of parameters - you must set up the supplied **csShell** executable as the client application and run the application from there. To do this, click the **Shell** data type in the Data Types screen, and check that the path to csShell is correct in this screen. You can then enter the path to the executable and any required parameters in the Text formatter preceding the Run exe module in the EMF Process, for example, **c\:\myfile.bat param1**. If you are using csShell on Windows, you must enter the path to the executable using the 8.3 (DOS) naming convention (for example, no long file names or spaces). Alternatively, you can use quotation marks around the path.

4. If the Executable client has been configured by the EMF administrator to save the data to a temporary file and then pass the file reference to the application (rather than passing the EMF Process data directly to the application), enter the extension that

should be used for the temporary file in the **Temporary File Extension** field. This allows applications such as web browsers to treat incoming files correctly.

5. Normally, the target application is launched immediately upon receipt of appropriate data. To prompt before launching the application (for example, in order to give you the opportunity to cancel the application launch), select the **Prompt before opening** option and then enter a required delay time in the **Delay in Seconds** field. The client will wait for the defined number of seconds and, if it has not been cancelled in this time, start the application automatically.

[Installing and running the Executable Service client](#)

[Configuring general settings for the Executable Service client](#)

[Adding the URLs of trusted servers](#)

[Defining Data types](#)

[Manually editing the Properties file](#)

Editing the EMF Executable Service Property Files

The Executable Client properties are saved and loaded from property files. It is possible to load these properties from a mappable network address (as long as the address makes sense in the platform's native file system). This allows any number of Executable Client PCs to be administered remotely, as they can all be configured to read from the same set of property files.

The following properties files are used by the Executable Client:

- **ExecutableService.properties:** Contains the locations and filenames of all the other properties files. The location/name of this file can be configured as a command-line parameter, for example, **props=.\\ExecutableService.properties**.
- **DataTypes.properties:** Contains the Data Types accepted by this client. This file should not normally be edited directly. It is updated via the [Data Types](#) tab in the Executable Client configuration dialog.
- **TrustedServers.properties:** Contains the addresses of the EMF servers from which this client will accept incoming connections. This file should not normally be edited directly. It is updated via the [Trusted Servers](#) tab in the Executable Client configuration dialog.
- **GeneralProperties.properties:** Contains the properties edited via the [General](#) tab in the Executable Client configuration dialog. These include the port number that the client should listen on, temporary file directory, and various log file options.

Location of Properties files: The Properties files can be found in the root directory of the Executable Client Service working folder (for example, D:\\RunExeClient).

[Installing and running the Executable Service client](#)

[Configuring general settings for the Executable Service client](#)

[Adding the URLs of trusted servers](#)

[Defining Data types](#)

The Add/Edit Trusted Server Screen

You can specify the machines that you want the EMF Executable Service client to accept incoming data from using the **Add/edit Trusted Server** screen of the **Configuration** dialog.

To define or modify a trusted server:

- Enter a name for the server in the **Server Name** field. This can be anything you like (e.g. free text) - it is used in the properties file, and can also be used as a hint to identify the server (e.g. "Mail_Server_In_Marks_Office").
- Enter the **Network Address** that is used to identify the server. This can be either the IP address or DNS name.

Important: No further authentication is performed on the connecting server.

[Installing and running the Executable Service client](#)

[Configuring general settings for the Executable Service client](#)

[Defining trusted servers](#)

[Defining Data types](#)

[Manually editing the Properties files](#)

How RunExe Works

The functions of the [Executable Output Service](#) and the [Executable Service Client](#), including [intercommunication](#) between the client and server are described in the topics of this section of the Help.

Executable Output Service Functionality

This section specifies functionality for the Executable Output Service (here referred to as the "server") that integrates with the Service Manager:

- The server handles multiple concurrent connections.
- Although the server is allowed to keep a session open to a client, it generally keeps sessions short-lived. The session functionality is included so that the server and client can maintain a short conversation without having to reconnect for each request.
- The server receives requests from the client, to notify it of the result of operations.
- It is possible to Escalate an EMF Process based on the client's feedback.
- The server fulfills all the common functions of an Output Service, in particular (but not limited to):

- If the server cannot connect to the client, this is logged as an error.
- If the server cannot connect to the client, the EMF Process can be escalated.

[Client Server Communication](#)

[Client Functionality](#)

[The Run Executable module](#)

[Go to start of Output Modules](#)

Client Functionality

The client needs to handle only one connection at a time and it handles requests from the server. The client looks up the requested data type to get the application that needs to be launched. It then launches the requested application, passing it any parameters that were supplied with the request. Parameters can be passed by value (in-memory) to the application, or by reference (where the parameter data is first saved to a temporary file, then the file reference passed to the application).

Before launching an application, the client conditionally displays a dialog (depending on how the application's data type has been configured). The dialog consists of:

- A **time-out value** which counts down. When the value reaches zero, the application launches.
- An **OK** button allowing the user to launch the application immediately.
- A **Cancel** button allowing the user to veto the launch.

The client can connect to a remote Listener, the address of which will be specified in the EMF Process request. Whether or not to connect to the Listener is also defined in the request.

The client will connect to the Listener if a condition specified by the EMF Process request is true. The possible conditions are:

- **Unconditional** - Must connect to the Listener to inform it of the final result.
- **Only connect if** the application launched successfully.
- **Only connect if** the application failed to launch (either there was an error or the user vetoed the launch).
- **Only connect if** the application launched and returned a return code to be specified in the request.
- **Only connect if** the application launched and did not exit after a specified number of seconds.
- **Only connect if** the application launched and exited before a specified number of seconds.

The client software includes a user interface that allows the user to do the following:

- **Start** and **stop** the client service.
- Specify the **port number** to listen on.
- Create a **list of Trusted Servers**.

The client will only accept connections from servers on this list, and each server can be specified as either an IP or domain address (for example, company.com:16000).

- Create a **list of data types**. The "data type" can be any text string, although it is encouraged to use MIME types. For each data type, the following must be specified:
 - The application associated with it (one per data type).
 - The local path to the application. The application could also be run from a server, as long as it can be launched from the client PC.
 - Are the parameters passed to the application by value, or by file reference? If by value, the parameters are passed verbatim to the application. If by file reference, the parameters are first saved to a temporary file, then the file path and name are passed to the application as a parameter.
 - Prefix (e.g. if the prefix is "/r " and the parameter data is "tmp0001.txt" then the command-line sent to the application would be "/r tmp0001.txt").
 - Postfix (e.g. a prefix of "/file{" and postfix of "}" would result in "/file {tmp0001.txt}").
 - Is the user prompted before launching the application?
 - If the user is prompted, how many seconds are they given before the application launches?
- There can be one or more data types specified per application (for example, the client could be configured so that Microsoft Word handles both "text/plain" and "application/msword" data types.
- All text displayed is taken from a Properties text file.

Properties Files

The main client properties are stored locally in various Properties text files. The locally stored properties file specifies the locations and filenames of each of the other Properties files. These default to a local path, but can be changed to point to a network path. Default Properties files are supplied as part of the client installation. The systems administrator can copy pre-configured Properties files to the client, for ease of administration. More than one client can be made to point to the same Properties file, as long as each of the clients has read access to the file.

[Client Server Communication](#)

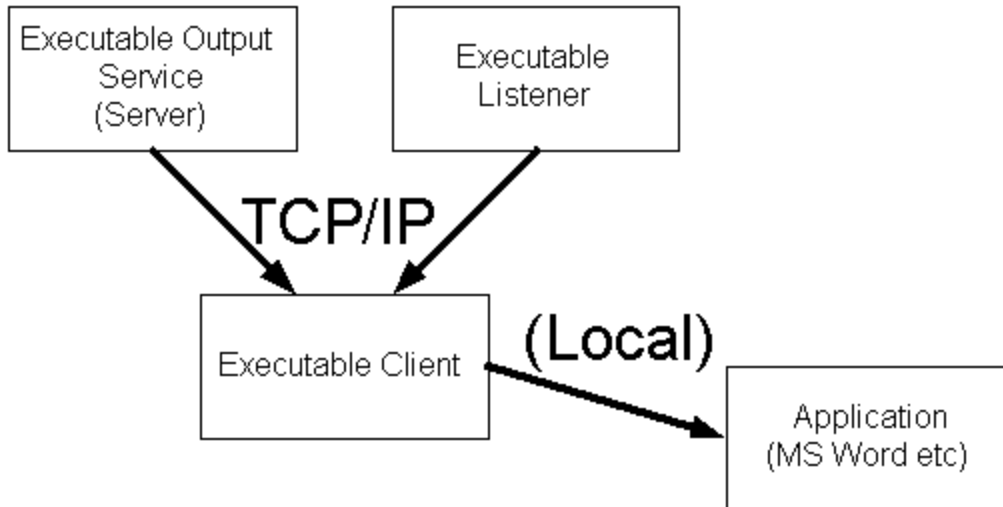
[Server Functionality](#)

[The Run Executable module](#)

[Go to start of Output Modules](#)

Client-Server Communication

Communication between the server and client takes place over TCP Sockets.



The Executable Service has the following intercommunication between client and server:

1. **The server establishes a session.**
2. **The server proves its identity to the client.** The server must prove its identity to the client at the start of each session, and the client may demand proof of identity from the server at random intervals during the session.
3. **The client proves its identity to the server.** The client must prove its identity to the server at the start of each session, and the server may demand proof of identity from the client at random intervals during the session.
4. **The server sends one or more requests to start an application on the client.** Each request consists of the MIME type of the data (or some other text field identifying the data - this can be anything, as long as it is recognized by the client - the client will use this field to identify which application it needs to execute) and the data that will be passed to the application as a parameter (or sequence of parameters).

The request includes the **data type** of the EMF Process (the application to be launched is derived from this), the EMF Process **data, instructions** on when and why to contact the Listener, and an EMF Process **instance ID** that may be used to identify this instance to the Listener.

The client attempts to launch a local application and, if required, initiates contact with the Listener and sends a request consisting of the EMF Process instance ID supplied by the server and the outcome of the executed process. Depending on the outcome, the Listener may cancel [Escalation](#) (if relevant to the EMF Process).

5. **The server closes a session.** The session will time out (i.e. expire) after a default 10 minutes, although the timeout interval can be configured on the server. The server specifies the required timeout interval at the start of the session, and it can send a session "keep-alive" request to the client, to prevent a session from timing out.

[Server Functionality](#)

[Client Functionality](#)

[The Run Executable module](#)

[Go to start of Output Modules](#)

Attaching Files from POP3IN_ATTACHMENTS

You can include one or more files contained in the POP3IN_ATTACHMENTS data section of an EMF Process in an XML-formatted message section. This message section can then be used later in an output module, for example the Queue out module. In order for a receiving application to parse the content of the message section, it must understand the XML schema you are using.

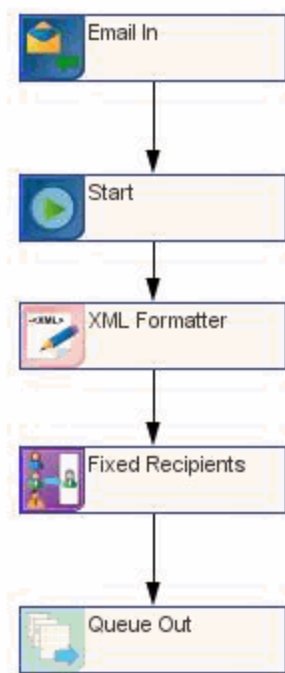
Note: POP3IN_ATTACHMENTS is populated when the POP3 listener associated with the [Email IN initiator module](#) receives an email message with attachments. Attachments are only saved to the data section if you have selected the **Save attachments as data section** option in the [POP3 listener screen](#).

For more information on attaching files from POP3IN_ATTACHMENTS to email messages, see [Attaching files to an email message from the POP3IN_ATTACHMENTS data section](#).

For all other output modules, see the following sections:

To include a named attachment in an XML-formatted message section:

1. Create the EMF Process, containing at least a POP3 initiator module, an XML formatter module, a recipients module (if required) and an output module. For example:



2. Double-click the **XML formatter** and add XML nodes and references to the attachments you want to include. For example:

```
<?xml version="1.0" ?>
<Root>
  <attachments>
    <file>
      <name>File1.doc</name>
```



```

<content>$GETATTACHMENT('File1.doc')$</content>
</file>
<file>
<name>MyFile.xls</name>
<content>$GETATTACHMENT('MyFile.xls')$</content>
</file>
</attachments>
</Root>

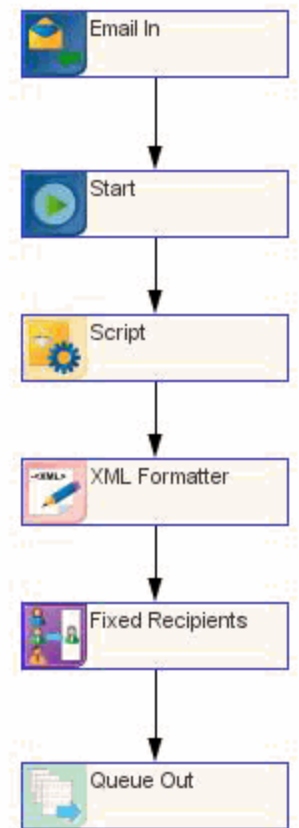
```

Note: if any of the attachments are binary, then the content will be Base64-encoded.

3. Configure the remaining modules as required, selecting the XML formatter message section in the output module.

To include all attachments in an XML-formatted message section:

1. Create the EMF Process, containing at least a POP3 initiator module, a Script module, an XML formatter module, a recipients module (if required) and an output module. For example:



2. Double-click the **Script** module and copy and paste the contents of this [pop3attachments script](#). By default, the script will write XML output with this structure:

```

<?xml version="1.0" ?>

<Root>

```

```
<Attachments>
<File></File>
<contentType></contentType>
<content></content>
</Attachments>
</Root>
```

3. Modify the script if required, for example, to specify names of nodes in your own XML schema, or to include additional information.

Note: The <Root> node is specified in the XML formatter module that follows, not in the script.

4. Double-click the **XML formatter** and add a reference to the content of the MyPOP3Attachments message section that was created in the **Script** module:

```
<?xml version="1.0" ?>
<Root>
$MESSAGE('MyPOP3Attachments')$
</Root>
```

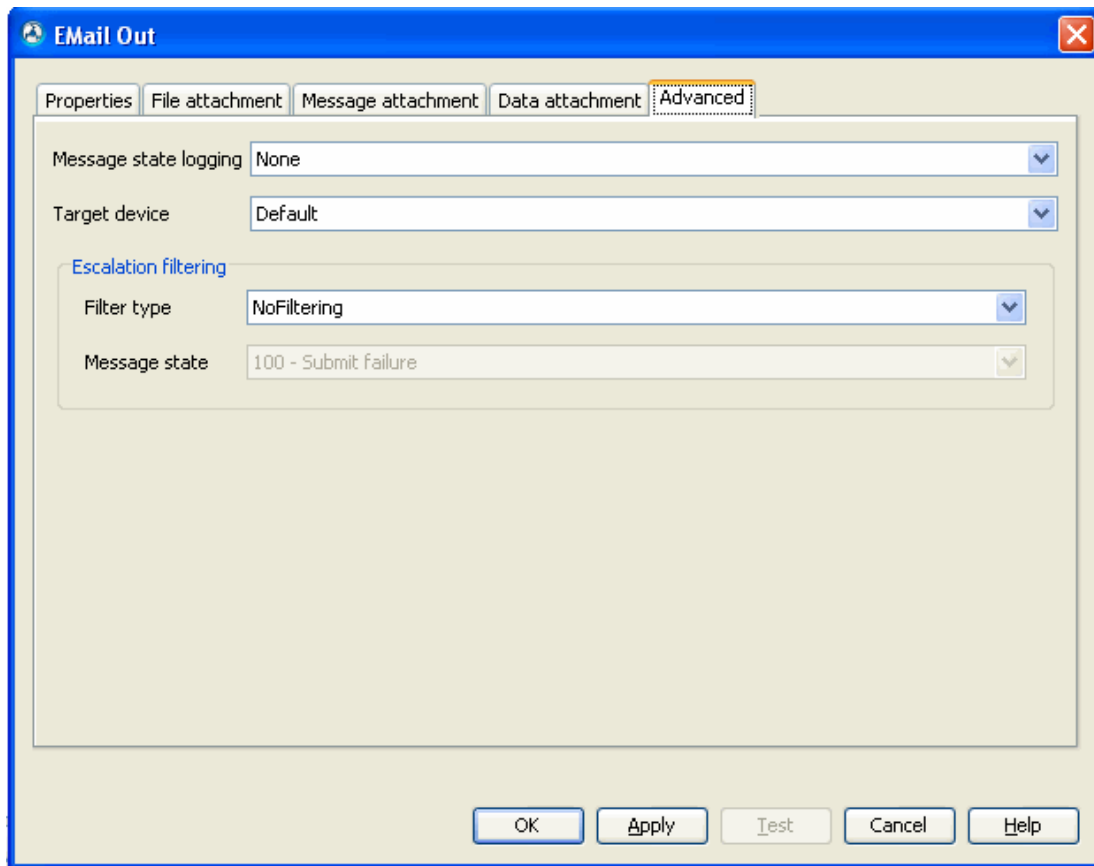
Note: If any of the attachments are binary, then the content will be Base64-encoded.

5. Configure the remaining modules as required, selecting the XML formatter message section in the output module.

[EMF Process Output Modules](#)

Advanced Settings for Output Modules

Although the EMF Process output modules have different functions and settings, they all share a common Advanced Properties page, shown below (The example shown is from the Pager output module; other modules will have a different title and may also have different or additional tabs.).



The settings on the **Advanced** tab are as follows:

- **Message state logging:** Allows you to select the level of detail for the log of messages that are sent out. If you select **Full**, all events are logged, whereas if you select **None**, no log file is created.

Important: You must set **Message State Logging** to **Full** if you want to use an [Escalation module](#).

- **Target device:** Allows you to select the devices to which the EMF process should be sent.
 - If you select **Default**, all recipients in all recipient sections will receive a message on their default device for that output module type. This enables the person building the EMF process to specify the device type that must be used to receive the message.
 - When **Selected** is chosen, the recipient (rather than the person building the EMF process) decides the device type on which they will receive the message. In this case, only those recipients who have selected a specific device for that output module type will receive a message, and they will receive it on the device they selected, rather than their default device. (For **Selected**, the EMF process includes multiple output module types so that the device type chosen by the recipient is available as an option.)

Note: You cannot select anything other than **Default** for the File Out and FTP Out modules, because recipients cannot have multiple devices of these types.

- **Escalation filtering:** Allows you to refine your escalation settings for the EMF Process. For further details, see [Message State Logging and Escalation Filtering](#).

The default setting is **No filtering**, in which case the module will not use previous Message State Logging information and all the recipients that should receive a message (based on other module settings) will receive one. If you select **No filtering**, the **Message state** field is not available.

The other options (**Greater than**, **Less than**, and **Equal to**) are used to select the recipients who will receive a message, based on Message State Logging information created by the previous Output modules.

- If you select **Equal to**, the message will only be escalated if the **Message state** is identical to that selected.
- If you select **Greater than**, the message will be escalated if the **Message state** is identical to or higher than that selected.
- If you select **Less than**, the message will be escalated if the **Message state** is identical to or lower than that selected.
- If you select **Equal to**, **Greater than**, or **Less than**, you must then select one of the following **Message states**:
 - **100 - Submit failure:** Escalation will occur if the message cannot be sent.
 - **101 - Submit success:** Escalation will occur if the message is sent successfully.
 - **200 - Accept failure:** Escalation will occur if the message was sent and delivered, but was rejected by the recipient.
 - **201 - Accept success:** Escalation will occur if the message was sent and delivered, and was accepted by the recipient.
 - **300 - Delivery failure:** Escalation will occur if the message could not be delivered.
 - **301 - Delivery success:** Escalation will occur if the message was delivered successfully.
 - **400 - Retrieval failure:** Escalation will occur if the message was delivered successfully, but a link within the message (e.g. a link to a Web page) was not followed.
 - **401 - Retrieval success:** Escalation will occur if the message was delivered successfully and a link within the message (e.g. a link to a Web page) was followed.
 - **500 - Reply failure:** Escalation will occur if the message was delivered successfully but no reply was received.
 - **501 - Reply success:** Escalation will occur if the message was delivered successfully and a reply was received.

[EMF Process Output Modules](#)

Message State Logging and Escalation Filtering

Escalation Filtering

You can use Escalation Filtering to send messages to a recipient depending on the outcome of a previous message sent to the same recipient. For example, you could use it to detect a missing reply to an email message, in which case the recipient could be sent an SMS voice message which is more likely to get an immediate response. Alternatively, escalation filtering can do the opposite and only send further messages to those recipients that *did* reply to an email (for example, when a customer requests further information about a product).

You can also use it to provide a guaranteed delivery of messages: if a message fails to be delivered to one out of ten recipients, then the message is sent again - but only to the recipient that failed to receive it. This is very flexible, and the EMF Process can be built to do whatever is required (for example, use a Delay module before the next output module to allow for periods when email servers are down. See example below).

Note: Escalation filtering requires multiple Output modules in a single EMF Process branch.

Important: To use Escalation filtering, you must enable **Message State Logging** for the Output modules.

Message State Logging

Enabling Message State Logging for an Output module causes the module to create a permanent record of the actions taken when sending messages. This log can then be used by other modules that appear later in the same branch of an EMF Process to enable the use of the [Escalation](#) module, the [Reply](#) module, and to guarantee delivery via Escalation Filtering (see above).

Each Output module type sends messages in a different way, but they all follow the same basic pattern that allows message state logging to define the semantics of five generic states (see below) to which each of them maps the actions they perform. Each message sent by an Output module is logged independently of any other as it advances through the states.

Note: Due to the performance cost of logging the information, Message State Logging is disabled by default.

The Message States

Note: Due to the generic nature of the states, not all Output modules use all states.

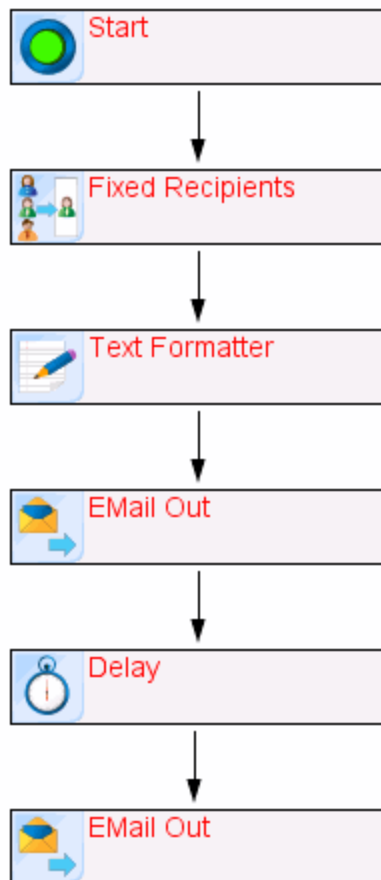
- **Submit** - EMF has forwarded the message to the server that will send the message.
- **Accept** - The message was accepted by the server that will send the message.

- **Delivered** - The message has been delivered to the recipient.
- **Retrieved** - The recipient either followed a link or downloaded further information via information contained in the message.
- **Replied** - The recipient replied to the message.

Note: For the reply message state to be reached, the message section that is sent to the recipient must contain the \$ESCALCODE\$ [dynamic function](#). Provided the message sent back to EMF includes the recipient's escalation code, EMF will be able to associate the reply with this recipient.

The reply can be sent back to EMF via any device, it is not necessary for the reply to be sent back via the device that received the message (provided a listener has been configured for the device type used).

An example of using Escalation Filtering



Here, the message will be sent to the fixed recipient by email. If it fails, for any reason, it will retry. In the example shown:

- All of the Output modules have the Target Device set to **Default**.
- The Output modules immediately after the branch have Message State Logging set to **Full**.
- The Output modules before the Delay module have Escalation Filtering set to **No Filtering**
- The Output modules after the Delay module have (for example) a filter type of **Less than** and a Message State of **200 - Accept Failure**

See also Examples of Escalation Filtering in Personalized EMF Processes.

[Advanced Settings for Output Modules](#)

[EMF Process Output Modules](#)

Edit Recipient Device

You can use the **Edit Device** screens to enter device-specific information about the various different delivery methods that you enable for each recipient.

To view the Edit Device screen for a recipient: Double-click the icon for the appropriate recipient in the **Recipient administration** section of the EMF tree view and select the **Devices** tab, and then double-click the required device.

The picture at the bottom of this Help topic shows the WAP device screen; other devices require slightly different sets of information, although the following three fields are common to all:

- **Name:** This is the information that will appear in the **Name** column on the System Recipient administration **Devices** tab.
- **Description:** This is the information that will appear in the **Description** column on the System Recipient administration **Devices** tab.
- **Delivery method:** When you add a new device to the **Devices** tab, you must specify the delivery method that it uses by selecting it from the list of those available. After you have added a device, you cannot change its delivery method, so this field cannot be changed when editing an existing device.

The following device types require specific information:

- **Email:** If you want to be able to send the recipient an EMF Process by email, enter their email address in the **Email address** field.
- **Fax:** No module in EMF currently uses this device settings.
- **HTTP:** If you want to be able to send an HTTP request to a recipient device, enter its **URL**.
- **IP:** If you want to use an EMF Process to trigger an executable program on the recipient's computer, you must enter its IP address in the **IP address** field.

Important: You must ensure that the information entered in the IP Address field has been confirmed or supplied by your network administrator.

RunExe Tip: Each recipient waiting for EMF Process information to be sent via RunExe is listening on a port defined by an [Executable Service](#) instance. If the port being used by the recipient is known, it can be added to the IP address by placing a colon followed by the port number. For example, if the IP address is 192.192.192.192 and it is listening on port 123, you should specify 192.192.192.192:123 in the IP field.

- **Pager:** No module in EMF currently uses this device settings.
 - **Queue:** If you want to be able to send an EMF Process to a queueing system, select the appropriate **Queue** and **Queue provider** from those available. If you want to specify a queue provider defined in EMF Administration System, you must also add it as an external queue provider in EMF Administration Services before you can associate it with a recipient.
 - **SMS:** If you want to be able to send the recipient an SMS, enter their SMS number in the **SMS number** field and their country code in the **SMS country code** field (leading zeroes should be removed from both numbers to ensure delivery of EMF Processes via SMS cellphone message. For example, 001 would be entered as 1 and 01372368800 would be entered as 1372368800. You must also select the **SMS provider** from the drop-down list.
-

Note: The SMS provider setting is not currently used. Hence any value set will be ignored.

- **Voice:** No module in EMF currently uses this device settings.
- **WAP:** No module in EMF currently uses this device settings.

[Recipients Administration Screen](#)

[Recipient Administration](#)

[Back to Start of Recipient Administration](#)

OPC DA Write

OPC DA (Open Process Communications Data Access) support in EMF allows the EMF server to act as an OPC client and read/write data to OPC servers. OPC is a standard typically used to allow industrial devices and control applications to communicate. For more information on OPC see <http://opcfoundation.org>. EMF supports OPC DA v2.0. OPC DA v3.0 and OPC UA (unified architecture) are not currently supported.

You can use the **OPC DA Write module** to write values to an OPC Server.

To use an OPC DA Write Module

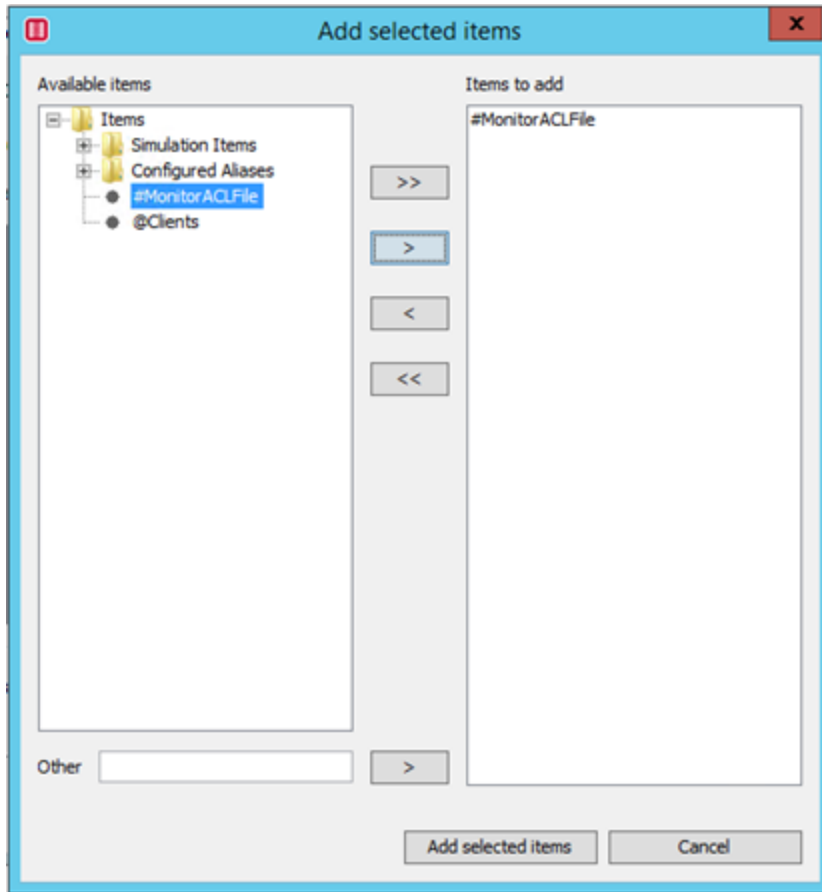
1. Drag an **OPC DA Write module** icon into the EMF Process you are creating, at the appropriate location:



2. Double-click on the module icon to display the **OPC DA Write** properties.

3. Select the OPC DA service that you need to use from the **OPC DA Service** drop-down list.
You can click **Edit** to modify the properties of the selected **OPC DA Service**.
4. Enter a **Group Name** to which the OPC items are added. Group names are unique per client connection. By default, the **Autoname** check box is selected which allows the group name to be automatically assigned.
5. In the **Selected Items** section, click **Add** to add the items that will be queried from the OPC server. This will open the **Add Selected items** window. Select the items to add from the available items and click **Add selected items**. If you do not currently have a connection to the OPC server, or an item you know exists is not displayed, or you wish to define the item dynamically at runtime using dynamic functions then you can manually enter the item in the **Other** field and add it.

You can also delete, or change the order of the items added using the **Remove**, **Remove all**, **Move up**, and **Move down** options.



6. Enter the new **Value** and **Type** for the items added in **Selected Items** section. Dynamic functions may be used to define the **Value**. If the item that you are writing to is an array, then select the **Is Array** check box. When writing an array, the value should be defined using standard CSV delimiter rules, where the " character is used as the text qualifier character. So, for example to write an array of 3 integers enter the following as the value field:

145,789,43533

To write an array of strings,

Hello World, Goodbye World, Another string

would write three values:

- Hello World
- Goodbye World
- Another string

Where as:

"Hello World, Goodbye World", Another string

would write two values:

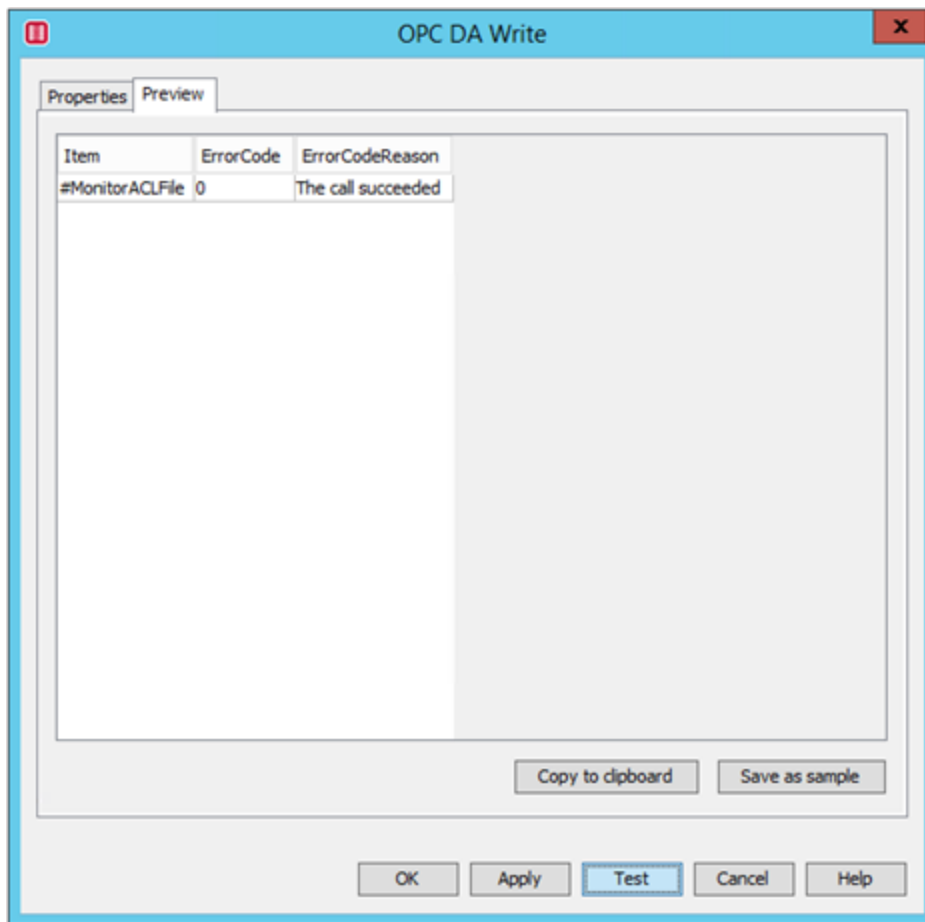
- Hello World, Goodbye World
- Another string

The exception to the rule is for an array of characters, which are always just a single character separated by a comma (even if the character to write in the array is a comma).

For values of type "Date" the value must follow the following format:

YYYY-MM-DD hh:mm:ss

7. Enter an appropriate name in the **Results Data Section** field.
8. Click **Test** to view the output data in the **Preview** tab. You can click **Copy to clipboard** button to copy the data to clipboard for future use. You can also click **Save as sample** button to save the data for testing purposes.



The data section created will contain the following columns:

- **Item**: the item (tag) name.
- **ErrorCode**: the error code reports any issues with trying to communicate with the OPC server. A successful call will return 0.
- **ErrorCodeReason**: the textual representation of the error code.

[Go to start of Output Modules](#)

File/FTP Utility Modules

EMF has the following modules that assist in performing operating system file manipulation such as copying or deleting files:

- [Copy Files Module](#): Allows files and directories to be copied. Wildcard filters can be applied.
- [Move Files Module](#): Allows files and directories to be moved. Wildcard filters can be applied. Moving files is similar to renaming files, so if the requirement is to rename a file, you can use this module.
- [FTP Move Files Module](#): Allows files/directories on an FTP server to be moved/renamed on the same server. Wildcard filters can be applied.
- [Delete Files Module](#): Allows files and directories to be deleted. Wildcard filters can be applied.
- [FTP Delete Files Module](#): Allows files/directories on an FTP server to be deleted. Wildcard filters can be applied.
- [Touch File Module](#): This is similar to the Unix "touch" command. It changes the last modified time on a file, if the file exists, or creates a new empty file if it does not exist.

Copy Files Module

This module allows files and directories to be copied in the operating system.

Note: Sufficient rights must be enabled in the operating system to copy files. Pressing the test button will cause the test to be run using the rights of the logged-in user. When the module is run through the server, the rights will be that of the user that the server is running as. On Windows, this is the user the service is running as- "Local System Account" by default.

To use the Copy Files module:

1. Drag the **Copy Files** icon onto the EMF Process Design graph at the appropriate place.



2. Double-click the **Copy Files** icon to display the **Copy Files** properties screen.

3. Select the **Copy file** radio button and perform the steps listed under **Copy file** if you want to copy a single file. Alternatively, select the **Copy directory contents** radio button and perform the steps listed under **Copy directory contents** if you want to copy a number of files or directories from one location to another.

- **Copy file**

- a. Enter or select the source name of the file to copy. If you enter a directory name as the source, rather than a file name, then the directory will be copied, but none of the contents of the directory. To copy the directory contents, use the **Copy directory contents** option.

- b. Enter or select the destination name of the file or directory that will be created.
- **Copy directory contents**
 - a. Enter or select the **From directory**. This is the directory whose contents will be copied.
 - b. Enter or select the **To directory**. This is the destination directory to which the items will be copied. If the directory does not exist, it will be created.
 - c. Select from the available **Filter** options to provide advanced control over the files that will be copied. If multiple filter options are selected, then the **AND** criteria is used to copy the directory contents (for example, the file must be older than the date/time specified in the **Files older than** field **AND** be newer than the date/time specified in the **Files newer than** field). If no filter options are specified, all files in the root of the **From directory** will be copied. No subdirectories or their contents will be copied.
 - **Files matching pattern:** If you select this option, files matching a specified pattern are copied. The pattern is only applied to files and not to directories. If the pattern is a regular expression, select the option to indicate it. If it is not a regular expression, the wildcard characters * and ? can be used to match filenames (these follow similar conventions to DOS, * matches 0 or more characters and ? matches a single character). So, to copy all files with the extension txt, enter *.txt.
 - **Files older than:** Allows a date and time to be specified, and files that were last modified before that time will be copied. The format for the date/time, must be YYYY-MM-DD HH:MM:SS.
 - **Files newer than:** Allows a date and time to be specified, and files that were last modified after that time will be copied. The format for the date/time, must be YYYY-MM-DD HH:MM:SS.
 - **Include subdirectories:** If selected, then any child directories (and any of their descendants) in the **From directory** will also be scanned for files that match the specified filter, and copied. The same directory structure in the **To directory** is maintained, so any subdirectories that need to be created will be created. For more information about whether empty directories will be re-created in the destination, see the **Create empty directories** option in **step4**.
4. Select the following additional options as appropriate:
 - **Copy Attributes:** Copies the file attributes. The exact list of attributes that are copied depends on the operating system and Java implementation.
 - **Replace existing:** Replaces existing files if they exist. If the file exists and this option is not selected, then the module will error. If the destination file is a symbolic link, then the symbolic link is replaced.
 - **Follow symbolic links:** If the source file/directory is a symbolic link, then this option indicates whether to copy the file that the link is pointing to (if

selected), or whether to copy the actual symbolic link itself (option not selected).

- **Create empty directories:** Whether, when subdirectories are copied, they are re-created even if they have no content (this may be because nothing matched the specified filter). This option is only available if **Copy directory contents** is selected. For example, if a subdirectory **dir1** contains the file "file.txt", and the filter specified to include subdirectories, but was matching a pattern *.xml, then the directory **dir1** would only be created under the destination directory if the **Create empty directories** option was selected (it would be empty as no files matched the filter).
5. The **Test** button can be used to test run the copy on the local machine (this may not be the same machine on which the server is running, so may give different results).

Move Files Module

This module allows files and directories to be moved in the operating system. This module can also be used to rename a file or directory.

Note: Sufficient rights must be enabled in the operating system to move files. Pressing the test button will cause the test to be run using the rights of the logged-in user. When the module is run through the server, the rights will be that of the user that the server is running as. On Windows, this is the user the service is running as - "Local System Account" by default.

To use the Move Files module:

1. Drag the **Move Files** icon onto the EMF Process Design graph at the appropriate place.



2. Double-click the **Move Files** icon to display the **Move Files** properties screen:

3. Select the **Move file** radio button and perform the steps listed under **Move file** if you want to move a single file. Alternatively, select the **Move directory contents** radio button and perform the steps listed under **Move directory contents** if you want to move a number of files or directories from one location to another.

- **Move file**

- a. Enter or select the source name of the file to move. If you enter a directory name as the source, rather than a file name, then the directory will be renamed. To move the directory contents, use the **Move directory contents** option.

- b. Enter or select the destination name of the file or directory that will be created.

- **Move directory contents**

- a. Enter or select the **From directory**. This is the directory whose contents will be moved.
 - b. Enter or select the **To directory**. This is the destination directory to which the items will be moved. If the directory does not exist, it will be created.
 - c. Select from the available **Filter** options to provide advanced control over the files that will be moved. If multiple filter options are selected, then the **AND** criteria is used to move the directory contents (for example, the file must be older than the date/time specified in the **Files older than** field **AND** be newer than the date/time specified in the **Files newer than** field). If no filter options are specified, all files in the root of the **From directory** will be moved. No subdirectories or their contents will be moved.
 - **Files matching pattern:** If you select this option, files matching a specified pattern are moved. The pattern is only applied to files and not to directories. If the pattern is a regular expression, select the option to indicate it. If it is not a regular expression, the wildcard characters * and ? can be used to match filenames (these follow similar conventions to DOS, * matches 0 or more characters and ? matches a single character). So to move all files with the extension txt, enter *.txt.
 - **Files older than:** Allows a date and time to be specified, and files that were last modified before that time will be moved. The format for the date/time, must be YYYY-MM-DD HH:MM:SS.
 - **Files newer than:** Allows a date and time to be specified, and files that were last modified after that time will be moved. The format for the date/time, must be YYYY-MM-DD HH:MM:SS.
 - **Include subdirectories:** If selected, then any child directories (and any of their descendants) in the From directory will also be scanned for files that match the specified filter, and moved. The same directory structure in the To directory is maintained, so any subdirectories will be created in the destination.
4. Select the following additional options as appropriate:
 - **Replace existing:** Replaces existing files if they exist. If the file exists and this option is not selected, then the module will error. If the destination file is a symbolic link, then the symbolic link is replaced.
 - **Atomic move:** Performs the move as an atomic file operation. If the file system does not support an atomic move, then the module will error. With an atomic move, you can move a file into a directory and be guaranteed that any process watching the directory accesses a complete file.
 5. The **Test** button can be used to test run the move on the local machine (this may not be the same machine on which the server is running, so may give different results).

FTP Move Files Module

This module allows files/directories on an **FTP** server to be moved/renamed on the same server.

To use the FTP Move Files module:

1. Drag the **FTP Move Files** icon onto the EMF Process Design graph at the appropriate place.



2. Double-click the **FTP Move Files** icon to display the **FTP Move Files** properties screen:

 The screenshot shows a dialog box titled "FTP Move Files" with a standard Windows-style title bar (minimize, maximize, close buttons). The dialog has a light gray background. At the top, there is a label "FTP service" followed by a red asterisk, a dropdown menu, and an "Edit" button. Below this, there are two radio buttons: "Move file/directory" (which is selected) and "Move directory contents". Under "Move file/directory", there are two text input fields labeled "Source name" and "Destination name", each followed by a red asterisk. Under "Move directory contents", there are two text input fields labeled "From directory" and "To directory". Below these is a section titled "Filter" which contains several checkboxes: "Files matching pattern" (with a sub-checkbox "Pattern is a regular expression" and a "Pattern" text field), "Files older than" (with a "Date" text field and a calendar icon), "Files newer than" (with a "Date" text field and a calendar icon), and "Include subdirectories". At the bottom of the dialog, there is a checkbox labeled "Overwrite existing files". At the very bottom, there are five buttons: "OK", "Apply", "Test", "Cancel", and "Help".

3. Select the FTP service that you wish to use from the [FTP service](#) drop-down list. You can click **Edit** to modify the properties of the selected [FTP service](#).

4. Select the **Move file/directory** radio button and perform the steps listed under **Move file/directory** if you want to move a single file or directory. Alternatively, select the **Move directory contents** radio button and perform the steps listed under **Move directory contents** if you want to move a number of files or directories from one location to another.
 - **Move file/directory**
 - a. Enter the source name of the file/directory to move. If you enter a directory name as the source, rather than a file name, then the directory will be renamed. To move the directory contents, use the **Move directory contents** option.
 - b. Enter the destination name of the file or directory that will be created.
 - **Move directory contents**
 - a. Enter the **From directory**. This is the directory whose contents will be moved.
 - b. Enter the **To directory**. This is the destination directory to which the items will be moved. If the directory does not exist, it will be created.
 - c. Select from the available **Filter** options to provide advanced control over the files that will be moved. If multiple filter options are selected, then the **AND** criteria is used to move the directory contents (for example, the file must be older than the date/time specified in the **Files older than** field **AND** be newer than the date/time specified in the **Files newer than** field). If no filter options are specified, all files in the root of the **From directory** will be moved. No subdirectories or their contents will be moved.
 - **Files matching pattern**: If you select this option, files matching a specified pattern are moved. The pattern is only applied to files and not to directories. If the pattern is a regular expression, select the option to indicate it. If it is not a regular expression, the wildcard characters * and ? can be used to match filenames (these follow similar conventions to DOS, * matches 0 or more characters and ? matches a single character). So to move all files with the extension txt, enter *.txt.
 - **Files older than**: Allows a date and time to be specified, and files that were last modified before that time will be moved. The format for the date/time, must be YYYY-MM-DD HH:MM:SS.
 - **Files newer than**: Allows a date and time to be specified, and files that were last modified after that time will be moved. The format for the date/time, must be YYYY-MM-DD HH:MM:SS.
 - **Include subdirectories**: If selected, then any child directories (and any of their descendants) in the From directory will also be scanned for files that match the specified filter, and moved. The same directory structure in the **To directory** is maintained, so any subdirectories will be created in the destination. Any source subdirectories that are empty will be moved to the new destination. Also, after matching files have been moved, if the source

subdirectory is empty then, it will be removed from the source file structure.

5. The **Test** button can be used to test run the move on the **FTP** server.
6. Select **Overwrite existing files** option to replace existing files.

Delete Files Module

This module allows files and directories to be deleted from the operating system.

Note: Sufficient rights must be enabled in the operating system to delete files. Pressing the test button will cause the test to be run using the rights of the logged in user. When the module is run through the server, the rights will be that of the user that the server is running as. On Windows, this is the user the service is running as - "Local System Account" by default.

To use the Delete Files module:

1. Drag the **Delete Files** icon onto the EMF Process Design graph at the appropriate place.



2. Double-click the **Delete Files** icon to display the **Delete Files** properties screen.

3. Select the **Delete file/directory** radio button and perform the steps listed under **Delete file/directory** if you want to delete a single file. Alternatively, select the **Delete directory contents** radio button and perform the steps listed under **Delete directory contents** if you want to delete a number of files in a directory structure.
 - **Delete file/directory**
 - a. Enter or select the Name of the file to delete. If you enter a directory name as the source, rather than a file name, then the whole directory structure will be removed.
 - **Delete directory contents**
 - a. Enter or select the Directory - this is the directory whose contents will be deleted.
 - b. Select from the available **Filter** options to provide advanced control over the files that will be deleted. If multiple filter options are selected, then the **AND** criteria is used to delete the directory contents (for example, the file must be older than the date/time specified in the **Files older than**

field **AND** be newer than the date/time specified in the **Files newer than** field). If no filter options are specified, all files in the root of the **From directory** will be deleted. No subdirectories or their contents will be deleted.

- **Files matching pattern:** If you select this option, files matching a specified pattern are deleted. The pattern is only applied to files and not to directories. If the pattern is a regular expression, select the option to indicate it. If it is not a regular expression, the wildcard characters * and ? can be used to match filenames (these follow similar conventions to DOS, * matches 0 or more characters and ? matches a single character). So, to delete all files with the extension txt, enter *.txt.
 - **Files older than:** Allows a date and time to be specified, and files that were last modified before that time will be deleted. The format for the date/time, must be YYYY-MM-DD HH:MM:SS.
 - **Files newer than:** Allows a date and time to be specified, and files that were last modified after that time will be deleted. The format for the date/time, must be YYYY-MM-DD HH:MM:SS.
 - **Include subdirectories:** If you select this option, any child directories (and any of their descendants) will also be scanned for files that match the specified filter, and deleted. Any empty subdirectories will also be deleted.
4. The **Test** button can be used to test run the delete on the local machine (this may not be the same machine on which the server is running, so may give different results).

FTP Delete Files Module

This module allows files/directories on an **FTP** server to be deleted.

To use the FTP Delete Files module:

1. Drag the **FTP Delete Files** icon onto the EMF Process Design graph at the appropriate place.



2. Double-click the **FTP Delete Files** icon to display the **FTP Delete Files** properties screen.

3. Select the FTP service that you wish to use from the [FTP service](#) drop-down list. You can click **Edit** to modify the properties of the selected [FTP service](#).
4. Select the **Delete file/directory** radio button and perform the steps listed under **Delete file/directory** if you want to delete a single file or directory. Alternatively, select the **Delete directory contents** radio button and perform the steps listed under **Delete directory contents** if you want to delete a number of files in a directory structure.
 - **Delete file/directory**
 - a. Enter the **Name** of the file/directory to delete. If you enter a directory name as the source, rather than a file name, then the whole directory structure will be removed.
 - **Delete directory contents**
 - a. Enter the **Directory** - this is the directory whose contents will be deleted.
 - b. Select from the available **Filter** options to provide advanced control over the files that will be deleted. If multiple filter options are selected, then the **AND** criteria is used to delete the directory contents (for example, the file must be older than the date/time specified in the **Files older than** field **AND** be newer than the date/time specified in the **Files newer than** field). If no filter options are specified, all files in the root of the **From directory** will be deleted. No subdirectories or their contents will be deleted.
 - **Files matching pattern:** If you select this option, files matching a specified pattern are deleted. The pattern is only applied to files

and not to directories. If the pattern is a regular expression, select the option to indicate it. If it is not a regular expression, the wildcard characters * and ? can be used to match filenames (these follow similar conventions to DOS, * matches 0 or more characters and ? matches a single character). So, to delete all files with the extension txt, enter ***.txt**.

- **Files older than:** Allows a date and time to be specified, and files that were last modified before that time will be deleted. The format for the date/time, must be YYYY-MM-DD HH:MM:SS.
- **Files newer than:** Allows a date and time to be specified, and files that were last modified after that time will be deleted. The format for the date/time, must be YYYY-MM-DD HH:MM:SS.
- **Include subdirectories:** If you select this option, any child directories (and any of their descendants) will also be scanned for files that match the specified filter, and deleted. Any empty subdirectories will also be deleted.

5. The **Test** button can be used to test run the delete on the **FTP** server.

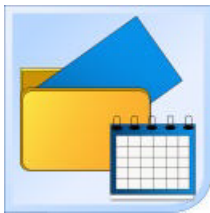
Touch File Module

This module is similar to the Unix “touch” command. It changes the last modified time on a file, if the file exists or creates a new empty file if it does not exist.

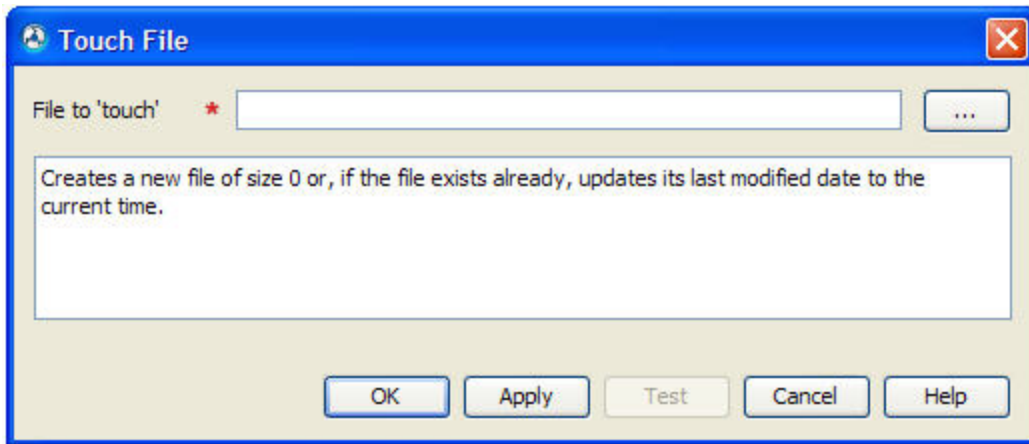
Note: Sufficient rights must be enabled in the operating system to create/modify files. Pressing the test button will cause the test to be run using the rights of the logged-in user. When the module is run through the server, the rights will be that of the user that the server is running as. On Windows, this is the user the service is running as (“Local System Account” by default).

To use the Touch File module:

1. Drag the **Touch File** icon onto the EMF Process Design graph at the appropriate place.



2. Double-click the **Touch File** icon to display the **Touch File** properties screen:



3. Select or enter the name of the **File to 'touch'**. This is the file that will be created if it does not exist. If it already exists, its modified date will be updated to the current time.

Note: If the file already exists, and it is set to read-only, the modified time can still be adjusted on some operating systems.

4. The **Test** button can be used to test run the touch module on the local machine (This may not be the same machine on which the server is running, so it may give different results.).

Audit Modules

EMF has modules that assist in the automation of internal and external audit processes. They are:

- [Benford's Law Module](#) - This module allows you to analyse a set of numbers and verify if their distribution complies with Benford's law and other expected distributions. It can be used to detect error and fraud in sets of numbers, and where a large amount of estimation has been used. This module can be useful in highlighting suspicious data, in potentially fraudulent transactions, that should be investigated further.
- [Audit Reporting Module](#) - The audit reporting module is used to log audit events to the EMF GRC (Governance, Risk and Compliance) Workspace. For example, an audit event may be a high transaction value on an account. Using other EMF modules, it is possible to detect this event. The purpose of the audit reporting module is to record this event in the GRC Workspace so that an auditor can verify if the transaction is legitimate.

Benford's Law Module

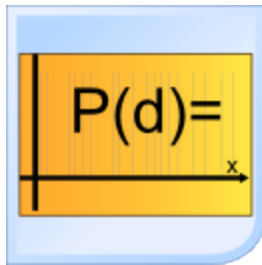
This module allows you to analyze a set of numbers to see if their distribution complies with Benford's law and other expected distributions.

What is Benford's Law? Wikipedia describes it as: "*Benford's law*, also called the first-digit law, states that in lists of numbers from many real-life sources of data, the leading digit is 1 almost one-third of the time, and larger numbers occur as the leading digit with less and less frequency as they grow in magnitude, to the point that 9 is the first digit less than one time in twenty. This counter-intuitive result applies to a wide variety of figures, including electricity bills, street addresses, stock prices, population numbers, death rates, lengths of rivers, physical and mathematical constants, and processes described by power laws (which are very common in nature)."

To analyze a set of numbers, a data section containing the necessary columns is necessary (for example, invoice amounts). After analyzing the data, the module will then generate two new data sections. One data section will contain the expected and actual occurrences found using the selected technique. The other data section will contain a number of statistical measures of how well the actual data fits the expected data. These data sections can then be used to decide whether further investigation is necessary.

To use Benford's Law module:

1. Drag the **Benford's Law** icon onto the EMF Process Design graph at the appropriate place.



2. Open the Benford's Law module to display its properties. By default, you have to double-click on the icon to open but you can configure mouse-click actions on the Options window. From the **Tools** menu, select **Options** -> **EMF**-> **Process Builder Behaviour** tab.

3. Select the data section that contains the data to be analyzed.
4. The **Column to analyse** section contains the data that is to be analyzed. The column should contain numerical values. Any non-numerical/blank values will be ignored during analysis. The column to analyze can either be selected via its name or index. Select the appropriate radio button and then enter either the column index or column label.

Note that the column numbering starts from zero, so for example the third column would be number 2.

5. Select the **digits to analyse** from the drop down list. This defines the type of analysis that will be performed.
 - **First Digit** counts the occurrences of the digit that each item of data analyzed starts with. Leading zeros are removed from the data.
 - **First 2 Digits** counts the occurrences of the first two digits that each item of data analyzed starts with. Leading zeros are removed from the data. This gives a more detailed idea of the location of the data for abnormalities as it generates easier to spot "spikes" in the results. Generally this test is used when the sample size is less than 10,000, if the sample size is greater than use first 3 digits.
 - **First 3 Digits** counts the occurrences of the first three digits that each item of data analyzed starts with. Leading zeros are removed from the data. Generally

this test is used when the sample size is greater than 10,000. If the sample size is less than this then use the first 2 digits.

- **Last Digit** counts the occurrences of the last digit that each item of data analyzed ends with. This test is not part of Benford's law but is useful in helping spot data that may have been estimated/fraudulently created. Each digit is expected to be evenly distributed.
 - **Last 2 Digits** counts the occurrences of the last two digits that each item of data analyzed ends with. Like the **Last Digit** test, this test is helpful in spotting data that may have been estimated/fraudulently created. Each pair of digits is expected to be evenly distributed.
 - **Second Digit** counts the occurrences of the second digit of each item of data analyzed. Leading zeros are removed from the data. This test gives an overview of how well the data fits Benford's law.
6. Enter the name of the data section that will be created in the **Results data** field that will contain the frequency distribution results.
- The columns in the returned data section are as follows:

Column Name	Description
Digits	The digit(s) that this row applies to.
ExpectedFreq	The expected frequency of the digit(s) in the data as a percentage.
ActualFreq	The actual counted frequency of the digit(s) in the data as a percentage of the number of data items successfully analyzed.
PercentageDifference	The percentage difference between the actual and expected frequency for that digit(s).
ExpectedOccurrences	The expected number of occurrences of that digit(s) that should exist in the data.
ActualOccurrences	The actual number of occurrences of that digit(s) that existed in the data.
OccurrencesDifferences	The difference between the actual and expected number of occurrences.

7. Enter the name of the data section that will be created in the **Results data summary** field that will contain a summary of the results. The summary data section contains information about how well the actual results matched the expected results. More information on Benford's law and interpreting the data can be found in the paper "Detecting Problems in Survey Data using Benford's Law" at

<http://www.aae.wisc.edu/schechter/benford.pdf>

The columns in the summary data section are as follows:

Column Name	Description
Samples	The number of data samples successfully analyzed. Not numeric values and empty cells are ignored. The value zero is ignored in some tests (those that analyse the first digit(s))
Correlation	The Pearson correlation coefficient.
ChiSquare	The Chi-square distribution.
KuiperTest	Kuiper's modified Kolmogorov-Smirnov goodness of fit test (VN), which is less sensitive to sample size and recognizes the circularity of the data

8. Click **Test** to preview the summary data and the results data.

[Audit Modules](#)

Audit Reporting Module

The audit reporting module is used to log audit events to the EMF GRC (Governance, Risk and Compliance) Workspace. Audit events can be categorized into one of three criteria- "failure", "warning" and "pass". It is up to the person constructing the EMF process to decide which category an audit event should fall in to.

A single audit reporting module is used to log events of a certain type e.g. High transaction value, vendor invoices paid after due date, manually overwritten prices in sales orders. Each event type may have individual events of the 3 criteria (failure, warnings, passes), or may only log events of 1 or 2 of the categories. For example, if you are monitoring high transaction events then you may log a high transaction as a failure or warning (warnings and failures are treated the same, it is purely a visual thing in how they are presented in the GRC Workspace). No "pass" events may want to be recorded as this may cause the recording of much information that will never be looked at. However, another business may want all transactions that are below the high transaction account value to be recorded as passes, to have a full audit trail to demonstrate that all transactions have been audited.

The audit report module needs the following data:

1. **Audit rule name** – The name of the test that is being performed. This is a fixed field that does not take dynamic functions. It allows the end user in the GRC workspace an easy way to filter for the results of all tests of this type.

2. **Rule description message section** – This is a text section containing an overview of a particular run of a test. The text section may contain dynamic functions. For example, you may want to record how many records were checked during a test, or any variable parameters that were used during the running of the test.
3. **Failures data section (optional)** – If failures are to be recorded, then this is a data section that contains one row for each failure that is to be recorded.
4. **Failures message section (optional)** – If failures are to be recorded, then this is a message section that contains the individual event text that should be recorded for each row in the failures data section. This message section typically contains the \$AUDITDATA\$ dynamic function that is used to write out the contents of the current row of data in the data section that is currently being processed. This message section can contain HTML mark-up so the formatting of the text displayed within the GRC Workspace can be controlled. Any data in the \$AUDITDATA\$ dynamic functions is automatically escaped for correct display in a Webpage.
5. **Warnings data section (optional)** – If warnings are to be recorded, then this is a data section that contains one row for each warning that is to be recorded.
6. **Warnings message section (optional)** – If warnings are to be recorded, then this is a message section that contains the individual event text that should be recorded for each row in the warnings data section. The same rules apply as those described above in the “Failures message section”.
7. **Passes data section (optional)** – If passes are to be recorded, then this is a data section that contains one row for each pass that is to be recorded.
8. **Passes message section (optional)** – If passes are to be recorded, then this is a message section that contains the individual event text that should be recorded for each row in the passes data section. The same rules apply as those described above in the “Failures message section”.

To use the Audit Reporting module:

1. Drag the **Audit Reporting** icon onto the EMF Process Design graph at the appropriate place.



2. Open the Audit Reporting module to display its properties. By default, you have to double-click on the icon to open but you can configure mouse-click actions on the Options window. From the **Tools** menu, select **Options -> EMF-> Process Builder Behaviour** tab.

3. Enter the **Audit rule name**. This is a fixed field that does not take dynamic functions. It allows the end user in the GRC workspace an easy way to filter for the results of all tests of this type.
4. Select a **Rule description message section**. This is a text section containing an overview of a particular run of a test. The text section may contain dynamic functions. For example, you may want to record how many records here checked during a test, or any variable parameters that were used during the running of the test.
5. Select the type of events that will be recorded by selecting the appropriate check boxes under **Enable reporting of**. When a check box is selected, the corresponding tab is enabled, allowing details to be entered.

[Failures/Warnings/Passes tab](#)

[Building your first audit rule](#)

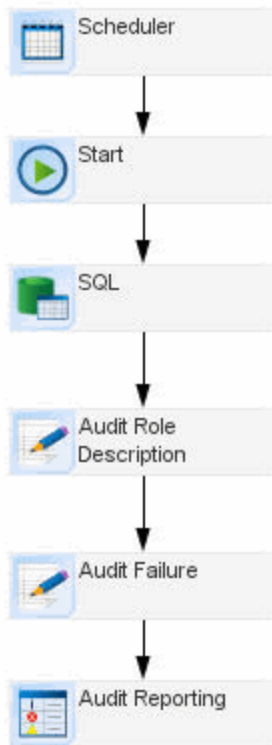
[Audit Modules](#)

Audit Reporting - Building your first audit rule

This page gives instructions to build an audit rule to display results in the EMF GRC Workspace.

This sample rule will notify users of all EMF processes that are on hold.

1. Create a new EMF Process, called "My First Audit Rule".
2. Place the modules shown below on the EMF Process.



3. In the SQL module, select the "Repository" connection and add the following query - this brings back all the EMF Processes that are on hold.

```
SELECT AlertProfile.AlertProfileID, AlertProfile.FolderID, AlertProfile.Name,
AlertProfile.Description, AlertProfile.APIName, AlertProfile.Status FROM AlertProfile
WHERE Status = 'H'
```

4. Name the the first text formatter as Audit rule description. This will be a description of the rule that the auditor will see. It can take dynamic functions, so the text can change at runtime.
\$ROWCOUNT('SQL')\$ processes are reported as being on hold.
5. Name the second text formatter as Failure message. This is the information needed by the auditor to understand what data has caused the audit rule to fail. *AlertProfileID*
"\$AUDITDATA('AlertProfileID')\$", Name "\$AUDITDATA('Name')\$"

6. In the audit report module, configure the settings as shown below.

The screenshot shows the 'Audit Reporting' dialog box with the 'Properties' tab selected. The 'Audit rule name' is set to 'Alerts that are on hold' and the 'Rule description message section' is set to 'Audit rule description'. Under the 'Enable reporting of' section, the 'Failures' checkbox is checked, while 'Warnings' and 'Passes' are unchecked. The dialog box has 'OK', 'Apply', 'Cancel', and 'Help' buttons at the bottom.

Audit Reporting

Properties Failures Warnings Passes

Audit rule name: Alerts that are on hold

Rule description message section: Audit rule description

Enable reporting of

- ☒ Failures
- ☐ Warnings
- ☐ Passes

OK Apply Cancel Help

The screenshot shows the 'Audit Reporting' dialog box with the 'Failures' tab selected. The 'Messages Section for per item failure message' is set to 'Failure message'. The 'Data section containing failures' is set to 'SQL'. Under 'Key columns in data section', 'Specify by name' is selected. A list box contains 'AlertProfileID'. To the right of the list box are buttons for 'Add', 'Remove', and 'Remove All'. At the bottom are 'OK', 'Apply', 'Cancel', and 'Help' buttons.

7. Run the EMF Process.
8. Login to the GRC Workspace, <http://localhost:8080/GRCWorkspace>, and you should see the results presented below.

Audit Summary Results (1 - 1 of 1)							
Action	Date Run	Date Modified	Test Name	Description	Failures	Warnings	Passes
Detail	27-Feb-2007 11:33:08	27-Feb-2007 11:33:08	Alerts that are on hold	0 alert(s) are reported as being on hold	0 (0) (0) (0)	0 (0) (0) (0)	0

9. If you had no EMF Processes that were on hold, then there probably won't be much to view. Create a couple of new EMF Processes and leave them "on hold". Run "My First Audit Rule" again and then press the "Refresh/Apply Filter" button in the GRC Workspace summary screen.

Audit Summary Results (1 - 2 of 2)							
<div> <div>↑↓</div> <div>✕</div> </div>							
Action	Date Run	Date Modified	Test Name	Description	Failures	Warnings	Passes
Detail	27-Feb-2007 11:42:34	27-Feb-2007 11:42:34	Alerts that are on hold	2 alert(s) are reported as being on hold	2 (2) (0) (0)	0 (0) (0) (0)	0
Detail	27-Feb-2007 11:33:08	27-Feb-2007 11:33:08	Alerts that are on hold	0 alert(s) are reported as being on hold	0 (0) (0) (0)	0 (0) (0) (0)	0

- Click the "Detail" button for the last run. Here you can view the details of the data that broke the rule.

Alerts that are on hold

	Failures	Warnings	Passes
New	2	0	
Investigating	0	0	
Resolved	0	0	
Total	2	0	0

Refresh Summary results Full History

Failures

Status	Count
New	2
Investigating	0
Resolved	0

[illegible]

11. Clicking on “Edit” for a particular item allows the auditor to enter information about the failure, and any corrective action taken. All failures and warnings can be marked as being in one of 4 states:
 - New - it’s a new failure/warning and no action has been taken.
 - Being investigated - someone is looking at the problem.
 - Corrected, done - the problem was fixed.
 - No action needed, done - there is no need to fix the problem.

When next to notify: ☒ Leave as is
☐ Continue to flag
☐ Do not flag again ever
☐ Do not flag again for specified time

Time not to flag for

Status

Notes

☐ Re-assign to Username

This simple sample, has only demonstrated logging of failures. The audit reporting module can also log data that should be marked as passed, and data that should be marked as a warning. E.g. The example alert could be modified to also query all “active” alerts and report them as passess, and the “incomplete” alerts and report them as “warnings”.

Every time you run “My First Audit Rule”, you’ll be notified of all failures. If the required behavior was to only notify once, then place a data filter module after the SQL module.

[Failures/Warnings/Passes tab](#)

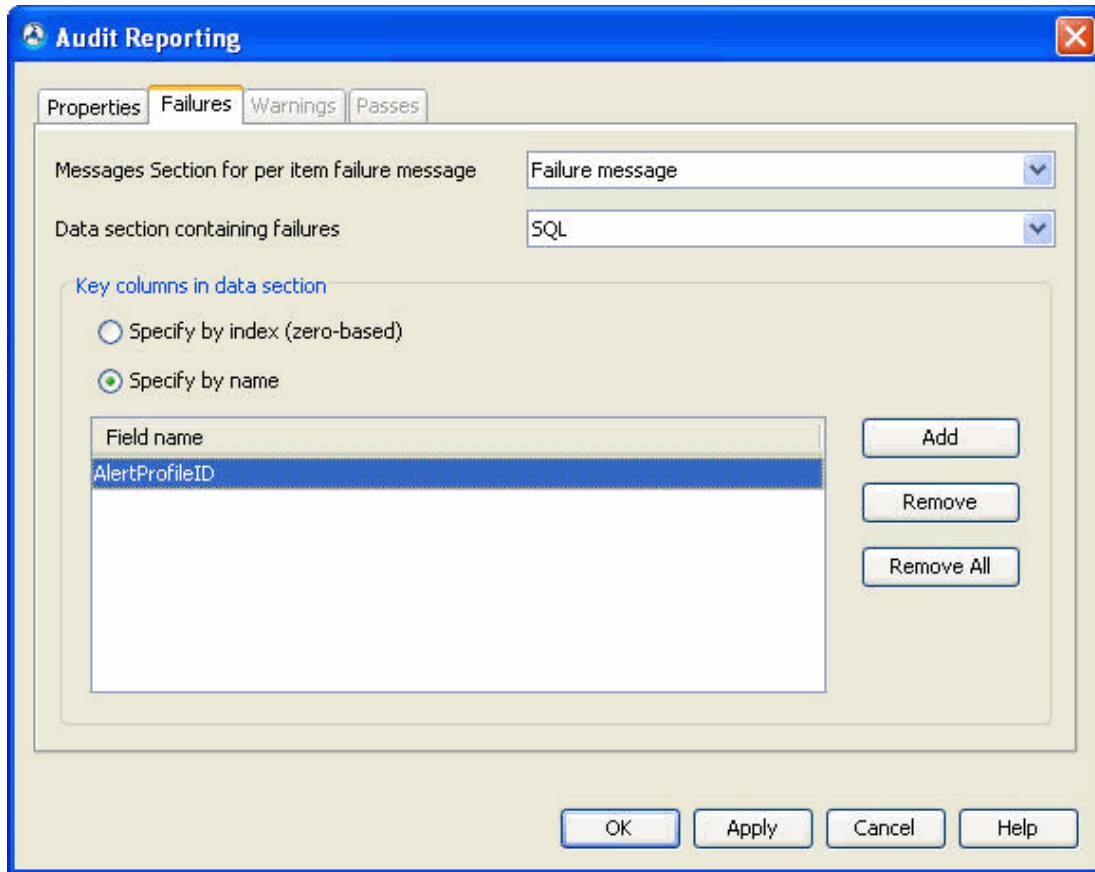
[Audit Reporting Module](#)

[Audit Modules](#)

Failures, Warnings and Passes tabs

The Failures, Warnings and Passes tabs are all identical, so are described here as one.

On each of these tabs, you configure the data that you want recorded for a particular run of a test.



1. For the **Message section for per item Failure/Warning/Pass message** select a message section that will contain the text to be display for each individual item that will be reported in a particular run of a test. i.e. for each row contained in the data section. Use the \$AUDITDATA\$ dynamic function within that message section to display the contents of a particular field in the row of data currently being processed. The same message section can be reused on the failures, warnings and passes tab, if appropriate.
2. In the **Data section containing Failures/Warnings/Passes** select the data section that will contain one row of data for each item that that is to be recorded on that run of the test. The data sections selected on the 3 tabs should be different, as it would make no sense to record the same data as a failure and a warning.
3. The **key columns** section needs to be configured if you wish to use the "When next to notify" functionality within the GRC Workspace.

When next to notify: ☒ Leave as is
☐ Continue to flag
☐ Do not flag again ever
☐ Do not flag again for specified time

Time not to flag for:

The "When next to notify" functionality allows a specific audit record to be marked by the auditor in the GRC Workspace so that it won't appear again either ever or for a period of time.

For EMF to identify which record is being marked, and to allow it to not record an audit event for the specified record - the record needs to be uniquely identifiable. To do this, the columns that will unique identify the row need to be added to the **key columns in data section**. Columns can be specified either by index (zero based - so the first column in the data section is column zero, the second column one and so on) - or by the column name.

The columns to add are typically the primary key columns for the data in the data section.

[Audit Reporting Module](#)

[Building your first audit rule](#)

[Audit Modules](#)

Routers Modules

The following topics describe the routers modules that decide which EMF Processes should be fired when a specific Listener server instance receives a message. The router modules available in EMF are:

- [Email Router Module](#)
- [SNMP Router Module](#)
- [HTTP Router Module](#)
- [Queue Router Module](#)

Email Router Module

The **Email Router** module decides which EMF Processes should be fired when an email message is received by a **POP3 Listener** instance or an **IMAP Listener** instance. It is included in the **POP3SystemAlert** System EMF Process, and should be added to any process that you create yourself, to use instead of POP3SystemAlert.

It does not contain any properties as these are defined in the POP3/IMAP Listener server instances that use the appropriate [System EMF Process](#).

The [POP3/IMAP Listener](#) waits for a message to be received in the specified mailbox. When a message is received the information is sent to the **Email Router**, the router then checks all your EMF Processes, and finds those that contain a POP3Emailinitiator that uses the same email listener that received the message. These are then fired and passed through the email message with the following data sections:

- **POP3IN** - a message section containing the text block of the email message received. This can be used by the [MESSAGE](#) Dynamic Function and Output Modules.

- **POP3IN_SD** - a Data section containing sender information from the header of a message routed by the POP3/IMAP Listener, i.e. **From**, **Subject**, **Email address**, **"Reply to" name**, **"Reply to" email address**, and **Date sent**. This can be used by the [DATA](#), [DATAFORMAT](#), [TABLEHEADER](#), [TABLE](#), [HTMLTABLE](#) and [ROWCOUNT](#) Dynamic Functions.
- **POP3IN_RD** - a Data section containing recipient information from the header of a message routed by the POP3/IMAP Listener. There will be one row for every recipient that the message was sent to, i.e. **Recipient's name**, **Recipient's email address**, **Date received**, **ID** of the POP3/IMAP Listener routing the message, and the Type of recipient (0 = TO, 1 = CC'ed, 2 = BCC'ed). This can be used by the [DATA](#), [DATAFORMAT](#), [TABLEHEADER](#), [TABLE](#), [HTMLTABLE](#) and [ROWCOUNT](#) Dynamic Functions.
- **POP3IN_ATTACHMENTS** - a Data section that contains attachments to emails.

Important: if you want to save email attachments into Data sections you must enable the "Save attachments as data section" option on the [POP3/IMAP listener Advanced Properties](#) page.

The [POP3IN_ATTACHMENTS](#) section has the following format:

Field name	Type	Size	Nullable
FileName	VarChar	255	No
	ar		
ContentType	VarChar	255	No
	pe ar		
Content	Binary	Variable	No
		le	

Note: If the content-type header is of type 'text/*', the content is transformed into UCS2 before persisting.

The [POP3IN_SD](#) and [POP3IN_RD](#) sections can be used with the Dynamic Recipient modules to send an email back to the original sender of the email and the other people the message was originally sent to. This is useful to build automated response mechanisms to emails.

[Email In Module](#)

[POP3 Listener](#)

[IMAP Listeners](#)

[Go to start of Initiator Modules](#)

SNMP Router Module

The **SNMP Router** module decides which EMF Processes should be fired when a queue message is received by an SNMP Listener server instance. It is included in the **SNMPSystemXalert** System EMF Process, and should be added to any process that you create to use instead of SNMPSystemXalert.

It does not contain any properties as these are defined in the **SNMP Listener** Server instances that use the appropriate [System EMF Process](#).

The [SNMP listener](#) waits for a message to be received in the specified queue and when a message is received, the information is sent to the **SNMP Router**. At this point, all EMF Processes are checked by the router to see if they contain an SNMP listener initiator module. On finding these, the router fires the EMF Processes and passes them the appropriate information.

Two data sections are passed to the EMF Process, and these can be manipulated by any dynamic functions that can take data sections. They are:

- **SNMP_DATA** - This data section contains details of the SNMP message, for example, request type.
- **SNMP_OIDS** - This data section contains the details of each object in the SNMP message's object list. These are the objects that the request is referencing.

For more information on these data sections, see [SNMP Listener Data Section Fields](#).

[SNMP Listener Module](#)

[Dynamic Functions](#)

[Back to Start of Initiator Modules](#)

HTTP Router Module

The **HTTP Router** module decides the EMF process to be fired when an HTTP Listener server instance receives an HTTP message.

It must be included in a [System EMF Process](#) that has been selected as the EMF Process to be fired when a request is made to the [HTTP Listener](#) (this is set up using the **Properties** tab of the [HTTP Listener Server](#)).

It does not contain any properties as these are defined in the **HTTP Listener** Server instances that use the appropriate system EMF Process.

The [HTTP Listener](#) waits for a request from an HTTP client and when a message is received, it creates data/message sections and passes the information to the **HTTP Router**. At this point, all EMF Processes are checked by the router to see if they contain the HTTP Listener (server instance) in the selected HTTP In module. On finding these, the router fires the EMF Processes and passes them the HTML message and the relevant data sections as defined by you in the HTTP Listener. For more information on the message and data sections, see [HTTP Listener Server](#).

[The HTTP In Module](#)

[Initiator Modules](#)

Queue Router Module

The **Queue Router** module decides which EMF Processes should be fired when a queue message is received by a Queue Listener server instance. It is included in the **QueueInSystemAlert** System EMF Process, and should be added to any that you create yourself to use instead of QueueInSystemAlert.

It does not contain any properties as these are defined in the **Queue Listener** Server instances that use the appropriate [System EMF Process](#).

The [Queue Listener](#) waits for a message to be received in the specified queue and when a message is received, the information is sent to the Queue Router. At this point, all EMF Processes are checked by the router to see if they contain a Queue Listener initiator. On finding these, the router fires the EMF Processes and passes them the appropriate information.

Queue messages can be in the Java Message Service (JMS) Text or Map Message format. A JMS text message contains a single string. This can include XML, if required. A JMS Map Message contains a list of Name/Value pairs. The names are in string format and the values are Java primitive types, for example Long or Boolean.

- **QUEUEIN** - This message section contains the text of the queue message (Text Messages only). This can be used by the MESSAGE dynamic function and output modules.
- **QUEUEIN_HEADERDATA** - This data section specifies the JMS header information (Both Map and Text Messages). For more information on the headers fields, see [QUEUEIN_HEADERDATA Fields](#).
- **QUEUEIN_DATA** - This data section contains the fields Name and Value (Map Messages only).

For a **Map** Message, both data sections are populated and there is no message section.

For a **Text** message, only QUEUEIN_HEADERDATA is populated. There is no second data section, while the message section contains the text of the Queue.

[Explanation of Queue Listener Modules](#)

[Overview of the Java Message Service](#)

[Go to start of Initiator Modules](#)

QUEUEIN_HEADERDATA Fields

For both JMS Text and Map messages, the Queue Router passes the QUEUEIN_HEADERDATA data section to the EMF Process.

Most of the fields in this data section correspond to the Java Message Service specification message header fields.

The following table gives the name and description for each QUEUEIN_HEADERDATA data section field.

Field Name	Description
QProviderName	The name of the queue provider to which the Queue listener listens
QueueName	The name of the queue to which the Queue listener listens
ListenerName	The name of the Queue listener server instance
Priority	The priority of the message received
ExpiryDate	The expiry date of the message received
UniqueIdentifier	The unique identifier of the message received
TimeStamp	The header field that contains the time a message was passed to a provider to be sent
DeliveryMode	<p>The delivery mode for the message. DeliveryMode can be set to PERSISTENT or NON_PERSISTENT by a JMS provider. The numeric values assigned to these constants vary from provider to provider.</p> <p>A queue message marked as persistent must be delivered once and only once. The message must be logged to stable storage so that if a provider fails, the message can be resent (however, a hardware failure may still cause the message to be lost).</p> <p>A queue message marked as non-persistent must be delivered at most once. It does not have to be logged to stable storage, so if it is lost en route to its destination, it will not be resent.</p>
CorrelationId	The correlation identifier for the message. This is used as a header field to link one message with another. A typical use is to link a response message with its request message.
ReplyQueueName	The reply queue name where the message should be sent.
ReplyQueueManager	The reply queue manager where the message should be sent.
Redelivered	This indicates whether the message is being redelivered.
ServerInstanceID	This is the ID of the queue listener server instance.

[Queue Router Icon](#)

[Queue Listener Module](#)

[Back to Start of Initiator Modules](#)

Event Plan Modules

The following modules are available to allow Events to be created and processed:

- [Publish Event module](#) – allows events to be created and published in the system.
- [Event Plan Initiator module](#) – starts a new process instance (event plan instance) when a particular event type is published.
- [Expect Event module](#) – waits for an event(s) in a process to be published. Different actions can be taken if the event is overdue/early/late.

Publish Event Module

The Publish Event module is used to create (publish) events that are used by the [Event Plan Initiator](#) and [Expect Event](#) modules. Typically, a process is created that will detect data change (for example, in a database) or wait for an incoming message (such as HTTP in request or Email In) and change it to an event. Events can be detected and created from multiple sources (for example, a file appearing on a file system can be converted to an event). The Publish Event module converts the detected message to an event for the [Event Plan](#).

To use the Publish Event module

1. Drag a **Publish Event** module into your EMF Process at an appropriate place.



2. Double-click the **Publish Event** module icon to display the **Publish Event** screen.

Publish Event

Properties Preview

Event Definition: New Transaction ... Edit

Simple Mapping

Event Property	Value
Transaction ID	
Some Data	

Action if nothing waiting for event

☐ Do nothing ☒ Store Event until required

Results data section: PUBLISH_RESULTS

OK Apply Test Cancel Help

Properties Tab

1. Select the **Event Definition** you want to publish.

Depending on the type of the **Event Definition** selected, the **Simple Mapping** or **Free Form Mapping** section will be enabled.

- **Simple Mapping** - A text value (including dynamic functions) is entered against each property. A value can be left blank. This allows incoming data to be mapped to properties of the event instance.

Publish Event

Properties Preview

Event Definition: New Transaction

Simple Mapping

Event Property	Value
Transaction ID	10022
Some Data	\$MESSAGE('Text')\$

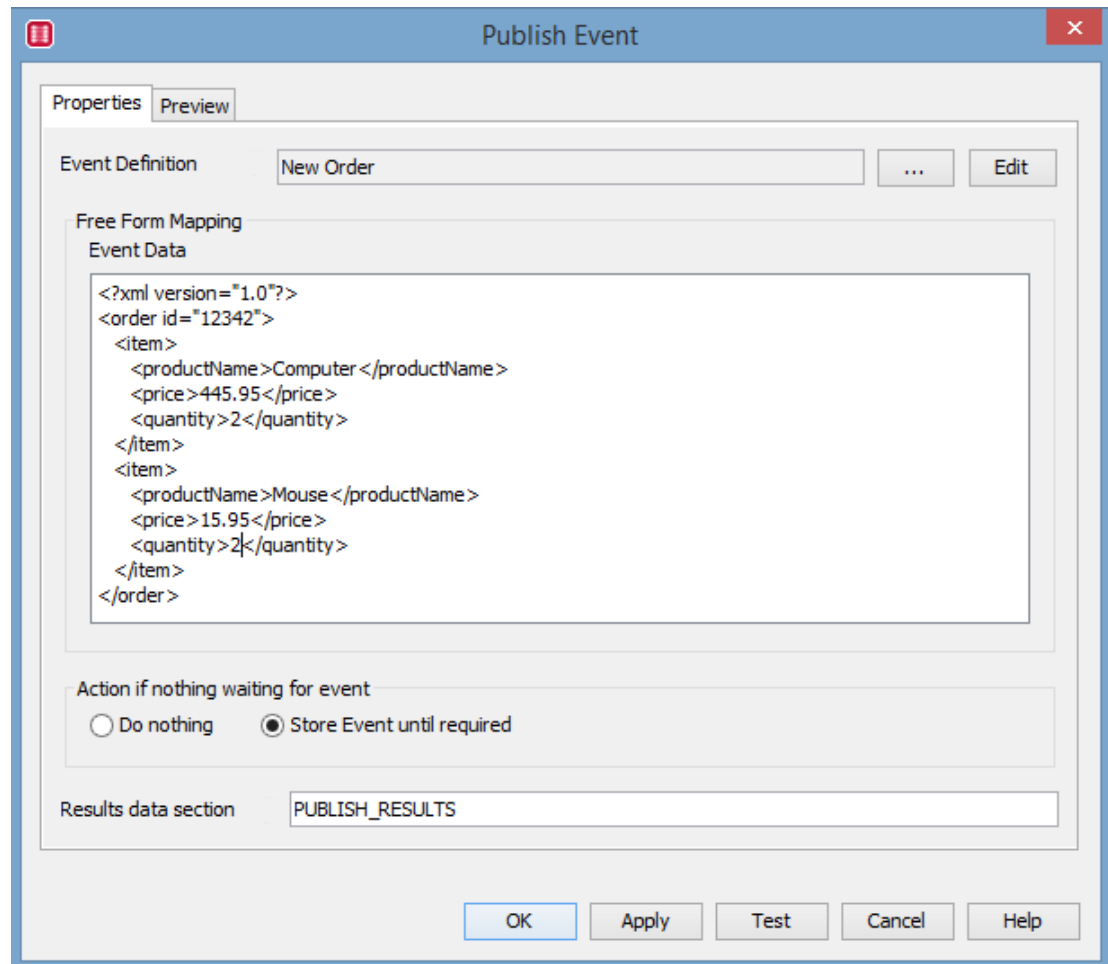
Action if nothing waiting for event

☐ Do nothing ☒ Store Event until required

Results data section: PUBLISH_RESULTS

OK Apply Test Cancel Help

- **Free Form Mapping** - A text section is created (including dynamic functions) that will contain the data which will be part of the event instance.



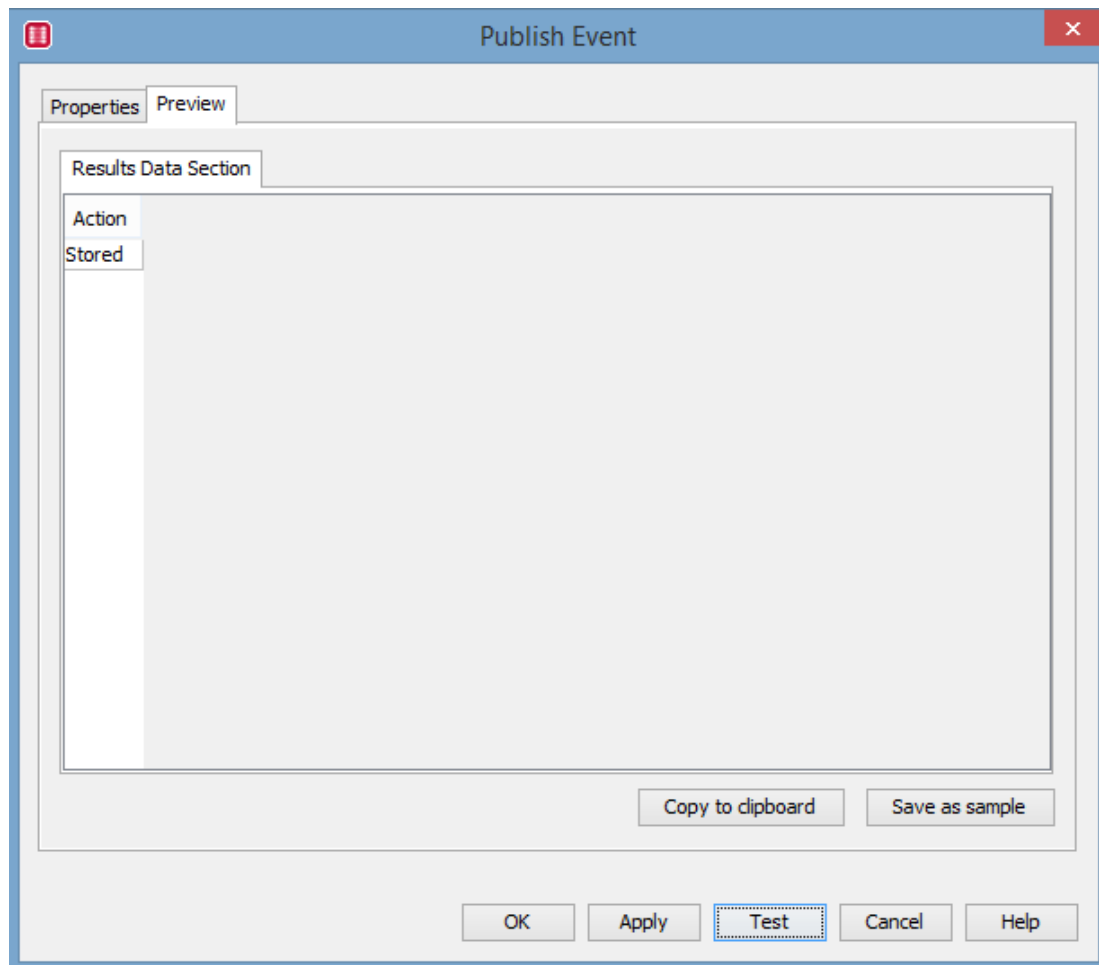
2. **Action if nothing waiting for event** - Define the action taken for the event if the event does not correlate against any [Expect Event](#) or [Event Plan Initiator](#) modules, that is, nothing is waiting for the event instance.
 - **Do Nothing** – The published event instance is discarded and no action is required.
 - **Store Event until required** – The event is stored until an [Expect Event](#) or [Event Plan Initiator](#) requires the event.
3. **Results data section** - The name of the data section that will store the result of the **Publish Event** action. It will be one of the following value:
 - **Processed** – One or more [Expect Event/Event Plan Initiator](#) modules consumed the event. The event was added to the [Published Event Store](#) and marked as **Processed**.
 - **Nothing** - Nothing was expecting the event. The event was discarded.
 - **Stored** - Nothing was expecting the event. The event was added to the **Published Event Store** and marked as **Stored**.

Preview Tab

- Click **Test** to publish the current event definition with the properties set.

The **Preview** tab will be enabled displaying the **Results data section**.

Note: If the event publishing causes a process to run, then the server queue provider must be running (which if using the internal queue provider in a standard configuration, means that the server needs to be running).



Event Plan Initiator Module

The Event Plan Initiator module is used to initiate a new process instance when an event is published. It can only be linked to the [Start](#) module.

When an instance of the selected [Event Definition](#) is published, a new process instance will be created and queued to run for all active process definitions with an **Event Plan Initiator** configured with the same event definition. A new [Event Plan](#) Instance will be created, associating the event instance data to the plan. This will be visible in the [Event Plan Monitor](#).

Note: Publishing a single event can cause multiple processes to run.

To use the Event Plan Initiator

1. Drag an **Event Plan Initiator** module into your EMF Process at an appropriate place.



2. Double-click the **Event Plan Initiator** module icon to display the **Event Plan Initiator** screen.

- **Event Definition** - Select which type of Event the Event Plan Initiator will be associated.
- **Event Data Section** - Specifies the name of the data section in which the instance data of the event will be stored. The event data section is created using the current process instance scope, meaning that all branches of this process have visibility to it. For more information on data section scopes, see [About EMF Data Store - Sharing Data between Processes](#).

The structure of the **Event Data Section** created depends on the type of the event definition:

- Simple Definitions - A single row data section is created. Each column in the data section is named after the property defined in the event definition.

Order Number	Order Date	Product ID	Quantity
123	5/8/2014	6423	4

- Free Form Definitions - A single cell data section is created. The column name is **event_data** and this will contain the data of the event.

event_data

```
<?xml version="1.0"?>
<order>
  <number>123</number>
  <date>5/8/2014</date>
  <product_id>6423</product_id>
  <quantity>4</quantity>
</order>
```

[Event Plans](#)[Event Definition](#)[Expect Event Module](#)[Publish Event Module](#)

Expect Event Module

The Expect Event module is used to cause a process to “wait” for a particular event (for example, a “delivery notification”). When an event arrives that matches (correlates) one that the module is waiting for, one or more links arriving from the module will be processed.

When the module first executes, it will check the [Published Event Store](#) to see if any matching unprocessed events already exist in the store. If there are, then these events will be used to match against the events the module will wait for. This deals with the scenario where events may arrive out of sequence before the Expect Event module is waiting for them.

It is possible to define the schedule when the event is expected to arrive, and cause different links to be run out of the module that indicate if the event arrived earlier or later than expected. It is also possible to define how many instances of an event to wait for.

To use the Expect Event Module

1. Drag an **Expect Event** module into your EMF Process at an appropriate place.



2. Double-click the **Expect Event** module icon to display the **Expect Event** screen.

Expected Event

Event Definition *

Event data dataset: Event Data

Correlation

Event Property	Value	Correlate

☐ Expect event is optional (Process can complete without this event arriving)

OK Apply Cancel Help

Expected Event Tab

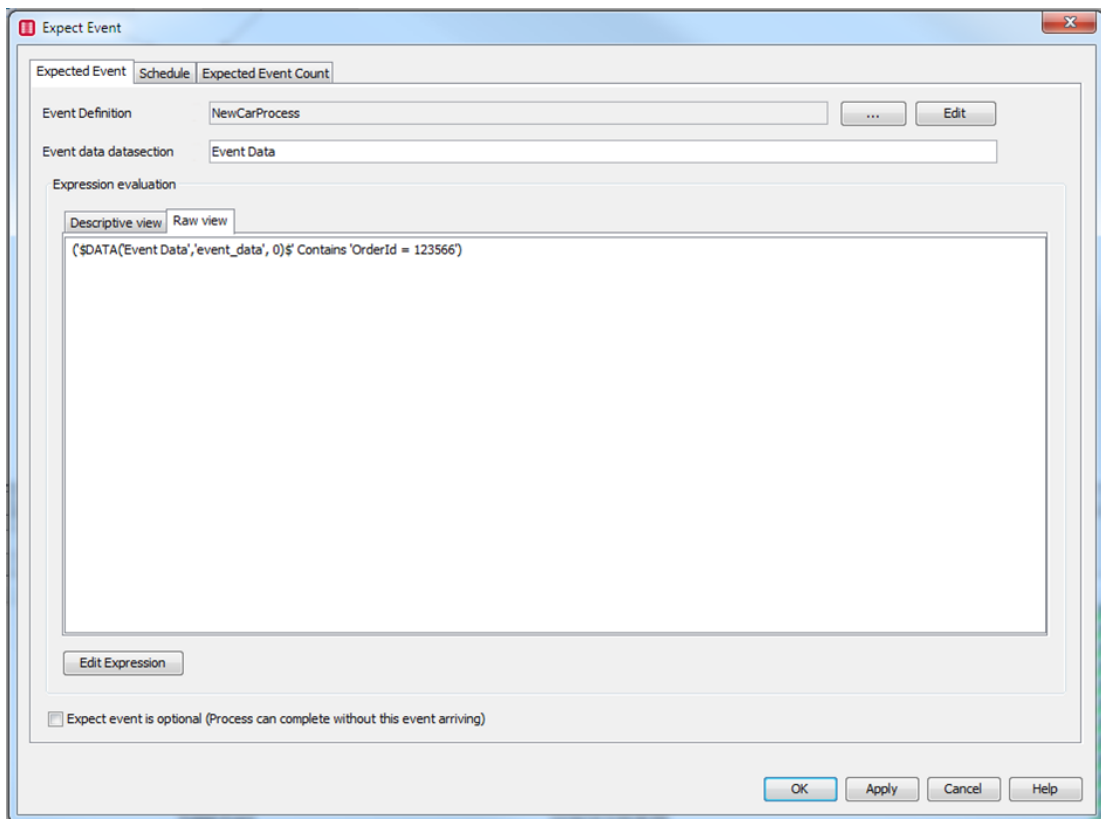
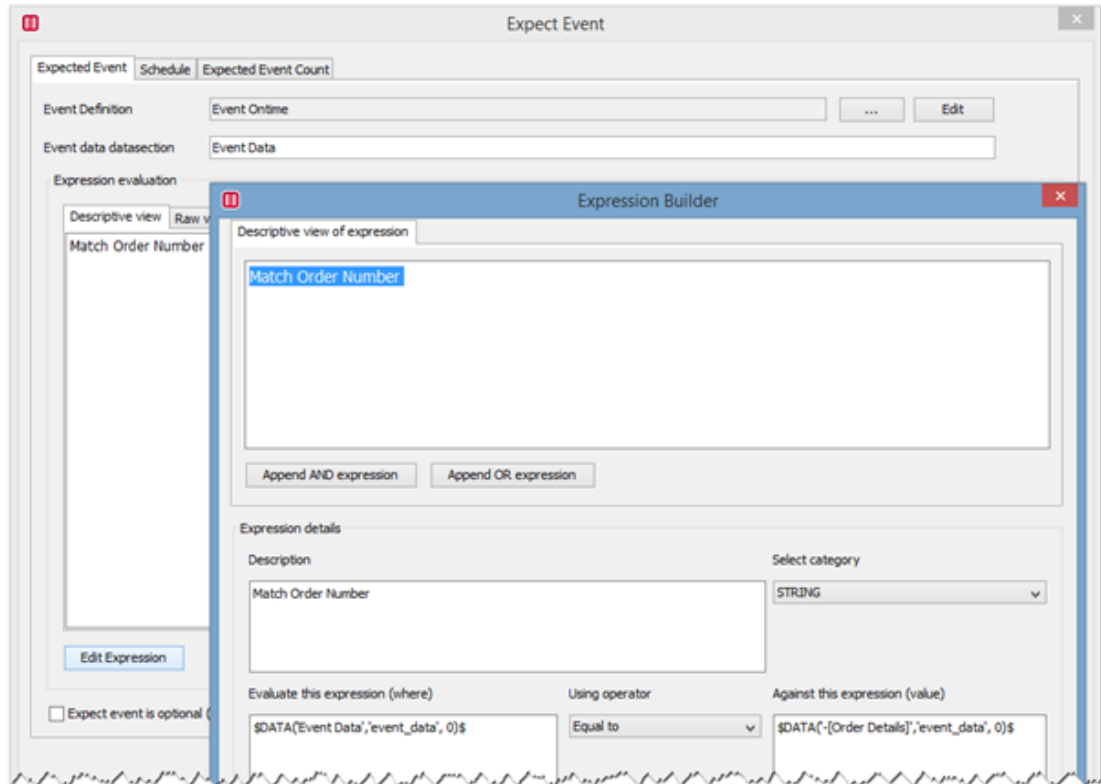
1. Select the required **Event Definition** you want to expect (see [Event Definitions](#) for more information).
2. Type the **Event data dataset** name that will store the data of the Event(s) this module matches with.
3. Depending on the **type** of the **Event Definition**, the **Correlation** or **Expression evaluation** section will be enabled.
 - **Correlation** – (The **Event Definition** is *Simple*). The correlation table ties published events to their appropriate event plans.

- The **Event Property** column will display all the properties defined in the **Event Definition**.
- The **Value** column is where the data to correlate will be entered. The incoming event must have the same value as the value entered in this field for this property to "correlate". Dynamic functions can be entered in this field.
- The **Correlate** column must be selected for the properties that you want to use in the correlation.

Multiple properties can be selected for correlation, and the expression for all must match against a published event for the correlation to be a positive match. For example, the following image displays an Expect Event waiting on a Published Event for "*New Car Process*" which must correlate on two of the three properties: "*Order Id*" and "*Type*".

Event Property	Value	Correlate
Order ID	1234	<input checked="" type="checkbox"/>
Type	Coupe	<input checked="" type="checkbox"/>
Model		<input type="checkbox"/>

- **Expression evaluation** – (The **Event Definition** is *Free Form*). It uses an Expression Builder similar to the one used for [Link Conditions](#). The Dynamic functions will display a special Data section named after the **Event Data dataset** that will be created at runtime that holds a single row/column of data called "event_data" that contains the data from the incoming event. It is then possible to use this data section to build a correlation expression that will decide at runtime if the incoming event is a positive match.



4. Select the **Expect Event is optional** check box if the Event Plan can be completed without the event arriving. In a process that has multiple branches, if other branches complete and the remaining incomplete branches are all waiting on "optional" event modules, the event plan will be marked as completed, and the expected event modules will be cancelled. This allows an event plan to wait for events that may or may not arrive (for example, a "delivery failure" event, may or may not happen in an event plan, but once the "delivery successful" event has arrived there would be no need to wait for a "delivery failure" event any more).

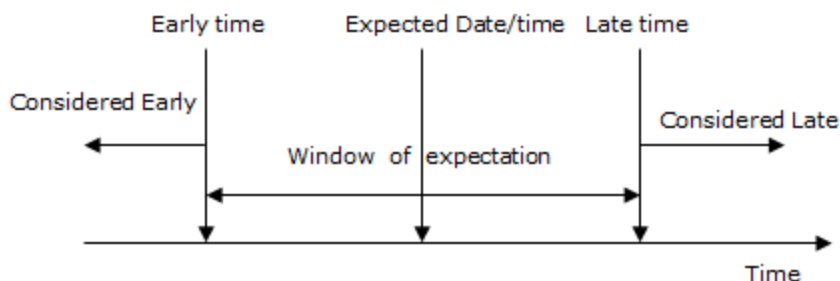
Note: As well as marking **Expect Events** as optional, it is also possible to cancel an expect event module from waiting for an event. This must be done from a branch within the same process instance. To cancel the expect event, you need to use the [Script](#) module, and enter the following:

```
cancelEventPlanStep("<name of expect event module to cancel>");
```

This will cancel all Expect Event modules currently waiting for events that are named after the module name to cancel. This is useful if an event is no longer expected to happen (such as an order is cancelled, hence the delivery event will never occur). It will allow the event plan to complete successfully without waiting on an expect event that would never arrive.

Schedule Tab

The schedule tab defines when the event is expected to happen. It allows a time window to be defined so that it can detect if an event is early/late/overdue. This window of time can be relative to another event arriving, the event plan starting, or absolute time. All time fields accept dynamic functions so the time can be dynamically defined dependent on some other information. Dynamic functions for the schedule tab are resolved at the point in time the module starts waiting.



1. **Early Time** – If enabled, defines the time as an **Offset from expected time** in which the event arriving before will be determined to be early (for example, if a truck arrives at a depot too early additional costs may be incurred). Early time may not be used if the **Maximum Expect Event count** is unlimited (See [Expected Event Count](#) tab).

2. **Expected Time** – Can be defined in two ways:

- **Relative** – When selected, the time is defined relative to the Event Plan Start Time, Itself (meaning when the current Expect Event started running), or another Expect Event.
- **Absolute** – An absolute date/time is entered.

The time must be in the format: **YYYY-MM-DD hh:mm:ss**

For example: November 13, 2020 at 6:34 pm would be written as: **2020-11-13 18:34:00**

This can also be generated using Dynamic Functions. For example, to say an event is expected today at 1:00 pm, an expression such as the following can be used:

\$DATETIME('yyyy-mm-dd 13:00:00','S',0,'now')\$

3. **Late Time** – If enabled, defines the time as an offset from the **Expected Time** in which the event arriving after will be determined as late.

Expect Event modules cannot have normal links coming out of the module. There are five special different links that can be used, so that different actions can be taken depending on whether the event arrived or not, and if arrived, at what time it arrived.

- **Event arrived early** – event has arrived, but before "expected time – early offset".
- **Event arrived on time** – event has arrived, after "expected time – early offset", but before "expected time + late offset".
- **Event arrived late** – event has arrived, but after "expected time + late offset".
- **Event arrived** – event has arrived, either early, on time or late. Hence, it is possible for both an *event arrived early* link and *event arrived* link to run simultaneously.
- **Event Overdue** – event has not arrived, and "expected time + late offset" has passed. However, even though the link runs, the **Expect Event** module will continue to wait for the event to arrive. This means two branches will be running, one branch to process the event overdue, and the one still waiting for the event. (See [Running Event Plan Manager](#) for handling overdue)

Expected Event Count Tab

The **Expected Event Count** tab defines how many of each event has to arrive before the module considers the "event" as arrived. For example, a factory production may require three widgets to be built before a process can continue.

This tab allows a configuration to be entered to wait for three "widget complete" events before continuing. All events are collective classed as either early/expected/late, so that if the process is waiting for two events and one arrives on time and one is late, then the collective is "late" and the *event late link* runs.

Expect Event

Expected Event | **Schedule** | Expected Event Count

Number of events to wait for

Minimum: 1

Maximum: 1 ☐ Unlimited

How long to wait for the events

☒ Maximum possible ☐ Minimum Required

Maximum possible - while the maximum number of events has not been met (but the minimum has), wait until the expected time (and late time, if enabled) has passed to get the maximum possible number of incoming events.

Minimum Required - as soon as the minimum number of events has been matched, continue processing.

OK Apply Cancel Help

1. **Number of Events to wait for** – Specify the **Minimum** and **Maximum** number of events that is required. When **Maximum** is unlimited, the **Early Time** in the **Scheduled** tab will be disabled as not possible to detect early events until the time for this Expect Event has expired.
2. **How long to wait for the events** – The following two different modes of operation are possible:
 - **Maximum Possible** – While the maximum number of events has not been met, wait until the "expected time + late time (if enabled)" has passed to get the maximum possible number of incoming events. If during the wait time the maximum number is reached, then it will fire any of the appropriate "arrived" links. If the minimum number of events has not arrived after the late time, it will run the overdue link and continue to wait.
 - **Minimum Required** – As soon as the minimum number of events has been matched, it will continue processing.

Note: When **Maximum unlimited** is selected for the **number of events to wait for** and "**maximum possible**" is selected for **how long to wait for the events**, then no links will run from this module until expected time + late has passed. It is the job of the [Running Event Plan Manager](#) to resume the process in this case.

Returned Data Section

Once matching events have arrived that meet the rules defined in the [Expected Event Count](#) tab a data section is created named after the **Event data datasection** using the current process instance scope, meaning that all branches of this process have visibility to it. For more information on data section scopes, see [About EMF Data Store - Sharing Data between Processes](#). The data section will contain a separate row of data for each event matched.

The structure of the **Event data datasection** created depends upon the type of the event definition:

- Simple Definitions – Each row contains data for matching event. Each column in the data section is named after the property defined in the event definition. The event definition properties that will be correlated will contain identical data. The properties that are not used for correlation will have different values. For example, refer the following table.

Order Number	Order Date	Product ID	Quantity
123	5/8/2014	6423	4
123	5/8/2014	6911	1

- Free Form Definitions – A single column data section is created. The column name is "event_data" and this will contain the data of each event. For example, refer the following table.

event_data

```
<?xml version="1.0"?>
<order>
  <number>123</number>
  <date>5/8/2014</date>
  <product_id>6423</product_id>
  <quantity>4</quantity>
</order>
<?xml version="1.0"?>
<order>
  <number>123</number>
  <date>5/8/2014</date>
  <product_id>6911</product_id>
  <quantity>1</quantity>
</order>
```

[Event Plans](#)

[Event Definitions](#)

[Event Plan Modules](#)



4

EMF Concepts and Components

Overview

If you wish to customize EMF you should start by referring to the [Fundamental EMF Technical Concepts](#) section of the Help, which provides a technical overview of what EMF is, its system architecture, and how it works.

Detailed instructions on how to develop new modules or customize the way EMF works are included in the two **Application Programming Interface** (API) guides, which are intended mainly for developers and system administrators.

Finally, [Configuring the EMF SOAP API](#) details the EMF support for **Simple Object Access Protocol** (SOAP). SOAP is a simple XML-based protocol for exchanging application data over the Web. The SOAP server components are included with the main EMF installation, but you need to install the SOAP client components from the EMF installation media (in the **SupportFiles** directory).

[Introducing EMF](#)

Fundamental EMF Technical Concepts

The **Fundamentals** Help topics are intended to provide a technical overview of what EMF is, how it works, and how to customize it.

Components	Provides details on the EMF Administrator, EMF Process Builder, and the EMF Server component.
Architecture	Provides details on the architecture of the EMF product.
How EMF Processes Work	Provides a description of how an EMF Process is constructed and works its way through the EMF system.
Binary Data	Specifies the areas of the product where data is maintained in the binary format.

[Go to Start of EMF Help](#)

EMF Components

[Administrator](#)
[EMF Process Builder](#)
[Server](#)

Administrator

The EMF Administrator is the graphical tool used for creating business rules, monitoring data sources, and carrying out general administrative work such as:

- Building EMF Processes
- Creating and editing user, group and alias information.
- Organizing EMF Process business rules definitions in various logical folders, including Search and filter facilities.
- Administering database connections, security and auditing.
- Managing the configuration of the EMF System.

The EMF Process, user, recipient and additional information that is created through the Administrator is stored in a repository that is shared by all EMF components.

[More Information about the Administrator](#)

EMF Process Builder

The workflow style EMF Process Builder sits within the Administrator, however its importance merits separate mention here.

The EMF Process Builder is the component which allows you to build custom EMF Processes. It uses modules as visual building blocks to define the functions of the EMF Process and simplify its presentation. Each block defines a task that is to be performed by the EMF Process, with links that define the order of processing. With this, you can easily change to order of the tasks to suit the specific need. The processing path can even be split into multiple branches.

Modules Include:

- **Initiating modules:** Scheduler, Inbound API, Inbound Email, Inbound Queue
- **Database modules:** SAP Read, SQL Read, HTML
- **Process flow modules:** Cascade, Escalation, Synchronization, Delay, Halt condition handler, Conditional links
- **Recipient modules:** Dynamic recipients, Fixed recipients, Aliased recipients, Deliver profile
- **Message formatters:** XML, WML, HTML, Text, Delimited
- **Output modules:** File output, Fax output, Email (SMTP) output, SMS, Pager, FTP, Queue, Run executable, WAP push, Voice
- Scripting
- **Logging:** Auditing, Debugging, Statistics, Billing

Server

The EMF Server component is the neural center of the product. This is the software engine

that runs 24 hours a day, 7 days a week, to keep an eye on your ever-changing business data, and generate the EMF Processes that are defined in the repository by the Administrator. The server runs in the background of the system and is configured through the Administrative tools.

Typically, the Server runs on a dedicated computer on the network, with access to the required databases, Email servers, modems and other components.

[Back to Start of Fundamentals](#)

Architecture

EMF Architecture is based on a message passing backbone that provides for scalable, fault tolerant and distributable messaging. The architecture has several components including queues, databases, service managers, initiators and output devices, and a diagrammatic representation is available [here](#).

The following is a description of the major components of the EMF architecture:

Repository Database

EMF requires databases to store system configuration and information about EMF Processes. The open database architecture that Aptean has implemented as its back end allows it to support a range of database products for this purpose. For a list of currently supported databases, see the *Event Management Framework 7 Installation and Configuration Guide*. If a client database is a supported type, then the repository database can be located on the client database server.

- **Repository:** the data repository, also known as the repository database, contains information specific to the EMF Process such as rules, contact information and trend analysis data. The Data Repository can be located on the same machine as the server managers, but is more commonly on a separate database server in the network.

Queues

EMF uses the Java Messaging Service (JMS) API and third party queueing software to manage the physical Queues in the system. Logical representations of the Queues are mapped to these physical queues. Each physical queue has at least one server manager servicing it, though there can be more than one logical queue on a physical queue.

EMF uses a number of logical queues, though not all of these queues are required in every situation. A general rule of thumb is that as you add to the volume of your EMF Process, the number of physical and logical queues must increase to accommodate the larger throughput required.

Logical queue types include:

- **Primary Input Queue (PIQ)** - This is the queue on which an EMF Process is first placed when it is initiated. EMF Processes that have not yet begun to run are taken off this queue by their server manager and put into processing. an EMF Process on the PIQ contains its ID, which is used to populate the EMF Process object with the list of modules to run and the links between them. Any data which have been passed to the EMF Process from the event listeners are also in the EMF Process at this stage. This queue must be present.
- **Work In Progress Queue (WIPQ)** - This is the queue where EMF Processes currently being processed are placed while waiting to be picked up by other server managers and allocated threads on which to run. The presence of this queue greatly increases scalability by allowing EMF Process processing to be spread over many threads running in multiple server managers. Items on the WIPQ can be run on any server, since all modules are constructed to have a common interface. This queue must also be present.
- **Output Queues** - These queues include all of the queues which are set up for designated output services such as email, SMS, FTP, file, pager, fax and executable EMF Processes. These queues are optional, as the work in progress queue can process these outputs directly if needed.
- **Dead-letter Queue (DLQ)** - Used for EMF Processes which have failed during processing. This queue must be present.
- **Debug Mode Queue (DBGQ)** - The Debug Queue is used in debug mode. EMF Processes on this queue are executed in single step mode from the administrator.

Server Managers

EMF processing is performed by server managers, each having multiple worker threads. Each server manager must be uniquely identified and there must be at least one server manager per physical queue in the queueing software. The server manager must be capable of processing any type of EMF Process module that is passed to it. Modules within an EMF Process are designed to have a generic interface so that they can be processed by any server manager in the system.

In a larger distributed system, there can be any number of server managers. They can be located on any physical machine within the network providing that they are registered within the configuration database.

Client Databases

The client database is any database in the client network used to store EMF Processes information. While the machine running a EMF server manager can be different from the client database server, it should be located as close as possible within the network to the database server. This reduces the number of network device interactions (hops) that the data packets will have to travel while being processed and therefore minimizes the load on the network. It is possible to have multiple database servers and different types of databases within the EMF system.

Output Devices

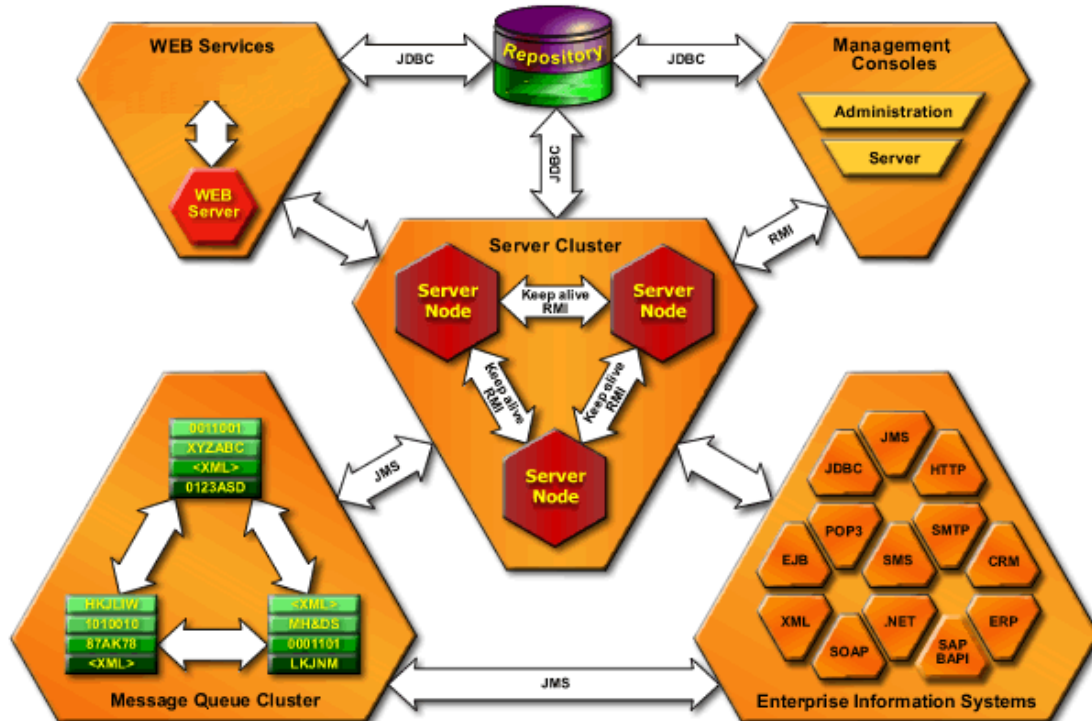
The output devices in your EMF system (e.g. the fax services, email server, SMS modems, web server etc.) can be located anywhere within your network. The worker threads for the output services must be able to access these devices to enable them to send EMF Processes.

[Architecture Diagram](#)

[Back to Start of Fundamentals](#)

Architecture diagram

The following diagram shows a graphical representation of the EMF architecture.



[Architecture Components](#)

[How EMF Processes Work](#)

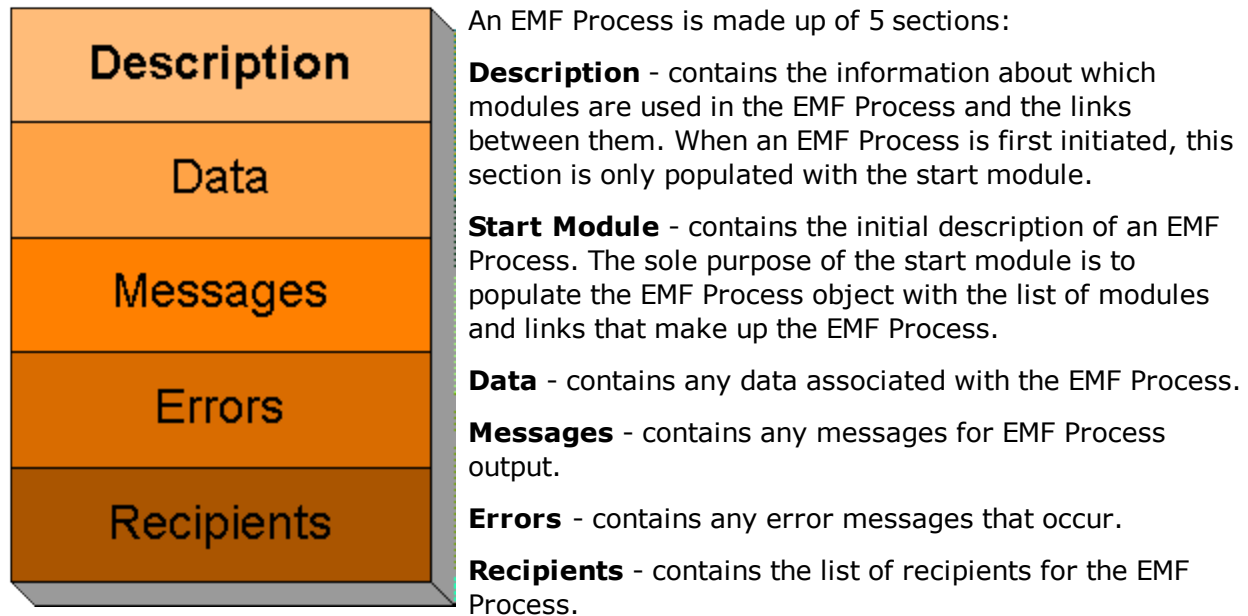
[EMF Process initiators](#)

[EMF Process Processing](#)

[Fundamental EMF Concepts](#)

How EMF Processes Work

EMF Processes are defined in the Administrator and scheduled to run at times specified by [initiator](#) modules. They then retrieve the necessary data, [process](#) it and send it to the recipients.



[Back to Start of Fundamentals](#)

EMF Process Initiators

An EMF Process can be initiated in four ways:

- EMF Processes are defined in the Administrator and scheduled to run at pre-specified times and retrieve the necessary data when they are run.
- EMF Processes are initiated when a trigger occurs within a database. The trigger invokes an EMF Process which is pre-defined in the EMF Administrator. The data that is captured by the trigger will be passed to the EMF Process and distributed to the recipients that are defined in the repository. In EMF, this function is supported for Oracle and Microsoft SQL Server databases.
- EMF Processes can be initiated by listeners, which receive incoming email or Queue messages and trigger the appropriate EMF Process.
- EMF Processes can be generated by any external application using the API.

See also [The Input Modules](#) for further information.

[Back to Start of Fundamentals](#)

EMF Process Processing

When an EMF Process is initiated, the EMF Process object contains only the Start Module. The EMF Process object is placed on the Primary Input Queue. (PIQ) This queue stores EMF Processes that are waiting to be processed and passes them on to the server manager in a FIFO (first in, first out) manner.

As an EMF Process begins processing, the server manager for the PIQ assigns a worker thread to the start module in the EMF Process object. The start module populates the description of the EMF Process object with the list of modules and links which will be run during the EMF Process execution. Once this is complete, the worker thread will begin work on the next module in the path of the EMF Process. The worker thread processes each module in sequence, which may involve reading EMF Process details from the repository, processing the associated data, or producing output.

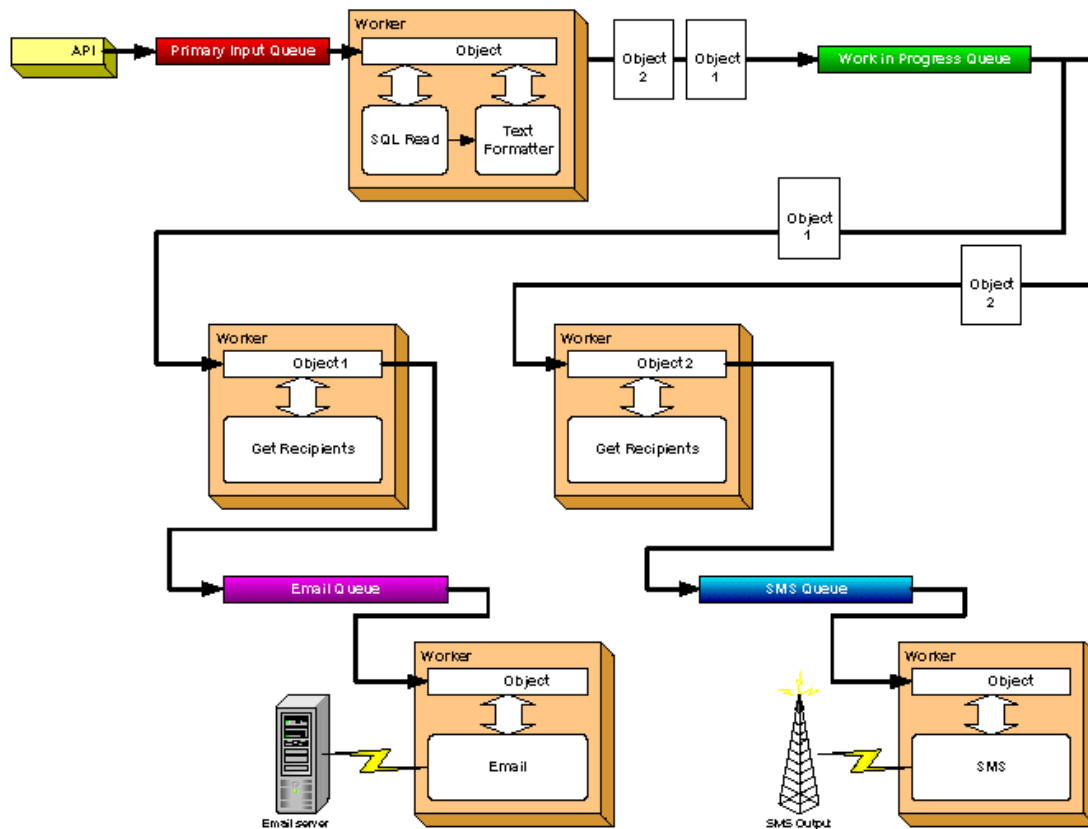
If the EMF Process processing path splits, a new EMF Process object for each of the paths is created and placed on the Work in Progress Queue. (WIPQ) From there, each of the EMF Process objects can be picked up by worker threads and each can continue processing independently until it reaches the end of its processing path.

Click [here](#) to see this process represented graphically.

[More about EMF Process Control and Processing Modules](#)

[Back to Start of Fundamentals](#)

EMF Process Processing Flow Chart



Binary Data

When data is received by an EMF Process it is stored internally as Unicode, ready to be sent out as a text message to some type of device. There are, however, times when the data that is being processed cannot be represented internally as text, but must be maintained as it's binary representation. A typically scenario for this would be a simple EMF Process that read an Excel spreadsheet file from an FTP site using the **File In** module and then saved it to a local drive using the **File Out** module. If during the running of this EMF Process the data was internally converted to Unicode, then the resultant file would have been corrupted.

To handle situations where the integrity of the incoming data must be maintained, EMF has specific points in the product where the data will be maintained in its binary form.

Note: Data that has been marked as binary cannot be output directly as a message, except at one of those points specified below where binary data is supported. Attempting to do so will result in the text of the message being rendered as "Message is non text and cannot be displayed".

The following areas of the product will take data in (or generate it) and maintain it in a binary format:

- The **File In** module where the option has been selected to **Read File Contents** and the option selected to **Store Contents as Binary** selected.
- Data received by an **HTTP Listener** where the message content has been **POSTed** and the content type of the message is marked as being **application/octet**.
- Binary data read from a database. This will typically be from fields of type VARBINARY, RAW, IMAGE, BLOB dependant on your database type.
- Data created using the Base64 decode dynamic function.
- The HTTP module when used in "request" mode can store the contents of the result as binary.
- The **Report Generator** and **Excel Writer** modules can create messages sections that are in binary format.

Once data is in its binary format it can be manipulated/outputted in the following ares of a EMF:

- The **File Out** module. Note that the file out module automatically detects if the data is binary, and if it is, then the **Save as unicode** option will have no affect.
- Data **POSTed** by the **HTTP** module where the content type is marked as being **application/octet**.
- Data written to a database. Typically it will be written to fields of type VARBINARY, RAW, IMAGE, BLOB dependant on your database type.
- Data can be encode using the Base64 encode dynamic function and then the encoded text message can be treated as a normal message section.

Some example usages:

- Storing files in a database
- Communicating with devices that send data as binary streams via HTTP.
- Decoding email attachments and writing them to a database/disk

[HTTP Module](#)

[File In Module](#)

[SQL Module](#)

[File Out Module](#)

[Report Generator Module](#)

[Excel Writer Module](#)

Administrator Main Screen

The Administrator main screen is split into two panes. The left pane consists of a tree view from which you can access all of the functions of the EMF system. The right pane displays additional options or information that pertain to the highlighted item in the left pane tree view. For example, when the EMF process node of the tree is selected, the right pane display the icons for the EMF process General Tasks.

Nodes in the EMF process tree include:

- **EMF Process Folder** - This is where all EMF processes and EMF process templates are stored in folders. [More Information](#)
- **System** - This node contains the information and utilities needed for system configuration and administration. [More Information](#)
- **Recipient Administration** - This node contains the information and utilities to perform recipient administration tasks. [More Information](#)
- **Data Sources** - This node is used to maintain connections to system and client databases.
- **Services** - This node contains the information on the input and output services and the means to access and configure these services. [More Information](#)

[Back to Start of Fundamentals](#)

Considerations for Sizing and Optimizing an EMF Deployment

Overview

Due to the broad functionality and wide variety of processes that can be supported by EMF, it can be difficult to recommend database and server configuration. This document will highlight items that you will need to consider when planning your EMF implementation. When designing a system for optimum reliability and performance the following factors will need to be considered:

- Complexity of the process
- Third party technologies
- Queuing technology and branching

Complexity of the process

There is an overhead in loading the process definition into memory when it is being executed, but typically this is small enough to be ignored and frequently used processes are cached by the EMF server for optimal performance.

Third party technologies

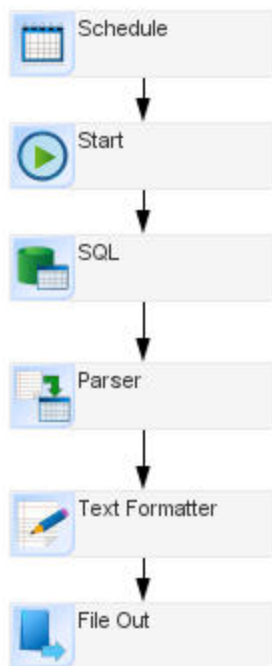
EMF is designed to integrate with numerous third party technologies and services that use standard or proprietary interfaces such as SAP via JCo, Oracle via JDBC, Email servers via SMTP, IMAP or POP3, and so on. These third party technologies are beyond the control of EMF but can have a significant effect on the execution of a process. A typical installation would require EMF to communicate with some of these third party applications via the LAN or WAN so when the network infrastructure is unreliable or slow this will cause

performance issues beyond EMF's control.

Queuing technology and branching

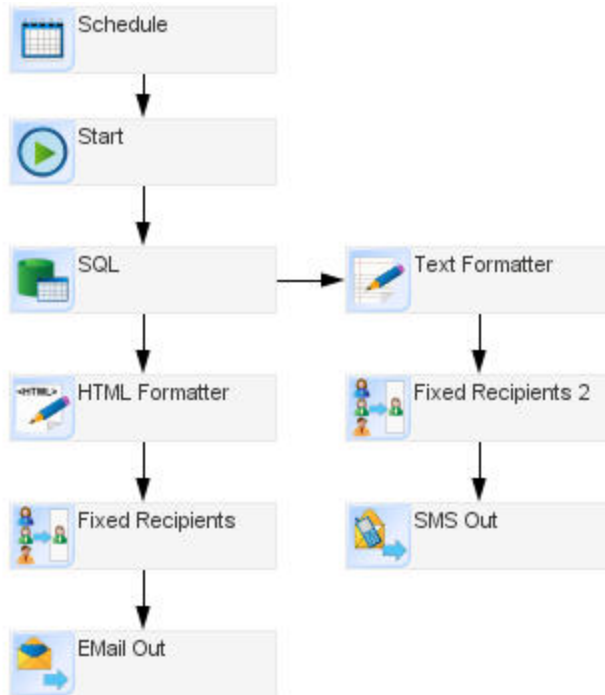
EMF is shipped with queuing technology (ActiveMQ) that performs adequately for the majority of EMF installations. In some high volume implementations, it may be necessary to use an alternative queuing technology. Understanding how EMF uses queues during a process will help you design more efficient processes that reduce the burden on the queuing technology and improve overall performance.

The figure below shows a process that has no branching and will execute one module after another without additional queuing.



This is an ideal situation and has the least additional impact on the queuing technology. When a process diagram branches unconditionally, i.e. neither branch has conditions applied, when the process execution reaches the branch, the EMF server will place work items onto the queue for each branch of the process. This functionality allows the server to recover gracefully from any errors that may occur during execution, but it does impose an overhead on the system.

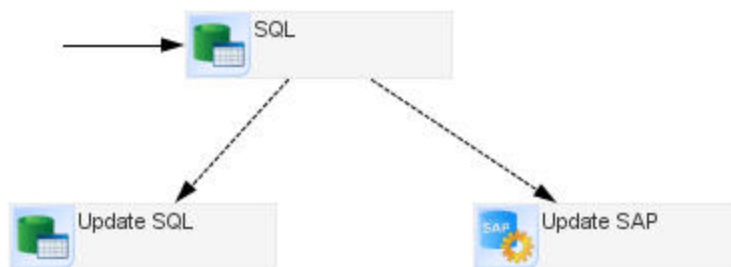
Figure 2 below is an example showing a branch after the SQL module which will cause work items to be placed onto the queue. Each branch will then pick up the item and continue processing.



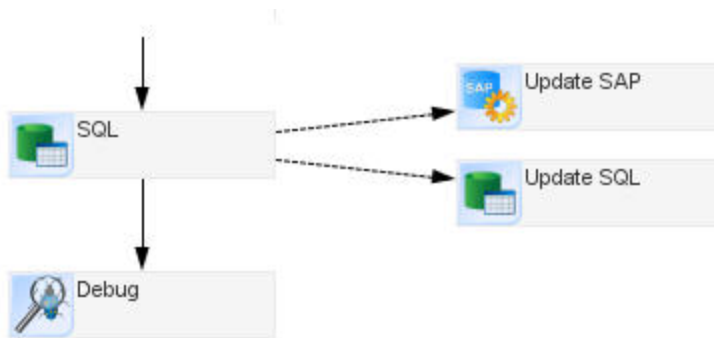
When designing a process to use branching, it's worth considering the reason for adding branching especially where high volumes of processes will be executed. So in the example above there is no benefit in forcing the process to branch after the SQL module and the process will perform less efficiently due to the additional queuing overhead than if it were built inline as shown below.



If you specify conditions on the links for each branch and only one of the links will be executed then this can be viewed as a straight through process and no additional queuing will be required. The example below shows the a link with a condition specified going to the SQL module and the link going to the SAP module will only execute if the left branch is not taken.



A common scenario is to add a debug module to capture a snapshot of the process to either aid with building the process or for capturing historic data. However, this module is often just added as a branch to the process as shown below.



Although the process is laid out differently it is the same process as shown in figure 4, but now includes a debug module. The debug module has now created a situation where there will always be two branches taken so the process will now incur the additional overhead of adding and removing two work items from the queue.

Tip 1 – Avoid branching unless it is required by the business process functionality.

Tip 2 – Remove unnecessary links before migrating process to a production environment. It is acceptable practise to leave orphaned modules on a process, that is modules that have no links connecting them to additional modules. These modules may be temporarily connected in the future for analysis.

Cascade Module

The cascade module allows you to call additional EMF processes from within a running process.

The two most commonly used modes of operation for the cascade module are:

- To launch a separate self contained process that requires no data from the parent process.
- To loop through a data section in the parent process and launch a second process once for each row of data in the data section. So for example you may retrieve a list of orders that have overdue delivery and the second process will retrieve the details for the order and send out notifications to relevant parties.

In both scenarios above you may choose to have the process return back to the parent and include any Messages or Data generated into the parent process, which will obviously increase the size of the parent process. The simple process below demonstrates the item listed in 2) above and will retrieve an initial list of orders that are overdue and will call a second process to get the order summary details. It has been configured to add the data it collects back into the parent process so that we can create a summary email. The configuration screen for the cascade module is shown below.

Tip 1 - There are still many situations where it makes sense to use the cascade module, but you should always consider using the Loop module for maximum performance.



Cascade

Properties | **Synchronization**

Synchronization method

- ☒ No wait/synchronization
- ☐ Wait for first cascaded process before continuing
- ☐ Wait for all cascaded processes before continuing

Timeout

Timeout period:

Timeout interval:

☐ Halt on timeout

Message section merge options

- ☒ Do not replace existing message sections if they already exist.
- ☐ Overwrite message sections if they already exist.

Data section merge options

- ☒ Do not alter existing data sections if they already exist.
- ☐ Add child data into existing data sections

Recipient section merge options

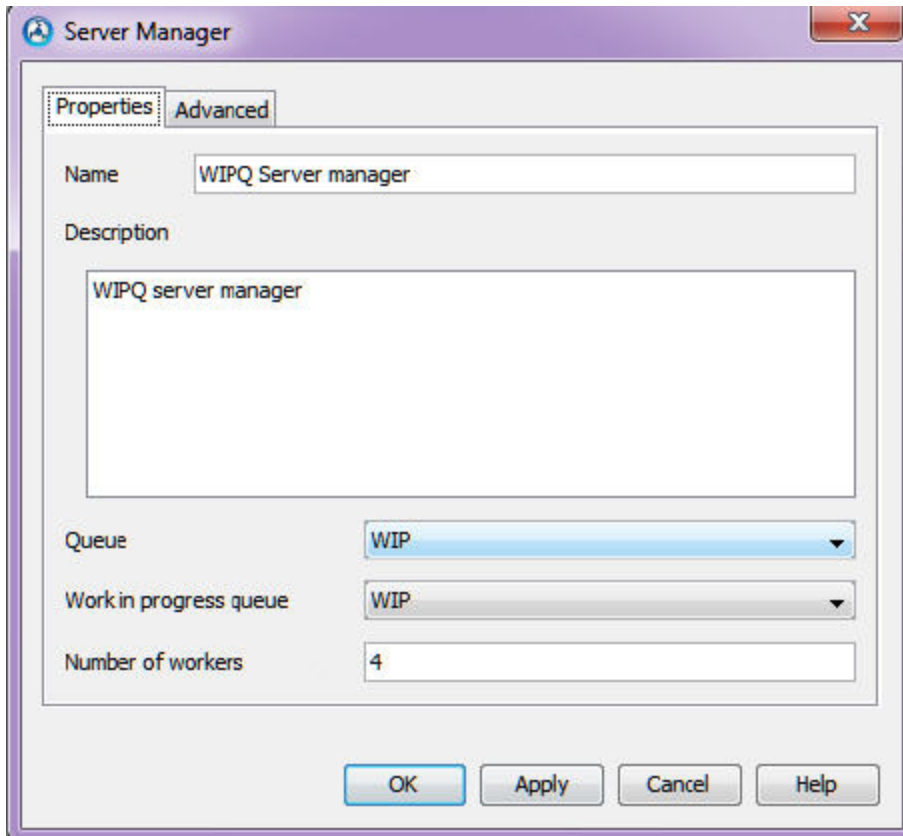
- ☒ Ignore recipient sections in child processes
- ☐ Add recipient sections from child processes
- ☐ Replace all recipient sections in parent with those from child processes

Results data section:

OK Apply Cancel Help

Multi-threading

EMF imposes a limit to the number of threads that can concurrently service a queue at any one time. By default this is set to 4, but can be changed using the server manger configuration. Note this is hidden by default and "show/Hide system listeners" will have to be enabled under the machines node in the EMF Process builder.



Decreasing the number of threads can help reduce the overall memory requirements by restricting the amount of concurrent processes that can be executed at any one time. It has the same affect on third party systems such as target database servers, web service hosts by reducing the number of concurrent activities that can occur. So for example in the previous cascade example, if the parent process returns 50 rows, using the default configuration the server will only execute 4 child processes at a time until all are complete. The optimum number of threads for a deployment is completely dependent on the server architecture and platform on which EMF is deployed and should be established during testing phase, There is a play-off between process throughput and resource requirements. Note that the vast majority of installations never touch this setting.

Data consumed by the process

When a process executes it may harvest data from a variety of sources and will store

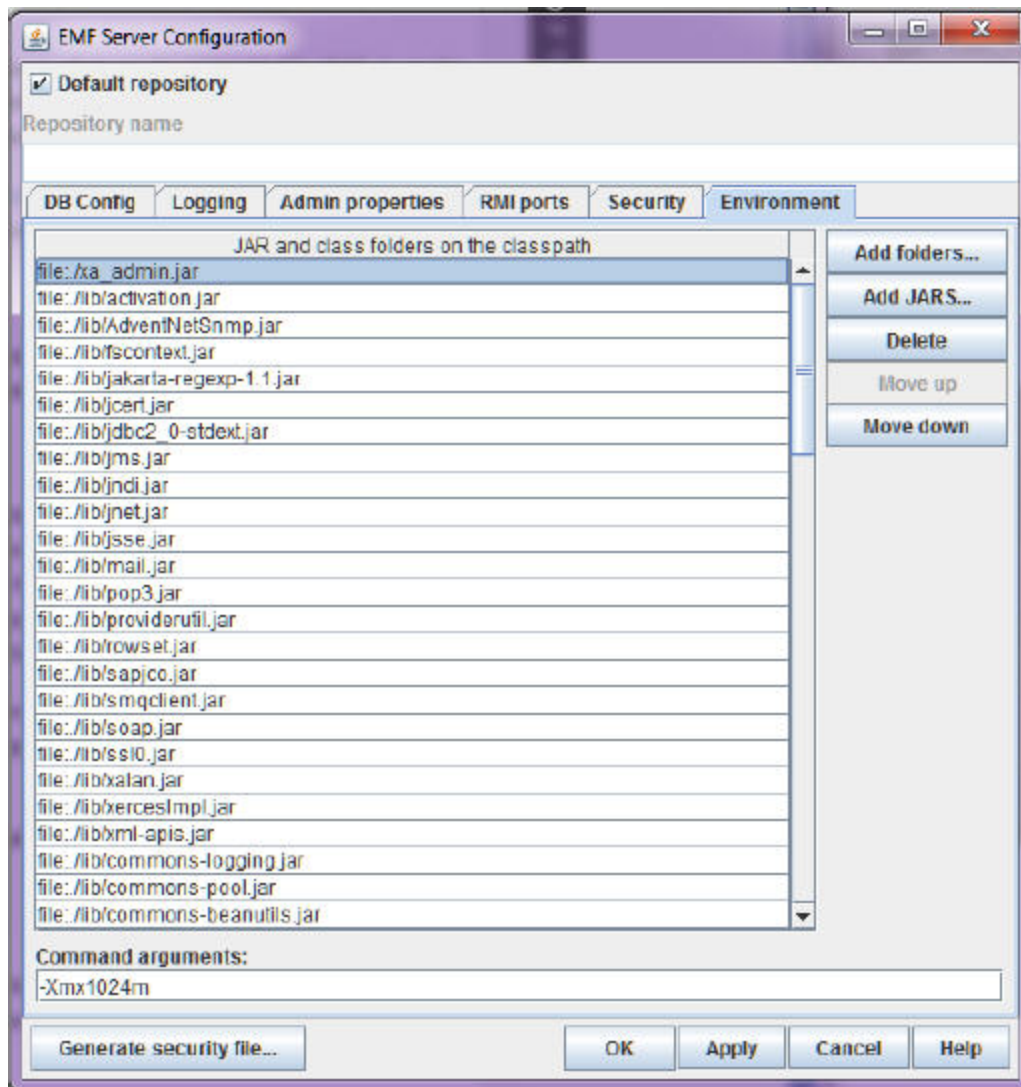
these in the running EMF process in Data, Message, Recipient and Error Sections. There needs to be sufficient memory (heap space) allocated to the EMF server for it to be able to hold all the data gathered by the sum of all running processes e.g. if there are ten processes running concurrently and each consumes 10MB of memory EMF will require 100MB of heap space to be able to execute them.

In systems where memory is an important consideration it is imperative that you understand the impact of specific modules and use techniques to free up memory as quickly as possible and reduce the amount of memory each module uses.

The amount of memory allocated to the EMF server is specified in the server configuration utility environment tab. The example in the screen grab below shows 1024MB (1GB) of ram allocated to the server. Note that due to constraints with 32bit architecture servers the maximum memory that can be allocated is in the region of 1.4GB.

The amount of memory allocated cannot be greater than the physical memory of the computer system, and for performance reasons it is recommended it should be at least half the physical memory of the system (to avoid excessive swapping to disk).

It is therefore strongly recommended that where processes will be handling large amounts of data, that 64 bit architecture is deployed where this restriction does not exist.



EMF can consume or create data from:

- External applications/processes
- EMF modules executed from within a process. Some of the more commonly used EMF modules that can create data in a running process are:
 - SQL
 - Parser
 - XML Parser
 - Data Filter
 - SAP
 - SAP Authorizations
 - SAP Data Browser
 - Script
 - Gap Detection

- Cascade
- Duplicates
- File In
- Email Listener
- Segregation of Duties
- All Recipient Modules
- Reply
- Cascade
- POP3

In some of these modules such as the data filter, data section toolbox, SQL modules have the ability to consume large amounts of data.

Data filter module

The data filter module has the ability to detect changes between two similar sets of data. Typically this is used to detect status changes or new items being added to a system but can also be used to look for items that have not changed between records sets. This module makes life incredibly easy for EMF users to build complex data monitoring functionality and is widely used. It is important to understand how it works in order to gauge the impact it has on hardware requirements.

Let's suppose that the data filter was being used to monitor a list of employees in an HR system and the process would automatically create an account for the user in the Pivotal CRM system for each new user as they were added. Let us also assume that the HR system has a current total of 1000 employees. In order for EMF to be able to detect a new employee, it has to be able to keep a snapshot of the current HR employees so that it can look for additional entries. So assuming it is a SQL query that returns the current set of users and that the size of the 1000 row resultset returned is 2MB, EMF will now take a snapshot of this resulting in a memory requirement of 4MB.

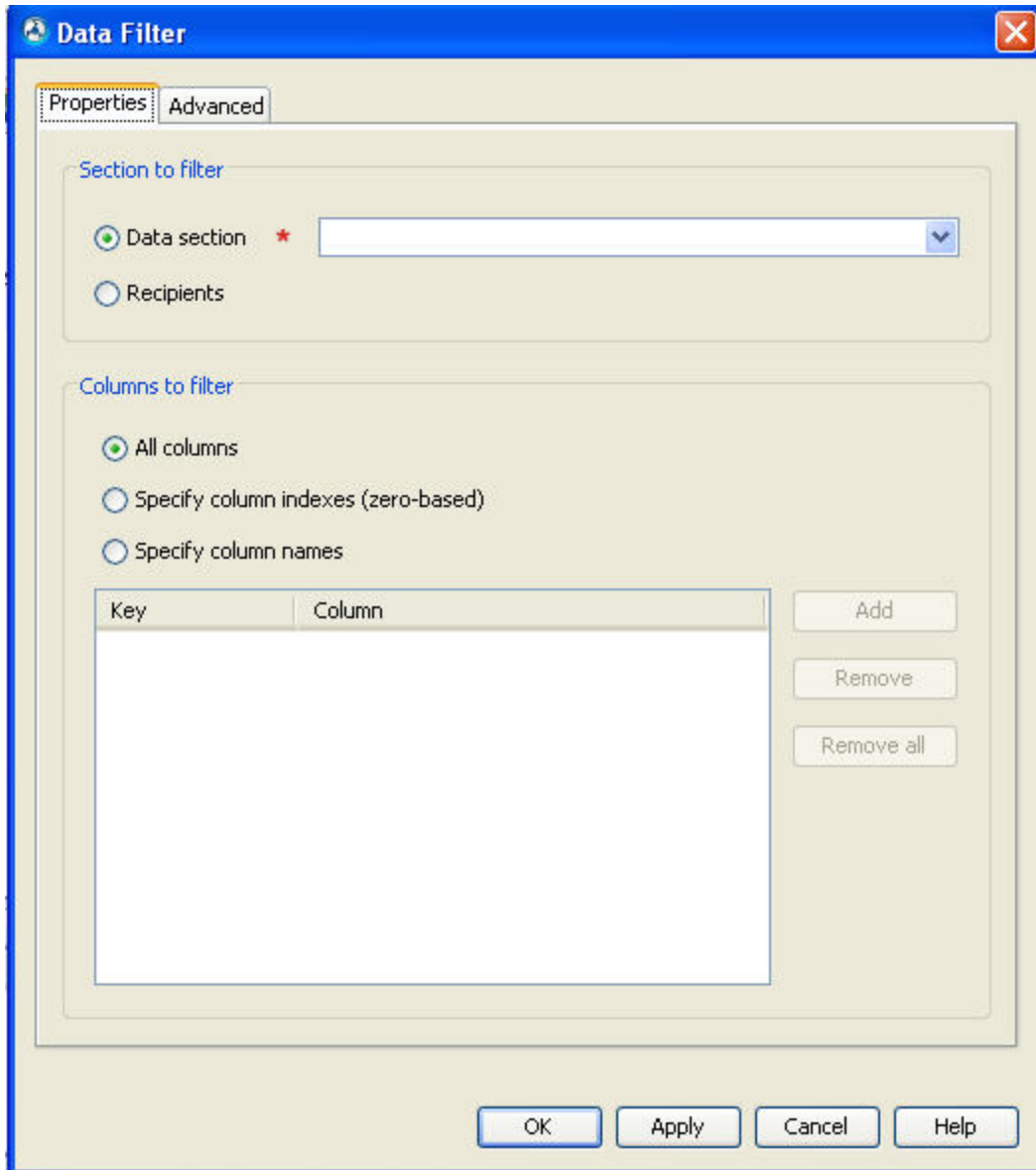
What if the company were a bank or a telco looking for changes to customer records then we are talking potentially millions of returned rows resulting in extremely large memory requirements which could be doubled by the data filter if incorrectly used.

Fortunately the Data filter has some options that can help reduce the memory it uses when executing and combined with good process design can massively reduce the overall memory requirements for the process.

Following are some best practise tips:

Tip 1 - Always identify a field or combination of fields than can be used to uniquely identify a specific row of data. If this is possible then under certain circumstances it may be possible to store only minimal data in order to perform the comparison. This is particularly relevant when you are looking for new records in which case ONLY the unique key values are needed in order to detect a new record added.

Tip 2 - Use specific column names to perform comparisons. If for example you have a recordset that contains 30 fields of order header details then configure the data filter to look at only specific columns such as the Order Status field which is all it would need in order to determine if an order has changed status. EMF will then only store the status values in it's snapshot and not the additional fields.



SQL Module

The SQL module has a non-replication function which behaves similar to the Data filter module. This is a legacy feature and is only retained for backwards compatibility. In practise it should never be used as the Data filter is more efficient and has additional options as described above.

Tip 1 - Select * is your enemy! Not specific to the data filter, but if memory is a potential issue always try to reduce the amount of data returned by the initial query. If the query returns half a million rows each with 30 fields but only 10 fields are ever used, change the query to return only the 10 rows you need.

Tip 2 - ALWAYS use SQL parameters whenever possible for dynamic queries! If for example we have the following query...

```
Select orderId, customerId, orderdate from orderstable where customerId = '123456CDC'
```

This could be converted into a dynamic query, i.e. one that takes a run time defined value for the customerId, by using dynamic functions such as

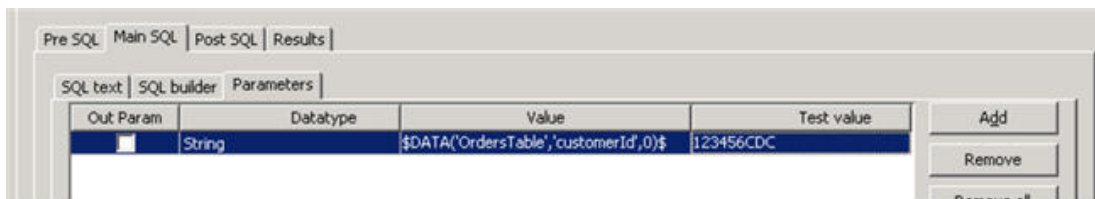
```
Select orderId, customerId, orderdate from orderstable where customerId = '$DATA('Orders','CustomerId',0)$'
```

This will function as expected, but if you were to execute this statement a 1000 times passing in different values for customerId each time then typically, the database engine would compile the query each time it executes it. In order to achieve maximum performance benefits from the database server, we can use it's ability to use pre-compiled queries by specifying SQL parameters.

Using SQL parameters, the query would then look like the one below and note the lack of quote marks around the ?&ldots;

```
Select orderId, customerId, orderdate from orderstable where customerId = ?
```

In EMF you would define a SQL parameter as shown in the screen grab below.

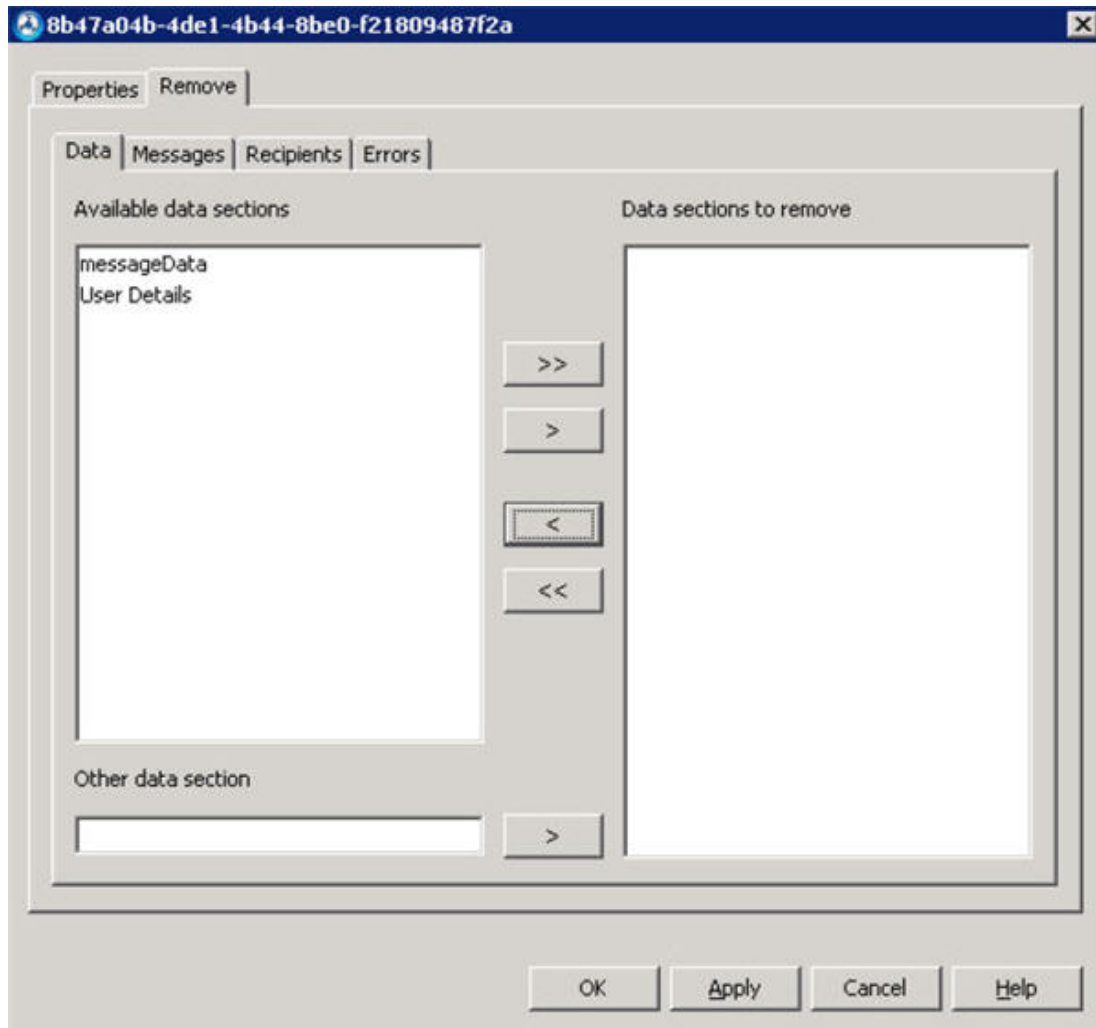


There are three other major benefits to using SQL parameters in EMF:

- You can specify a value to use for testing the query in the 'test Value' field. This will only be used when the test button is pressed, when the query executes for real it will use the value returned by the dynamic function.
- Parameters support real data typing so you do not need to worry about the string handling. If for exam you wanted to perform an SQL update and update a comment to a record with 'Customer says thank's for your quick response', you would find that it would break the SQL due to the inclusion of the quote mark and you would need to escape the string before passing it into the SQL query. SQL Parameters take care of this for you and ensure that the correct values are passed to the database.
- It is more secure and will prevent SQL Injection attacks on your system http://en.wikipedia.org/wiki/SQL_injection.

Remove data when no longer required

When a process no longer requires some data you can tell the process to delete the data from the running process. In order to do this you configure the link properties and specify which pieces of data you no longer need. EMF Data, messages, recipients and error sections can all be removed.



Parser Module

The Parser Module transforms content, usually a text based document such as a CSV file and converts it into an EMF data section for use in the process. In some instances the documents passed in can contain many thousands of rows of data that will need to be converted. The converted document will increase in size slightly as the converted data will have real data types associated with it. In addition there are options to specify specific data types for column values and also to force column widths to be a minimum

size. Frequently there are many more fields passed in than are required by the process later on so here are two tips for improving performance.

Tip 1 - Ensure that you uncheck all columns that are not actually required for the downstream business processing. The screen below demonstrates this by using only the OrderId and OrderStatus values, all other values will be ignored and the resulting data section will contain only two columns.

The screenshot shows the 'Parser' dialog box with the 'Properties' tab selected. The 'Destination data section' is set to 'ParsedOrders'. The 'Source text' field contains the command '\$DATA('FileIn','CONTENTS', 0)\$'. The 'Trim spaces' checkbox is checked. The 'Text qualifier' field is empty. The 'Use column names from first row of source' checkbox is unchecked. The 'Column type' section has 'Delimited' selected. The 'Delimiters' section shows 'Column delimiter' as 'Comma' and 'Row delimiter' as 'CRLF'. The 'Column definitions' table lists several columns, with 'OrderId' and 'OrderStatus' checked for use.

Use	Name	Length	Type	Format
<input type="checkbox"/>	UniqueID	0	Varchar	
<input type="checkbox"/>	AuditDate	0	Date	
<input type="checkbox"/>	AuditedBy	0	Varchar	
<input type="checkbox"/>	LastModified	0	Varchar	
<input checked="" type="checkbox"/>	OrderId	0	Varchar	
<input checked="" type="checkbox"/>	OrderStatus	0	Varchar	
<input type="checkbox"/>	Company	0	Varchar	
<input type="checkbox"/>	ContactName	0	Varchar	
<input type="checkbox"/>	OrderDate	0	Varchar	

Tip 2 - Once you have extracted the data using the parser module and no longer need to manipulate the source document use the link properties to remove the message or data section from the process after parsing the contents.

Scheduling process execution

When you build a process that is set to run at specified scheduled intervals you have the option to specify when the process should run, for example at 10 minute intervals starting at 9am and finishing at 6pm. The EMF server will execute the processes at the time you have specified, or a calculated interval from your start time so therefore it follows that if you specify the same schedule for all your processes then you will force the server to try and execute all processes at the same time which will clearly be less than ideal.

Tip 1 - Do not always use the default settings for start time when you configure a scheduled process

Tip 2 - Do not use rounded times or frequencies for scheduling intervals unless absolutely necessary. So for example set a process to start at 13:27:13 and with a frequency of 11 minutes rather than a start time of 13:30:00 with a frequency of 10 minutes.

Recipient Administration

You can view and manage details of the recipients of the EMF system using the **Recipient administration** section of the EMF tree view. You can:

- [Add new recipients, and edit or remove details of existing recipients](#)
- [Create groups of recipients](#)
- [Create aliased recipients](#), so that recipient lists are created dynamically at EMF Process runtime

Adding, editing and managing recipients

You can use the following manner to get recipient information into the EMF system:

- By adding each recipient's details individually using the Recipients section of Recipient Administration in the EMF tree view. For an overview of adding and managing EMF recipients, see [Managing Recipients](#).

When you have all the details of your users in the EMF system, you can create groups (see [Managing Groups of Recipients](#)).

For general information, see [Managing Recipients](#).

[Creating and Managing Aliased Recipients](#)

Managing Recipients, Groups and Aliases

You can use the **Recipient administration** section of the EMF tree view to manage your EMF users, including creating and removing users, and creating groups.

The **Recipient administration** section contains three icons:

- **Recipients** contains details of all the individual recipients that can be used when sending messages, including name and contact information and details of all the devices that EMF can use when sending them messages. You can add, remove and modify recipient details, as well as assign alias values. See [Managing System Recipients](#).
- **Groups** contain details of all the recipient groups that you have. You can create and remove groups and alter their membership. See, [Managing Groups of Recipients](#).
- **Aliases** allow you to create and delete "aliased recipients" so that the recipients of EMF Processes are selected dynamically when the EMF Process runs. See, [Creating and Managing Aliased Recipients](#).

[Recipient Management](#)


Managing Recipients

To add, delete or modify EMF recipients:

1. Double-click the **Recipient administration** icon in the EMF tree view and then double-click the **Recipients** icon to display a list of all the existing recipients in the EMF system.
2. **To delete a recipient**, right-click the recipient name and select **Delete** to remove the selected recipients from the repository. All outstanding mail messages, and any other references to those recipients, are also removed.
3. **To create a new recipient**, right-click in the list view and select **New recipient**. **To modify an existing recipient**, double-click its name in the **Recipient name** column. The **Recipient** screen is displayed.

Recipient

Recipient Devices Aliases Digital ID

Recipient name 

Title

First name


Last name

Salutation

Organisation

Department

Telephone number

Preferred language 

Address line 1

Address line 2

City

State

Zip code

Country

OK Apply Cancel Help

4. The **Recipient** tab of the Recipient screen is for personal details (including the address) on the individual that is to be brought into the EMF system (the [Aliases](#) tab is used to define different areas of information about recipients, and the [Devices](#) tab is used to specify all the different devices and services that can be used to contact the recipient). Enter the relevant information as follows:

- **Recipient name:** The name that the person that will receive the output from the process. It is used purely for identification purposes and need not be the person's real name.

Note: The only mandatory fields on this screen are **Recipient name**, which is used for identification in EMF, and **Language**, which is the primary language that the recipient will be using. All the other fields on this screen are purely for your information.

- **Language:** This is the default language that will be used to send EMF processes to the recipient.

The following fields are optional and primarily for your information. However, if an output module can include this information (for example, if an output module uses a postal address), it will be used.

- **Title:** If you enter a title (for example, Mr, Ms), the recipient's name will be preceded by this when an output module that can use this information sends an EMF Process.
- **First name:** If you enter a first name (for example, Matilda), the recipient's first name will be used when an output module that can use this information sends an EMF Process.
- **Last name:** If you enter a last name (for example, Smith), the recipient's last name will be used when an output module that can use this information sends an EMF Process.
- **Organization:** If you enter an organization (for example, Smith Software), this will be used when an output module that can use this information sends an EMF Process.
- **Department:** If you enter a department (for example, Research & Development), this will be used when an output module that can use this information sends an EMF Process.
- **Telephone number:** If you enter a standard telephone number, this will be used when an output module that can make a standard telephone call sends an EMF Process.
- **Salutation:** If you enter a salutation (for example, Hello), this will be used when an output module that can use this information sends an EMF Process.
- **Address Line 1:** The house number (or building name) and street name associated with the recipient.
- **Address Line 2:** The name of the district where the recipient lives.
- **City:** The name of the town or city where the recipient lives.
- **State:** The name of the state, county or area.
- **Zip Code:** The zip code or post code.
- **Country:** The country where the recipient lives.

Note: The country has no effect on time zone settings. To allow a recipient to specify their time zone, see [Creating a Recipient Profile](#).

5. Select the [Devices](#) tab to enter details of the various different EMF Process mechanisms for the recipient, and select the [Aliases](#) tab to view and edit the recipient's aliases.

[Recipients Administration Devices Tab](#)

[Recipients Administration Aliases Tab](#)

[Recipients Administration Digital ID Tab](#)

[Back to Start of Recipient Administration](#)

Defining a Recipient's Devices

You can use the **Devices** tab of the Recipient screen to specify details of all the various different delivery methods that can be used to notify a recipient with an EMF Process.

To view the Devices tab:

1. Double-click the **Recipient administration** icon in the EMF tree view and then double-click the **Recipients** icon to display a list of all the existing recipients in the EMF system.
2. Double-click the required recipient to open the [Recipient screen](#), and then select the **Devices** tab.
3. When you create a new recipient, the standard delivery method types (Email, Fax, IP, Pager, Queue, HTTP, and SMS) are automatically added to the **Devices** tab. These are the default devices for each delivery type. You can add another instance of a delivery type (for example, a second email address), but you cannot change the default device. You can, however, modify the details of the default device to point to a different device.
4. **To enter or modify details -or a delivery method:** Click the **Advanced** tab, highlight the item in the list and click **Edit** to display the [Edit Recipient Device Screen](#).
5. If you do add further instances of a delivery method and later find you no longer require them, you can remove them by selecting them and clicking **Delete**. You cannot remove the default devices, although you do not have to enter any information about them (For example, if the recipient does not have an SMS device, you must leave the empty item in the list).

[System Recipient Administration](#)

[System Recipients Administration Recipients Tab](#)

[System Recipients Administration Screen](#)

[Recipient Management](#)

Assigning Aliases to System Recipients

You can use the **Aliases** tab of the Recipient screen to assign one or more aliases to your users. These aliases are then used by [Aliased Recipients](#) modules to determine the recipients of an EMF Process dynamically when it runs.

To view the Aliases tab:

1. Ensure that you have created one or more Aliases using the [Aliases](#) section of the EMF tree view.
2. Double-click the **Recipient administration** icon in the EMF tree view and then double-click the **Recipients** icon to display a list of all the existing recipients in the EMF system.
3. Double-click the required recipient to open the [Recipient screen](#), and then select the **Aliases** tab.

The **Aliases** tab shows the association of a value, such as an account number that is unique to a recipient, with an alias, such as customer account.

Note: The **System alias** is an alias automatically defined for each recipient in the system. The value of the system alias is the same as the **Recipient name**.

4. **To Assign an Alias to a Recipient**, click the **Add** button to display the **Add alias value** screen.
5. Select the required **Alias** from the drop-down list of the aliases that you have created, and then enter an **Alias Value** for this recipient. Click **OK** to close the Add alias value screen.

[Creating and Managing Aliased Recipients](#)

[System Recipients Administration Screen](#)

[System Recipient Administration](#)

[Back to Start of Recipient Management](#)

Assigning a Digital ID to a Recipient

A **Digital ID** for a recipient is a recipient's own unique identity. The **Digital ID** can be used to encrypt messages so that the message can be read only by the recipient it was intended for. In EMF, currently, only the email module supports encrypting of messages.

A **Digital ID** is created by importing a certificate into EMF. EMF supports the main standards of certificates/encryption, which are, **S/MIME** and **OpenPGP**.

S/MIME (Secure / Multipurpose Internet Mail Extensions) is a standard for public key encryption and signing of e-mail encapsulated in MIME. S/MIME is similar to, but incompatible with, OpenPGP. S/MIME functionality is built into the vast majority of modern email software and interoperates between all of the following (and others):

- Microsoft Outlook (since Outlook 98)
- Microsoft Outlook Express (since 1999)
- Apple Mail (Since Mac OS X v10.3 Panther)
- Mozilla (mail) (all releases after 0.9.7)
- Mozilla Thunderbird (all releases)
- Netscape Communicator (since 3.0)
- Lotus Notes (since release 5.0)
- NovellGroupWise (since 1998 with the 5.5 release)

Certificates for **S/MIME** must be obtained from a trusted source, such as [Thawte](#) or [VeriSign](#).

Certifications for **OpenPGP** can be created using freely available tools, such as [GnuPG](#). Plugins implementing OpenPGP functionality are available for many popular e-mail applications (such as Outlook, Outlook Express, Eudora, Evolution, Mutt, Mozilla Thunderbird, Apple Mail, and many others).

To assign a Digital ID:

1. Double-click the **Recipient administration** icon in the EMF tree view and then double-click the **Recipients** icon to display a list of all the existing recipients in the EMF system.
2. Double-click the required recipient to open the [Recipient screen](#), and then select the **Digital ID** tab.
3. To assign an S/MIME certificate, press the **Import** button next to S/MIME and select the recipients S/MIME DER encoded binary X.509 certificate to import.

To assign an OpenPGP certificate, press the **Import** button next to OpenPGP and select the OpenPGP public keyring file that contains just the single public key required. This file should be binary and not ASCII (so if exporting the key from using gpg do not use the -a option).

[Obtaining a S/MIME certificate and assigning it to a recipient](#)

[System Recipients Administration Screen](#)

[Email Module](#)

[System Recipient Administration](#)

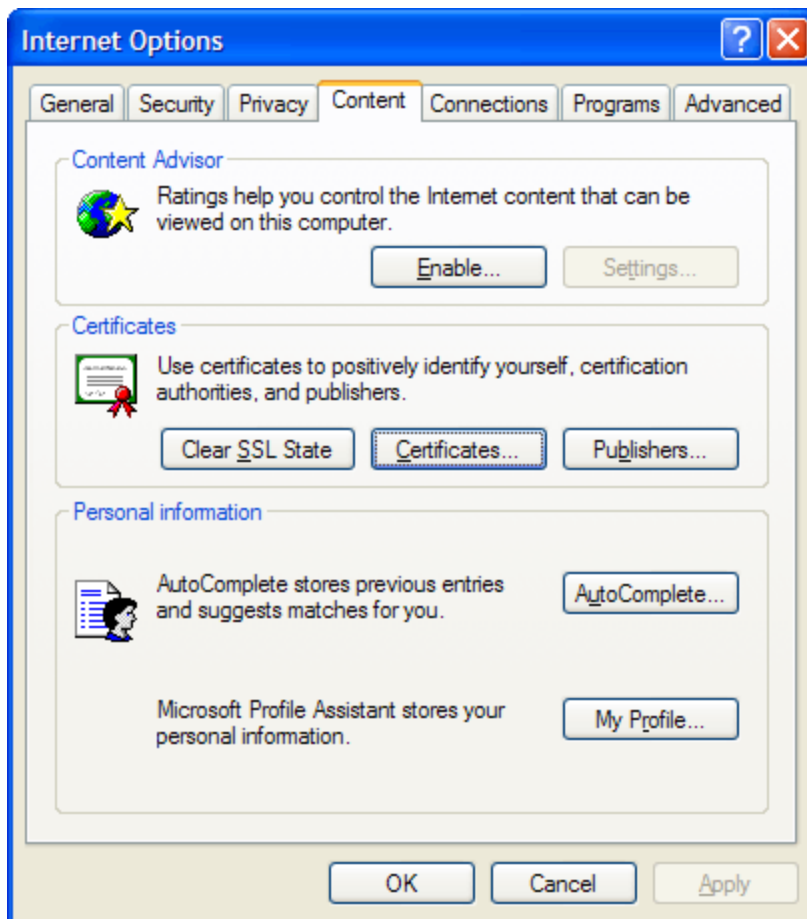
[Back to Start of Recipient Administration](#)

Example of obtaining a S/MIME certificate and assigning it to a recipient

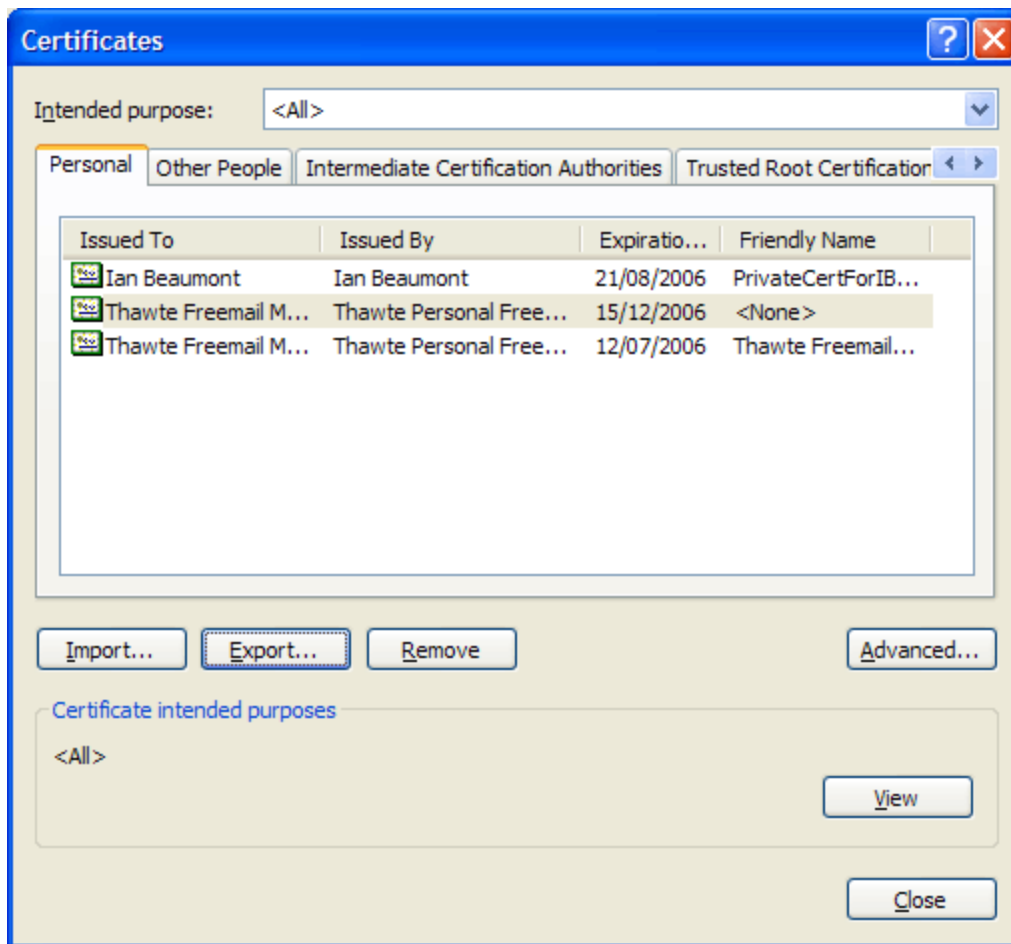
The following steps describe the procedure to obtain a S/MIME digital certificate and assign the public key from it to a recipient in EMF. This can then be used to send an encrypted email to that recipient.

1. S/MIME certificates must be issued by a trusted source. One such source is [Thawte](#), they also issue free personal certificates which we will use for this example. Go to the Thawte site and sign up for a [Personal certificate](#). After you have gone through all the necessary steps and followed the instructions, the certificate will eventually be saved into your certificate store on your computer.
2. To extract the public key certificate that can then be associated with an EMF recipient, in Microsoft Internet Explorer, select **Tools->Internet Options** and then select the

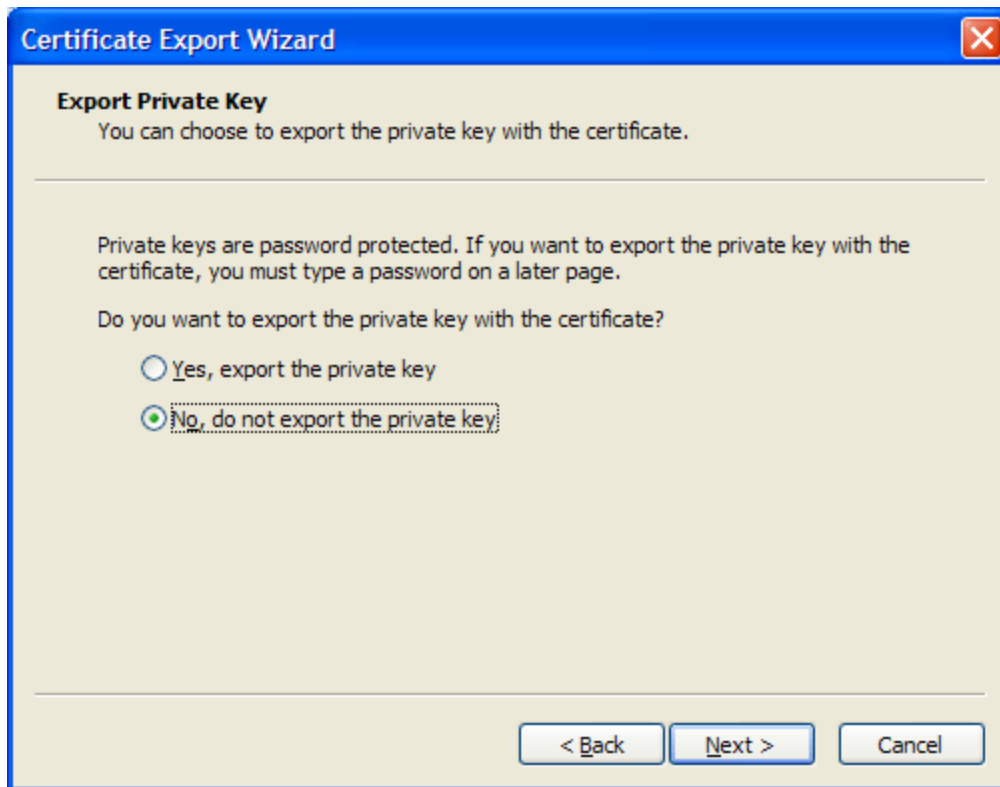
Content tab.



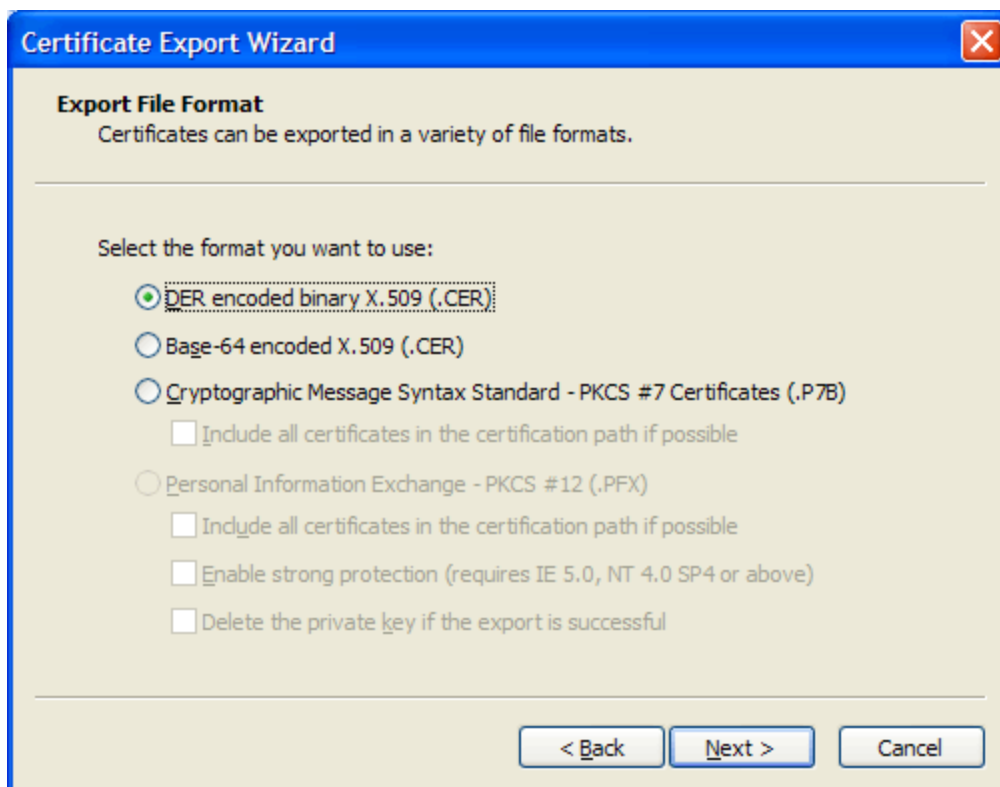
3. Press the **Certificates** button and then select the certificate that you want to use.



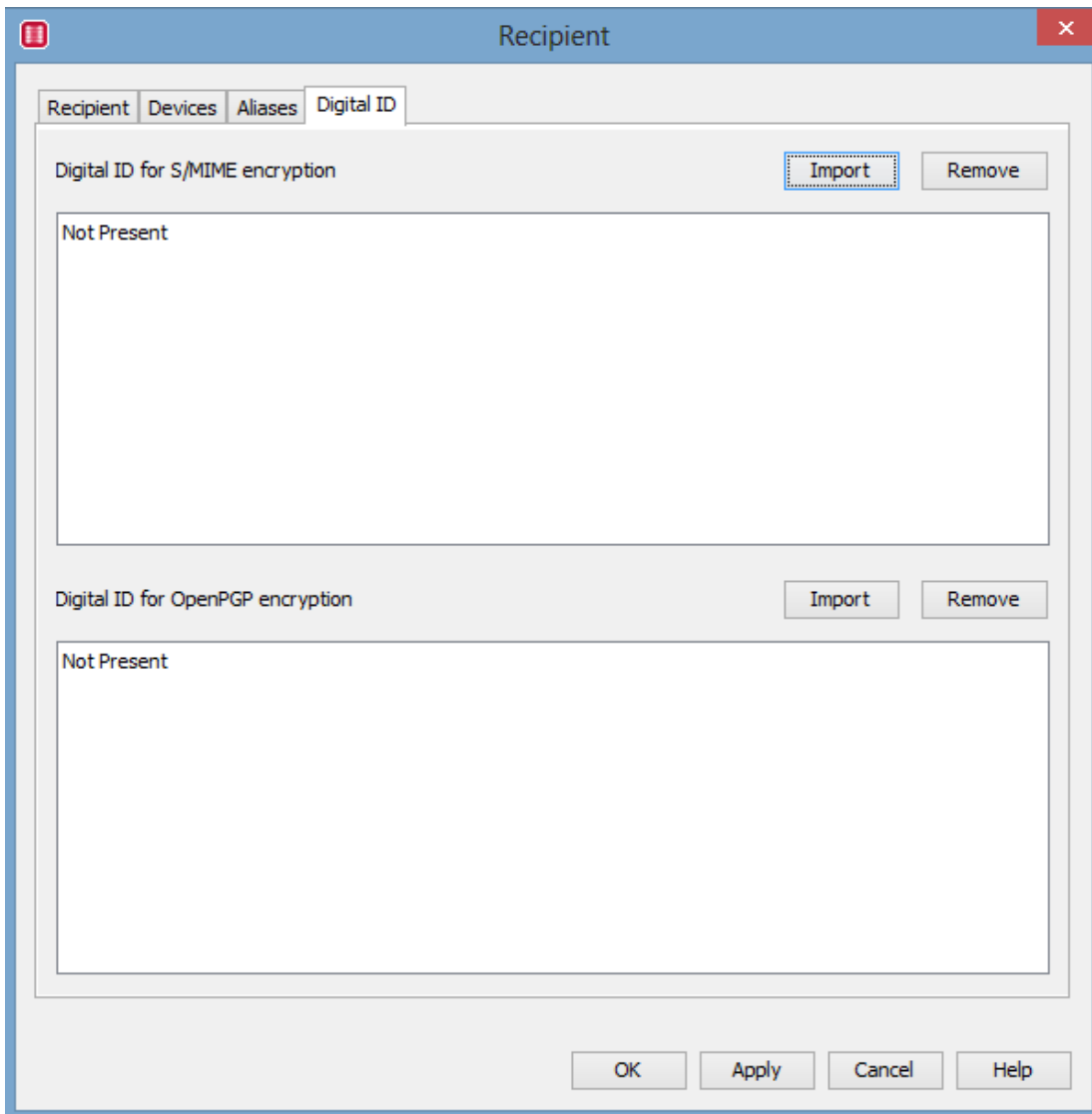
4. Press the **Export...** button and then when you get the option to "Export Private Key", select "No". Only the public key is required by EMF.



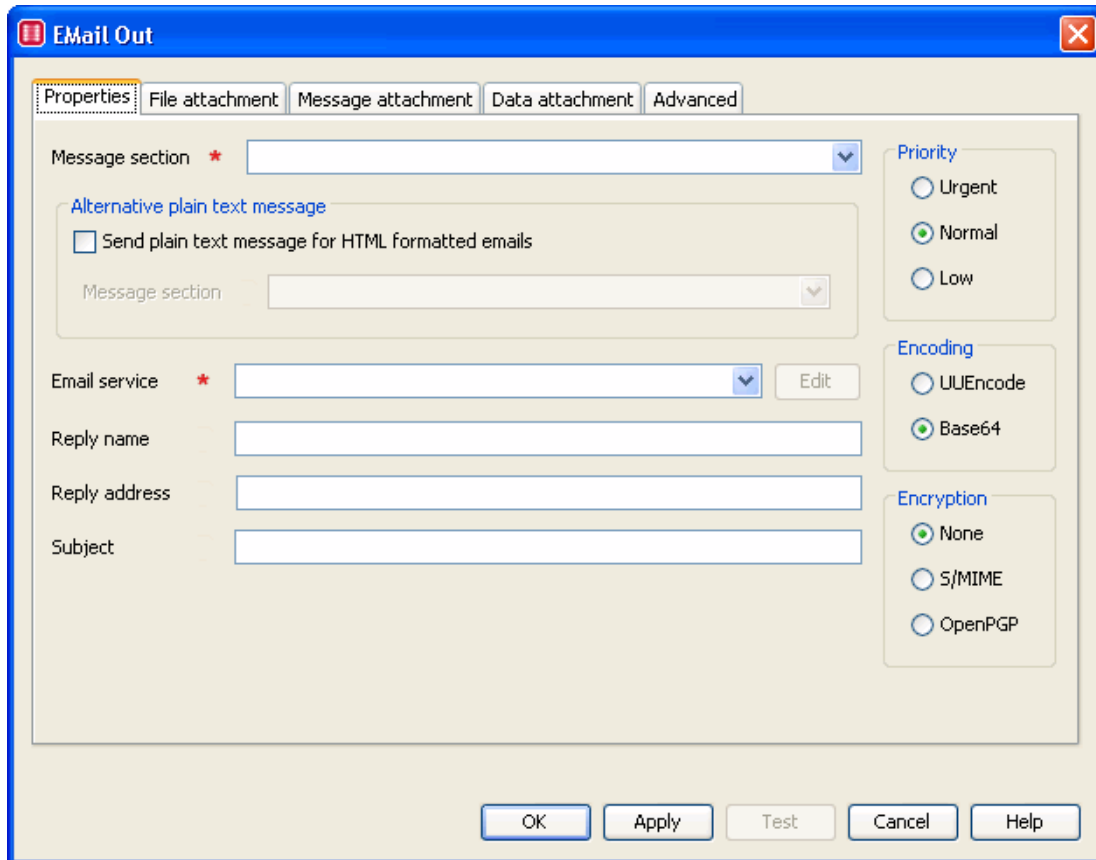
5. When you get the option on "Export File Format", select "DER encoded binary X.509". You should now be able to save off the certificate file.



6. Transfer this certificate to a computer that has access to the EMF administrator console. Select the user profile dialog in the administrator console for the recipient with the email address that you created the certificate for. Select the **Digital ID** tab and press the **Import** button for S/MIME encryption. Select the certificate you created.



7. You should now be able to build an alert and send this recipient an S/MIME encrypted email. The only extra thing that needs to be done over a normal alert that sends an email is to select the **S/MIME** radio button in the **Email output module**.



[Digital IDs](#)

[Email Module](#)

[System User Administration](#)

Managing Groups of Recipients and Groups

You can use **groups** to send the same EMF Process to several recipients and groups of recipients at the same time. You can add a new group, or edit or remove the details of an existing group, using the **Groups** screen (in Recipient administration) in the EMF Tree view.

When you first create a recipient, the recipient is assigned by default to the **All recipients** group. The **All recipients** group can be used to populate a **Recipient Section** within an EMF Process. All system recipients will be sent the EMF Process.

You can create as many recipient groups as you wish, and each recipient can belong to any number of groups.

There is only one default group in EMF:

- **All recipients:** New recipients are automatically added to the **All recipients** group.

[Groups Screen](#)

[Back to Start of Recipient Administration](#)

Naming and Describing a Group

You can use the **Properties** tab of the Groups screen to enter a name and descriptive information for the group. This name is used throughout the EMF system.

To view the Properties tab:

1. Double-click the **Recipient administration** icon in the EMF tree view and then double-click the **Groups** icon to display a list of all the existing groups in the EMF system.
2. Double-click the required group to open the **Groups** screen. The **Properties** tab is displayed (for information about the other tabs, see [Recipients](#), and [Groups](#)).
3. Enter a **Name** for the group that will be used as a reference to the group throughout EMF.
4. If you want, you can enter a **Description** that defines what the group is to be used for.

[Group Administration](#)

[Back to Start of Recipient Management](#)

Adding Recipients to a Group

You can use the **Recipients and Groups** tab of the [Group screen](#) to define which recipients and groups should be placed in the group.

To view the Recipients tab:

1. Double-click the **Recipient administration** icon in the EMF tree view and then double-click the **Groups** icon to display a list of all the existing groups in the EMF system.
2. Double-click the required group to open the Groups screen and select the **Recipients** tab (for information about the other tabs, see [Properties](#) or [Groups](#)).

The **All recipients** panel on the left of the **Recipients** tab shows all available recipients, and the **Selected recipients** panel on the right shows the recipients that are currently in the group.

- **To add a recipient to the group:** Select the required recipient and click the > button.
- **To add several recipients to the group:** Hold down the SHIFT or CTRL keys while selecting the recipients, and then click the > button.
- **To add all recipients to the group:** Click the >> button.
- **To remove recipients from the group:** Select the required recipients in the right-hand pane and use the < or << buttons.

[Groups Screen](#)

[Group Administration](#)[Back to Start of Recipient Administration](#)

Adding Groups to a Group

You can use the **Groups** tab of the [Group screen](#) to define which groups should be placed in the group.

To view the Groups tab:

1. Double-click the **Recipient administration** icon in the EMF tree view and then double-click the **Groups** icon to display a list of all the existing groups in the EMF system.
2. Double-click the required group to open the Groups screen and select the **Groups** tab (for information about the other tabs, see [Properties](#) or [Recipients](#)).

The **All groups** panel on the left of the **Groups** tab shows all available groups, and the **Selected groups** panel on the right shows the groups that are currently in the group.

- **To add a group to the group:** Select the required group and click the > button.
- **To add several groups to the group:** Hold down the SHIFT or CTRL keys while selecting the groups, and then click the > button.
- **To add all groups to the group:** Click the >> button.
- **To remove groups from the group:** Select the required groups in the right-hand pane and use the < or << buttons.

[Groups Screen](#)[Group Administration](#)[Back to Start of Recipient Administration](#)

Creating and Managing Aliased Recipients

Sometimes you may need to create and run an EMF Process where the recipients cannot be known in advance, or may change over time. In a situation like this, you can use **Aliases** (previously known as Dynamic Aliasing) to dynamically identify the recipients that should receive the EMF Process and then use an [Aliased Recipients](#) module to add them to the list of recipients.

You can use an Alias to define custom information that uniquely identifies each recipient, such as a customer account number. The Aliased Recipients module then compares the alias values against the results returned in a defined column of a data section and, when a match is found, the system recipient associated with the alias value is added to the list of recipients. For an example usage, [see below](#).

To create a new alias:

1. Right-click in the **Recipient Administration > Alias** section of the EMF Tree view and select **New Alias** to display the **Aliases** screen.
2. The Aliases screen allows you to define the alias names that can be used by [Aliased Recipients Modules](#). Recipients are associated with the aliases and can have messages delivered to them by output modules which receive their recipient lists from these Aliased recipients modules.
3. Assign a unique **Name** to the alias.
4. If you want, you can enter a brief **Description** of the Alias. Note that this is an optional field and is purely for your reference, it is not used elsewhere.
5. Click on the **OK** button to save your changes and close the screen.
6. Add this alias to all of the recipients that might require the EMF Process to be sent to them using the [System Recipients Administration Alias Tab](#), and in each case enter a unique Alias value.

You can now add an Aliased Recipients module to your EMF Process.

An example of using Aliases:

Your bank wishes to send an EMF Process to its customers who have less than \$500 in their accounts. When the EMF Process is built, there is no way of knowing which customer will receive the EMF Process on any given day, so you build the EMF Process using Aliased Recipients.

Following the procedure above, you create an Alias called e.g. "Bank account number". You then add this alias to the recipient information Alias tab of all the recipients who have an account with your bank and in each case enter their account number as the Alias value.

You then create an EMF Process containing a data section (called e.g. "Balances") that will return a list of the bank account numbers with a balance of less than \$500, and then add an [Aliased recipients](#) module to the EMF Process after the "Balances" data section.

- In the Aliased recipients module: Select **Balances** from the drop-down list of the data sections in the EMF Process that precede the Aliased Recipients module, select **Bank account number** from the drop-down list of the Alias types that you have created, and specify the **Lookup column** in the data section that contains the account number.

When the EMF Process runs, EMF will use the bank account number alias values to cross-reference with the system recipients in order to correctly identify which customers should receive the EMF Process.

[Aliased Recipients Module](#)

[Back to Start of User Management](#)

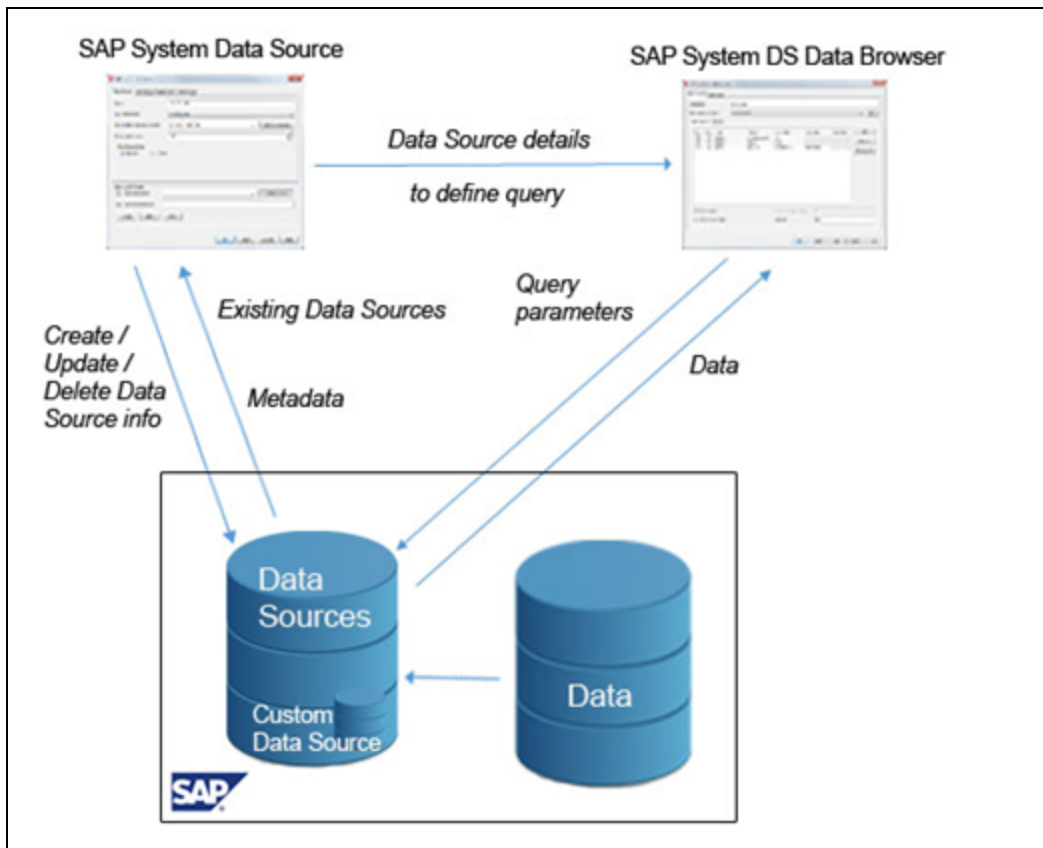
Overview of SAP System Data Sources

The SAP System Data Source Management allows you to set up a data source on the SAP System which can be then used to extract data from one or multiple SAP tables using joins.

The [SAP System DS Data Browser module](#) has a number of advantages over the [SAP Data Browser module](#) as listed:

- Ability to extract data from multiple tables
- Shorter Processing Time
- More efficient SAP Function Modules used to extract table data
- No limit to the length of each extracted row
- SAP System based batch processing

The SAP System Data Source defines the table(s) fields to return in the output and fields to use as query options to filter the output data. The SAP System DS Data Browser module then uses the SAP System Data Source, filtering data with the query options if required, to return SAP table data back to the EMF process.



For the SAP System Data Source to work, the SAP Data Extraction component has to be installed on the SAP System. For more information, refer to [Setting up the EMF SAP System Data Source on the SAP System](#).

Note: EMF requires SAP Java Connector 2 (SAP JCo 2) or SAP Java Connector 3 (SAP JCo 3)

to connect to the SAP back-end. SAP JCo supports connections to 4.5 B, 4.6B, 4.6C, 4.6D, 4.7, 6.20 and higher. The SAP Authorizations required for creating the SAP System Data Source are detailed in the *Setting up the EMF System Data Source on the SAP System Guide*.

To extract data using the SAP System Data Source:

1. Ensure you have installed the [Integrated SAP Transport](#) on the SAP System.
2. Create an SAP Connection and setup the SAP Java Connector, as detailed in the [Creating SAP Connections](#).
3. Create an SAP System Data Source to define the query for the data to extract as detailed in the [Creating SAP System Data source](#).
4. Build a process using the [SAP System DS Data Browser Module](#) to extract the data required. The module uses the data source defined in **Step 3**.
5. Run the new process as any other EMF process.

[Creating SAP System Data Source](#)

[Examples](#)

Importing Integrated SAP Transport

The transport files are created using SAP ERP 6.0. The minimum version of SAP required to import these transport files is SAP ERP 6.0 (Technology Platform SAP NetWeaver 7.0).

The following is a general procedure of how to import an ABAP solution. A similar approach can be used to import the Extraction Framework in an SAP ERP System.

Note: This is typically an SAP Basis Activity and must be performed by a Basis Administrator only.

To import an ABAP solution

1. Extract the transport file
`SupportFiles\SAPTransport\SAPDataExtractFramework.zip` (found on the EMF install media) containing Cofiles and Datafiles.

This will result in the following two files:

- **K Type Transport**

This file starts with the letter K followed by some numbers and a Q11 file extension (e.g. K900521.Q11). This is a Cofile which does not contain much data. It has the attributes of the data file stored in it as well as command or change request information files that include information on the transport type, object classes, required import steps, and post-processing exit codes.

- **R Type Transport**

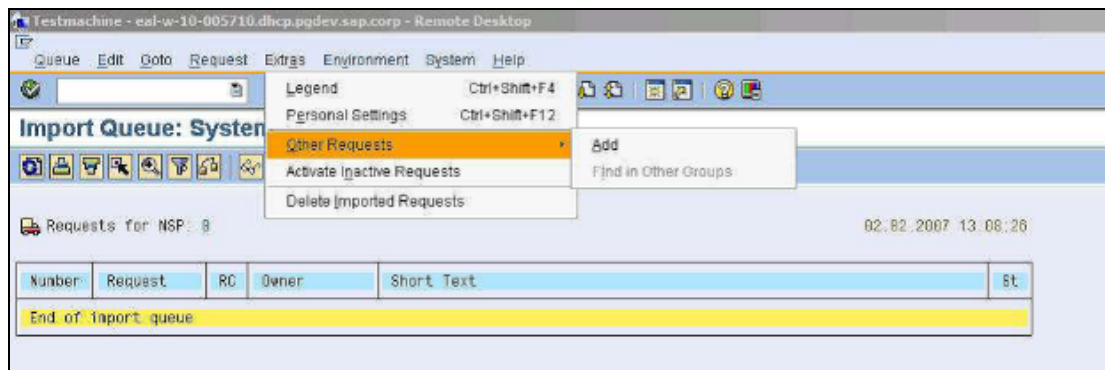
This file starts with the letter R followed by some numbers and a Q11 file extension (e.g. R900521.Q11). This is a Datafile which contains the actual data for the transport – the changes will be made in your system.

2. Navigate to the following directory on the SAP Development Server:

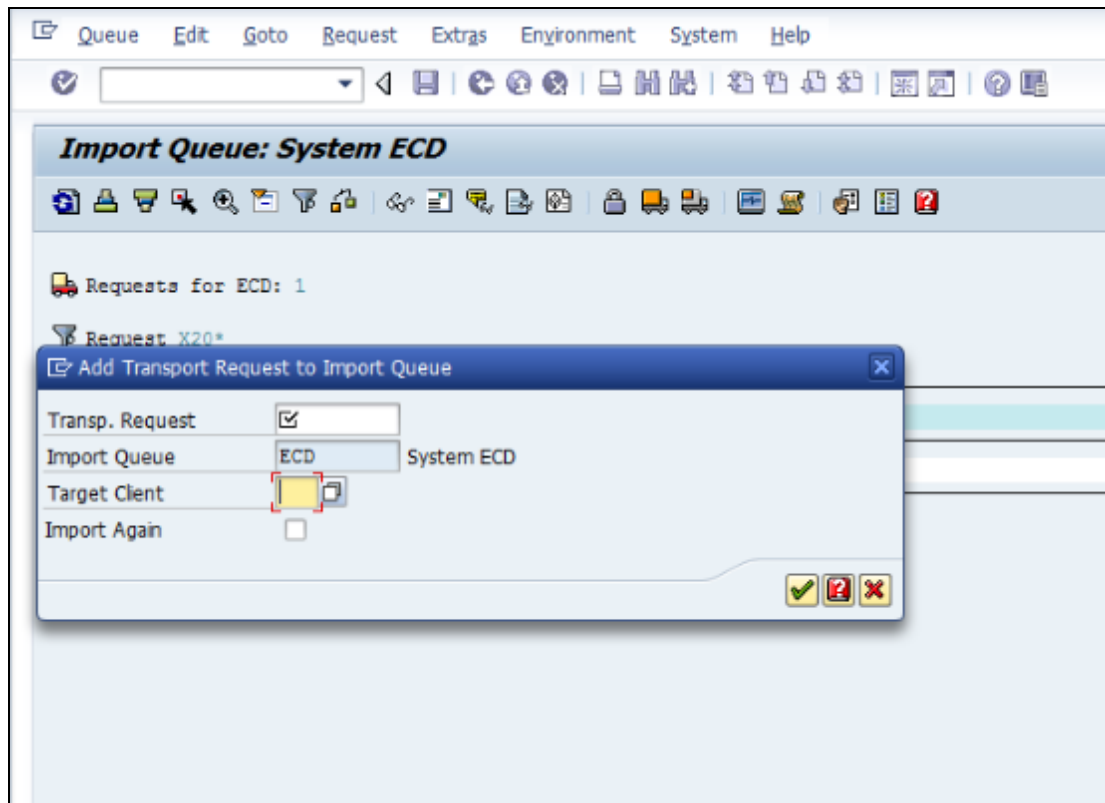
\usr\sap\trans

- Copy the K Type Transport to Cofile folder located in \usr\sap\trans\cofile
- Copy the R Type Transport into the data folder located in \usr\sap\trans\data

3. Select transaction STMS in the SAP ECC Development System.
4. Proceed to import queue (Press **F5** key and then select **Dev System**).
5. From the **Extras** menu, select **Other Requests > Add**.

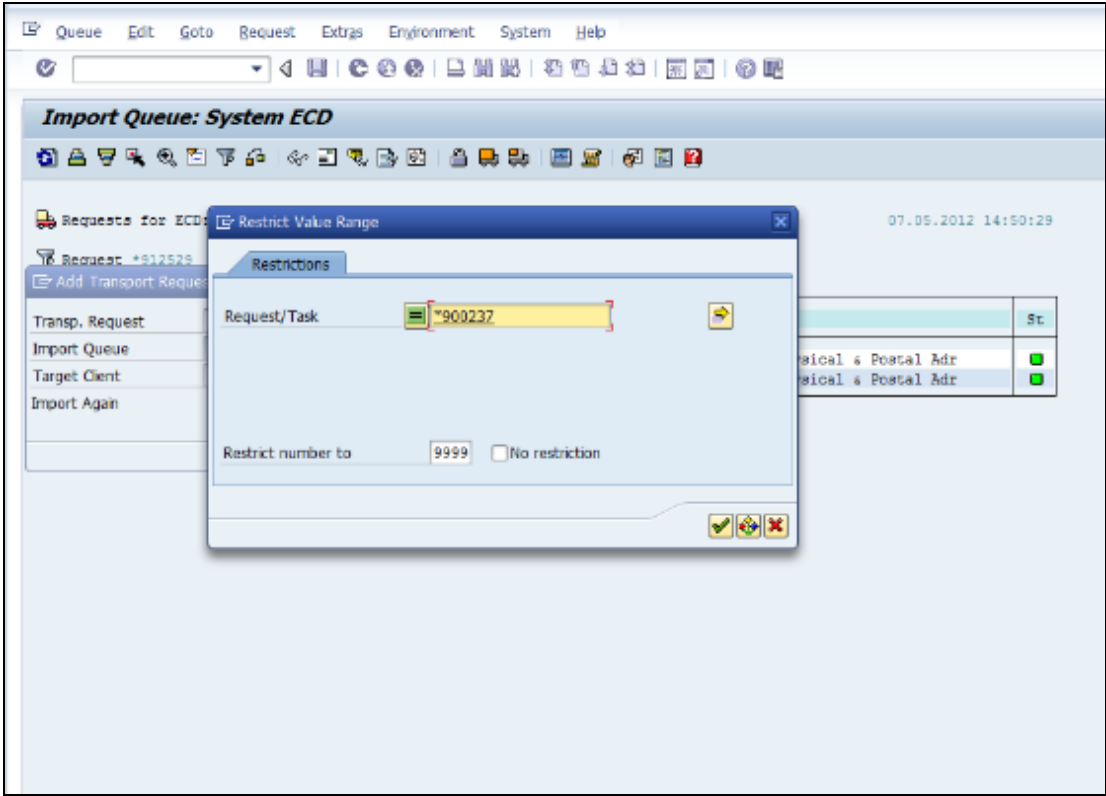


6. Enter the target client and select the **Transport Request** search button.

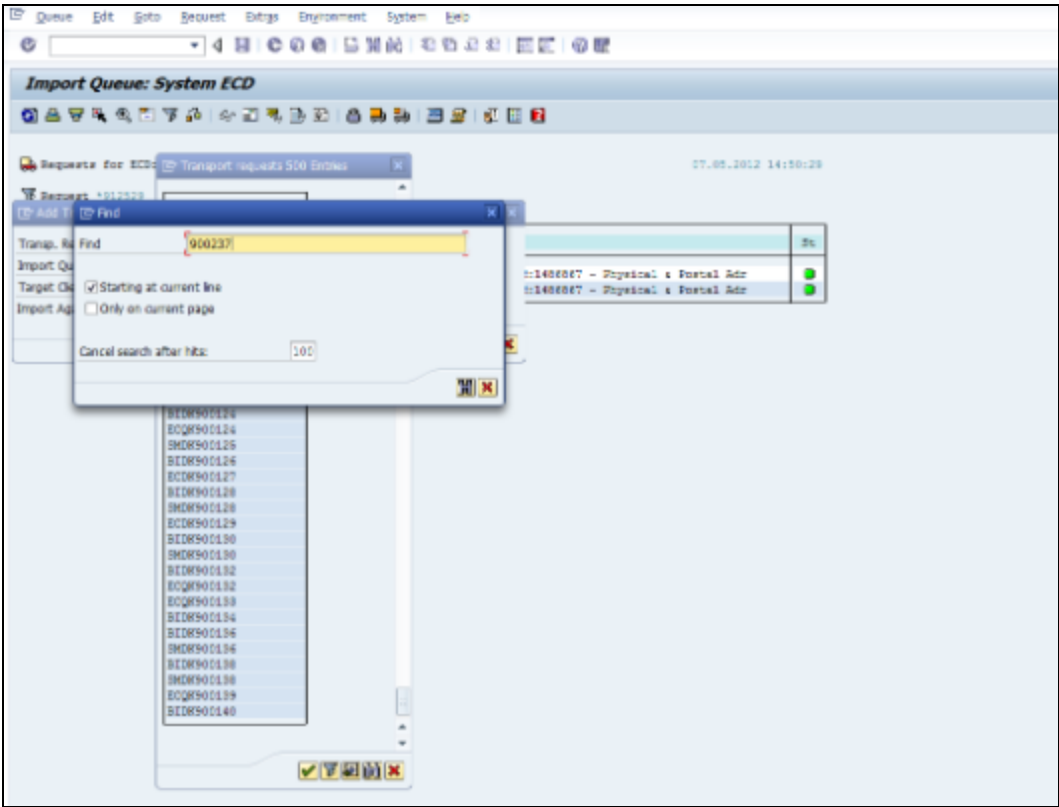


7. Search for your transport. Type a * at the start for a wild card and then type the digits in the transport file – for example, if the Transport file name is K900521.Q11, then type *900521. Ensure you increase your number restriction to a value that will give you the preferred result.

Warning: Do not include the .Q11 extension, this will likely show up as the transport prefix.

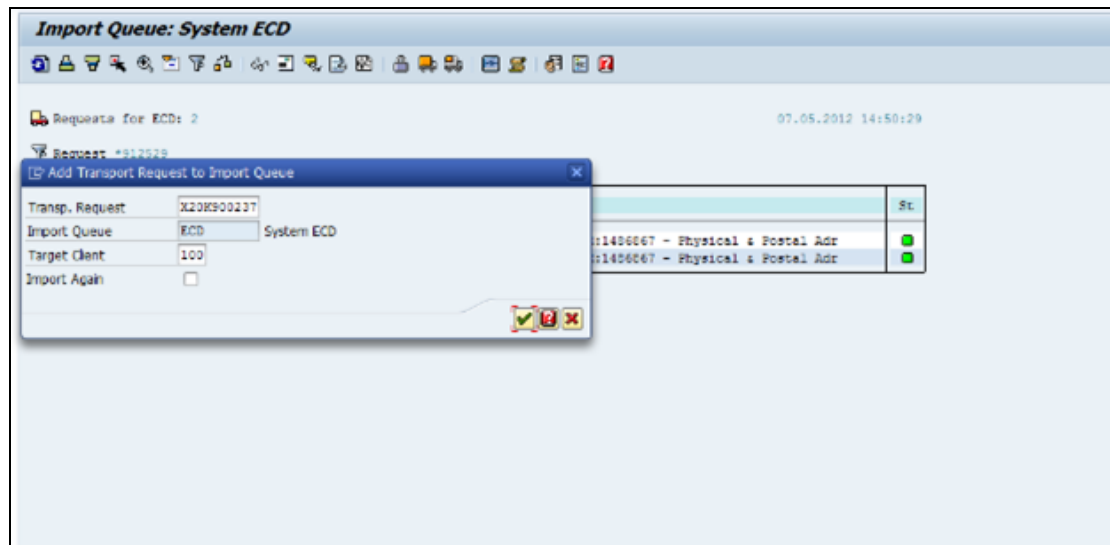


8. Find your transport number and select it.

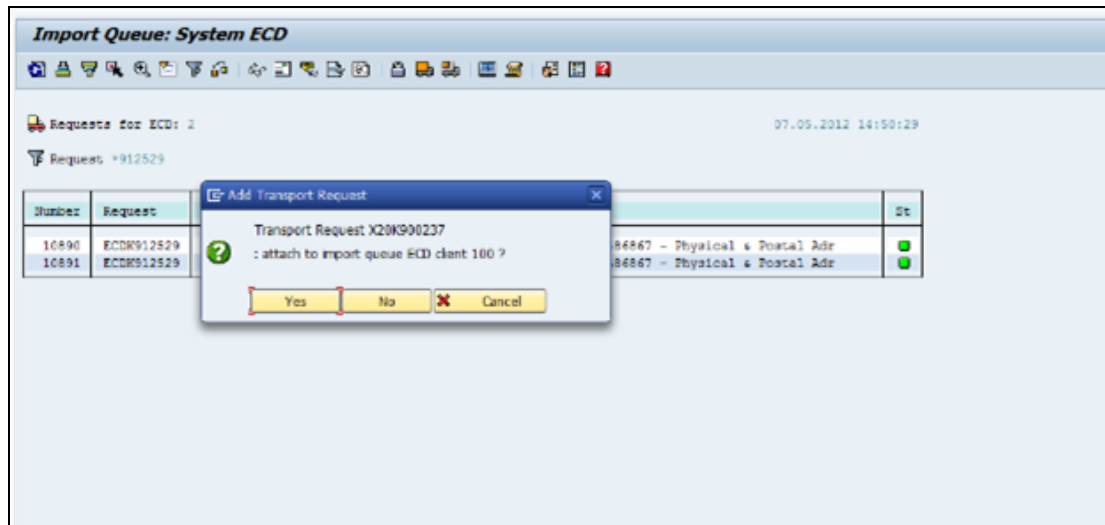


9. The following screen illustrates the data populated with the transport request, target client, and import queue.

Note: You can select import again, if you are importing the same transport

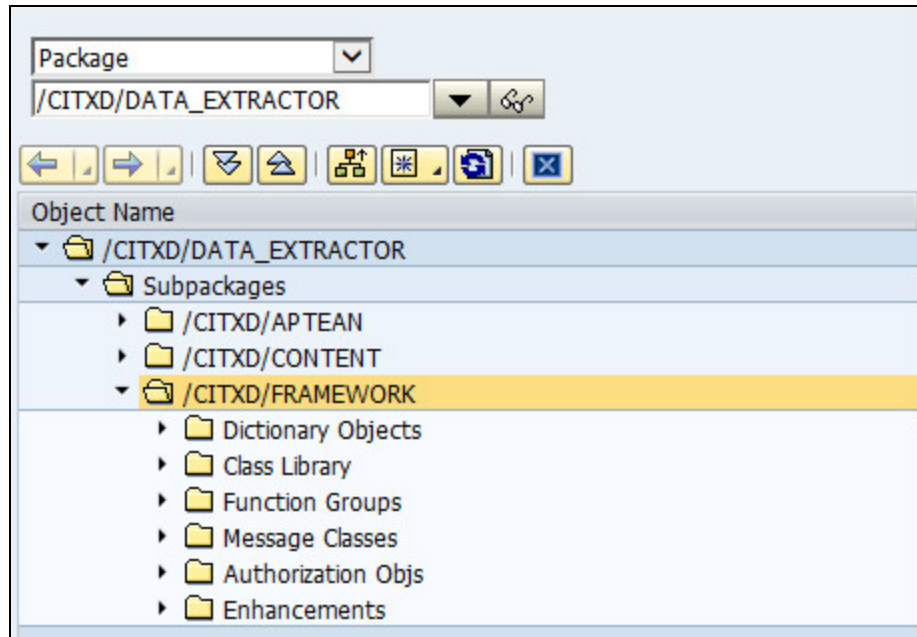


10. Click **Yes** when prompted to attach to import queue.



11. Once it is attached to the import buffers, import the transport.

After the transport has been imported, the ABAP solution will reside in the package CITXD/DATA_EXTRACTOR as displayed:



Setting up SAP User Authorizations for use in SAP System Data Source

The SAP System Data Source requires users to be granted a set of SAP authorizations dependant on the role they will be performing. There are three distinct roles that an EMF user will perform. Different users may be created in the SAP system for these different roles, or the required SAP Authorizations may be assigned to existing users (even a single user). The roles are:

- Create/maintain SAP system data sources - this is needed for all design time work in the creation and maintenance of SAP system data sources.
- Create custom SAP System data sources – this is needed at design time to create custom SAP System data sources. If custom SAP system data sources are not being created then this role is not required. The user assigned these authorizations has to be a dialog user.
- Run and extract data for SAP System data source – this is needed at runtime to run a SAP System data source that has been created to extract data.

Note: The users can only be created after the Apteian SAP Extract Solution has been installed on the SAP System, as it relies on some objects that are created as part of the installation.

Create/maintain SAP system data sources role

The user assigned the following set of authorizations will have rights to create/update/delete and query the SAP System data sources. Although the user can have any name, we shall refer to it as APTEAN_CPIC throughout.

User: APTEAN_CPIC

Password: Minimum of 11 alphanumeric characters. Other password policies may apply depending on your SAP System setup.

User Type: Communication 'C'

Following authorizations are required for the roles:

Role			
Authorization Object	Field Name	'From' Value	'To' Value
Role: /CITXD/R_DESIGNTIME			
ZCXD_DSGTM	/CITXD/DSC	*	
	ACTVT	*	
Role: Z_APTEAN_DS_ACCESS			

S_RFC	RFC_TYPE	FUGR
	RFC_TYPE	FUNC
	RFC_NAME	/CITXD/EXTRACT_DATA
	RFC_NAME	/CITXD/EXTRACT_DATA_ BATCH
	RFC_NAME	/CITXD/FG_DATA_ EXTRACTOR
	RFC_NAME	/CITXD/FG_DS_ DESIGNTIME
	RFC_NAME	/CITXD/FM_CHANGE_ DATASOURCE
	RFC_NAME	/CITXD/FM_CREATE_ DATASOURCE
	RFC_NAME	/CITXD/FM_DELETE_ DATASOURCE
	RFC_NAME	/CITXD/READ_RUNTIME_ LOGS
	RFC_NAME	/CITXD/SETUP_EXTRACT
	RFC_NAME	BAPI_TRANSACTION_ COMMIT
	RFC_NAME	BAPT
	RFC_NAME	DDIF_FIELDINFO_GET
	RFC_NAME	RS_COMPONENT_VIEW
	RFC_NAME	RFC1
	RFC_NAME	RFCPING
	RFC_NAME	RFC_GET_FUNCTION_ INTERFACE
	RFC_NAME	RFC_READ_TABLE
	RFC_NAME	SDIFRUNTIME
	RFC_NAME	SDTX
	RFC_NAME	SEU_COMPONENT
	RFC_NAME	SYST
	ACTVT	16
S_CTS_ADMI	CTS_ADMFCT	TABL
S_CTS_SADM	DOMAIN	<Domain where transport files are to be stored>
	DESTSYS	<Logical system name>
	CTS_ADMFCT	TABL
	PROGRAM	SAPLSTRF
S_DATASET	ACTVT	34
	FILENAME	<location where transport files will be stored>e.g. \\Q11ERP70\sapmnt\trans*

	DICBERCLS	&NC&
S_TABU_DIS	DICBERCLS	SS
	ACTVT	03
	ACTVT	03
S_TABU_NAM	ACTVT	02
	TABLE	TADIR
	DOMAIN	''
S_SYS_RWBO	DESTSYS	<logical system name>
	TTYPE	DTRA
	ACTVT	01
S_TRANSPRT	TTYPE	DTRA
	ACTVT	01
Role: /CITXD/R_RUNTIME		
ZCXD_RUNTM	/CITXD/DSC	*
	ACTVT	*

Create custom SAP system data sources role

A user only requires these authorizations if creating Custom data sources. They need developer access in order to create the ABAP code that will run the Custom SAP System data sources for non-transparent tables like BSEG. This user can either be created from new or an existing SAP user with developer access can create the ABAP code as required.

Use an existing SAP User:

The existing SAP user has to be given the following:

- Developer access for writing ABAP code
- Authorization to create, read, update and delete objects in the Aptean Solution package /CITXD/DATA_EXTRACTOR

Create a new SAP user:

If creating a new SAP user, use authorizations below. Note that the authorizations below are recommended, not mandatory. It is up to each SAP System admin to determine what access a developer needs to be able to write ABAP code in the relevant business objects area and packages. Roles can also be created or changed, provided the authorization objects below are used. Although the user can have any name, we shall refer to it as APTEAN_TEST throughout.

User: APTEAN_TEST

Password: As per password policies in your SAP System setup.

User Type: Dialog 'A'

Role

Authorization Field Object	Name	'From' Value	'To' Value
-----------------------------------	-------------	---------------------	-------------------

Role: /CITXD/R_DESIGNTIME
 ZCXD_DSGTM /CITXD/DSC *
 ACTVT *
 Role: Z_APTEAN_ABAP_ACCESS
 S_CTS_ADMI CTS_ TABL
 ADMFACT
 DEVCLASS Z*
 OBJTYPE Z*
 S_PACKSTRU OBJNAME DEVC
 P_GROUP *
 ACTVT 02
 DEVCLASS *
 OBJTYPE *
 S_DEVELOP OBJNAME *
 P_GROUP *
 ACTVT 03
 DEVCLASS Z*
 OBJTYPE *
 S_DEVELOP OBJNAME Z*
 P_GROUP *
 ACTVT L0
 DOMAIN *
 DESTSYS <logical system name where
 Aptean solution installed>
 S_SYS_RWBO TTYPE DTRA
 TTYPE TASK
 ACTVT 01
 TTYPE DTRA
 S_TRANSPRT TTYPE TASK
 ACTVT 01
 Role: Z_APTEAN_ABAP_ACCESS_1
 TCD SE16
 SE19
 S_TCODE SE37
 SE80
 SLG1

	DOMAIN	*
	DESTSYS	<Logical system name where Aptean ABAP solution installed>
	CTS_ ADMFCT	EPS1
	CTS_ ADMFCT	EPS2
S_CTS_SADM	CTS_ ADMFCT	INBX
	CTS_ ADMFCT	INIT
	CTS_ ADMFCT	PROJ
	CTS_ ADMFCT	TABL
	CTS_ ADMFCT	TADM
	PROGRAM	*
S_DATASET	ACTVT	33
	ACTVT	A6
	FILENAME	*
	PROGRAM	SAPLLOCAL_EDT1
	ACTVT	33
S_DATASET	ACTVT	34
	ACTVT	A6
	ACTVT	A7
	FILENAME	Z*
	DICBERCLS	ZAPT
S_TABU_DIS	ACTVT	02
	ACTVT	03
	DICBERCLS	/CI*
S_TABU_DIS	ACTVT	02
	ACTVT	03
	DEVCLASS	/CITXD
	OBJTYPE	*
	OBJNAME	*
S_DEVELOP	P_GROUP	*
	ACTVT	03
	ACTVT	16

	DEVCLASS	\$*
	DEVCLASS	T*
	DEVCLASS	Y*
	DEVCLASS	Z*
	OBJTYPE	*
S_DEVELOP	OBJNAME	*
	P_GROUP	*
	ACTVT	01
	ACTVT	02
	ACTVT	03
	ACTVT	06
	ACTVT	07
	DOKU_ACT	RAW_VERS
S_DOKU_AUT	DOKU_	TEST
	DEVCL	
	DOKU_	MAINTAIN
	MODE	
S_FOBU_MTH	ACTVT	*
	DEVCLASS	*
	TTYPE	PIEC
	TTYPE	TASK
	ACTVT	02
S_TRANSPRT	ACTVT	03
	ACTVT	05
	ACTVT	06
	ACTVT	43
	ALG_	
	OBJECT	/CITXD/MO
	ALG_	
S_APPL_LOG	SUBOBJ	/CITXD/DT
	ALG_	
	SUBOBJ	/CITXD/RT
	ACTVT	03
	ACTVT	06
Role: Z_APTEAN_DS_ACCESS1_FM		
S_TCODE	TCD	SE37
		SE38
	PROGRAM	*
S_DATASET	ACTVT	*
	FILENAME	*

	DEVCLASS	*
	OBJTYPE	*
S_DEVELOP	OBJNAME	/CITCD/*
	P_GROUP	*
	ACTVT	03
	ACTVT	16
	DEVCLASS	\$*
	DEVCLASS	T*
	DEVCLASS	Y*
	DEVCLASS	Z*
	OBJTYPE	FUGR
S_DEVELOP	OBJTYPE	PROG
	OBJNAME	/CITXD/*
	P_GROUP	*
	ACTVT	01
	ACTVT	02
	ACTVT	03
	ACTVT	06
	P_GROUP	*
S_PROGRAM	P_ACTION	BTCSUBMIT
	P_ACTION	SUBMIT
	P_ACTION	VARIANT
Role: /CITXD/R_RUNTIME		
ZCXD_RUNTM	/CITXD/DSC	*
	ACTVT	*

Following standard profiles also form part of the APTEAN_TEST user:

- HR250_ALL_BC_ENDUSER
- SAP_BC_ENDUSER
- TEC_BC_ENDUSER
- R3_FUNC, S_TABU_ANZ

Run and extract data from data sources in SAP System role

A user assigned the following SAP Authorizations has "Run time" rights to be able to run the queries created on the SAP System data source and extract the SAP data. This user can have any name but we shall refer to it as APTN_CPICRUN throughout.

User: APTN_CPICRUN

Password: As per password policies in your SAP System setup.

User Type: Communication 'C'

Following authorizations are required for the roles:

Role			
Authorization Object	Field Name	'From' Value	'To' Value
Role: /CITXD/R_RUNTIME			
ZCXD_RUNTM	/CITXD/DSC	*	
	ACTVT	*	
Role: Z_APTEAN_DS_ACCESS_RUN			
S_RFC	RFC_TYPE	FUGR	
	RFC_TYPE	FUNC	
	RFC_NAME	/CITXD/EXTRACT_DATA	
	RFC_NAME	/CITXD/EXTRACT_DATA_BATCH	
	RFC_NAME	/CITXD/FG_DATA_EXTRACTOR	
	RFC_NAME	/CITXD/READ_RUNTIME_LOGS	
	RFC_NAME	/CITXD/SETUP_EXTRACT	
	RFC_NAME	BAPI_TRANSACTION_COMMIT	
	RFC_NAME	BAPT	
	RFC_NAME	DDIF_FIELDINFO_GET	
	RFC_NAME	RFC1	
	RFC_NAME	RFCPING	
	RFC_NAME	RFC_GET_FUNCTION_INTERFACE	
	RFC_NAME	RFC_READ_TABLE	
	RFC_NAME	SDIFRUNTIME	
	RFC_NAME	SDTX	
	RFC_NAME	SYST	
	ACTVT	16	

Note: APTEAN_CPIC user and APTEAN_TEST authorizations can be combined, if required. The concept of different users is to allow the separation of users who are only allowed to run the SAP System data sources and those who are allowed to update the SAP System data source. For example, a Development department could be tasked with creating the SAP System data sources for use by an EMF user. In this case, the Development user would have APTEAN_CPIC and APTEAN_TEST user authorizations, while the EMF user would have APTN_CPICRUN user authorizations added.

Configuring Data Sources

Creating Connections to Data Sources

To run your EMF against client databases, you must create connections to them using either JDBC or SAP.

To view your existing data source connections:

- Select **Data sources** in the EMF tree view and then select either **JDBC connections**, **SAP connections** or **SAP System Data Sources**.

To create a data source connection:

1. Select **Data sources** in the EMF tree view and then select either **JDBC connections**, **SAP connections** or **SAP System Data Sources** as required.
2. Right-click in the list view and select **New**.
3. Enter the required information, depending on whether you created a [JDBC Connection](#), [SAP Connection](#) or an [SAP System Data Source](#).

Note: If you wish you can create a trigger to access SQL Server data sources. This trigger is fired when specified conditions are met within that data source. For SQL Server, the trigger information is sent to the EMF API, which in turn sends it via SOAP to the EMF Server.

Note: EMF uses two types of databases:

- The **Repository** databases are EMF repository databases that contain information on the EMF system (e.g. details of your email server) and any EMF Processes that you create.
- The **Client databases** include all the databases against which EMF Processes are run. EMF can connect to JDBC or SAP data sources (see [JDBC Connections](#), [SAP Connections](#) and [SAP System Data Sources](#)) and use the information that is contained within them to create Recipient sections and Data sections.

To create a new Repository database, refer to the *Event Management Framework 7 Installation and Configuration Guide*.

If you upgrade your EMF installation (or apply a service patch), you may need to upgrade your repository databases to work with the newer version - see [Upgrading your repository databases](#).

[Configuring the EMF System](#)

Creating JDBC Connections

JDBC connections are used to connect to JDBC data sources. Prior to setting up an EMF process, you must configure connections to any data sources that you will be using in the **JDBC Data source** screen of the EMF tree view.

Note: In order to create a new JDBC connection, or edit an existing one, you must have permissions to access the database. This is set up as part of your security settings, as set up by your system administrator using the [Groups](#) screen.

To create a JDBC connection

1. Click on **JDBC Connections** in **Data Sources** in the EMF tree view. Any existing JDBC connections will be displayed in the list view.
2. Right-click in the list view and select **New** to display the **JDBC Data source** screen.

JDBC Data source

Connection name: Repository Refresh metadata Clear metadata

Description:

Driver/Data source: net.sourceforge.jtds.jdbcx.JtdsDataSource

User name: EMF_3

Password:

Connect settings

URL:

Properties

Name	Value
useLOBs	false
databaseName	EMF_3
xa_verifyQuery	SELECT 1
instance	SQLExpress
serverName	banprabathpc

Schema: Browse

OK Apply Test Cancel Help

The **JDBC Data source** screen defines connections to data sources available within the computer network. Once connected to a data source, all information therein can be passed through an EMF process.

3. Enter a unique name that you wish to assign to this JDBC connection in the **Connection name** field. When selecting a JDBC connection from other areas of EMF, the value assigned to this field will be used to distinguish between multiple connections.

Warning: The name used must be unique or the connection will not be allowed.

4. Use the **Refresh metadata/Clear metadata** buttons to refresh or clear metadata, i.e., data that describes the database objects in the database being connected to (e.g. tables, views, and fields in views).

EMF stores metadata for the database to allow easy building of SQL queries. Metadata is used in the **SQL Builder** tab of the [SQL Module](#) to allow graphical building of SQL queries. If new tables are added or the database structure is updated, the metadata will need to be refreshed so that the correct structure and queries are built within the SQL Builder of the SQL Module.

5. Enter information in the **Description** field if required, in order to remind you why the connection was created. This is optional and is not used elsewhere.
6. Select the fully qualified name (the format of this string is available on your driver documentation) of the proprietary JDBC driver class that is to be used to access the data or the name of the datasource class from the **Driver/Data source** drop-down list.
7. Enter the **User name** and **Password** in the fields provided.
8. Enter the connection URL for the JDBC driver in the **URL** field. The syntax for this value is driver-dependent; refer to your driver documentation for further information. Refer to [JDBC Driver Details](#) for examples.

Important: Leave this field blank if using a Data source.

In order to connect to the repository database, and any other data sources, you must use a supported JDBC driver. See [JDBC Configuration Details](#) for more information.

9. Click **Add/Remove** to set the connection properties.

Each property name must be entered in accordance with the driver's documentation. For example: if the driver requires the database name to be specified as a property, then 'databaseName' should be entered in the **Name** column and the database's name in the **Value** column.

10. To add a schema, click **Browse** next to the Schema field. Navigate to the schema you want to add, select it, and click **OK**.
11. When all the required information has been entered, click **Test**. If the connection is successful, this will return a positive result.

[The SQL module](#)

Additional Properties and Pooling Options

You can enter extra options on the [JDBC Connection Properties](#) to force unused connections to close down after a certain period of time, and to limit the maximum number of connections that can be used (the system can then wait for a connection to become available). This can be useful where the database/driver only supports a limited number of connections.

The following additional properties are applicable and optional:

- **xa_minPoolSize**: the number of connections that will initially created in the pool, and held in the pool at all times. The default value is **-1**, which means an unlimited number.
- **xa_maxPoolSize**: the maximum number of connections that the pool will give out. The default value is **-1**, which means an unlimited number. **If this value is set to 0**, attempts to gain further connections after the minimum number of connections have been retrieved will fail. **If this value is set to any number greater than 1**, when the maximum number of connections has been retrieved the pool will wait for a connection to become available. If no connection becomes available within the time specified by `xa_maxWait` (see below) the request will fail.
- **xa_maxIdleTime**: the maximum amount of time (in seconds) that a connection can sit in the pool unused. Any connections that are still in the pool and unused after this time will be released. The default value is **600** (10 minutes).
- **xa_propertyCycle**: how frequently (in seconds) to check for connections in the pool that have been idle for the time specified in `xa_maxIdleTime` and should be removed. The default value is **600** (10 minutes).
- **xa_oracleFailoverCheck**: allows you to specify whether a check should be made against every Oracle connection to check that the connection is still valid. Enabling this option will allow the EMF server to recover in the event that the network or database fails, but can have a detrimental effect on performance. The default setting for this is **True**.
- **xa_disableInternalPooling**: prevents the server from using its own internal pooling. This allows you to use a third-party datasource that has its own pooling (e.g. `oracle.jdbc.pool.OracleConnectionCacheImpl`). The default setting for this is **False**.
- **xa_maxWait**: how long (in seconds) to wait for a connection to become available (see `xa_maxPoolSize`). If set to 0 or less it will wait indefinitely as long as `xa_maxPoolSize` is set to greater than 0. The default value is **0**.
- **xa_ignorePooledDatasourceInterface**: only appropriate for the Microsoft SQL Server 2000 Driver for JDBC. This mandatory property must be set to **True** in order for the driver to work correctly with EMF.

[Creating JDBC Connections](#)

EMF JDBC Data Source Example

JDBC Data source

Connection name:

Description:

Driver/Data source: ☒

User name:

Password:

Connect settings

URL:

Properties

Name	Value
instance	SQLExpress
databaseName	EMF_3
serverName	banprabathpc
useLOBs	false
xa_verifyQuery	SELECT 1

Schema:

[JDBC Driver Configuration Details](#)

Creating SAP Connections

If you want to retrieve or update data in a SAP data source for use in an EMF Process, you must first create and establish a SAP connection to the provider.

Note: EMF requires SAP Java Connector 2 (SAP JCo 2) or SAP Java Connector 3 (SAP JCo 3) to connect to the SAP back end. The SAP Java Connector supports connections to 4.0B, 4.5B, 4.6B, 4.6C, 4.6D, 4.7, 6.20 and higher. The SAP authorizations required for running the RFCs in the SAP Data browser and the SAP Authorization modules are

Auth Object: S_RFC

ACTVT: 16

RFC_NAME: SYST, SUSO, RFC1, STUW, SDTX, SUSR , BAPT, SDIFRUNTIME, SEU_COMPONENT

RFC_TYPE: FUGR

Auth Object: S_TABU_DIS

ACTVT: 03

DICBERCLS authorization group for all tables that must be downloaded: &NC&, SC, SS

SUSR : This is required to make authorization checks from the EMF server (Show Missing Authorizations is not selected in the SAP authorization module). If you are using only the local copies of the UST , USR tables, then it is not required.

SEU_COMPONENT: To get list of tables in the databrowser.

&NC& : This is required as it allows access to UST tables, unless you already have user-defined authorization groups which give access to UST tables only.

Important: Before you can test and use the SAP Data source, you must have first installed the SAP Java Connector.

Setting Up SAP Java Connector for EMF

The SAP Java Connector is available free of cost as a download from the SAP site for registered users. It is important to download the correct version of the Java Connector (JCo) for the hardware you are running on (32bit x86, 64bit x64 or Intel Itanium IA64). EMF currently supports JCo 2 and JCo 3.

If running on 32-bit platform, the x86 JCo connector is required. If running on 64-bit platform, the x86 JCo connector is required for the EMF UI and the x64 (or IA64, depending on processor type) JCo connector is required for the EMF server.

Setting Up JCo 2 on 32-bit Operating System

For a Windows installation, the following files need to be copied to the appropriate directories:

- **librfc32.dll** (for 32-bit JRE) needs to be copied to the EMF/SupportDLL directory. However, if you have other SAP software already installed, replace any older version of librfc32.dll in the {windows-dir}\system32 directory with the version that came with Java Connector.
- **sapjcorfc.dll** (for 32-bit JRE) needs to be copied to the EMF/SupportDLL directory. If you have recently installed EMF, please reboot at the end of the installation. Otherwise, the DLLs will not be detected.
- **sapjco.jar** (for 32-bit JRE) needs to be copied to the EMF/auto_jar directory and the EMF/server/lib directory.

Setting Up JCo 2 on 64-bit Operating System

The EMF UI runs on a 32-bit JRE. Therefore, on 64-bit systems, both the 32-bit and 64-bit versions of the SAP Java Connector are needed, as the EMF server uses the 64-bit JRE.

The above steps for setting up JCo 2 on a 32-bit operating system need to be followed first (except copying to the server/lib directory). Then, do the following for the EMF Server:

For a Windows installation, the following files need to be copied to the appropriate directories:

- **librfc32.dll** (for 64-bit JRE) needs to be copied to the EMF/server directory.
- **sapjcorfc.dll** (for 64-bit JRE) needs to be copied to the EMF/server directory.
- **sapjco.jar** (for 64-bit JRE) needs to be copied to the EMF/server/lib directory.

Setting Up JCo 3 on 32-bit Operating System

For a Windows installation, the following files need to be copied to the appropriate directories:

- **sapjco3.dll** (for 32-bit JRE) needs to be copied to the EMF/SupportDLL directory. If you have recently installed EMF, please reboot at the end of the installation. Otherwise, the DLLs will not be detected.
- **sapjco3.jar** (for 32-bit JRE) needs to be copied to the EMF/auto_jar directory and the EMF/server/lib directory.

Setting up JCo 3 on a 64-bit operating system

The EMF UI runs on a 32-bit JRE. Therefore, on 64-bit systems, both the 32-bit and 64-bit versions of the SAP Java Connector are needed, as the EMF server uses the 64-bit JRE.

The above steps for setting up JCo 3 on a 32-bit operating system need to be followed first (except copying to the server/lib directory). Then, do the following for the EMF Server:

For a Windows installation, the following files need to be copied to the appropriate directories:

- **sapjco3.dll** (for 64-bit JRE) needs to be copied to the EMF/server/lib directory.
- **sapjco3.jar** (for 64-bit JRE) needs to be copied to the EMF/server/lib directory.

Note: Before installing JCo 3, install the latest Visual Studio 2005 runtime libraries as described in note SAP Note 684106. If running on 32-bit platform, the x86 Visual Studio redistributable is needed. If running on 64-bit platform, the x86 Visual Studio redistributable is needed for the EMF UI and the x64 (or IA64, depending on processor type) Visual Studio redistributable is needed for the EMF server.

How to Create a SAP Connection

Note: To create a new connection, you must have access to the SAP system and security rights for SAP connections (your system administrator can set this up for you, using the User administration [Groups](#) screen).

You must also have the required connection and configuration details (see below); your SAP administrator should be able to supply you with these.

1. Click the **SAP Connections** icon in the **Data** sources section of the EMF tree view. Any existing SAP Connections are displayed in the right-hand pane.
 2. Right-click in the right-hand pane and select **New SAP connection** to display the **SAP connection** screen.
 - The **SAP connection** section is where you define, and establish a connection to, a SAP provider.
 - The **BAPI metadata** section displays all the information that is available for selection from that provider.
 3. Enter a name for the profile in the **Name** field. This can be anything you like, but ideally should be a meaningful name that will allow you to choose the correct SAP Connection Profile when adding a SAP Module to an EMF Process.
 4. Enter the name of the **Application Server** that the SAP R/3 instance is running on. If a SAP router string is defined (you can determine this by checking the SAP login screen in the software that your company normally uses to access your SAP Connection), you should precede the server name with this - e.g. if the Application Server name is **cpcat01** and the SAP Router String is **/H/111.004.1.111/S/3297/H/** you would enter **/H/111.004.1.111/S/3297/H/cpcat01**.
 5. Enter the **System number** for the SAP R/3 System, as obtained from your SAP Logon Properties.
 6. Enter the **Client number** that indicates the client for the SAP R/3.
 7. Enter the **User ID** with which to log on to the SAP R/3 System.
 8. Enter the **Password** that the above user uses to log on to the SAP R/3 System.
 9. Enter a **Language** (as defined in the SAP R/3 system).
-

Note: If you need any help with the above details, your SAP R/3 administrator should have the necessary information.

Implementing User-Defined RFC for use in SAP Data browser module

EMF allows you to change the SAP RFC (Remote Function Call) used to read information from the tables in the SAP Data browser.

The standard RFC used within EMF is the RFC_READ_TABLE. However, there are situations when a SAP user may have restricted rights to certain objects within the SAP System, and the SAP admin may have created an RFC to allow the user to bring back information from tables, without needing the rights to that particular object. Such user-defined RFCs can be used within EMF, instead of the standard RFC. In order to use this user-defined RFC, the

Config.xml file (normally located in the All Users\Application Data\emf\6 folder for Windows XP or in the Users\Public\emf\6 folder for Windows 7) has to be edited.

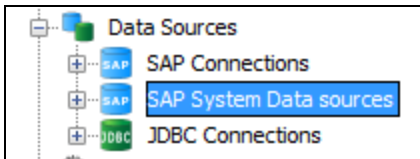
Insert the following text between `</system>` and `</configuration>` tags:

```
<sap>
<databrowser>
<rfcreadtable>RFC_READ_TABLE</rfcreadtable>
</databrowser>
<auth>
<rfcreadtable>RFC_READ_TABLE</rfcreadtable>
</auth>
</sap>
```

[Using the SAP Module](#)

Creating EMF SAP System Data Source

The SAP System Data Source defines the SAP table(s) fields to return in the output and fields to use as query options to filter the output data when extracting data from the SAP system using the [SAP System DS Data Browser Module](#). The SAP System Data Source can be accessed from the Data Sources node in the EMF tree view.



Any existing SAP System Data Sources are displayed in the right pane.

- Right-click the **SAP System Data Sources** icon and select **New**. The **SAP System Data Source** properties dialog is displayed.

The **SAP System Data Source** properties dialog consists of the following tabs:

- [Data Source Tab](#)
- [Tables to join Tab](#)
- [Data fields Tab](#)
- [Fields for join Tab](#)

Important: After the SAP System Data source is created, it needs to be applied to the SAP System as detailed in the [Applying to SAP System](#).

[Examples](#)

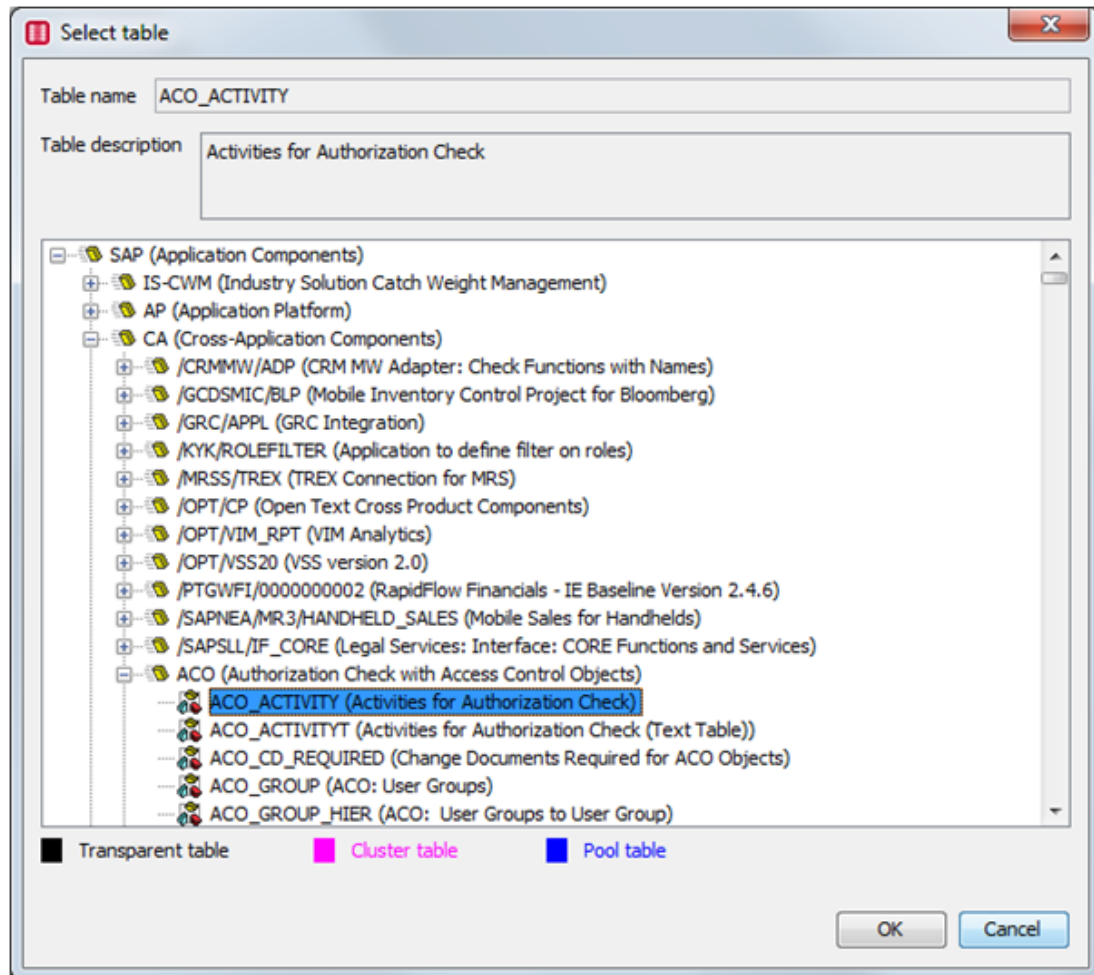
[Overview of SAP System Data Sources](#)

Data Source Tab

The screenshot shows the 'SAP System Data Source' dialog box with the 'Data Source' tab active. The 'Name' field contains 'COST CENTRES'. The 'SAP connection' dropdown is set to 'APTEAN_CPIC'. The 'SAP System data source name' dropdown is set to 'ZJR_COST_CENTRES', with a 'Get Data Sources' button to its right. The 'Primary table name' field contains 'CSKS' and has a browse button (...). The 'Data Source type' section has 'Standard' selected with a radio button, and 'Custom' is unselected. Below this is the 'Apply to SAP System' section, which includes a 'SAP Transport number' dropdown and a 'Get transports' button, followed by a 'SAP Transport description' text field. At the bottom of this section are 'Create', 'Update', and 'Delete' buttons. The very bottom of the dialog has 'OK', 'Apply', 'Cancel', and 'Help' buttons.

- **Name:** This name will be referenced throughout the EMF System to identify this instance of a SAP System Data Source.
- **SAP Connection:** Used to connect to the SAP System when querying or manipulating the SAP System Data Source. The drop down list shows all of the **SAP connections** that have been set up in the **SAP Connections** node.
- **SAP System data source name:** Name given to the SAP System Data Source which will be saved on the SAP System. The name cannot contain any blanks, has to start with either Y or Z and has to be upper case. The Data Source information is written into tables (created as part of the Extraction component set up on the SAP System) on the SAP System. Any existing SAP System Data Sources already defined in the SAP System will be available for selection in the drop down box, once populated by the **Get Data Sources** button. If a data source is selected from the drop-down list, then all relevant fields/grids in the dialog will be populated with the information from the selected data source.
- **Get Data Sources:** Click this button to get the data sources already defined in the SAP System and populates the Sap System Data Source Name combo box. Note that a valid SAP connection is required as the Get Data Sources gets the data from the SAP System.

- **Primary table name:** The primary table used for fetching table data from. The ellipse button opens up a dialog window from which the table can be selected. The Select table dialog allows the selection of the table using the pre-defined SAP Business Objects hierarchy (AP/CA/FI, and so on).



The following three SAP table types are supported:

- **Transparent Table** – This is the standard type of table (used in the Standard data source type). It exists with the same structure both in ABAP dictionary as well as in database exactly with the same data and fields. A transparent table is a table that stores data directly. You can read these tables directly on the database from outside SAP with for instance, an SQL statement.
- **Cluster Table** – This is supported in the Custom Data Source type. Cluster tables are logical tables that must be assigned to a table cluster when they are defined. Cluster tables can be used to store control data. They can also be used to store temporary data or texts, such as documentation. A clustered and a pooled table cannot be read from outside SAP because certain data are clustered and pooled in one field. These tables are shown in a different color.

.....	BSBM (Document Valuation Fields)
.....	BSBMT (Text for Valuation Adjustment or Deductible per Item)
.....	BSBW (Document Valuation Fields)
.....	BSEC (One-Time Account Data Document Segment)
.....	BSED (Bill of Exchange Fields Document Segment)
.....	BSEG (Accounting Document Segment)
.....	BSES (Document Control Data (Obsolete))
.....	BSET (Tax Data Document Segment)
.....	BSE_CLR (Additional Data for Document Segment: Clearing Information)

- **Pooled Table** - This is supported in the Custom Data Source type. Pooled tables are logical tables that must be assigned to a table pool when they are defined. Pooled tables are used to store control data. Several pooled tables can be combined in a table pool. The data of these pooled tables are then sorted in a common table in the database. A clustered and a pooled table cannot be read from outside SAP because certain data are clustered and pooled in one field. These tables are shown in a different color.

.....	T001D (Validation of Accounting Documents)
.....	T001E (Company Code-Dependent Address Data)
.....	T001F (Company code-dependent form selection)
.....	T001G (Company Code-Dependent Standard Texts)
.....	T001N (Company Code - EC Tax Numbers / Notifications)

- **Data source type:** Two types of Data Sources are used:
 - **Standard** – used for transparent tables in SAP.
 - **Custom** – used for cluster/pool tables like BSEG.

Note: The Custom data sources need extra ABAP code to be written in the SAP System. See [Creating Custom Data sources](#).

[Creating EMF SAP System Data Source](#)

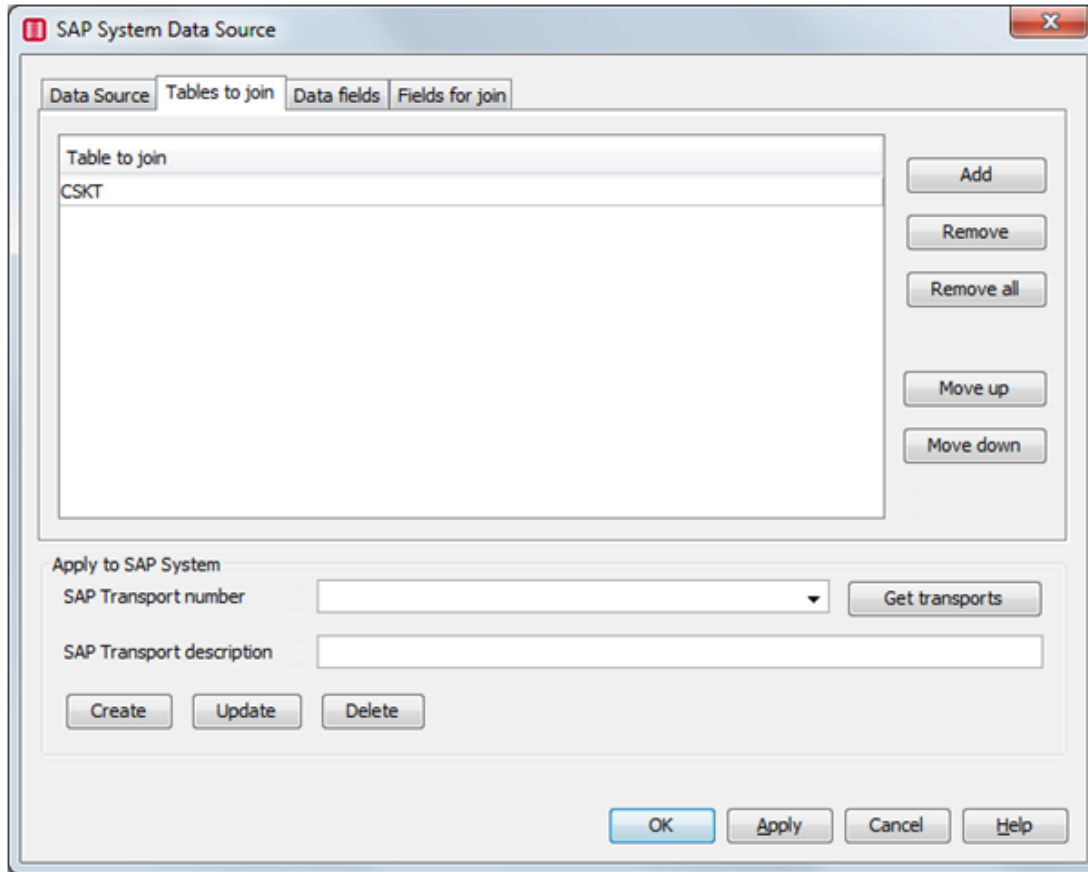
[Tables to join Tab](#)

[Data fields Tab](#)

[Fields for join Tab](#)

Tables to join Tab

Tables to join tab is used to select the tables that will be used if data is to be retrieved from more than one table.



- **Table to join:** Use this section to retrieve additional fields in addition to the primary table fields. The same [Select table](#) dialog is used to select the **Table to join**, as in the Primary table.
- **Add/Remove/Remove all:** These buttons allow you to add/remove the selected tables from the data source definition.
- **Move up/Move down:** These buttons allow you to move the selected table up or down in the list.

Note: The tables selected here will be "joined" in the order listed.

[Creating EMF SAP System Data Source](#)

[Data Source Tab](#)

[Data fields Tab](#)

[Fields for join Tab](#)

Data fields Tab

Data fields tab allows you to specify what data to retrieve and what fields to use in the query criteria (in the **SAP System DS Data Browser** module). The grid does not allow you to change or enter information directly into the grid.

- Click the **Select Fields** button to get the fields that will be output in the **SAP System DS Data Browser** module and to define what fields can be used in the query criteria in the **SAP System DS Data Browser** module.

If no fields are selected for **Use in Query**, then the query criteria will not be selectable in the **SAP System DS Data Browser** module and all data for the selected tables will be returned from the SAP System.

Note: Currently only ten query fields are supported and each query field can have ten parameters.

Return field	Use in Query	Key	Table	Field	Description
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKT	SPRAS	Language Key
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKS	KOKRS	Controlling Area
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKS	KOSTL	Cost Center
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKS	PRCTR	Profit Center
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKS	VERAK	Person Responsible
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	CSKS	DATBI	Valid To Date
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKS	DATAB	Valid-From Date
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKT	KTEXT	General Name
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKT	LTEXT	Description

Apply to SAP System

SAP Transport number:

SAP Transport description:

Create Update Delete

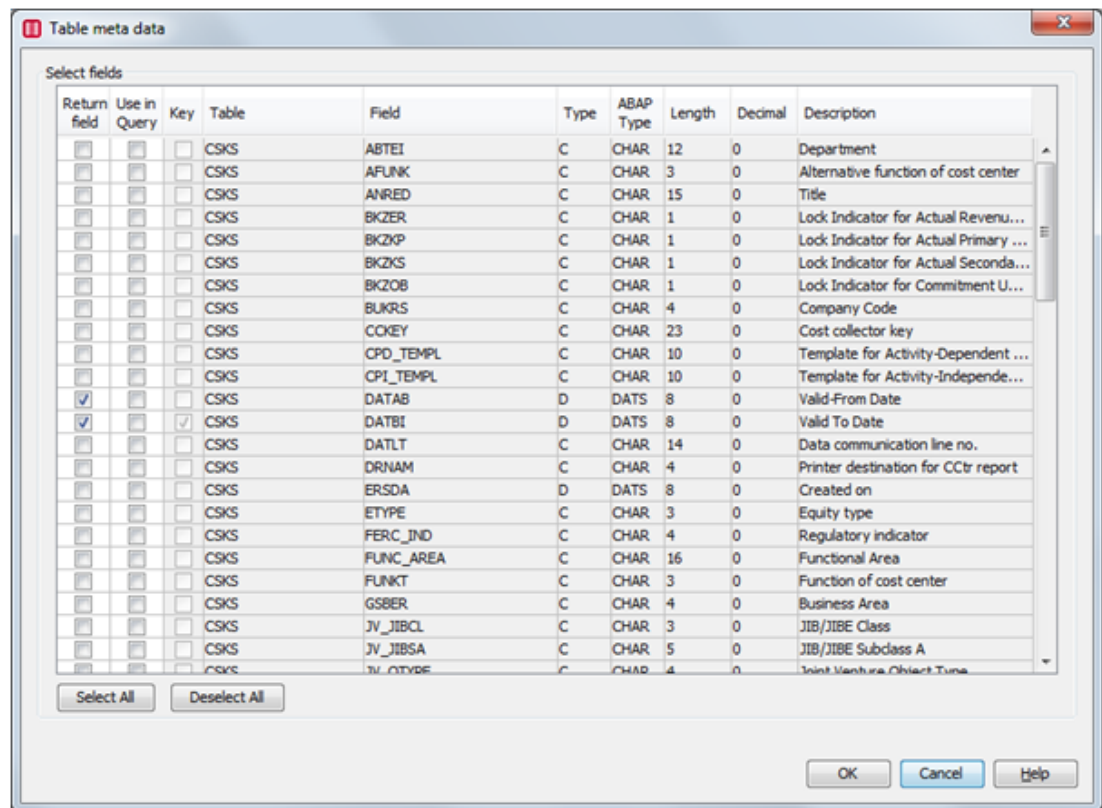
OK Apply Cancel Help

- Return field:** Indicates if the field will be included in the list of fields in the output.
- Use in Query:** Indicates if the field will be used in the query criteria in the **SAP System DS Data Browser** module.
- Key:** Indicates whether the field is a Key field in the table.

- **Table:** The table name for the field to be retrieved (Select) and/or queried on (Query). The table displayed can either be the primary table or a join table.
- **Field:** The field name in the table shown in the **Table** column.
- **Description:** Description of the field in the language specified in the [SAP Connection](#).
- **Move up/down:** Moves the relevant field up/down in the grid.

Note: The order in which the fields are displayed is the order in which they will appear in the output.

- **Select Fields:** Click this button to launch the **Table meta data** dialog (shown below) that allows you to select the fields to return and the fields to query from the primary and join tables. The dialog displays all the fields from all selected tables, together with the fields' metadata. Once the fields have been selected, click **OK** to return to the **Data fields** tab, populating the grid with the selected fields.



- **Return field:** Select this option if the field is to be included in the list of fields in the output.
- **Use in Query:** Select this option if the field is to be used in the query criteria in the **SAP System DS Data Browser** module. This option can be selected on its own, without selecting the **Return field** option.

- **Key**: Indicates whether the field is a Key field in the table.
- **Table**: The table name for the field to be retrieved (Select) and/or queried on (Query). The table shown here can either be the primary table or a join table.
- **Field**: The field belonging to the table displayed in the **Table** column.
- **Type**: SAP Data type for the field. Used mainly within ABAP programs. When you refer to data types from the ABAP Dictionary in an ABAP program, the predefined Dictionary types are converted to the data types, as the ABAP processor uses the data types in the work area for data.
- **ABAP Type**: This is a four character definition of the data type in the ABAP Dictionary.
- **Length**: The length of the field.
- **Decimal**: Specifies the number of decimals past the decimal point for those fields defined as floating point numbers. Else, all other fields will always display zero.
- **Description**: Description of the field in the language specified in the SAP Connection.
- **Select All**: Selects all the fields for output (**Return field**).
- **Deselect All**: Deselects all the fields for output (**Return field**).

[Creating EMF SAP System Data Source](#)

[SAP System DS Data Browser Module](#)

[Data Source Tab](#)

[Tables to join Tab](#)

[Fields for join Tab](#)

Fields for join Tab

Fields for join tab allows you to specify the join conditions for the join tables selected in the **Tables to join** tab. The join used is an INNER JOIN.

The screenshot shows the 'SAP System Data Source' dialog box with the 'Fields for join' tab selected. The dialog has four tabs: 'Data Source', 'Tables to join', 'Data fields', and 'Fields for join'. The 'Fields for join' tab contains a table with four columns: 'From table', 'From field', 'To table', and 'To field'. The table has two rows of data. To the right of the table are buttons for 'Add', 'Remove', 'Remove all', 'Move up', and 'Move down'. Below the table, there is a section for 'Apply to SAP System' with a dropdown for 'SAP Transport number', a text field for 'SAP Transport description', and buttons for 'Create', 'Update', and 'Delete'. At the bottom of the dialog are buttons for 'OK', 'Apply', 'Cancel', and 'Help'.

From table	From field	To table	To field
CSKS	KOKRS	CSKT	KOKRS
CSKS	KOSTL	CSKT	KOSTL

- **From table:** Select the table to join from the drop-down list (which shows the tables selected for the join and the primary table).
- **From field:** Select the field to join on for the **From table**, from the drop-down list.
- **To table:** Select the table to join from the drop-down list.
- **To field:** Select the field to join on for the **To table**, from the drop down list.
- **Add/Remove/Remove all** buttons: Allows you to add/remove joins for the data source definition.
- **Move up/Move down** buttons: Allows you to move the selected join condition up or down in the list.

Note: The order of the join is determined by the order of the tables in the **Tables to join** tab.

[Creating EMF SAP System Data Source](#)

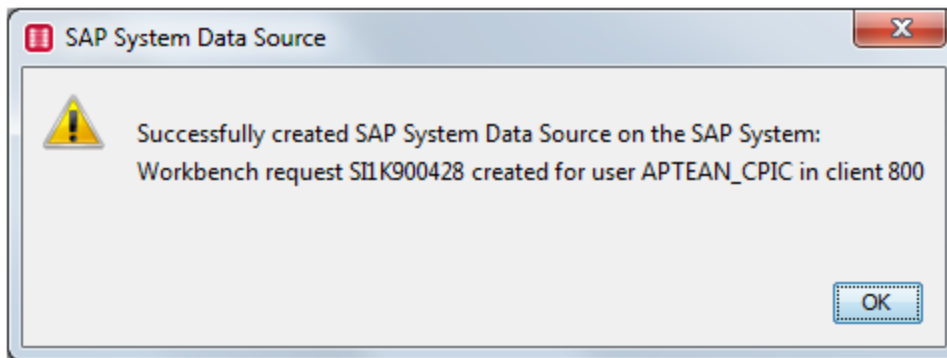
[Data Source Tab](#)

[Tables to join Tab](#)

[Data fields Tab](#)

Applying to SAP System

The SAP Data Source defined has to be created and updated on the SAP System, else you will not be able to run the **SAP System DS Data Browser** module in the Process Builder. The data entered in the SAP System Data Source is entered into the SAP System using the **Create** button and supplying the **Transport Description** at the same time. If the SAP System successfully creates the data source, a Transport number is returned to you, else an error dialog is displayed.



Note: It is possible to create an SAP System Data Source within the EMF system without initially creating it in the SAP System. This has the advantage of being able to define the Data Source fully before entering it into the SAP System. However, as mentioned above, the SAP System Data Source will need to be created on the SAP System for the data extraction to work.

It is important to sync the local version of the SAP System Data Source with the SAP System, else the mismatch of the **SAP System Data Source** definition between local and SAP System will cause errors when the **SAP System DS Data Browser** module is run. The **Update** button allows this sync to happen. If the **SAP System Data Source** definition is changed in the dialog, it is important to Update the SAP System with the changes if the SAP System Data Source is used in a **SAP System DS Data Browser** module. An existing Transport number has to be used when updating.

When a **SAP System Data Source** is no longer required, it can be deleted from the SAP System using the **Delete** button.

Note that the **SAP System Data Source** is referred to by the **SAP System data source name** within the SAP System. It is possible to reference the **SAP System data source name** by one or more **SAP System Data Source** entities within EMF.

It is therefore, important when deleting the **SAP System Data Source** on the SAP System that the **SAP System data source name** is not used by any other **SAP System Data Source** entity within EMF.

- **SAP Transport number:** A transport request is normally associated when making changes or creating entities in the SAP system, to enable the SAP system to move over changes from one system to another easily. Any existing transport number can be used when sending changes/deleting data sources from the SAP system. A Transport number is required when updating/deleting the SAP System Data Source from the SAP System.

- **Get transports:** Gets the list of transport requests available for the user specified in the [SAP Connection](#).
- **SAP Transparent description:** A description used to identify the Transport number being created.
- **Create:** This button is used to create the SAP system data source in the SAP system.
- **Update:** This button is used to change the existing data source in the SAP system.
- **Delete:** This button is used to delete the data source from the SAP System.

Note: The **Create/Update/Delete** buttons only apply to the SAP System. The **OK/Apply/Cancel** buttons behave normally and save/change data in the EMF system. This allows the user to work offline and create a SAP system data source in the SAP system when they are ready to upload it to their SAP system.

Examples

The following examples illustrates how to set up the SAP System Data Source for a number of scenarios. Click on each example to view how to do the set up.

- [Example 1: How to Set up a SAP System Data Source](#)
- [Example 2: How to Update a SAP System Data Source](#)
- [Example 3: How to Delete a SAP System Data Source](#)
- [Example 4: How to Update to Latest Version of SAP System Data Source \(a manual synchronization\)](#)

[Creating SAP System Data Source](#)

[Overview of SAP System Data Sources](#)

Example 1: How to Set up a SAP System Data Source

1. Right-click on the **SAP System Data Sources** node in the tree view and select **New**. The following dialog is displayed:

2. In the **Name** field, type "Cost Centres" for the SAP System Data Source.

Note: This is the name of the entity within the EMF system.

3. Select the **SAP Connection** from the list of available connections.

Note: You must have a **SAP Connection** defined within EMF. If it has not been set up, click **Cancel** to exit from this dialog and create a SAP Connection and ensure that you can connect to a SAP System.

4. In the **SAP System data source** field, type "ZAPT_COST_CENTRES" or an appropriate name.
5. In the **Primary table name** field, type "CSKS".
6. In the **Data Source type** section, select the **Standard** option.

The following dialog illustrates the data entered:

The screenshot shows the 'Data Source' configuration window. It has four tabs: 'Data Source', 'Tables to join', 'Data fields', and 'Fields for join'. The 'Data Source' tab is active. It contains the following fields and controls:

- Name:** Text box containing 'Cost Centres'.
- SAP connection:** Dropdown menu showing 'APTEAN_CPIC'.
- SAP System data source name:** Dropdown menu showing 'ZAPT_COST_CENTRES'. To its right is a 'Get Data Sources' button.
- Primary table name:** Text box containing 'CSKS' with a small '...' button to its right.
- Data Source type:** Two radio buttons: 'Standard' (selected) and 'Custom'.

7. Click the **Tables to join** tab.

The screenshot shows the 'Tables to join' configuration window. It has the same four tabs as the previous window. The 'Tables to join' tab is active. It contains:

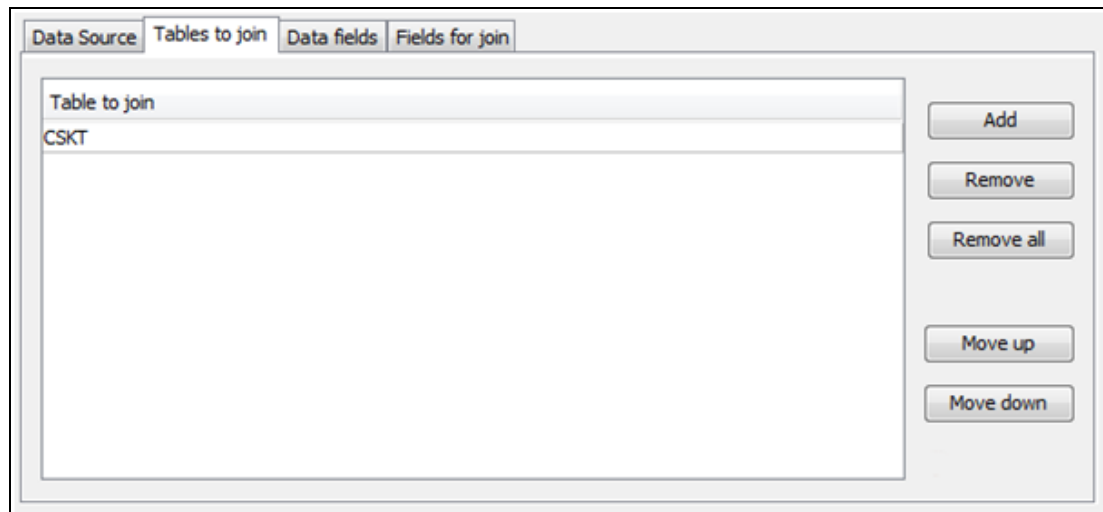
- Table to join:** A large empty list box for adding tables.
- Buttons:** A vertical stack of buttons on the right: 'Add', 'Remove', 'Remove all', 'Move up', and 'Move down'.

8. Click **Add** to insert an empty row into the **Table to join** table list.
 9. Double-click on the empty row to allow editing.

This is a close-up of the 'Table to join' list. It shows a single empty row. On the right side of the row, there is a small downward-pointing arrow, indicating a dropdown menu for table selection.

10. Type "CSKT" into the cell. You can use the down arrow to invoke the table selection dialog, if the table name is not known.

The following dialog illustrates the added table:



11. Click the **Data fields** tab.



12. Click **Select Fields** to display the available fields for the tables: CSKS and CSKT.
13. Select the fields as shown in the image below, ensuring the **Use in Query** fields are also selected:

Return field	Use in Query	Key	Table	Field	Type	ABAP Type	Length	Decimal	Description
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	CSKS	DATAB	D	DATS	8	0	Valid-From Date
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKS	DATBI	D	DATS	8	0	Valid To Date
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKS	KOKRS	C	CHAR	4	0	Controlling Area
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKS	KOSTL	C	CHAR	10	0	Cost Center
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKS	VERAK	C	CHAR	20	0	Person Responsible
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKT	KTEXT	C	CHAR	20	0	General Name
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKT	LTEXT	C	CHAR	40	0	Description
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKT	SPRAS	C	LANG	1	0	Language Key

14. Click **OK** to display the data in the **Data fields** tab grid.

Return field	Use in Query	Key	Table	Field	Description
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	CSKS	DATAB	Valid-From Date
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKS	DATBI	Valid To Date
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKS	KOKRS	Controlling Area
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKS	KOSTL	Cost Center
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKS	VERAK	Person Responsible
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKT	KTEXT	General Name
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKT	LTEXT	Description
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKT	SPRAS	Language Key

15. Click the **Fields for join** tab.
16. Click **Add** to add an empty row into the grid. Double-click the **From table** cell to allow editing.

17. Click the down arrow to display the list of the available tables (primary table + join tables) that can be used in the join.
18. Select "CSKS" from the drop-down list in the **From table**. It is assumed that you know the primary keys for the two tables from the **Select tables** dialog earlier, so it is relatively simple to select the fields that will be joined on, in this case.
19. Double-click the **From field** cell and display the drop-down list of all the fields for the selected **From table**. Select "KOKRS" from the list of fields.
20. Repeat steps above for **To table** and **To field**. Select or type "CSKT" for **To table** and "KOKRS" for **To field**.

You have now created a part of the join for the two tables: CSKS and CSKT.

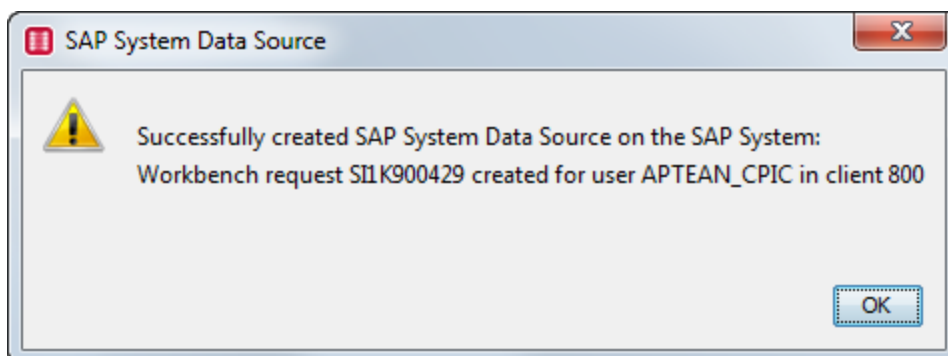
21. Click **Add** to create a new empty row. Repeat the above steps and enter the following data in this new row:
 - **From table** = CSKS
 - **From field** = KOSTL
 - **To table** = CSKT
 - **To field** = KOSTL
22. You have now completed the join.

The following dialog illustrates the completed grid:

From table	From field	To table	To field
CSKS	KOKRS	CSKT	KOKRS
CSKS	KOSTL	CSKT	KOSTL

23. Click **Apply** to save the SAP System Data Source entity into the EMF system.
24. Save the **SAP System Data Source** into the SAP System, so that the **SAP System DS Data Browser** can run this data source, when referenced.
The **Apply to SAP System** deals with saving and removing the **SAP System Data Source** from the SAP System.

25. In the **SAP Transport description** field, type "Aptean Cost Centres".
26. Click **Create** to create the **SAP System Data Source** in the SAP System. The following confirmation message will be displayed (the request number, user and client will be different in your case):



[Examples](#)

[Creating SAP System Data Source](#)

Overview of SAP System Data Sources

Example 2: How to Update a SAP System Data Source

1. Click the **SAP System Data Sources** node in the tree view and select the **SAP System Data Source** you want to edit.
2. Right-click on the selected **SAP System Data Source** and select **Edit**, (or double-click the selected **SAP System Data Source**) to display the following dialog:

The screenshot shows the 'SAP System Data Source' dialog box with the 'Data Source' tab selected. The fields are as follows:

- Name: Cost Centres
- SAP connection: APTEAN_CPIC
- SAP System data source name: ZAPT_COST_CENTRES
- Primary table name: CSKS
- Data Source type: Standard (selected), Custom

Buttons visible include 'Get Data Sources', 'Get transports', 'Create', 'Update', 'Delete', 'OK', 'Apply', 'Cancel', and 'Help'.

3. Click the **Data fields** tab.
4. Click **Select Fields** to display the **Table metadata** dialog. Select the field "BUKRS" from table "CSKS" and check the **Return** field check box, as shown below:

<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKS	BUKRS	C	CHAR	4	0	Company Code
-------------------------------------	--------------------------	--------------------------	------	-------	---	------	---	---	--------------

5. In the **Table metadata** dialog, click **OK** to update the **Data fields** grid.

Return field	Use in Query	Key	Table	Field	Description
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	CSKS	DATAB	Valid-From Date
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKS	DATBI	Valid To Date
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKS	KOKRS	Controlling Area
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKS	KOSTL	Cost Center
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKS	VERAK	Person Responsible
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKT	KTEXT	General Name
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKT	LTEXT	Description
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKT	SPRAS	Language Key
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKS	BUKRS	Company Code

6. The order of fields returned in the output will now be changed. Select the field "KOKRS" and move it to the top using the **Move up** button. Move the other fields so that the final order is as shown:

Return field	Use in Query	Key	Table	Field	Description
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKS	KOKRS	Controlling Area
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKS	KOSTL	Cost Center
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKS	BUKRS	Company Code
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKT	KTEXT	General Name
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	CSKS	DATAB	Valid-From Date
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKS	DATBI	Valid To Date
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKS	VERAK	Person Responsible
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CSKT	LTEXT	Description
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CSKT	SPRAS	Language Key

7. Click **Apply** to save it to the EMF system.
8. Click **Get transports** to get a list of available transports that can be used when updating the SAP System. Select a **Transport number** from the drop-down list.

Note: You can also use the **Transport request number** returned from the SAP System when setting up the SAP System Data Source.

Apply to SAP System

SAP Transport number

SI1K900429

▼

Get transports

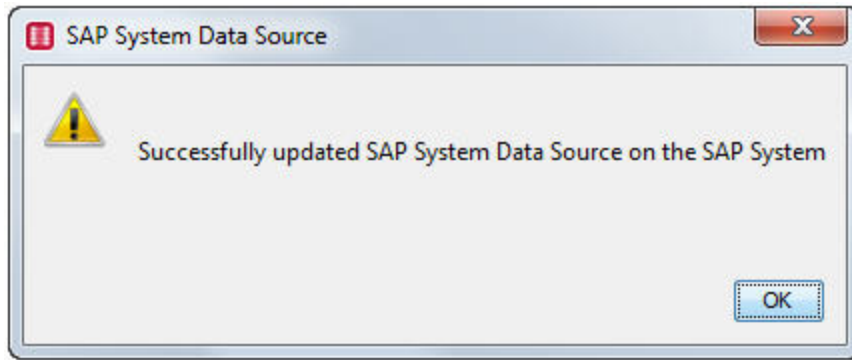
SAP Transport description

Create

Update

Delete

9. Click **Update** to update the **SAP System Data Source** into the SAP System. After a successful update, the following dialog is displayed:



[Examples](#)

[Creating SAP System Data Source](#)

[Overview of SAP System Data Sources](#)

Example 3: How to delete a SAP System Data Source

1. Click the **SAP System Data Sources** node in the tree view and select the **SAP System Data Source** you want to delete.

Note: If the **SAP System Data Source** entity in the EMF system will not be used for another SAP System data source name, then ensure that the **SAP System Data Source** entity is not used anywhere else in the EMF System. This can be ascertained by using the **Find Usages** feature on the **SAP System Data Source** entity.

2. Right-click on the selected **SAP System Data Source** and select **Edit** (or double-click the selected SAP System Data Source) to display the following dialog:

SAP System Data Source

Data Source | Tables to join | Data fields | Fields for join

Name: Cost Centres(1)
 SAP connection: APTEAN_CPIC
 SAP System data source name: ZAPT_COST_CENTRES
 Primary table name: CSKS
 Data Source type: ☒ Standard ☐ Custom

Apply to SAP System
 SAP Transport number:
 SAP Transport description:
 Create Update Delete

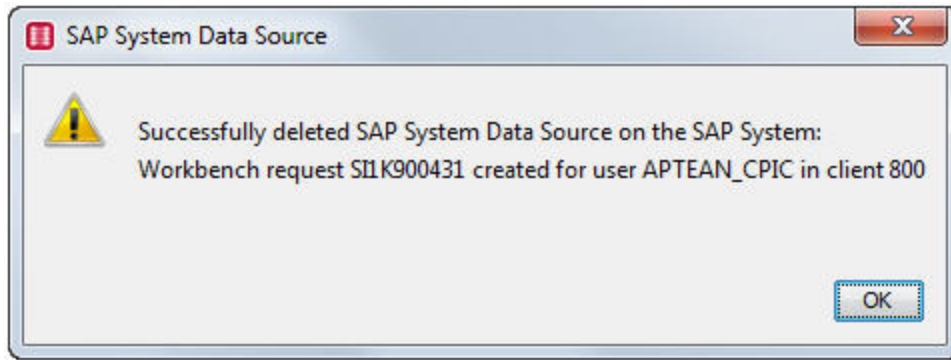
OK Apply Cancel Help

3. Click **Get transports** to get a list of available transports that can be used when updating the SAP System.
4. Select a **Transport number** from the drop-down list. You can also enter a **SAP Transport description**.

Apply to SAP System
 SAP Transport number: SI1K900429
 SAP Transport description:
 Create Update Delete

Note: You can also use the **SAP Transport request number** returned from the SAP System when setting up the SAP System Data Source.

5. Click **Delete** to delete the SAP System Data Source from the SAP System. After the delete, the following dialog is displayed:



Note: This only deletes the SAP System Data Source from the SAP System and not from the EMF System. Use the standard procedure to delete the SAP System Data Source from the EMF System.

[Examples](#)

[Creating SAP System Data Source](#)

[Overview of SAP System Data Sources](#)

Example 4: How to update to latest version of SAP System Data Source (a manual synchronization)

There can be instances where a different EMF user has updated the SAP System Data Source into the SAP System, using their own EMF repository.

To get the latest updates, you can either:

- export the SAP System Data Source entity from the other user's repository and import it into your own EMF repository.
- update SAP System Data Source entity using the SAP System Data Source entity dialog.

To update to latest version of SAP System Data Source

1. Click the **SAP System Data Sources** node in the tree view and select the **SAP System Data Source** you want to edit.
2. Right-click the selected **SAP System Data Source** and select **Edit** (or double-click the selected SAP System Data Source) to display the following dialog:

SAP System Data Source

Data Source | Tables to join | Data fields | Fields for join

Name: Cost Centres
 SAP connection: APTEAN_CPIC
 SAP System data source name: ZAPT_COST_CENTRES [Get Data Sources]
 Primary table name: CSKS [...]
 Data Source type: ☒ Standard ☐ Custom

Apply to SAP System
 SAP Transport number: [Get transports]
 SAP Transport description:

Create Update Delete

OK Apply Cancel Help

- Click **Get Data Sources** and select the **SAP System data source** name from the drop-down list.

This will retrieve the **SAP System Data Source** from the SAP System and update the **SAP System Data Source** entity with the latest version.

SAP System data source name: ZAPT_COST_CENTRES
 Primary table name: ZAPT_COST_CENTRES
 Data Source type: ZCLUSTER1

- Click **OK/Apply** to save it into the EMF System.

[Examples](#)

[Creating SAP System Data Source](#)

[Overview of SAP System Data Sources](#)

Configuring Services

Most of the EMF [Output modules](#) require you to create a corresponding **service** to send information. You can view, configure, create, and delete EMF services using the screens that are available from the **Services** icon in the EMF tree view.

Note: HTTP and [JNDI service](#) are not used by an Output module. They are used by the HTTP and JNDI modules, which retrieve data from third party systems.

You need to only configure those services that you intend to use.

To add or edit a service:

1. Click the plus sign (+) next to the **Services** icon in the EMF tree view to display the list of available service types. These include [Executable](#), [File](#), [FTP](#), [HTTP](#), [JNDI](#), [Queue](#), [SNMP](#), [SMTP](#), [Respond Service](#), [Web Service service](#), [SMS](#), and [OPC DA Service](#).

Note: EMF includes example instances of **File**, **FTP**, and **Executable** services.

2. Select the required service type to display a list of the existing services of that type in the right pane.
3. Do the following:
 - **To create a service:** Right-click on the service group and select **New**. Fill in the details on the new service's **Properties** pages as appropriate (for more information, click the **Help** button on each page).
 - **To modify or configure a service:** Double-click the icon for the relevant service and make changes as required.
 - **To delete a service:** Select the required service, right-click and then choose **Delete**.

[Configuring the EMF System](#)

JNDI Service

The JNDI service is used in multiple places in EMF:

- The [JNDI query module](#) in the [EMF Process Builder](#)
- The [External directory recipient module](#) in the EMF Process Builder
- In the [Roles](#) dialog and Login dialog to configure external operators to log in to EMF

EMF supports the version of JNDI in Java 2 SDK 1.3.1 or later. For more information about JNDI, see the [Oracle](#) website.

To specify a JNDI service:

1. Double-click the **JNDI services** icon in the **Services** section of the EMF tree view.

- To **modify an existing JNDI service**, double-click its icon.
- To **create a new JNDI service**, right-click in the list view and click **New JNDI service**.
- The JNDI service **Properties** screen is displayed.

2. Enter a **Name** for the service. This is the name that will be used within the EMF system to identify this instance of a JNDI service.
3. Enter the **Initial context factory** name for the JNDI naming context. This information determines the service provider that EMF should use.

Together with JNDI, a service provider communicates with the naming/directory server. Service providers for LDAP, CORBA Common Object Services (COS) naming service and RMI Registry are included with Sun's Java 2 SDK version 1.3.1. If you want to use other service providers, refer to the documentation that accompanies them for the initial factory name you should use.

Examples of Initial Context Factories for Sun Service Providers

LDAP	com.sun.jndi.LdapCtxFactory
COS	com.sun.jndi.cosnaming.CNCTXFactory

RMI	com.sun.jndi.rmi.registry.RegistryContextFactory
-----	--

4. Type the connection details for your naming/directory server in the **Initial context** field.

Examples of Initial Contexts for Sun Service Providers

LDAP	ldap://192.0.0.2:389
COS	iiopname://192.0.0.5:1900
RMI	rmi://192.0.0.10:1099

5. Type the **Context root** of the Initial context entered above. If you are using LDAP, the context root is the base DN. For example, ou=my company,dc=intranet,dc=company,dc=com.

Note: The initial context and context root together make the "provider URL" for the directory service.

6. If your naming/directory server requires authentication, select **Simple** from the **Authentication** drop-down list. Then enter the **Username** and **Password** in the textboxes below.

Note: Simple authentication sends a clear-text password. EMF does not currently support Security Layer (SASL) for authentication.

7. Select the following options:
 - **Enable service to be used by external directory recipient module** if you want this service to be available in the [external directory recipient module](#).
 - **Enable operators to log in against this service** if you want to add external operators to [Roles](#) and allow them to log in to EMF. This check box may not be enabled if you do not have the required license rights.

If either of these options are selected, the [External recipient/operator LDAP attribute mappings](#) and [External recipient/operator LDAP queries](#) tabs will be enabled and will need populating.

It is advisable not to have both check boxes selected together. They are configuring very different things within EMF, so it is better to define two JNDI Services and configure one for the external directory recipient module, and one for external operators.

8. Click the **Test connection** button to test the connection to your JNDI service provider. This will check whether the initial context exists. Depending on your JNDI provider it may also test the authentication details (if you entered any).

[Configuring Services](#)

[Configuring the EMF System](#)

[External recipient/operator LDAP attribute mappings tab](#)

[External recipient/operator LDAP queries tab](#)

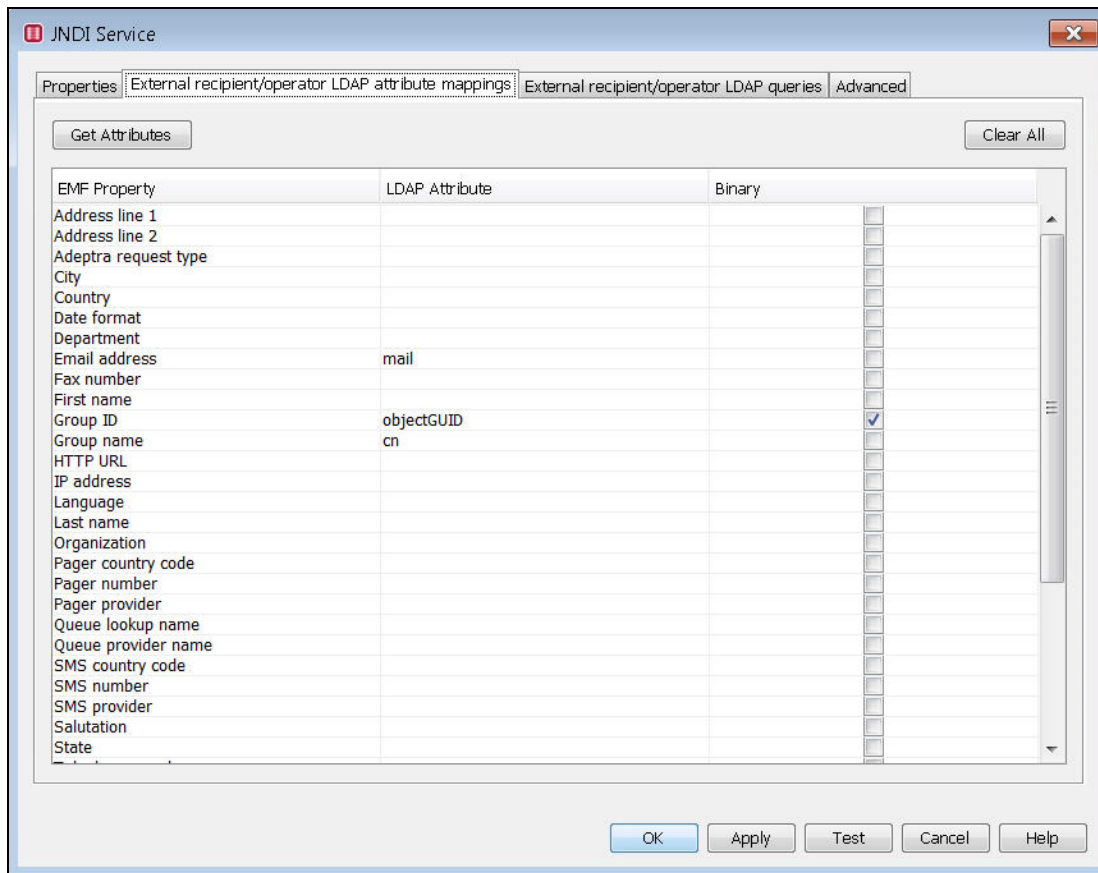
JNDI Advanced

External recipient/operator LDAP attribute mappings

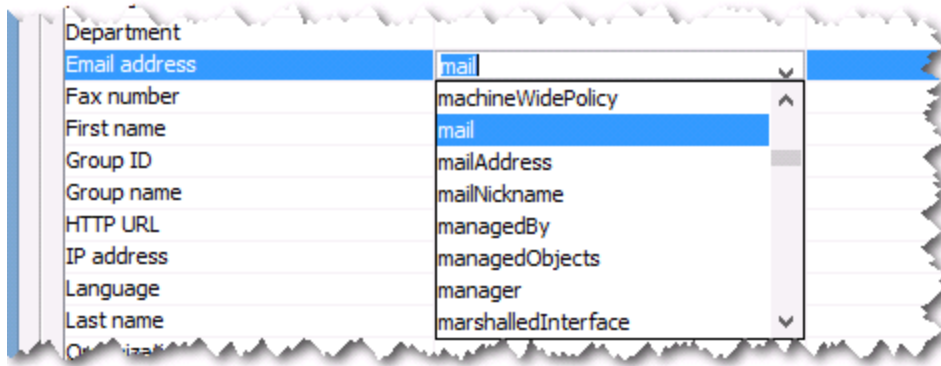
The **External recipient/operator LDAP attribute mappings** tab allows you to specify the mappings between the JNDI attributes and the EMF user properties. These are needed if you plan to use the external directory recipient module, or allow external operators to log in.

Note: Microsoft Active Directory is the only tested and supported LDAP store for using External Recipient and External operators with. Other LDAP stores may work but have not tested. External Operator Groups will only work against Microsoft Active Directory.

1. After setting up the connection details to the JNDI properties tab, press the **Get Attributes** button to query the JNDI store for a list of available attributes.



2. To map a property, you need to select the attribute from the drop-down list of available attributes to the corresponding EMF property. So for example, the Active Directory email address attribute is **mail**. So to be able to send external recipients (those in an LDAP store) emails, you would need to have mapped the EMF Property **Email address**, to the LDAP attribute **mail**.



3. As a minimum, you must map the following four EMF Properties to JNDI properties:
- User name
 - User profile ID
 - Group name
 - Group ID

Mapping to User profile ID and Group ID must allow a user/group to be uniquely identified. If using Microsoft Active Directory, it is recommended to use the **objectGUID** attribute to map to **User profile ID** and **Group ID**. This field contains a binary value, and therefore the **Binary** check box for the appropriate row also needs to be selected.

User name should map on to an attribute that identifies the user. Suggested values are **cn**, **mail** or **userPrincipalName**. The value here is the value that will be displayed for an external recipient or operator in the roles dialog or external directory recipient module. **If you are allowing external operators to log in, this is also the field that will contain their user name. So you would need to decide whether you want users to log in with a name, like John Smith, or you may prefer them to use their email address such as jsmith@company.com.**

The group name should map on to an attribute that identifies the group. Suggested value is **cn**.

4. Any mapped attributes that return binary values need to have the **Binary** check box selected in the appropriate row.

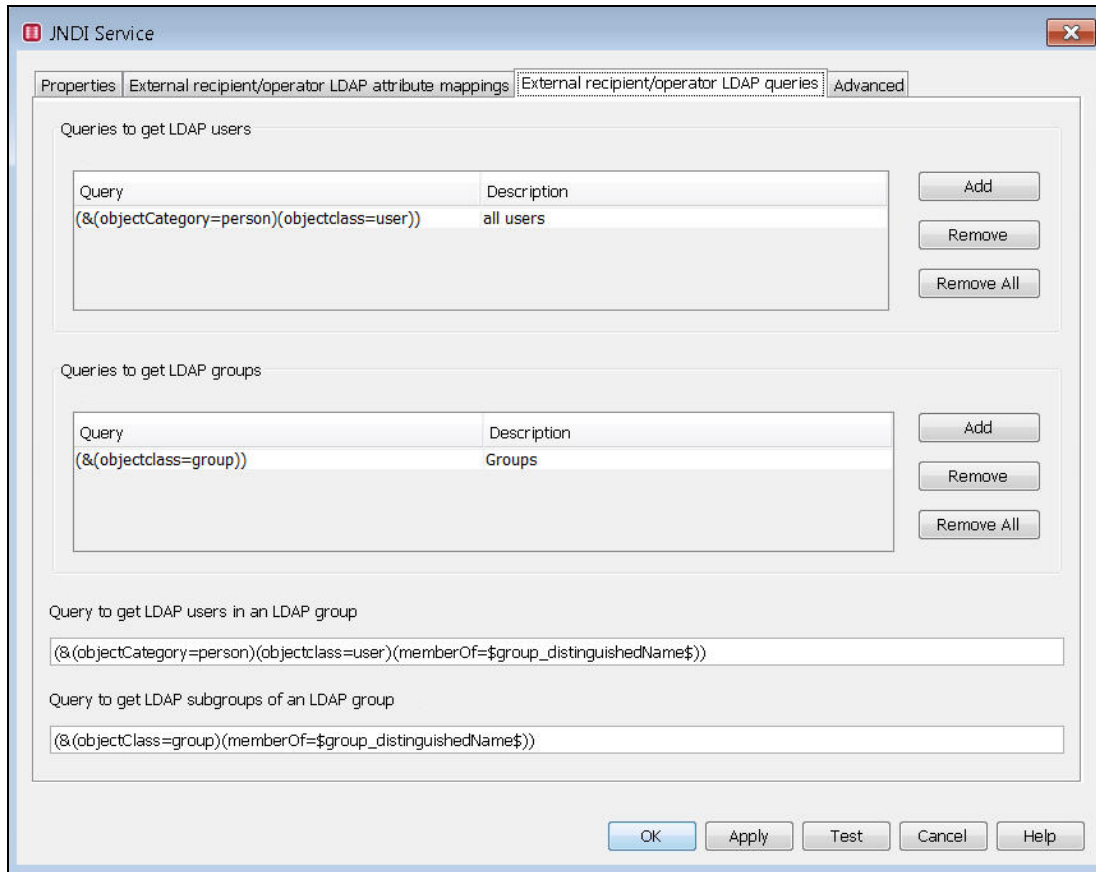
[JNDI Service](#)

[External recipient/operator LDAP queries tab](#)

External recipient/operator LDAP queries

The **External recipient/operator LDAP queries** tab defines the LDAP queries to get lists of users and groups from the directory service. The queries for the recipients and groups decide what recipients/operators and groups are available within the EMF UI to allow the process builder to select from.

It also defines the queries needed to extract user and subgroup information from a group for this particular LDAP store.



1. To add a query to display a list of recipients, press the **Add** button in the query for recipients section of the dialog and enter the required information.

For example, to return all users for Active Directory, enter:

```
(&(objectCategory=person)(objectclass=user))
```

You may however want to restrict this list to a certain set of users, or people within a certain group (for example, Sales).

```
(&(objectCategory=person)(objectclass=user)(
memberOf=CN=Sales,OU=Users,DC=YourDomain,DC=com))
```

It is then possible to add multiple entries, so you could include Sales, Marketing, and so on.

2. To add a query to display a list of groups, press the **Add** button in the query for groups section of the dialog and enter the required information.

For example, to return all groups for Active Directory, enter:

```
(&(objectclass=group))
```

This would allow any group to be selected as recipients of messages (or any recipient in any selected group to be able to log into EMF).

To limit the groups available, alter the query. Multiple queries can be added.

3. Edit the **Query to get LDAP users in an LDAP group** - this LDAP query is run to get the users associated with a group (at that level). The recommended query for Active

Directory is:

```
(&(objectCategory=person)(objectclass=user)(memberOf=$group_distinguishedName$))
```

Place holders can be entered that are replaced at execution time. These are entered as follows:

```
$group_<attribute_name>$
```

What then happens at execution time is that each group the query is executed against will replace the token with the attribute value for that group. So in the example above, at execution time, to find a list of recipients in the **Sales** group it will replace \$group_distinguishedName\$ with the distinguishedName attribute value for the sales group. So the query that would execute would be something like this:

```
(&(objectCategory=person)(objectclass=user)
(memberOf=CN=Sales,OU=Users,DC=YourDomain,DC=com))
```

which would return all the recipients in the **Sales** group.

4. Similar to the above, a query should be set to **Query to get LDAP subgroups of an LDAP group**. This then allows groups within groups to be found in the LDAP store. The recommended query for Active Directory is:

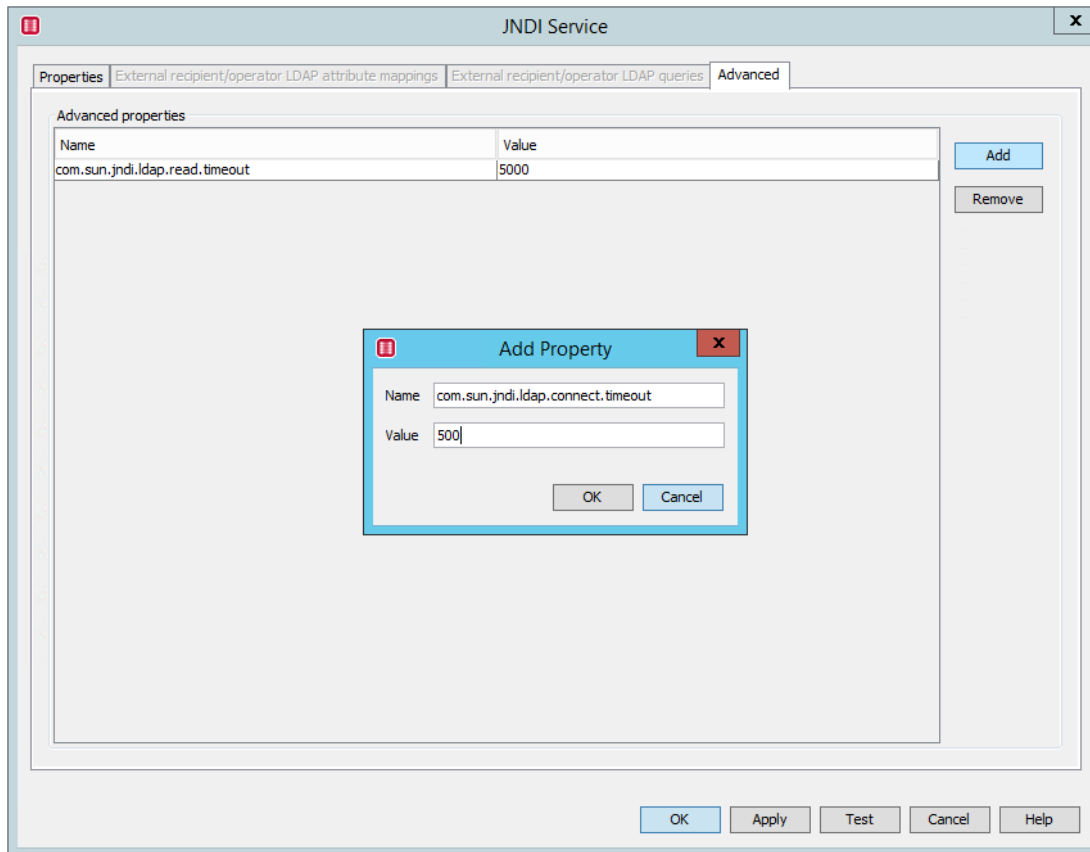
```
(&(objectClass=group)(memberOf=$group_distinguishedName$))
```

[JNDI Service](#)

[External recipient/operator LDAP attribute mappings tab](#)

JNDI Service - Advanced

The **Advanced** tab is used to set the Advanced properties in the JNDI Service Module.



The **Advanced** tab is used to define the JNDI environment properties:

- Click **Add** to enter the property **Name** and **Value** to be defined in the **Advanced properties** section. See, <https://docs.oracle.com/javase/8/docs/technotes/guides/jndi/jndi-ldap.html> for the list of properties. For example, to eliminate the issues with Connection timeouts, the `com.sun.jndi.ldap.read.timeout` and `com.sun.jndi.ldap.connect.timeout` properties are used, and to configure pooling options to improve performance `com.sun.jndi.ldap.connect.pool` property is used.
- Click **Remove** to remove the selected row property.

[JNDI Service](#)

Executable Service

Before you can use a [Run Executable output module](#) in the [EMF Process builder](#) you must configure an Executable Service from the Services section of the EMF tree view.

To specify an executable service:

1. Double-click the Executable services icon in the Services section of the EMF tree view.
 - To modify an existing executable service, double-click its icon.
 - To create a new executable service, right-click in the list view and click New executable service.

The **Executable Service** screen is displayed.

2. Enter the following information:

- **Name** - the value entered here will be used throughout the EMF System to refer to this instance of an Executable Service.
- **Queue** - this is the queue that the service will be using.

Note: Unless you are an advanced user and specifically need to distribute the load, you should always leave the setting as **<default>**.

- **Timeout (seconds)** - this is the amount of time that the executable is to attempt to run before timing out. This is measured in seconds.
- **Port number** - this is the default port number that is to be used. This should be set to the same value as that of the Executable Service clients.

[Configuring Services](#)

File Service

Before you can use an [output module](#) that writes directly to a file (e.g. XML, HTML, Flat File, or WML) in the [EMF Process Builder](#) you must configure a **File service** from the **Services** section of the EMF tree view.

To specify a file service:

1. Double-click the **File services** icon in the **Services** section of the EMF tree view.
 - **To modify an existing file service**, double-click its icon.
 - **To create a new file service**, right-click in the list view and click **New file service**.

The **File Service** properties screen is displayed.

2. Enter a **Name** for the service. This is the name that will be used within the EMF system to recognize this instance of a file service.
3. Select a **Queue** from the drop-down list of those available. The choice of queue defines how the information should be dealt with by the server.

Note: Unless you are an advanced user and specifically need to distribute the load, you should always leave the setting as **<default>**.

[Configuring Services](#)

[Configuring the EMF System](#)

FTP Service

Before you can use any FTP module ([FTP In](#), [FTP Out](#), [FTP Move Files](#) and [FTP Delete Files](#)) in the [EMF Process Builder](#) you must configure an **FTP service** from the **Services** section of the EMF tree view. The FTP service can be used to configure connections to FTP (File Transfer Protocol), FTPS (File Transfer Protocol Secure), and SFTP (SSH File Transfer Protocol or Secure File Transfer Protocol) servers.

To specify an FTP service:

1. Double-click the **FTP services** icon in the **Services** section of the EMF tree view.
 - **To modify an existing FTP service**, double-click its icon.
 - **To create a new FTP service**, right-click on the **FTP services** icon and click **New**.

The **FTP Service** properties screen is displayed.

The screenshot shows the 'FTP Service' configuration window. The 'Properties' tab is selected. The configuration fields are as follows:

- Name:** ExampleFTPSERVICE
- Queue:** <default>
- User name:** anonymous
- Password:** (empty)
- Remote server:** localhost
- Port number:** 21
- Protocol:** Plain FTP
- User directory is the root directory:** ☐
- Default directory:** ./outputs/ftp
- FTP Server time zone:**
 - ☐ Define FTP server time zone
 - FTP Server time zone: (GMT+05:30) Asia/Calcutta - India Standard Time
 - ☒ Adjust for daylight saving

Buttons at the bottom: OK, Apply, Test, Cancel, Help.

2. Enter a **Name** for the FTP service.
3. Select a **Queue** from the drop-down list of those available. The choice of queue defines how the information should be dealt with by the server.

Note: Unless you are an advanced user and specifically need to distribute the load, you should always leave the setting as **<default>**.

4. Enter the **User name** and **Password**, if required, to access the remote FTP Server.
5. Enter the address of the remote FTP Server in the **Remote Server** field.
6. Enter the remote port to connect to in the **Port number** field. The default values for **Plain FTP/FTP implicit**, **FTP explicit**, and **SFTP** protocols are 21, 990, and 22 respectively.

7. Select any of the following Protocol:
 - **Plain FTP:** The [FTP/FTPS Advanced tab](#) is enabled if you select this option.
 - **SFTP:** The [SFTP Advanced tab](#) is enabled if you select this option.
 - **FTPS explicit:** The [FTP/FTPS Advanced tab](#) is enabled if you select this option.
 - **FTPS implicit:** The [FTP/FTPS Advanced tab](#) is enabled if you select this option.
8. If you select the **User directory is the root directory** check box, then the users home directory on the FTP server becomes the root directory. If the **User directory is the root directory** check box is not selected, then either the FTP server root directory will be the root directory, or the users home directory will be the root directory depending on how the FTP user account has been configured.
9. Enter the **Default directory** where FTP output should be sent.
10. Select the **Define FTP server time zone** check box if the FTP Server is running in a different time zone to the EMF Server. There is no standard way to detect the time zone an FTP Server is running in, but one method is detailed here:

https://wiki.filezilla-project.org/Server_timezone_offset

Select the **FTP Server time zone** and the **Adjust for daylight saving** as appropriate. If the FTP Server time zone isn't correctly configured, then:

- FTP In will report the incorrect last modified time.
- FTP Move Files/FTP Delete Files – if moving/deleting files using the *older than/newer than* settings then the correct files will not be moved/deleted.

[FTP/FTPS Advanced](#)

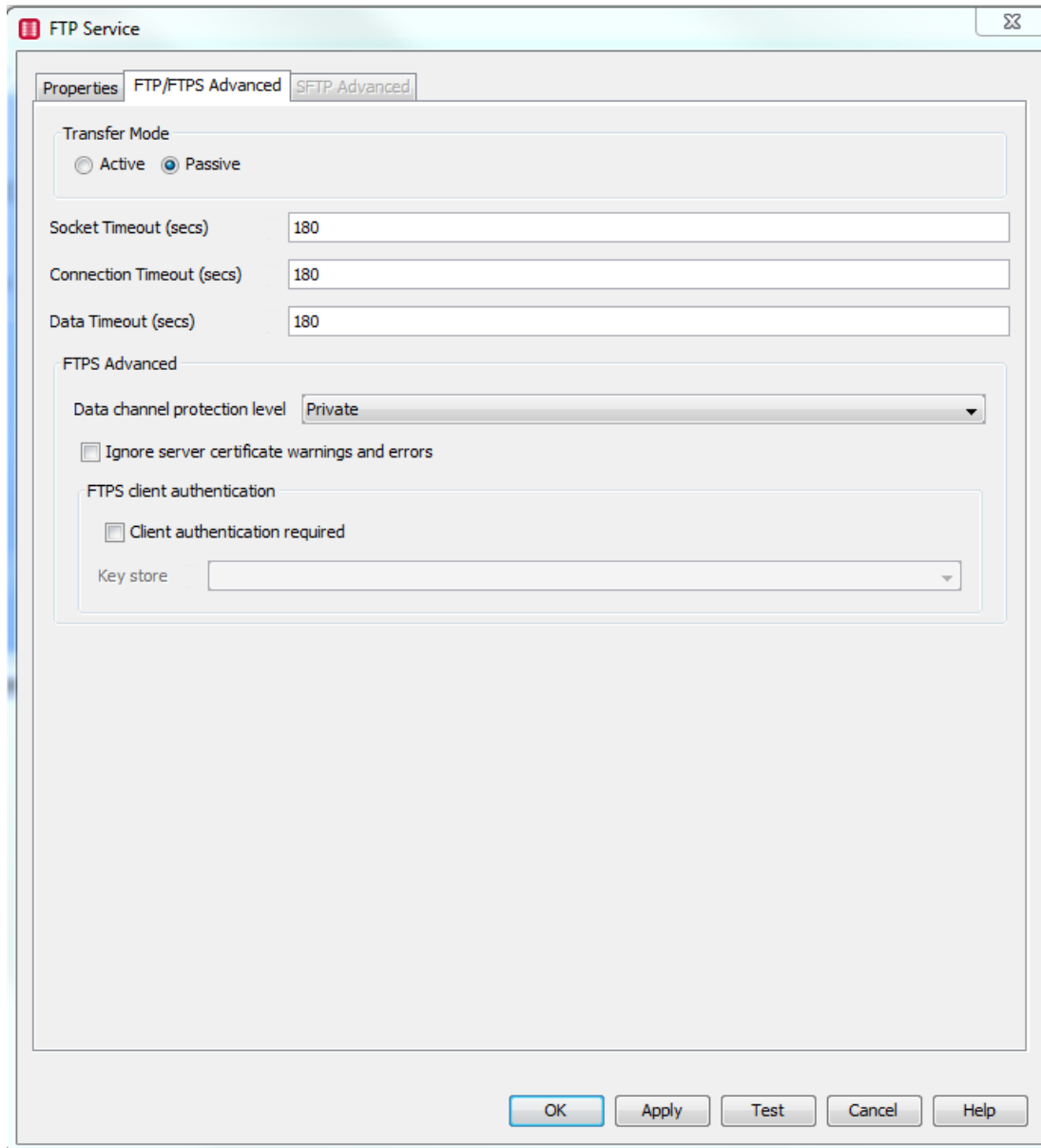
[SFTP Advanced](#)

[Configuring Services](#)

[Configuring the EMF System](#)

FTP/FTPS Advanced

You can use the FTP/FTPS Advanced tab to select the Transfer Mode and configure the Timeout details for the FTP protocol.



1. Select **Active** or **Passive** transfer mode.
2. **Socket Timeout (secs)**: Specify the time in seconds from when the socket connects until the connection breaks.
3. **Connection Timeout (secs)**: Specify the time in seconds to wait for an initial connection.
4. **Data Timeout (secs)**: Specify the time in seconds that the server must wait to receive data before timing out.

If **FTPS implicit** / **FTPS explicit** is selected as the Protocol, then the FTPS Advanced items are enabled.

1. Select the **Data channel protection level**, i.e., Clear, Safe, Confidential, and Private, from the drop down to specify the level of security used when transferring data between the EMF server and the FTPS server. For security, it is recommended to always use

"Private" – which means that the data is encrypted and its integrity ensured. Other levels may not be supported by all FTPS servers.

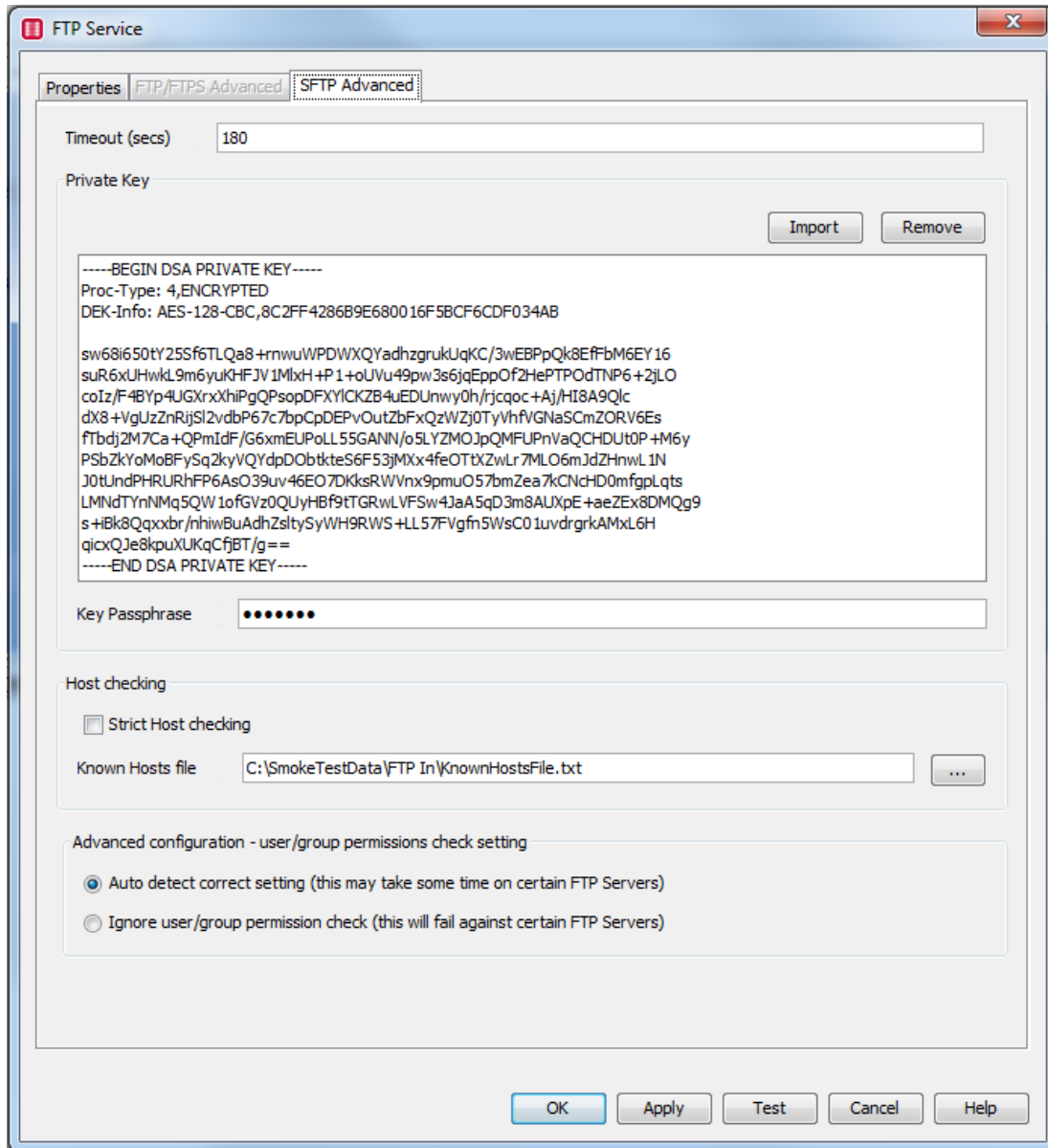
2. Select the **Ignore server certificate warnings and errors** check box if you want to bypass the service certificate warnings and errors. For instance, if the FTPS server is using a self-signed certificate then this option needs to be selected to prevent the EMF server from refusing the connection.
3. If the FTPS server has been configured as needing Client Certificate Authentication, then select **Client authentication required** check box, the **Key Store** drop down is enabled and a valid certificate should be selected to validate the user connecting. The client certificate is a digital certificate used to authenticate who is connecting and would typically be provided to you by the administrator of the FTPS server.

Note: Do not confuse the client certificate with the server certificate that is used to create the secure connection.

[FTP Service](#)

SFTP Advanced

You can use the SFTP Advanced tab to configure the Timeout details, Private Key, and Host checking for the SFTP protocol.



1. **Timeout (secs):** Specify the time in seconds to wait for an initial connection.
2. **Private Key** option is an alternative to a password to authenticate with the server that you are authorized to connect. You can **Import** a private key or **Remove** the existing private key. **Key Passphrase** allows you to enter the passphrase for the private key.

The following ciphers are supported:

- blowfish-cbc
- 3des-cbc
- aes128-cbc
- aes192-cbc
- aes256-cbc
- aes128-ctr

- aes192-ctr
- aes256-ctr
- arcfour
- arcfour128
- arcfour256

Note: By default, aes192 and aes256 are not supported. To support aes192 and aes256, you need to install the **Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files** for your Java Runtime Environment (JRE). These cannot be shipped with EMF due to export regulations. The files for Java 8 can be downloaded from <http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html>. The included README.txt in the download explains how to install the files, but it should be just copying the included files to the **Aptean\EMF\jreX.X.X.X\x64\lib\security** and the **Aptean\EMF\jreX.X.X.X\x86\lib\security** directories, having backed up the existing files first.

The following key types are supported:

- ssh-dss
- ssh-rsa
- ecdsa-sha2-nistp256
- ecdsa-sha2-nistp384
- ecdsa-sha2-nistp521

The EMF server will try to authenticate using the private key first, and then the User password.

3. **Host Checking** provides a level of security to check that the server you think you are connecting to is really the server you are connecting to.

The **Known Hosts** file is used to authenticate servers using public key cryptography. Whenever SSH is configured on a new server it always generates a public and private key for the server. Every time you connect to an SSH server, it sends the public key. If you do not have the public key of the server, then if **Strict Host checking** is enabled, the connection will fail. If **Strict host checking** is not enabled, then the key will be added to the **Known Hosts** file if it does not exist. If the Known hosts file does not exist, and **Strict host checking** is not enabled, then the file will be created. The three situations that a key mismatch typically happens are:

- The key changed on the server. This could be from reinstalling the operating system, or on some operating systems the key gets recreated when updating the SSH.
- The hostname or IP address you are connecting to belongs to a different server.
- Malicious activities also causes key mismatch.

If a key mismatch occurs, then the EMF module using the FTP service will display an error. This can then be captured using an error link, or the process can be allowed to fail. If the mismatch is genuine (because the server key has changed), then edit the **Known Hosts** file and update/remove the key entry for that server.

- **Known Hosts file:** Select the file that contains the Known Hosts. The file must be accessible by the EMF server. If no **Known Hosts** file is specified then host checking is not performed.

Note: If using the test button in the FTP service and the EMF UI and EMF server are on a different machine, then the testing is happening on the EMF UI machine. At runtime, it would be running on the EMF Server machine, and a different **Known Hosts** file would be used.

Note: Typically, the easiest way to configure the **Strict Host checking** is to run the process once on the server (or use the **Test** button if the server and UI are on the same machine) and this will create the **Known Hosts file** (if the file does not exist) with the server details. Then the **Strict Host checking** option should be enabled.

4. The **Advanced configuration - user/group permission check setting:** is necessary to make sure that the EMF server communicates to the FTP server correctly. FTP servers on different operating systems define permissions slightly differently. By default, the setting will try to **Auto detect** the correct option. This typically means that if it is connecting to a *nix (e.g. Linux) based FTP server there will be no delay. If it is a Windows based FTP server, there may be a delay during the operation whilst it detects the correct option to use. If the process does run with a delay, then enabling "verbose" logging on the process will identify if you can set the option to **Ignore user/group permission check** to improve performance. If the log shows the entry below, then it is advisable to select the **Ignore user/group permission check** option:

```
<VERBOSE> File/folder "amlt" found, storing details. [Process Details- Name: "Move
<VERBOSE> "Check the user/group permissions" is updated from "true" to "false" [Pr
<VERBOSE> Folder "/" found, reading contents. [Process Details- Name: "Move single
```

Note: When running in [Production mode](#) once the correct setting has been auto-detected, the server will continue to use that setting until it is restarted.

[FTP Service](#)

HTTP Service

Before you can use an [HTTP module](#) in the [EMF Process Builder](#) you must configure an **HTTP service** from the **Services** section of the EMF tree view.

To specify an HTTP service:

1. Double-click the **HTTP services** icon in the **Services** section of the EMF tree view.

- **To modify an existing HTTP service**, double-click its icon.
- **To create a new HTTP service**, right-click in the list view and click **New HTTP service**.

The **HTTP Service** properties screen is displayed.

2. Enter a **Name** for the service. This can be anything that you choose, and it is what will be used to identify this unique service when configuring the [HTTP module](#).
3. Select the required **Queue** from the drop-down list of those configured on your system. Any HTTP modules that are to be processed will be placed on the selected queue.

Note: Unless you are an advanced user and specifically need to distribute the load, you should always leave the setting as **<default>**.

4. Optional, enter a **Default URI** that you can use as the default Internet address (you can always specify a different one when configuring your HTTP modules).
5. Optional, enter a **Request timeout**. If the Web server fails to respond during the specified number of seconds, the connection will close.

[The HTTP module](#)

[The Internet Security Tab](#)

[How to create a self-signed certificate](#)

[How HTTP Works in EMF](#)

[Configuring Services](#)

[Configuring the EMF System](#)

HTTP Internet Security Settings

The [HTTP](#) services have the **Internet Security** tab.

1. Select the **Authentication** tab and enter the following information:
 - **Type:** From the **Type** drop-down list, select the type of authentication the HTTP service should use when trying to connect:
 - **None:** when no authentication is required.
 - **Basic/Digest:** for HTTP basic or digest authentication.
 - **Windows Authentication (NTLM):** to authenticate using the Microsoft Windows credentials.
 - **User name:** This is the user name that you use to log on when Basic/Digest or Windows authentication is required.
 - **Password:** This is the password that accompanies the user name.
 - **Workstation:** The workstation for your Windows authentication.
 - **Domain:** The domain for your Windows authentication.

2. Select the **SSL Settings** tab and enter the following information:
 - **Ignore server certificate warnings and errors:** Select this check box if you want to bypass the certificate warnings and errors. For instance, if the server is using a self-signed certificate, then this option needs to be selected to prevent the EMF server from refusing the connection.
 - **Client Authentication Required:** Select this check box if client authentication is required for the HTTP settings.
 - **Key Store:** The key store lists all the certificates for client authentication that you have installed (using the [Certificates](#) node in the **Security** section of the EMF tree view). Select the required key store from the list; in this case, the EMF server is the client to an external resource.
3. If you need to use a proxy server, select the **Proxy** tab and click the **Use Proxy** check box. Then enter the **Proxy host** and **Proxy port** in the appropriate text boxes.

[Configuring Services](#)

[Configuring the EMF System](#)

Configuring Web Service Services

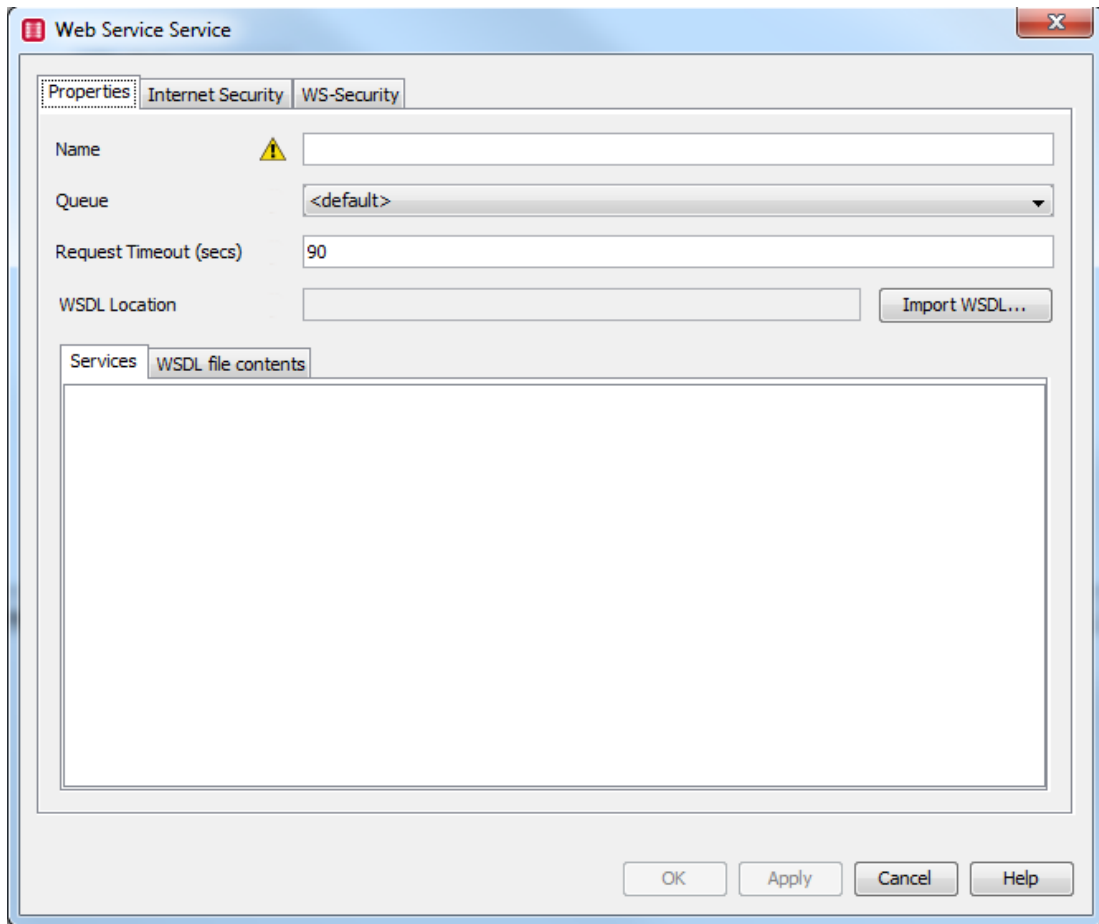
Before you can use a [Web Services Module](#) in the EMF Process Builder, you must configure a Web Service service from the **Services** section of the EMF tree view. The Web Service service is configured with the Web Service Definition Language (WSDL) file that contains the list of operations for the web service you are configuring, and therefore must be accessible.

Notes: Only WSDL 1.1 is supported, and not WSDL 2.0. Trying to import a WSDL 2.0 file will result in an error.

The web service module does not support old style SOAP-RPC calls. So, even though the WSDL file will import successfully, the operations will fail when trying to use them in the Web Services module.

To specify a Web Service Services:

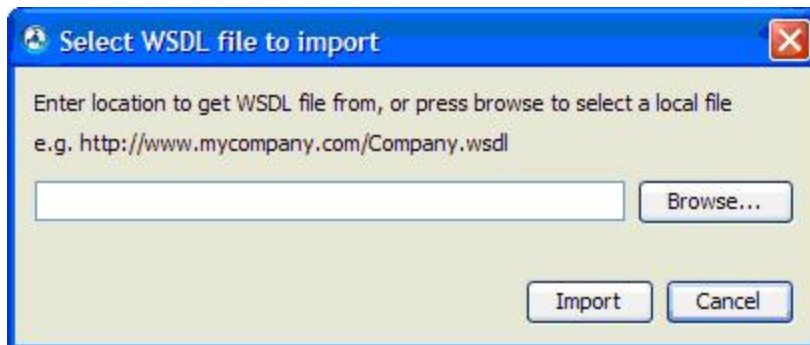
1. Right-click the Web Service services icon in the Services section of the EMF tree view and select **New**. The **Web Service Service** properties dialog is displayed.



2. Enter a **Name** for the service. This can be anything that you choose, and is what will be used to identify this unique service when configuring the Web Service module.
3. Select the required **Queue** from the drop-down list of queues configured on your system. Any Web Service modules that are to be processed will be placed on the selected queue. The default queue is WIP and it is recommended that you do not change that.

Note: Unless you are an advanced user and specifically need to distribute the load, you should always leave the setting as **<default>**.

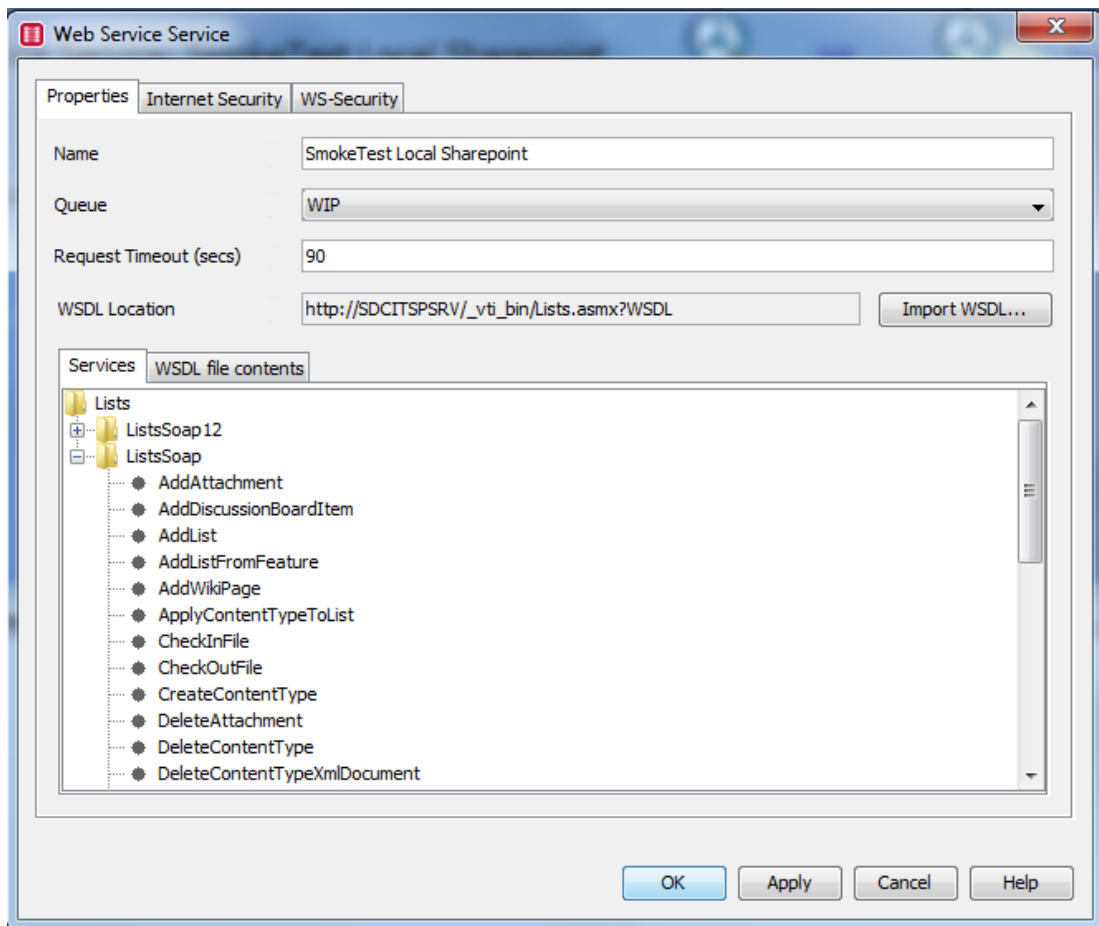
4. Press the **Import WSDL** button to select the WSDL file to use.



In the dialog box that appears, you can either enter the URL of the WSDL file to use, or press the **Browse** button to browse the local file system and select a WSDL file on your local drive. If for some reason you cannot always access a remote WSDL file, then it is possible to save that file to your local drive and import it. Press the **Import** button to complete the import.

Note: While the WSDL file is imported into the EMF repository, any other files it may reference (i.e. XSD files or other import WSDL files) are not. They are still referenced in their remote locations. Therefore, they must be available to the EMF server and EMF web service module when running a process or using the builder.

5. On successfully importing the file, the file is processed and a list of available operations are displayed in the **Services** tab. The actual contents of the WSDL file, or whatever was imported, can be viewed in the **WSDL file contents** tab (this can be useful to check if an error occurs as it may be because the file was not accessible and this tab may have an error to indicate the cause).



The Internet Security Tab

Select the **Internet Security** tab to enter the security settings needed to access the WSDL file, any associated files, and the actual web service call. Currently, it is only possible to

have one set of security settings for all of these items. The settings on this tab have no effect when importing the WSDL into the Web service service. For the import to work, the WSDL file must be accessible to the computer that the import operation is happening from.

The WS-Security Tab

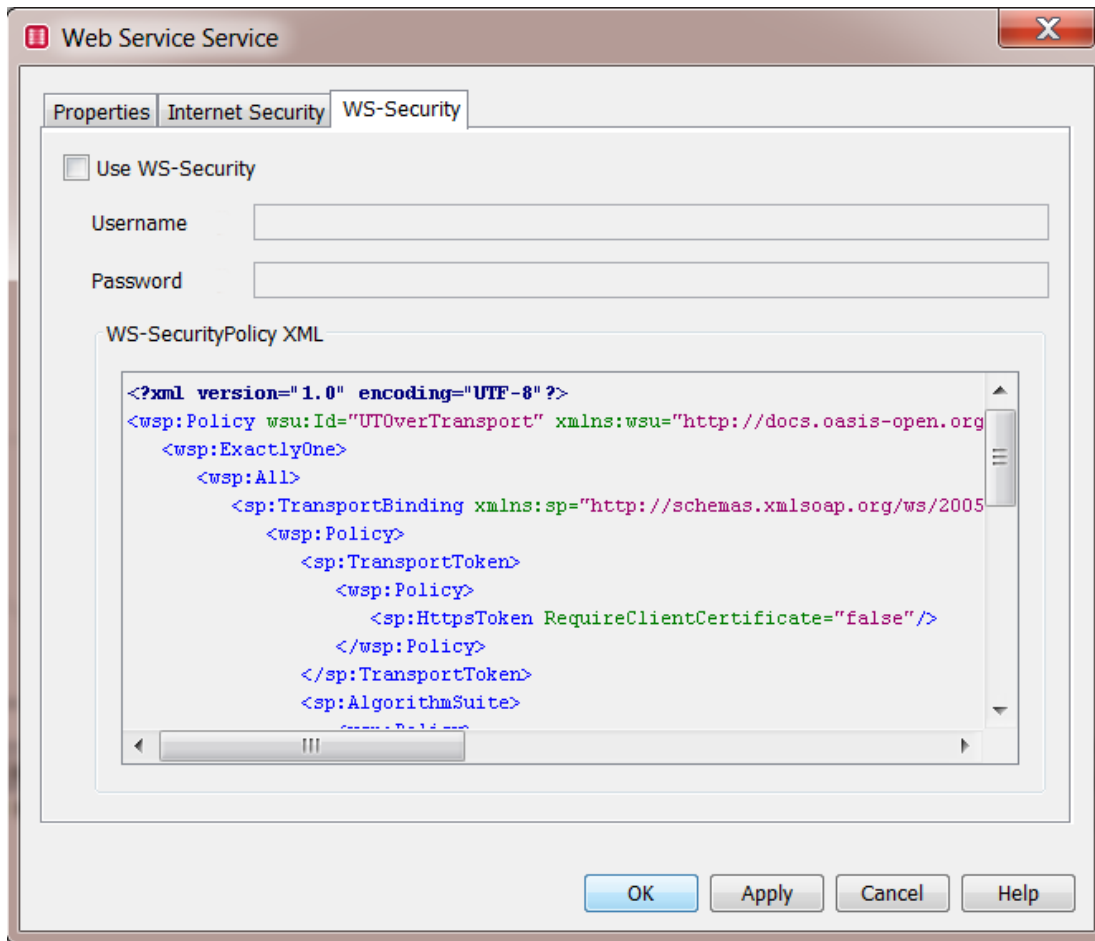
Select the [WS-Security](#) tab to provide additional security information the web service may need. WS-Security is an extension to SOAP to allow security to be defined on a SOAP web service call. If the webservice that you are calling needs WS-Security information to successfully make the call then you need to define the policy XML that defines what security information should be sent, and how it is sent.

[Web Services Module](#)

Webservices Security Settings

The **WS-Security** tab allows both the webservice security policy and, if required, the username/password needed to make the webservice call to be set. The username/password is not the same as defined on the [Internet Security](#) tab, and should only be set if a WS-Security policy is known to be needed to make the webservice call.

Click the **Use WS-Security** checkbox to enable the **Username**, **Password**, and **WS-SecurityPolicy XML** fields. Once the fields are enabled, enter the username and password if they are required by the policy.



WS-SecurityPolicy XML

The WS-SecurityPolicy XML defines the security protocol to be used for the web service. By default, the **WS-SecurityPolicy XML** text box contains a pre-defined XML. The XML has to be changed to match the security policy needed for the web service that is being called. For example, the web service may require the password to be encrypted using a particular algorithm. The XML allows this to be defined.

The following code gives a sample policy XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsp:Policy wsu:Id="UTOverTransport" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:TransportBinding
xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
        <wsp:Policy>
          <sp:TransportToken>
            <wsp:Policy>
              <sp:HttpsToken RequireClientCertificate="false"/>
            </wsp:Policy>
          </sp:TransportToken>
        </wsp:Policy>
      </sp:TransportBinding>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

```

</sp:TransportToken>
<sp:AlgorithmSuite>
  <wsp:Policy>
    <sp:Basic128/>
  </wsp:Policy>
</sp:AlgorithmSuite>
<sp:Layout>
  <wsp:Policy>
    <sp:Lax/>
  </wsp:Policy>
</sp:Layout>
</wsp:Policy>
</sp:TransportBinding>
<sp:SignedSupportingTokens xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702">
  <wsp:Policy>
    <sp:UsernameToken sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702/IncludeToken/AlwaysToRecipient">
      <wsp:Policy>
        <sp:HashPassword/>
      </wsp:Policy>
    </sp:UsernameToken>
  </wsp:Policy>
</sp:SignedSupportingTokens>
</wsp:All>
</wsp:ExactlyOne>
</wsp:Policy>

```

For more information on the XML structure for WS-SecurityPolicy, refer <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.html>

Note: EMF supports only a limited subset of WS-SecurityPolicy 1.1 and WS-SecurityPolicy 1.2.

Webservices Internet Security Settings

The **Internet Security** tab of the Web Service service enables you to enter the following security settings needed to access the WSDL file, any associated files, and the actual Web service call:

1. Select the **Authentication** tab and enter the following information:
 - **Type:** From the **Type** drop-down list, select the type of authentication the Web service should use when trying to connect:
 - **None:** when no authentication is required.
 - **Basic/Digest** for basic or digest authentication.
 - **Windows Authentication** (NTLM) to authenticate using the Microsoft Windows credentials.

- **User name:** This is the user name that you use to log on when Basic/Digest or Windows authentication is required.
 - **Password:** This is the password that accompanies the user name.
 - **Host:** The host computer for your Windows authentication.
 - **Domain:** The domain for your Windows authentication.
2. Select the **SSL Settings** tab and enter the following information:
 - **Ignore server certificate warnings and errors:** Select this check box if you want to bypass the certificate warnings and errors. For instance, if the server is using a self-signed certificate, then this option needs to be selected to prevent the EMF server from refusing the connection.
 - **Client Authentication Required:** Select this check box if client authentication is required for the Web Service settings.
 - **Key Store:** The key store lists all the certificates for client authentication that you have installed (using the [Certificates](#) node in the **Security** section of the EMF tree view). Select the required key store from the list; in this case, the EMF server is the client to an external resource.
 3. If you need to use a proxy server, select the **Proxy** tab and click the **Use Proxy** check box. Then enter the **Proxy host** and **Proxy port** in the appropriate text boxes.

SMTP Service

Before you can use an [Email Out module](#) in the [EMF Process builder](#), you must configure an **SMTP service** in the **Services** section of the EMF tree view.

SMTP Service

Name: Test SMTP

Queue: EMAIL

From: [Warning Icon] [Empty]

From address: [Warning Icon] [Empty]

Email server: 147.183.128.50

Port number: 25

Timeout (secs): 60

Connection: Normal

Authentication: None

User name: [Empty]

Password: [Empty]

☐ Automatically Trust Secure Connection Certification

OK Apply Test Cancel Help

- **Name** - The name entered here will be referenced throughout the EMF System to identify this instance of a SMTP Service.
- **Queue** - this is the queue that the SMTP service will use. When an email is sent via this server, the queue selected will process the requests for emails to be sent to recipients.

Note: Unless you are an advanced user and specifically need to distribute the load, you should always leave the setting as **<default>**.

- **From** - The name entered here will be added to the email as a specification of who the EMF Process information has been sent by.
- **From address** - Enter the email address to be used to send EMF Process information to Recipients.
- **Email Server** - Enter the name of the computer that is used to send the email from. This can be in the form of an IP address or computer name.
- **Port number** - This is the port number that will be used.
- **Timeout (secs)** - This is the limit to the amount of time that will be spent attempting to connect to the SMTP server. Entering 0 as the timeout value will force the connection to wait forever.

- **Connection** - This specifies the type of connection that is required for the SMTP Service. There are three choices of connection available from the drop down list:
 - **Normal**
 - **SSL/TLS**
 - **STARTTLS**
- **Authentication** - This specifies the type of authentication that is required to ensure the message is authorized. There are two choices of authentication available from the drop down list:
 - **None** - No authentication will take place before the email is sent.
 - **Standard** - The authentication mechanism is chosen among those supported by the SMTP server.
- **User Name** - User name for the sender's mail account.
- **Password** - Password for the sender's mail account.
- **Automatically Trust Secure Connection Certification** - Select this check box to automatically trust any **self-signed SSL Certificates**.

Note: If the **Automatically Trust Secure Connection Certification** check box is not selected and if the SSL certificate is not signed by a trusted authority, then the application returns an error.

Example Email Settings

The following examples explain how to connect to Gmail and Hotmail using different connections:

- Configuring Gmail using SSL
- Configuring Gmail using STARTTLS
- Configuring Hotmail

Configuring Gmail using SSL

To configure Gmail using SSL, enter the following settings in the **SMTP Service** dialog box:

1. In the **Name** field, type "Gmail (yourname) SSL".
2. From the **Queue** drop down list, select **WIP**.
3. In the **From** field, type your first name and last name.
4. In the **From** address field, type your gmail address - "yourname@gmail.com".
5. In the **Email server** field, type "smtp.gmail.com".
6. In the **Port number** field, type "465".
7. In the **Timeout (secs)** field, type "60".
8. From the **Connection** drop down list, select **SSL/TLS**.
9. From the **Authentication** drop down list, select **Standard**.
10. In the **User name** field, type your username - "yourname@gmail.com".

11. In the **Password** field, type your email password if required.
12. Click **Apply** and then **OK**, to save the settings.

Configuring Gmail using STARTTLS

To configure Gmail using STARTTLS, enter the following settings in the **SMTP Service** dialog box:

1. In the **Name** field, type "Gmail (yourname) STARTTLS".
2. From the **Queue** drop down list, select **WIP**.
3. In the **From** field, type your first name and last name.
4. In the **From** address field, type your gmail address - "yourname@gmail.com".
5. In the **Email server** field, type "smtp.gmail.com".
6. In the **Port number** field, type "587".
7. In the **Timeout (secs)** field, type "60".
8. From the **Connection** drop down list, select **STARTTLS**.
9. From the **Authentication** drop down list, select **Standard**.
10. In the **User name** field, type your username - "yourname@gmail.com".
11. In the **Password** field, type your email password if required.
12. Click **Apply** and then **OK**, to save the settings.

Configuring Hotmail

To configure Hotmail, enter the following settings in the **SMTP Service** dialog box:

1. In the **Name** field, type "Hotmail (yourname)".
2. From the **Queue** drop down list, select **WIP**.
3. In the **From** field, type your first name and last name.
4. In the **From** address field, type your Hotmail address - "yourname@hotmail.com".
5. In the **Email server** field, type "smtp.live.com".
6. In the **Port number** field, type "587".
7. In the **Timeout (secs)** field, type "60".
8. From the **Connection** drop down list, select **STARTTLS**.
9. From the **Authentication** drop down list, select **Standard**.
10. In the **User name** field, type "yourname@hotmail.com".
11. In the **Password** field, type your email password if required.
12. Click **Apply** and then **OK**, to save the settings.

[Configuring Services](#)

[Configuring the EMF System](#)

SNMP Service Screen

Before you can use an SNMP module in the [EMF Process builder](#) you must configure an **SNMP service** in the **Services** section of the EMF tree view.

Note: the **SNMP services screen** has two tabs, but only two of the fields (**Name** and **Remote address** on the **Properties** tab) are mandatory. All of the other fields are either optional, or else contain default values. If you need to change these default settings you will need to know details of the **Port numbers** that the service must use, the **Community strings** (SNMPv1 or SNMPv2c), and the **Security level, authentication** and **privacy** details (SNMPv3). If you need to change the default settings but do not know these details, consult your system administrator.

The **Properties** tab allows you to type a descriptive name for the SNMP service that will enable users to identify it in the **Name** field, and to select the queue on which incoming or outgoing SNMP messages should be placed from the **Queue** dropdown list. The default queue selected is WIP.

Note: Unless you are an advanced user and specifically need to distribute the load, you should always leave the setting as **<default>**.

The screenshot shows the 'SNMPService' configuration window with the 'Properties' tab selected. The window is divided into two main sections: 'Service properties' and 'SNMP properties'. In the 'Service properties' section, the 'Name' field is empty with a yellow warning icon to its left, and the 'Queue' dropdown menu is set to 'WIP'. The 'SNMP properties' section contains several fields: 'Community' and 'Write community' are both set to 'public'; 'Remote address' is empty with a yellow warning icon; 'SET port' is set to '161'; 'TRAP port' is set to '162'; 'Retries' is a spinner box set to '0'; and 'Timeout (ms)' is set to '5000'. At the bottom of the window are four buttons: 'OK', 'Apply', 'Cancel', and 'Help'.

Fill in the details as follows:

- **Community** and **Write community**: Enter data in these fields only if you wish to use the basic authentication provided by SNMPv1 and SNMPv2c. Type the community string for the read only community in the **Community** textbox. Type the community string for the read-write community in the **Write community** textbox. An SNMP element (manager or agent) will accept a message from another SNMP element only if the message contains the correct community string.
- **Remote address**: Enter the network address of the SNMP agent to which requests will be sent. This can be an IP address or a network name.
- **SET port**: Type the port that Set, Get, Get next and Get bulk requests should be sent to. The default value is 161.
- **TRAP port**: Type the port that Trap and Inform messages should be sent to. The default value is 162.
- **Retries**: Type the number of times EMF should try resending a failed request before giving up.
- **Timeout (ms)**: Type the length of time (in milliseconds) that EMF should wait for a response before timing out.

Note: If you want to use SNMPv3 security, see [Security tab](#).

[SNMP Overview](#)

SNMP Service Properties Tab

You can use the **SNMP properties** tab to set SNMP-specific properties for the SNMP output service.

Note: If you want to use SNMPv3 security, see [Security tab](#).

Fill in the details as follows:

- **Community** and **Write community**: Enter data in these fields only if you wish to use the basic authentication provided by SNMPv1 and SNMPv2c. Type the community string for the read only community in the **Community** textbox. Type the community string for the read-write community in the **Write community** textbox. An SNMP element (manager or agent) will accept a message from another SNMP element only if the message contains the correct community string.
- **Remote address**: Enter the network address of the SNMP agent to which requests will be sent. This can be an IP address or a network name.
- **Set port**: Type the port that Set, Get, Get next and Get bulk requests should be sent to. The default value is 161.
- **Trap port**: Type the port that Trap and Inform messages should be sent to. The default value is 162.
- **Retries**: Type the number of times EMF should try resending a failed request before giving up.

- **Timeout:** Type the length of time (in milliseconds) that EMF should wait for a response before timing out.

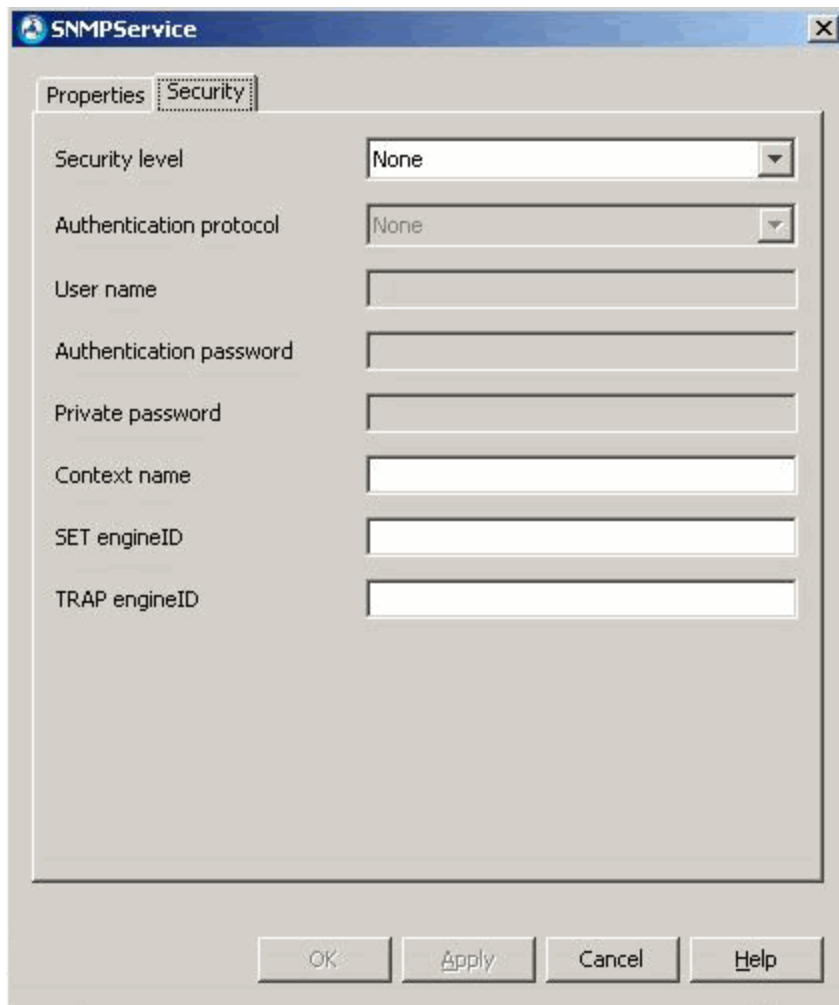
[Security tab](#)

[SNMP Service Screen](#)

SNMP Service Security Tab

In the **Security** tab you can enter the details for using SNMPv3 security.

You should only enter data in this tab if you wish to use SNMPv3 security features.

A screenshot of the 'SNMPService' dialog box, specifically the 'Security' tab. The dialog has a title bar with the text 'SNMPService' and a close button. Below the title bar are two tabs: 'Properties' and 'Security', with 'Security' being the active tab. The main area of the dialog contains several labeled input fields: 'Security level' (a dropdown menu currently showing 'None'), 'Authentication protocol' (a dropdown menu currently showing 'None'), 'User name' (a text box), 'Authentication password' (a text box), 'Private password' (a text box), 'Context name' (a text box), 'SET engineID' (a text box), and 'TRAP engineID' (a text box). At the bottom of the dialog are four buttons: 'OK', 'Apply', 'Cancel', and 'Help'.

Fill in the details as follows:

- **Security level:** Select the SNMPv3 security level from this drop-down list. The available options are **NoAuthNoPriv** (no authentication, no privacy), **AuthNoPriv** (authentication, no privacy) and **AuthPriv** (both authentication and privacy). AuthPriv offers the highest level of security.

- **Authentication protocol:** If you selected AuthNoPriv or AuthPriv as the security level, select **SHA-1** or **MD5**.
- **User name:** Type the user in whose name the requests or traps are being sent. The user must exist on both the source and destination SNMP network elements. In addition, the user's security settings on both source and destination elements must match.
- **Authentication password:** If you selected an authentication protocol, type the authentication password for the user here.
- **Private password:** If you selected AuthPriv as the security level, type the encryption password here.
- **Context name:** Type the SNMP context name here. This is an octet string that together with the EngineID uniquely identifies a particular SNMP context, or set of management information. Management information can exist in more than one context, and an SNMP entity can access multiple contexts.
- **SET EngineID:** Enter the ID of the authoritative SNMP engine for Set and Get requests here. This is usually the SNMP agent. If this is left blank, the EMF server will retrieve the EngineID automatically.
- **TRAP engineID:** Enter the ID of the authoritative SNMP engine for Trap and Inform requests here. This is usually the SNMP agent. If this is left blank, the EMF server will retrieve the EngineID automatically or generate its own.

[SNMP Service Screen](#)

[Configuring the EMF System](#)

Queuing Services

Before you can use a [Queue Listener](#) or [Queue Output module](#) in the [EMF Process Builder](#) you must configure both an **External Queue Provider** and an **External Queue** in the **Services** section of the EMF tree view.

- You can use the [External queue provider screen](#) to specify details for connection to an JMS external queue provider. When you have done this you can configure an **external queue**.
- You can use the [External queue screen](#) to specify the details of an external queue (this is the logical queue to which messages will be delivered).

Important: These queues are used by the [Queue Listener](#) and [Queue Output](#) modules in the EMF Process Builder; they are not the same as the EMF system queues. To configure the EMF system queues, see [Configuring the EMF Queues](#).

Note: The communication between EMF and the queueing system is handled by the Java Message Service (JMS). For further information on JMS, see [Messaging and the Java Message Service](#) or consult the JMS home page at <http://java.sun.com/products/jms/>.

[Creating Queues](#)

[Configuring the EMF Queues](#)

[Configuring the EMF System](#)

External Queue Provider Service

Important: You must configure both an external queue provider service and an [external queue service](#) before you can use a [Queue Listener](#) or [Queue Output module](#).

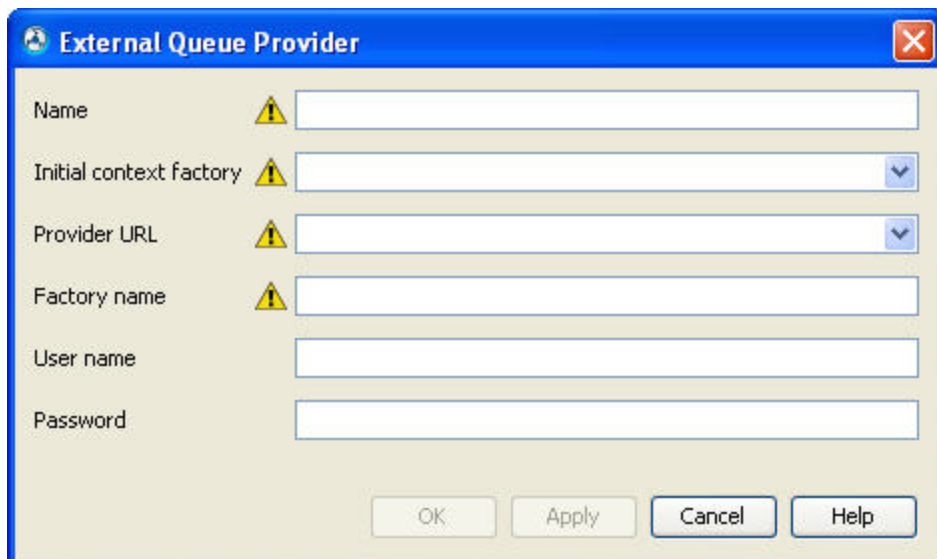
You can use the **External queue provider screen** to specify details for connection to an JMS external queue provider. When you have done this you can then configure an external queue.

Note: The queues used by the Queue Listener and Queue Output modules in the EMF Process Builder are not the same as the internal queueing system that EMF runs on (although you can use the same provider, if you wish). For information on configuring an internal queueing system, see [Configuring the EMF Queues](#).

To specify an external queue provider service:

1. Double-click the **Queue services** icon in the **Services** section of the EMF tree view.
2. Double-click the **External queue providers** icon to display the list of existing queue providers, then:
 - **To modify an existing queue provider**, double-click its icon.
 - **To create a new queue provider**, right-click in the list view and click **New external queue provider**.

The **External Queue Provider** properties screen is displayed.

The image shows a Windows-style dialog box titled "External Queue Provider". It has a blue title bar with a close button (X) in the top right corner. The main area is light gray and contains several input fields, each preceded by a yellow warning triangle icon. The fields are: "Name" (text box), "Initial context factory" (dropdown menu), "Provider URL" (dropdown menu), "Factory name" (text box), "User name" (text box), and "Password" (text box). At the bottom of the dialog, there are four buttons: "OK", "Apply", "Cancel", and "Help".

3. Enter a descriptive **Name** for the service that will help users to identify it.

4. Enter the **Initial context factory** name for the JNDI naming context.

An example of an initial context factory name is [com.sun.jndi.fscontext.RefFSContextFactory](#). EMF uses the initial factory and the provider URL to perform a JNDI lookup to find the provider configuration information.

5. Enter the URL of the root directory of the JNDI naming context where the lookup should be performed in the **Provider URL** field. For example, to specify your local C: root directory, type file:///c:/.

Important: On a Windows platform, the .bindings file is used for the lookup. Before you can use this external queue provider, you must update the .bindings file in the **Provider URL** location.

6. Type the name of the queue factory, e.g. QCF, in the **Factory name** field. EMF resolves this within the JNDI naming context.
7. If required, enter the **User name** and **Password** to log in to the external queue provider.

Important: If you modify any of the settings for an existing queue, you must shut down and restart the EMF Server in order for your changes to take effect.

Note: The communication between EMF and the queueing system is handled by the Java Message Service (JMS). For further information on JMS see [Messaging and the Java Message Service](#) or consult the JMS home page at <http://java.sun.com/products/jms/>.

[External Queue Screen](#)

[Queue Service](#)

[Configuring the EMF System](#)

External Queue Screen

Important: you must configure both an external queue service and an [external queue provider service](#) before you can use a [Queue In](#) or [Queue Output module](#).

You can use the **External queue screen** to specify the details of an external queue. This is the logical queue to which messages will be delivered. Before you can do this, you must first configure an external queue provider in the [External queue provider screen](#).

Note: the queues used by the Queue In and Queue Output modules in the EMF Process Builder are not the same as the internal queueing system that EMF runs on (although you can use the same provider, if you wish). For information on configuring an internal queueing system, see [Configuring the EMF Queues](#).

To specify an external queue service:

1. Double-click the **Queue services** icon in the **Services** section of the EMF tree view.

2. Double-click the **External queues** icon to display the list of existing queues, then:
 - **To modify an existing queue**, double-click its icon.
 - **To create a new queue**, right-click in the list view and click **New external queue**.

The **External Queue** properties screen is displayed.

3. Enter an appropriate **Name** for the queue that will allow you to identify it.
4. Select the **External queue provider** that provides access to this queue from the drop-down list of those that you have created using the [External Queue Provider](#) screen.
5. Enter the JNDI lookup name for the queue in the **Lookup name** field. This allows EMF to locate its configuration information within the JNDI naming context.
6. If required, enter a **Description** for the queue (e.g. you may want to explain the purpose of the queue so that users will know when they should use it) and click **OK** to close the External queue screen.
7. Select **Exclusive Queue** to run the queue in [Exclusive](#) mode.
8. Now you must update your queue provider's JNDI bindings to make this queue available under the appropriate lookup name. If you are using SwiftMQ you can do this using Swift Explorer, otherwise you should use whichever tool is supplied with the JNDI service that you are using.

Important: if you modify any of the settings for an existing queue, you must shut down and restart the EMF Server in order for your changes to take effect.

Note: the communication between EMF and the queueing system is handled by the Java Message Service (JMS). For further information on JMS see [Messaging and the Java Message Service](#) or consult the JMS home page at <http://java.sun.com/products/jms/>.

[External Queue Providers Screen](#)

[Queue Service](#)

[Exclusive Queue - Single Thread Operation/Message Order Dependency](#)

Messaging and the Java Message Service

This topic provides an overview of JMS (the Java Message Service), which is used by EMF to communicate with queueing systems such as SwiftMQ. The following background information is intended to help you in configuring queue services and queue listeners for external queue providers.

Introduction

Applications can use messages to send event notifications and data to other applications. Messages between message clients are sent through an enterprise messaging product,

such as SwiftMQ.

Messaging is typically asynchronous, which means that message clients can send and receive messages at any time - there is no need to wait for an application to respond to a message before sending or receiving other messages.

Message clients sending messages are known as message producers. Message clients receiving messages are known as message consumers. A message client can be both a message producer and a message consumer.

Java Message Service (JMS) is a vendor-independent Java API that can be used by applications to create, send, receive and read messages. Messaging products that support JMS have a JMS provider that allows JMS clients to send and retrieve JMS messages through the messaging system.

JMS Messaging Styles

The JMS API supports two messaging models:

- Point-to-point (PTP)
- Publish and Subscribe (Pub/Sub)

In PTP messaging, the message producer puts a message on a queue. Only the message consumer(s) for that message can retrieve the message from the queue (and acknowledge receipt).

In Pub/Sub messaging, the message producer broadcasts the message to a topic. Message consumers can subscribe to a topic. All subscribers to a topic receive the message.

Note: EMF supports only point-to-point messaging.

JMS Messages

A JMS message consists of the following:

- **Message header:** These fields are used to identify and route messages.
- **Message properties** (optional): These fields contain proprietary headers, or standard optional headers.
- **Message body:** The message body can be in one of five defined formats.
 - **TextMessage:** plain text string, java.lang.String (in XML, if desired).
 - **MapMessage:** Name/value pairs, where names are strings and the values are Java primitive types, such as Boolean or Long.
 - **BytesMessage:** An uninterpreted bytestream.
 - **StreamMessage:** A stream of primitive values.
 - **ObjectMessage:** A serializable object.

The JMS API

JMS has two administered objects:

- ConnectionFactory
- Destination

The objects are part of the JMS specification, but they are administered by the JMS provider. The configuration information in these objects is provider-specific.

The ConnectionFactory information is used by a JMS client to establish a connection to a JMS provider.

The Destination information specifies the destination of messages being sent and source of messages received. In the case of PTP messaging, the destination will specify a queue name.

Both the ConnectionFactory and Destination information required by JMS clients is typically placed in a JNDI (Java Naming and Directory Interface) namespace. The clients then perform a JNDI lookup to obtain these details.

After the client has obtained the connection configuration information through a JNDI lookup, it can use this to establish a connection to the JMS provider and the destination. A session is then created and the connection is started.

The client can use message selectors to filter messages based on the contents of their message headers.

For more information on the Java Message Service, refer to the JMS documentation on the Oracle web site: <http://www.oracle.com/us/sun/index.htm>

[Configuring the EMF Queues](#)

[Creating Queues](#)

[Configuring the EMF System](#)

SMS Service

Before you can use an [SMS output module](#) in the [EMF Process builder](#) with the **SMS Service**, you must configure the service.

Note: Before you can use the SMS Service, you must register for an SMS account. Customers should register at www.twilio.com to create a Twilio account for use for sending SMS messages (note: charges apply when sending messages).

To configure the SMS service:

1. Double-click the **SMS Service** icon in the **Services** section of the EMF tree view.
 - **To modify an existing SMS service**, double-click its icon.
 - **To create a new SMS service**, right-click in the list view and click **New**.

The **SMS Service** properties screen is displayed.

2. Enter an appropriate **Name** to use when the service is referenced by EMF Process modules.

3. Select a **Queue** to handle the SMS information from the drop-down list of those available.

Note: Unless you are an advanced user and specifically need to distribute the load, you should always leave the setting as **<default>**.

4. Select the **SMS Provider** from the drop-down.
5. Follow the below steps based on the selection of the SMS Provider.

For Apteian SMS Provider:

- a. Enter a name in the **SMS from** field. The name that you enter will be displayed on the recipient device as the sender of the message.
- b. Do not enter any value in the **Retries** field, since Apteian SMS does not currently support retry. This option, when it becomes available, will specify how many more times to try to send the message if the first attempt fails.
- c. Enter your Apteian SMS **Customer ID**, **Customer Name**, and **Customer Password** (provided by Apteian).

For Twilio SMS Provider:

- a. Enter the phone number in the **SMS from number** field.
- b. Enter your Twilio **Account SID** and **Authentication token (AUTH TOKEN)**. These credentials are available from your Twilio account settings at www.twilio.com.

Note: You can hide the authentication token by checking the **Hide Authentication token value** checkbox.

[Configuring Services](#)

Apteian Respond Service

Respond is a case management tool sold by Apteian. EMF can be used to automate the creation and updating of cases in Respond. For example, a process can be built in EMF to take customer complaint records from one system and automatically create case records in Respond for each complaint. A minimum version of Respond v5.9 is needed to operate with EMF. The user should have read and be familiar with the "Respond Web Services User Guide.pdf" available with Respond.

Before using the [Apteian Respond Module](#), a Respond service needs to be created. This service defines the instance of Respond that EMF will communicate with. This is done in the **Services-> Apteian Respond Service** section of the EMF tree. The following properties can be defined:

Respond Service

Name: WIN2008-CAT - Forms

Queue: WIP

Respond Database: EMF4

Webservice URL: http://win2008-cat/Respond/ws

Locale: en-GB

Authentication: Forms

User name: Runner

Password: ••••••••

Domain:

Request Timeout (secs): 90

OK Apply Cancel Help

- **Name:** The name entered here will be referenced throughout the EMF System to identify this instance of a Respond Service.
- **Queue:** This is the queue that the Respond service will use. When the Respond Module runs, it will be run by a server manager that is listening to the selected queue.
- **Respond Database:** The name of the Respond Database to connect to.
- **Webservice URL:** The URL of the root Respond Web service address. This will be of the following form:
http(s)://<server name>/<virtual directory>/ws
- **Locale:** Use the .Net culture names; these follow the RFC 1766 standard in the format: "<languagecode2>-<country/regioncode2>", where <languagecode2> is a lowercase two-letter code derived from ISO 639-1 and <country/regioncode2> is an uppercase two-letter code derived from ISO 3166. For example, U.S. English is "en-US".
- **Authentication:** The authentication mechanism Respond is configured to accept either "Integrated", "HMAC", or "Forms". For more information about how each of these is configured in Respond, see the Respond documentation.
- **User name:** The Respond user name of a service user with rights to make Web service calls.
- **Name:** The name entered here will be referenced throughout the EMF System to identify this instance of a Respond Service.
- **Domain:** If "Integrated" authentication is selected, then the Windows domain to which the user belongs must be entered.

- **Request Timeout (secs):** The amount of time (in seconds) to try to make the Web service call before timing out.

[Configuring Services](#)

[Configuring the EMF System](#)

Webservices Internet Security Settings

The **Internet Security** tab of the Web Service service enables you to enter the following security settings needed to access the WSDL file, any associated files, and the actual Web service call:

1. Select the **Authentication** tab and enter the following information:
 - **Type:** From the **Type** drop-down list, select the type of authentication the Web service should use when trying to connect:
 - **None:** when no authentication is required.
 - **Basic/Digest** for basic or digest authentication.
 - **Windows Authentication** (NTLM) to authenticate using the Microsoft Windows credentials.
 - **User name:** This is the user name that you use to log on when Basic/Digest or Windows authentication is required.
 - **Password:** This is the password that accompanies the user name.
 - **Host:** The host computer for your Windows authentication.
 - **Domain:** The domain for your Windows authentication.
2. Select the **SSL Settings** tab and enter the following information:
 - **Ignore server certificate warnings and errors:** Select this check box if you want to bypass the certificate warnings and errors. For instance, if the server is using a self-signed certificate, then this option needs to be selected to prevent the EMF server from refusing the connection.
 - **Client Authentication Required:** Select this check box if client authentication is required for the Web Service settings.
 - **Key Store:** The key store lists all the certificates for client authentication that you have installed (using the [Certificates](#) node in the **Security** section of the EMF tree view). Select the required key store from the list; in this case, the EMF server is the client to an external resource.
3. If you need to use a proxy server, select the **Proxy** tab and click the **Use Proxy** check box. Then enter the **Proxy host** and **Proxy port** in the appropriate text boxes.

OPC DA Service

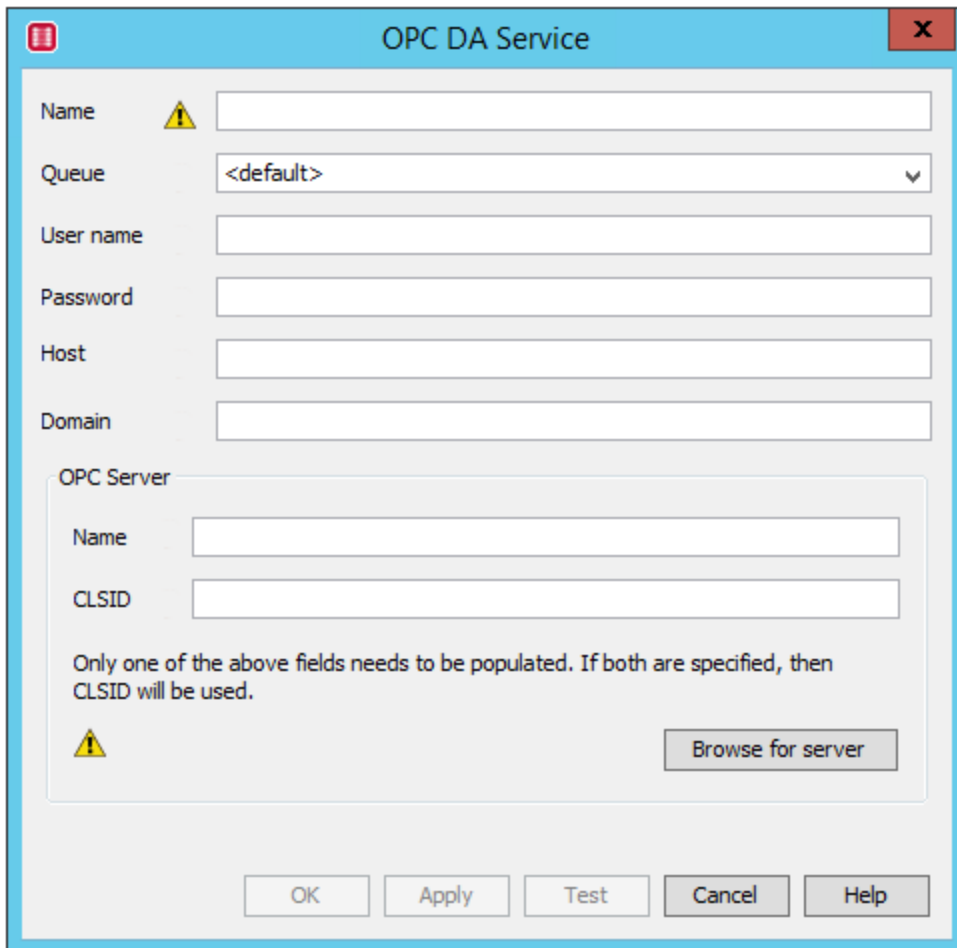
OPC DA (Open Process Communications Data Access) support in EMF allows the EMF server to act as an OPC client and read/write data to OPC servers. OPC is a standard typically used to allow industrial devices and control applications to communicate. For more information on OPC see <http://opcfoundation.org>. EMF supports OPC DA v2.0. OPC DA v3.0 and OPC UA (unified architecture) are not currently supported.

Before you can use the [OPC DA Read module](#) or [OPC DA Write module](#) in the EMF Process Builder you must configure an **OPC DA service** from the **Services** section of the EMF tree view. The OPC DA Service defines the connection details needed to connect to an OPC server.

To specify an OPC DA service:

1. Double-click the **OPC DA service** icon in the **Services** section of the EMF tree view.
 - To modify an existing OPC DA service, double-click its icon.
 - To create a new OPC DA service, right-click in the list view and click **New**.

The **OPC DA Service** properties screen is displayed.



The screenshot shows the 'OPC DA Service' configuration dialog box. It has a title bar with a red 'X' button. The dialog contains several input fields: 'Name' (with a yellow warning icon), 'Queue' (a dropdown menu showing '<default>'), 'User name', 'Password', 'Host', and 'Domain'. Below these is a section titled 'OPC Server' containing 'Name' and 'CLSID' fields. A note states: 'Only one of the above fields needs to be populated. If both are specified, then CLSID will be used.' There is a 'Browse for server' button next to the note. At the bottom are buttons for 'OK', 'Apply', 'Test', 'Cancel', and 'Help'.

2. Enter a **Name** for the OPC DA service.

3. Select a **Queue** from the drop-down list of those available. Typically you will leave the queue as "<default>".

Note: Unless you are an advanced user and specifically need to distribute the load, you should always leave the setting as **<default>**.

If selecting a different queue, then you need to make sure a [server manager](#) is defined to pick-up messages on that queue. The choice of queue can be used to define which machine processes the process in a multi-server environment.

4. Enter the **User name** and **Password** to access the OPC Server.
5. Enter the **Host** name that the OPC server is running on.
6. Enter the **Domain** to which the user belongs.
7. Click **Browse for server** to select the OPC server. You can edit the following:
 - **Name** - The Windows DCOM program Id of the server.
 - **CLSID** - The Windows DCOM class ID of the server.

Note: Only one of the above fields needs to be populated. If both are specified, then CLSID will be used.

8. Click the **Test** button to test the connection to the OPC Server.

Important: The machine that the OPC server is running on must be correctly configured to allow clients to communicate with it. This includes correctly configuring any firewalls and making sure the user has been assigned the correct rights. Refer to your OPC server documentation for more information.

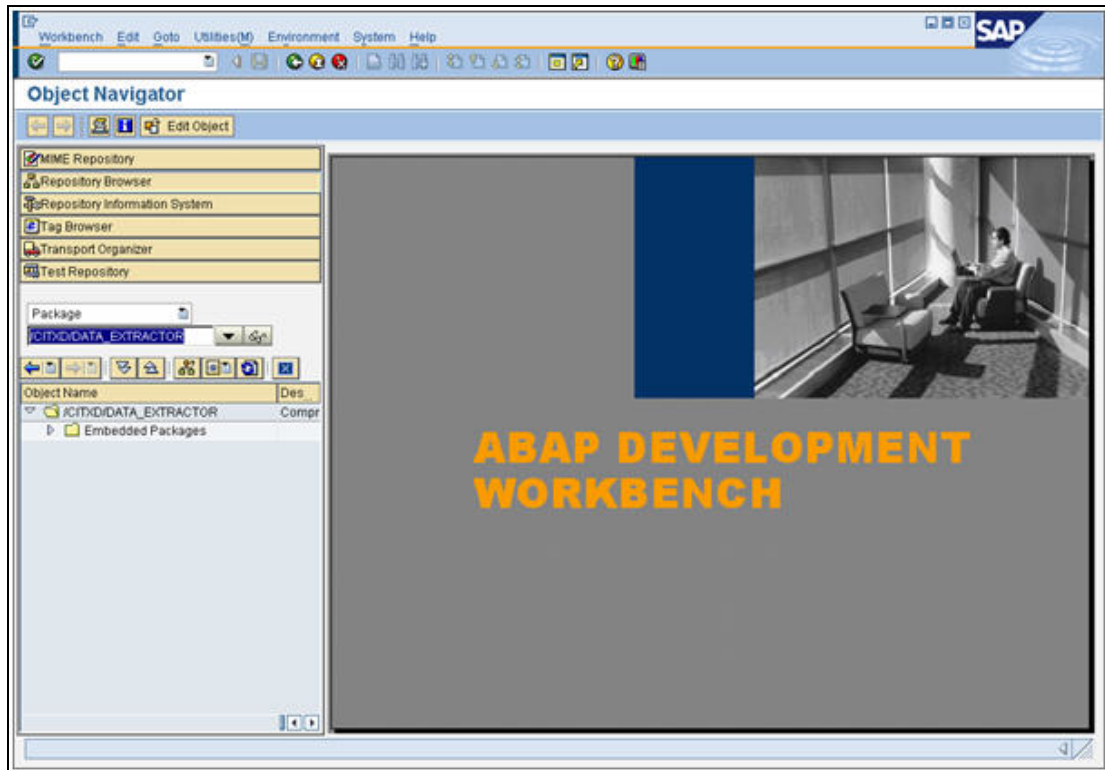
[Configuring Services](#)

[Configuring the EMF System](#)

How to Create SAP RFC Custom DataSource

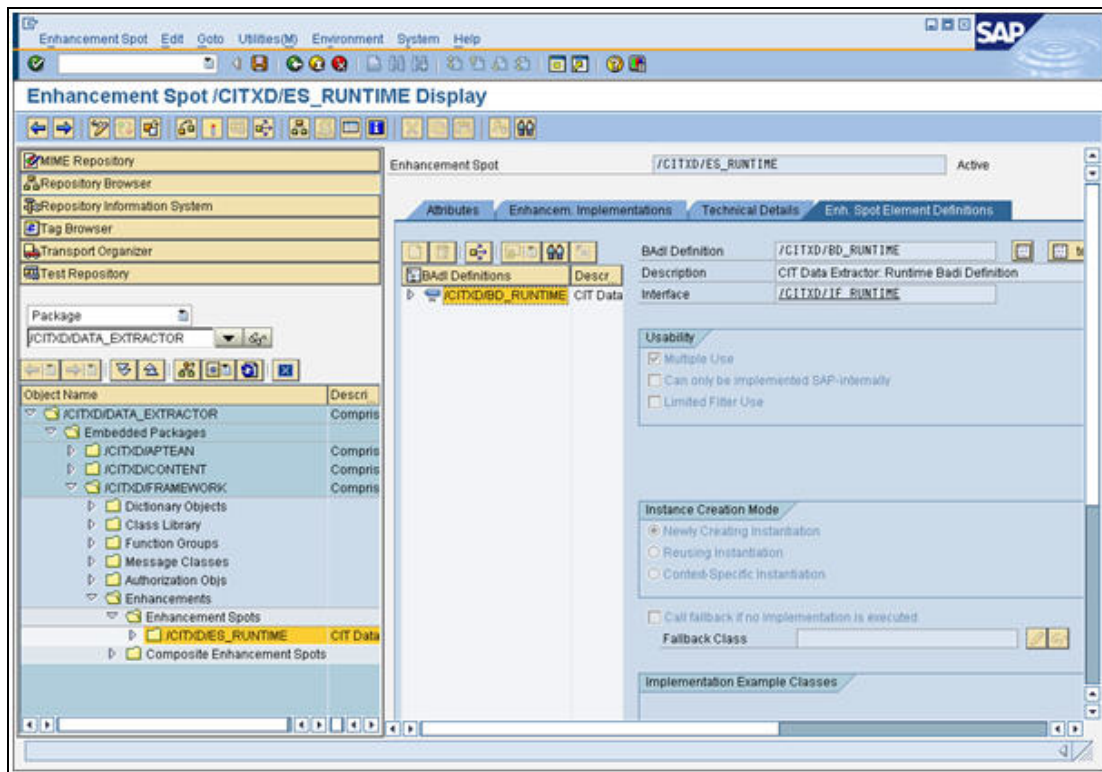
Setting up the Package


1. Create the Custom data source. Example: ZCUSTOM_DS_EXAMPLE that will contain fields for which data that needs to be extracted from the cluster tables. (Example: Table BSEG, BKPF). For an example Custom Data source set up, see [Appendix B](#).
2. In the SAP GUI, use transaction SE80 to navigate to **Object Navigator**.
3. Create a user package ZJR_CUSTOM4, or use an existing user package if one has already been created.
4. Navigate to /CITXD/DATA_EXTRACTOR package (enter the package name in the free-form entry field below the package and press the **Enter** key or click on the **Display** icon) as illustrated in the image below:

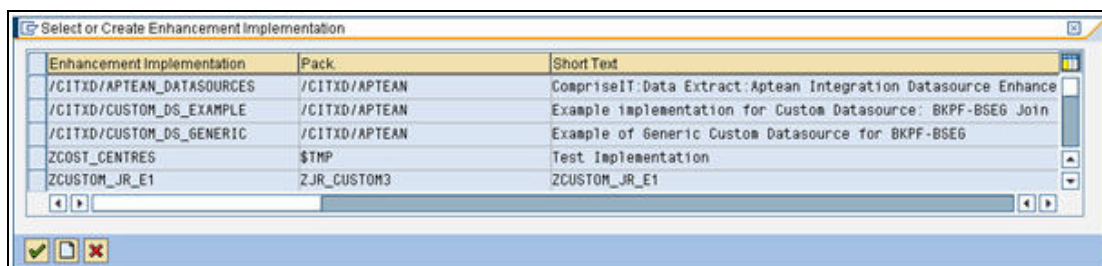



Creating the BAdI Implementation

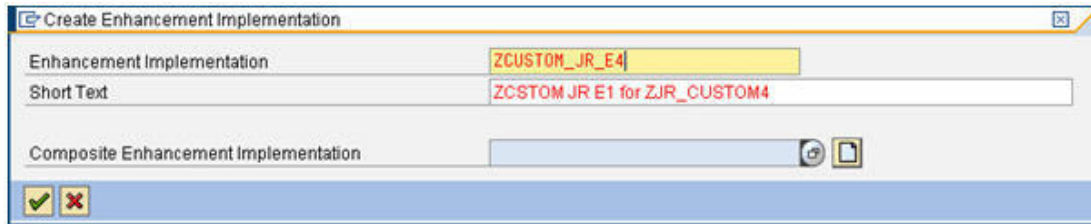
5. In the **Object Name** column, expand the /CITXD/DATA_EXTRACTOR node and select /CITXD/ES_RUNTIME in the **Enhancement Spots** node (as illustrated in the following image). If the right pane is not populated as shown below, right-click on the node and select **Display/In Same Window**.



6. In the right pane > **BAdI Definitions** column, select the **/CITXD/BD_RUNTIME BAdI** definition.
7. Click the **Create BAdI Implementation** icon  or right-click on the selected BAdI definition (/CITXD/BD_RUNTIME) to create a BAdI implementation. The following pop-up dialog is displayed:



8. Create a new **Enhancement Implementation** or select an existing one from the list. For illustration, let us create a new Enhancement Implementation.
9. Click the **Page** icon . Enter the name in the dialog as "ZCUSTOM_JR_E4".




Enhancement Implementation: ZCUSTOM_JR_E4

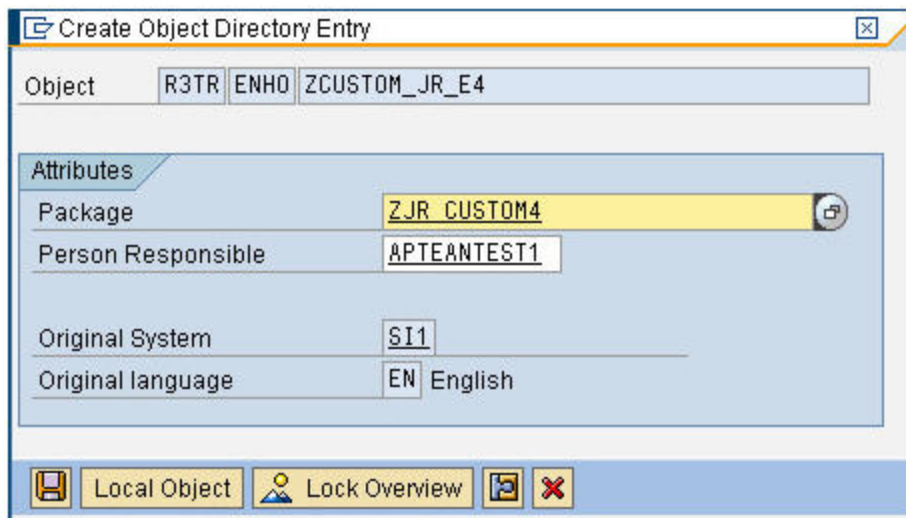
Short Text: ZCSTOMJR E1 for ZJR_CUSTOM4

Composite Enhancement Implementation: [icon]

[Checkmark] [X]

10. Click **Continue** .
11. Save the object in the package created above, ZJR_CUSTOM4.

Note: You can change the **Person Responsible** value, if required.



Object: R3TR ENHO ZCUSTOM_JR_E4

Attributes:

Package: ZJR_CUSTOM4

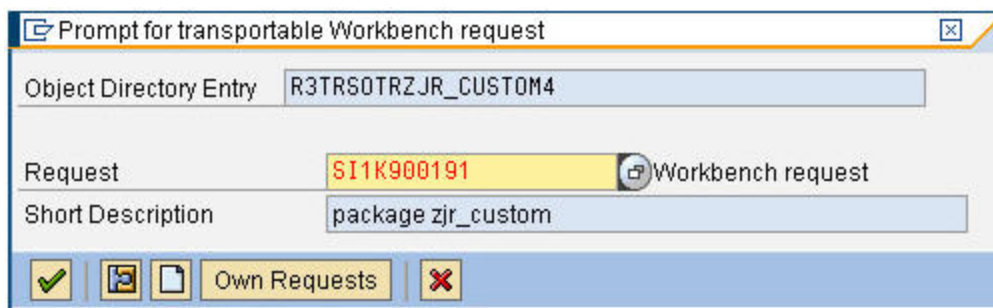
Person Responsible: APTEANTEST1

Original System: SI1

Original language: EN English

[Save] Local Object [Lock Overview] [X]

12. Create a transport, if prompted for transportable Workbench request. An existing Request can be used or a new one created.



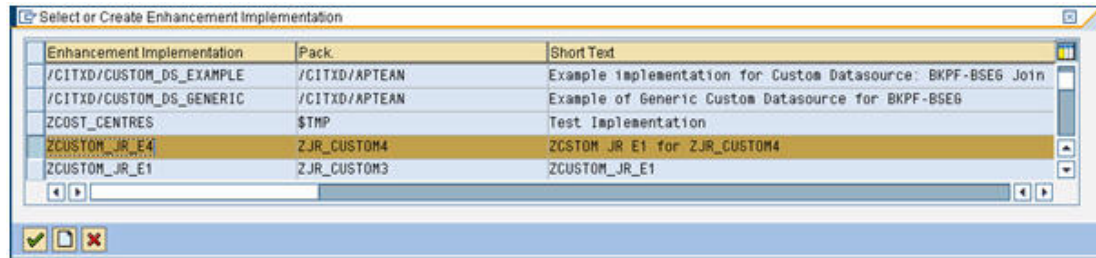
Object Directory Entry: R3TRS0TRZJR_CUSTOM4

Request: SI1K900191 Workbench request

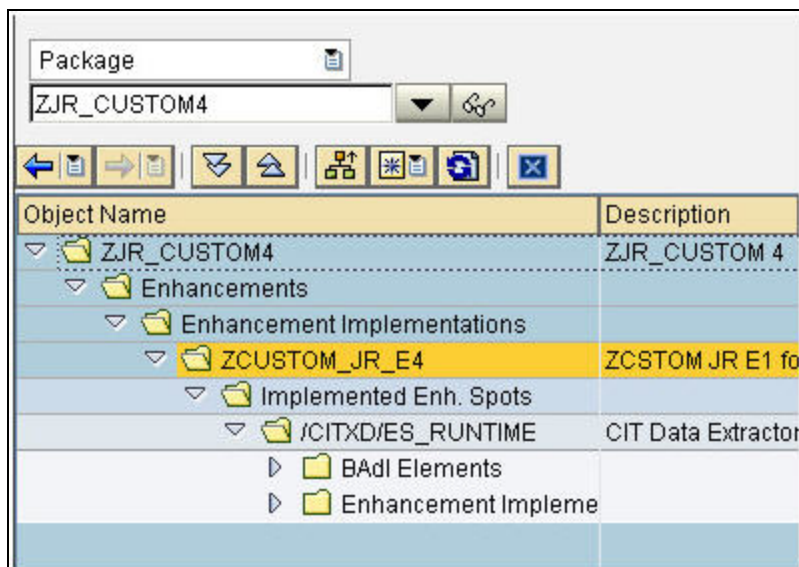
Short Description: package zjr_custom

[Checkmark] [Save] [Own Requests] [X]

The object will be created and shown in the list below:



13. Select the newly created object and click . The Create BAdI Implementation screen will be displayed. Click to cancel the creation as we will create the BAdI implementation later.
14. Navigate to the user package ZJR_CUSTOM4 and select the Enhancement Implementation that has just been created in the previous step (ZCUSTOM_JR_E4).



Tip: Expand the node to verify that this implementation has been implemented from the /CITXD/ES_RUNTIME Enhancement Spot. Activate (right-click on the ZCUSTOM_JR_E4 and select **Activate**) the Enhancement Implementation **ZCUSTOM_JR_E4**.

15. Double-click on the Enhancement Implementation **ZCUSTOM_JR_E4** to display the details of it in the right pane. Select **Change** mode (click or right-click **ZCUSTOM_JR_E4** and select **Change**).
16. In the right pane, create BAdI implementation (click). In the pop-up window, enter the name of the BAdI Definition /CITXD/BD_RUNTIME (or click the Selection List icon and select from the available list).

Definition	
Enhancement Spot	/CITXD/ES_RUNTIME
BAdI Definition	/CITXD/BD_RUNTIME

Implementation	
BAdI Implementation	ZCUSTOM_JR_BI
Implementing Class	ZCL_CUSTOM_JR
Short Text	ZCL CUSTOM JR - implementing class

17. Enter the BAdI implementation details:

BAdI Implementation – ZCUSTOM_JR_BI

Implementing Class – ZCL_CUSTOM_JR

18. Click **Continue** to create the BAdI implementation.

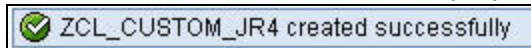
If a class is not created, you will be prompted to create one. Click **Y** and save the class in the package defined above ZJR_CUSTOM4.

Object	
R3TR	CLAS ZCL_CUSTOM_JR4

Attributes	
Package	ZJR_CUSTOM4
Person Responsible	APTEANTEST1
Original System	S11
Original language	EN English

19. Save transport request, if prompted.

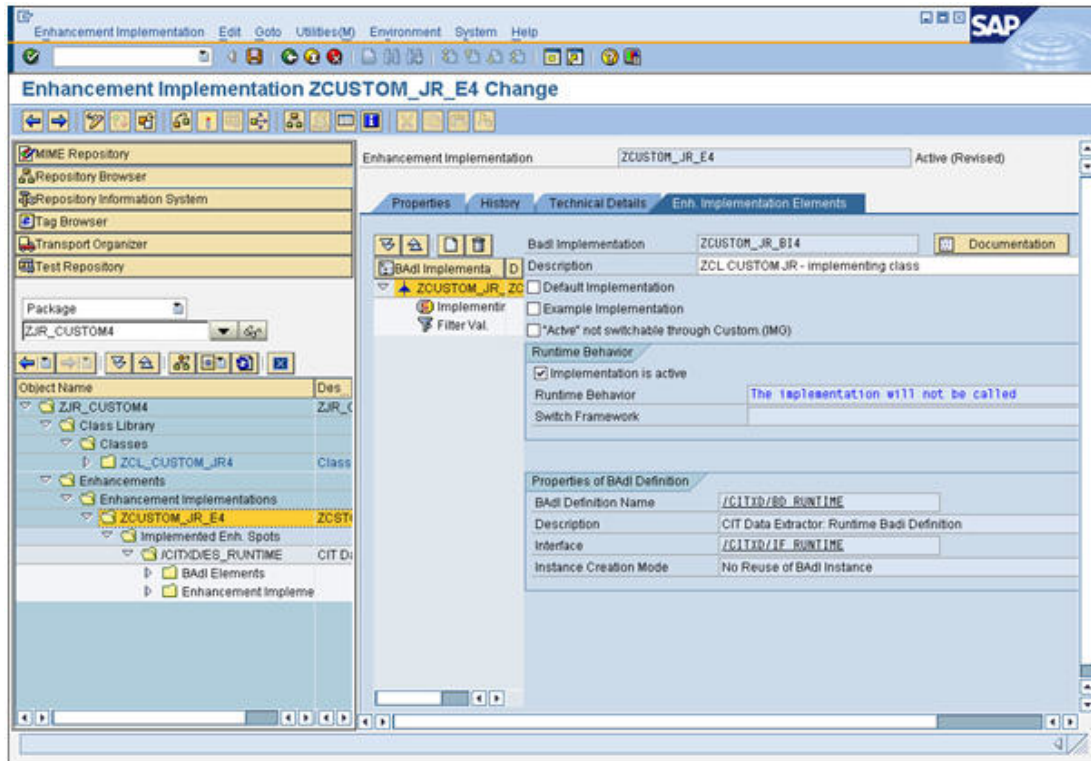
The status bar at the bottom of the SAP GUI displays "ZCL_CUSTOM_JR4 created successfully"



20. Expand the **BAdI Implementation** node (of the created BAdI Implementation) in the right pane.

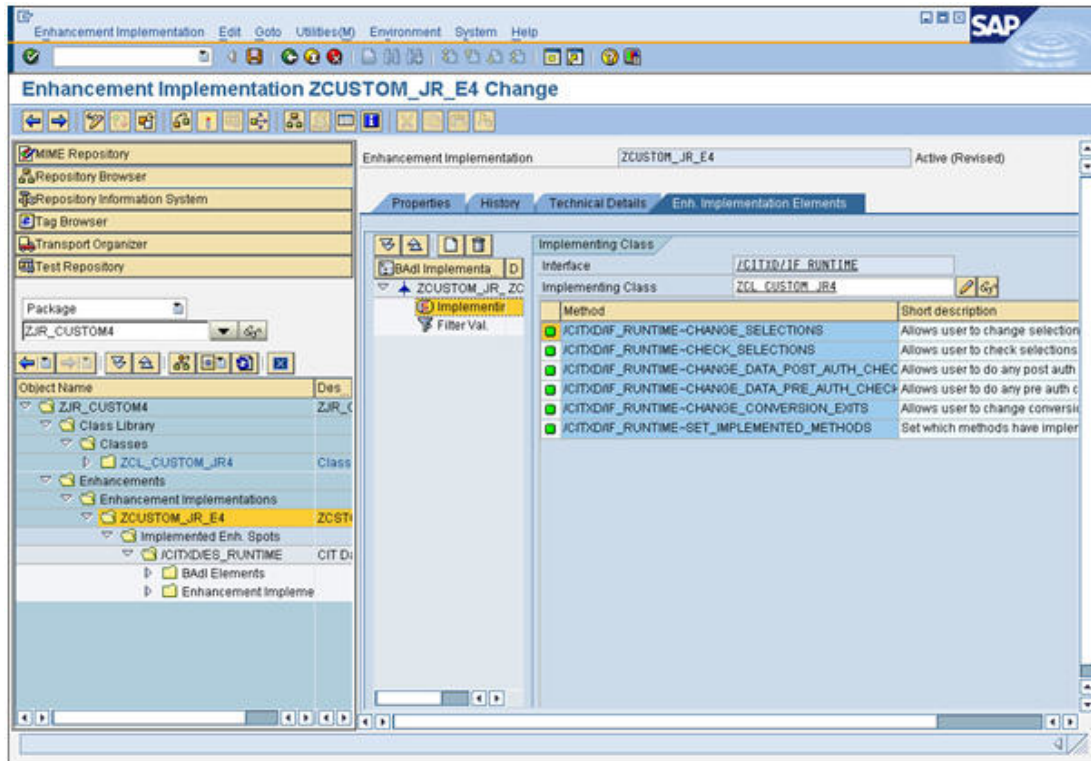
Notice that there are two sub-nodes: Implementing Class and Filter Val

21. Expand the **Class Library** node in the left pane and notice that the ZCL_CUSTOM_JR4 class has also been created.

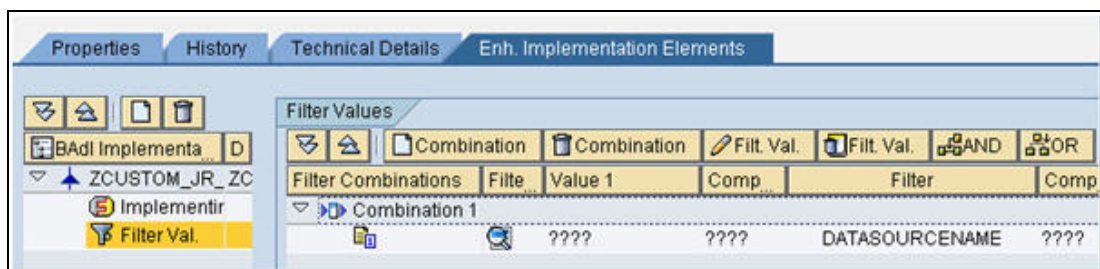


22. Double-click the **Implementing Class** node in the right pane to display the methods for that class. Ensure that all the relevant methods are listed.

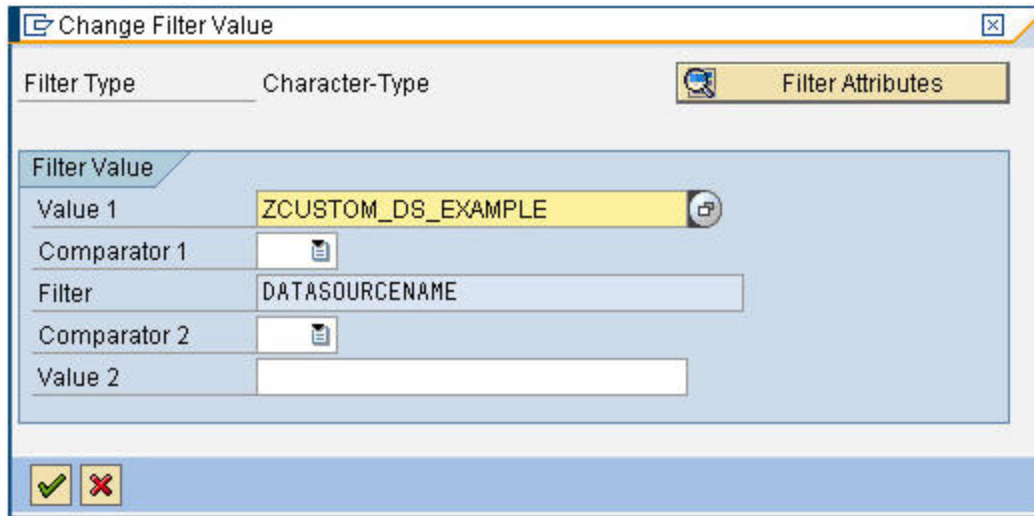
Note: The methods will be empty at this stage. You must populate the methods with the relevant ABAP code.



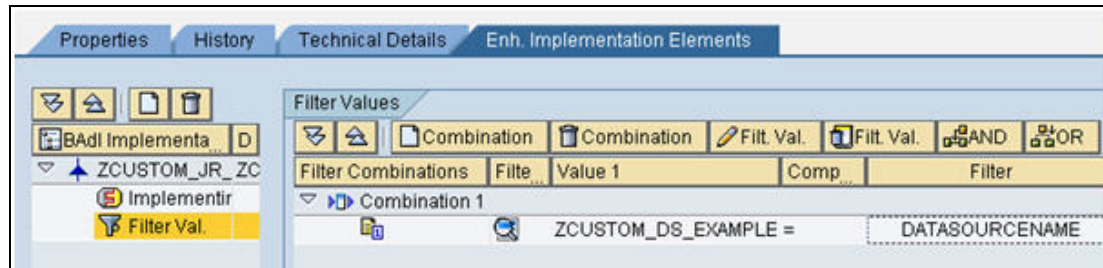
23. In order to restrict this BAdI Implementation to the data source created above, you must specify it in the **Filter** node. Double-click the **Filter** node and click **Combination**. An empty row will be created.



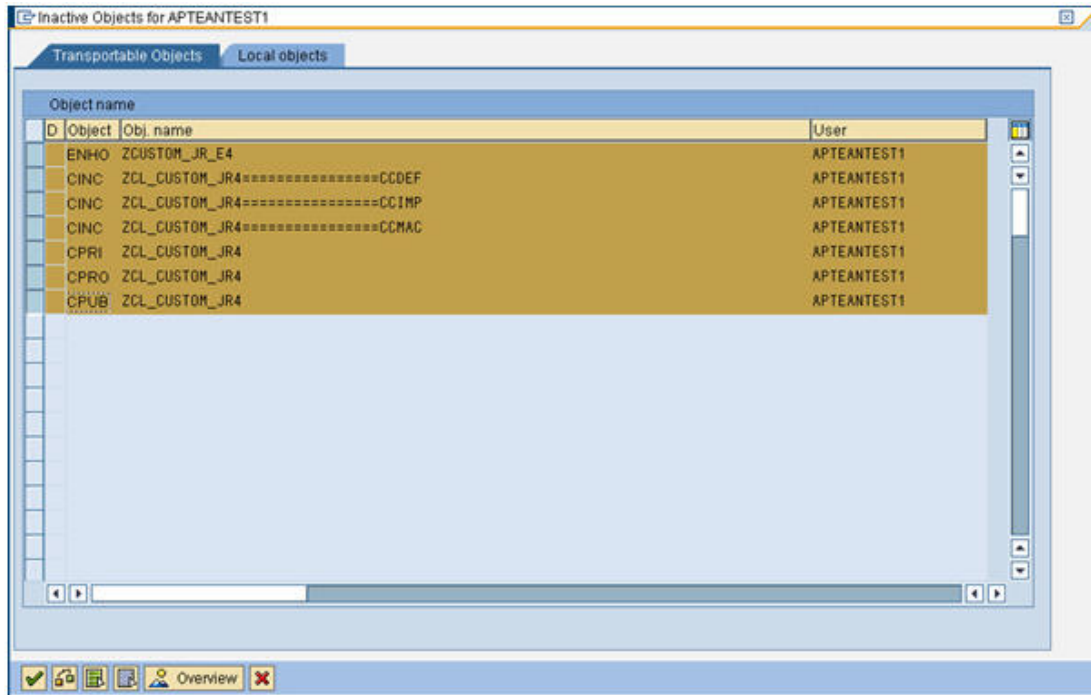
24. Double-click the newly added row and the **Change Filter Value** window is displayed. Enter the DataSource name (in **Value 1** field) for which this BAdI Implementation will be valid/used.



The following screen illustrates the updated screen:



25. Click **Save (Ctrl + S)**. A confirmation message "Data saved successfully" is displayed.
26. Activate (**Ctrl + F3**) the objects created. Select the created EI Implementation.

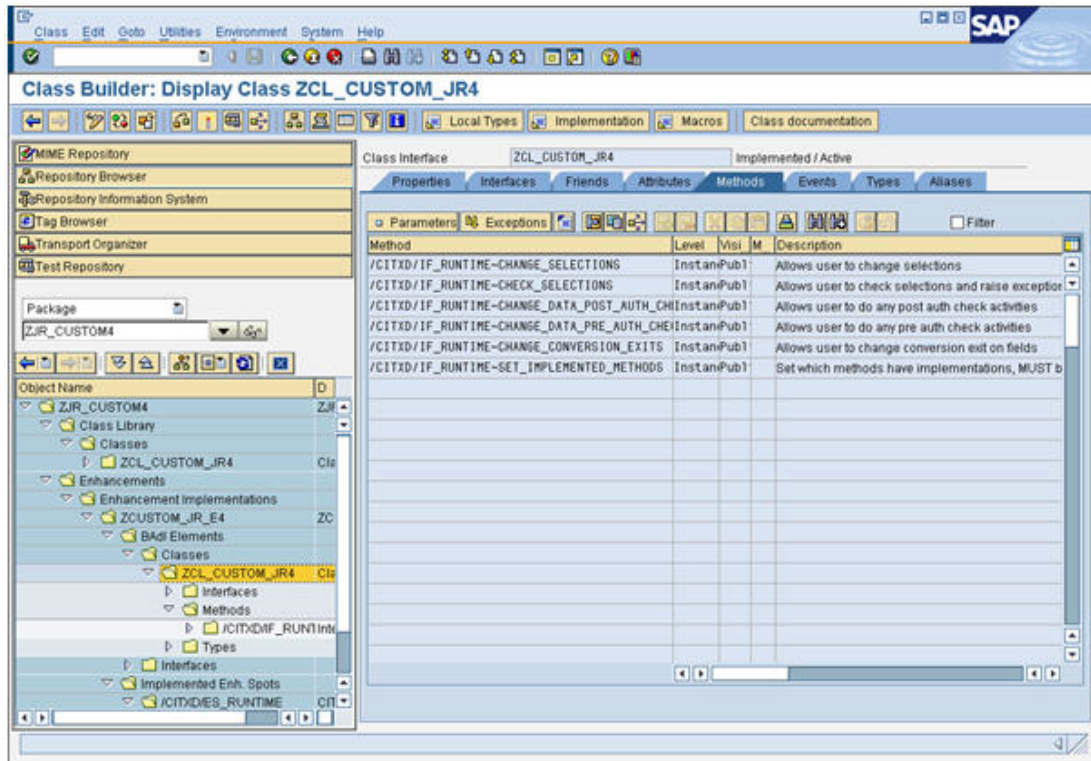


The message "Object(s) activated" is displayed at the bottom status bar.

Write ABAP code into the Class methods

Write ABAP code into the methods of the BAdI implementing Class ZCL_CUSTOM_JR4.

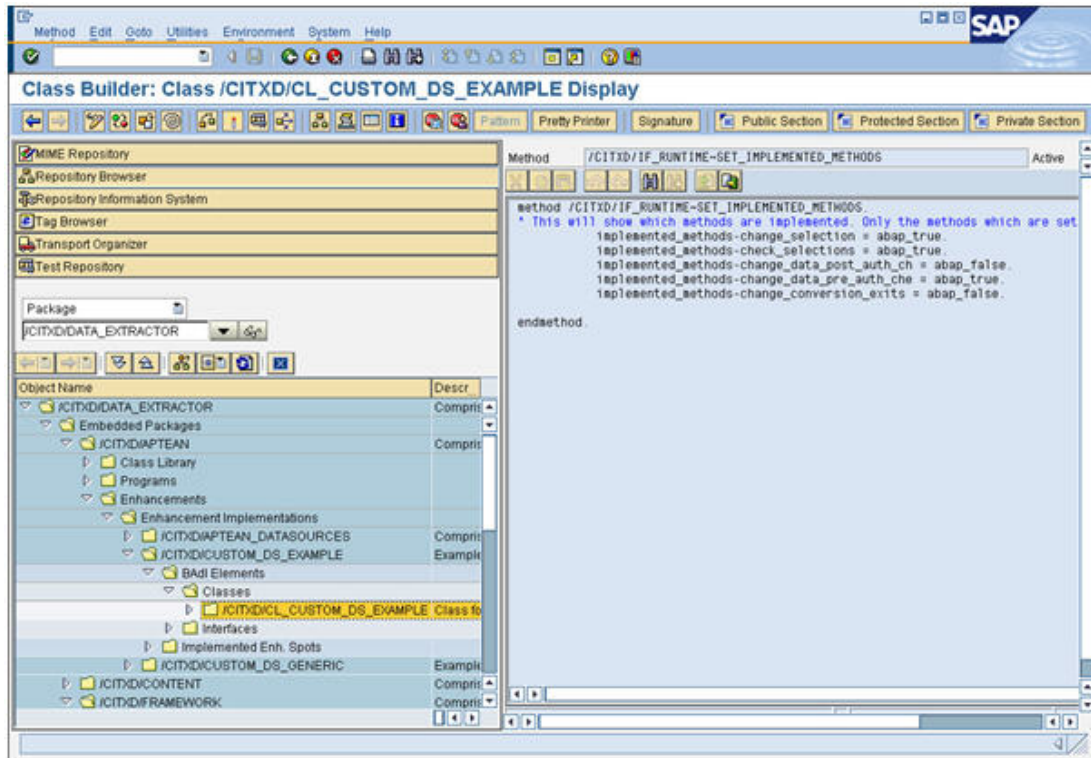
27. Navigate to the **BAdI elements Classes** node and select the **ZCL_CUSTOM_JR4** class.
28. Double-click on the **Implementing class** to display the Methods.



The methods are described in detail in Appendix A.

29. Edit the `/CITXD/IF_RUNTIME~SET_IMPLEMENTED_METHODS` (change to **Edit** mode and double-click on the `/CITXD/IF_RUNTIME~SET_IMPLEMENTED_METHODS`). Depending on what path is used to edit the method, the error "Interface method has not yet been implemented" can occur. This implies that the method needs to be implemented (change to **Edit** mode and open the method). An empty body with just the begin and end method statements will be displayed.

Note: The best way to populate the ABAP code is to copy the code from the existing example class methods (`/CITXD/CUSTOM_DS_EXAMPLE`). This code can be used as a base.

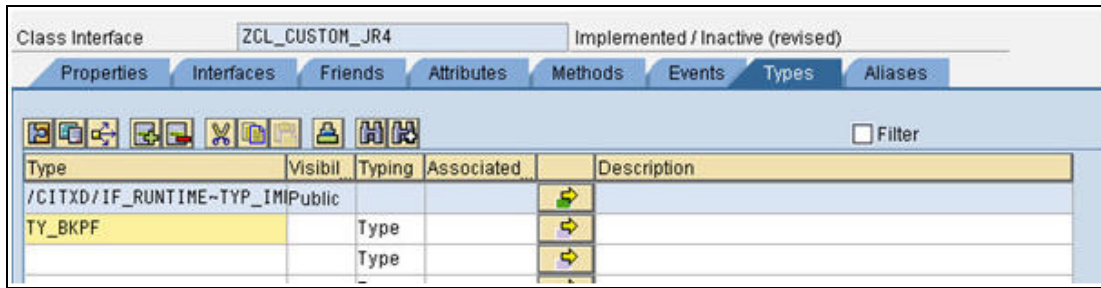



30. As a minimum, the following methods must be populated:

- /CITXD/IF_RUNTIME~SET_IMPLEMENTED_METHODS
- /CITXD/IF_RUNTIME~CHANGE_DATA_PRE_AUTH_CHECKS

Note: The /CITXD/IF_RUNTIME~CHECK_SELECTIONS method must be selected as `abap_true` in the /CITXD/IF_RUNTIME~SET_IMPLEMENTED_METHODS method. Else, a message "no selections for datasource" can occur on the on the SAP runtime logs.

31. Copy the code from the example class into these methods as stated in **step 29**. See [Appendix A](#) for more information on the sample code.
32. The data types used in the CHANGE_DATA_PRE_AUTH_CHECKS and other methods must be defined. These types will be the same as those used in the join tables/fields selected for selection and extraction in the DataSource ZCUSTOM_DS_EXAMPLE. Copy the types code from the example class, as shown below.



35. Click , the **Direct Type Entry** button and paste the code from the example type or type in directly to create new types.


Once the types have been created, they can be used in any of the methods in this class (note that they are private). The types used in this example are:

- **ty_bkpf**: For fields you need to extract from BKPF.
- **ty_bseg**: For fields you need to extract from BSEG.
- **ty_result**: For your final result output structure.
- **ty_bsec**: If you want to extend this to third table join (Join with BSEC as well).

These types can be related to the data fields defined in the DataSource in [Appendix B](#). The only field used from table BKPF is BUKRS – which is used to define the join between BKPF and BSEG tables. Type ty_bkpf, therefore, contains one entry only, that is, bukr. The associated type for bukr (bkpf-bukr) is obtained from the SAP System.

Similarly, the other types can be related back to the datasource.

36. If any other checks are required, they need to be done in the CHANGE_DATA_PRE_AUTH_CHECKS method before any authorizations are run.

37. Activate the  class.

The custom DataSource is now ready to be run for the cluster tables. This DataSource can be run as a normal DataSource.

Restricting the Number of Records

If the numbers of records need to be restricted (Example: for testing purposes), then the select statement shown in **Step 3** of *Appendix A* can be changed to include “up to max rows”.

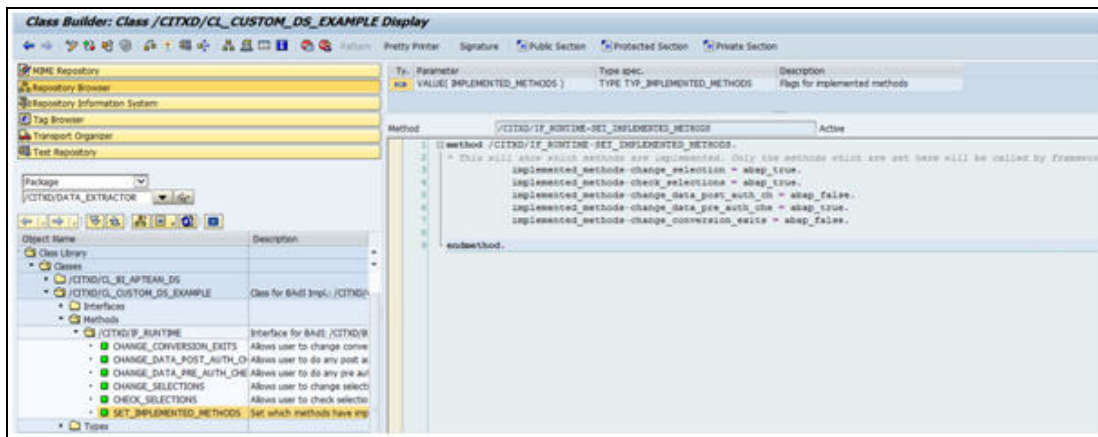
```
select belnr gjahr from bseg into corresponding fields of ls_bseg up
to max rows
```

The max value is passed through in the FM that is used to run this custom datasource.

Appendix A – BAdI implementation Class Methods

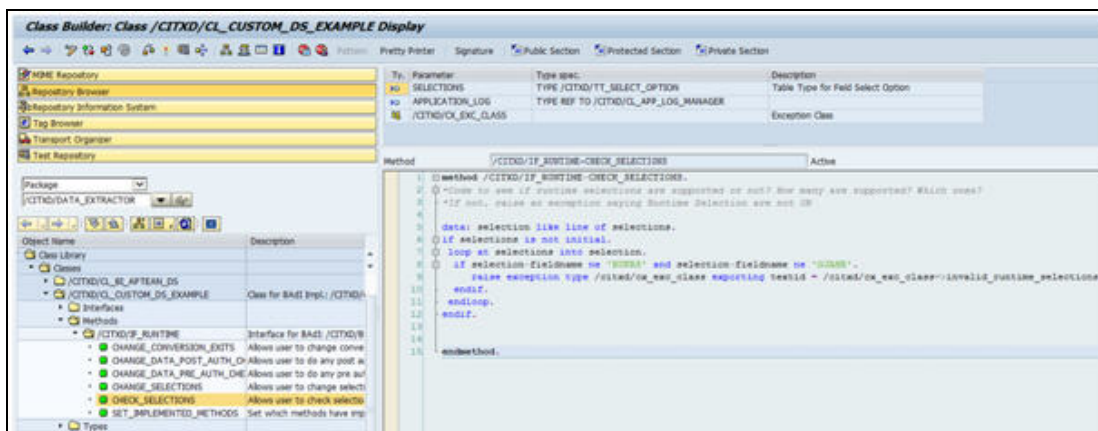
Implement method /CITXD/IF_RUNTIME~SET_IMPLEMENTED_METHODS of class:

This defines which methods of the BAdI will be implemented for this custom DataSource. The framework will only call those methods that are defined. To allow the defined methods to be called, set the value to `abap_true`, else set them to `abap_false`, as shown below:



Implement method `/CITXD/IF_RUNTIME~CHECK_SELECTIONS` of class:

This method allows the runtime selections to be checked. For example, a check can be made to see if the passed-in selections are supported at runtime for this DataSource. If false, a pre-defined exception can be thrown as shown below:



Implement method `/CITXD/IF_RUNTIME~CHANGE_SELECTIONS` of class (Only if needed):

Selections in addition to the runtime selections can be added into this method. For example, any mandatory selections if missing from runtime selection list or a selection which should be always be applied.

Ty.	Parameter	Type spec.	Description
Ⓜ	APPLICATION_LOG	TYPE REF TO /CITXD/CL_APP_LOG_MANAGER	
Ⓜ	VALUE(SELECTIONS)	TYPE /CITXD/TT_SELECT_OPTION	Table Type for Field Select Option
Ⓜ	/CITXD/CX_EXC_CLASS		Exception Class

Method	/CITXD/IF_RUNTIME-CHANGE_SELECTIONS	Active
1	method /CITXD/IF_RUNTIME-CHANGE_SELECTIONS.	
2	data: selection like line of selections.	
3		
4	* selection-fieldname = 'BUKRS'.	
5	* selection-sign = 'I'.	
6	* selection-option = 'EQ'.	
7	* selection-low = 1900.	
8		
9	*append selection to selections.	
10		
11	endmethod.	

Implement method /CITXD/IF_RUNTIME~CHANGE_DATA_PRE_AUTH_CHECKS of class:

This method is used to perform the main processing of the DataSource. The Output data object of this method which is passed on to the framework is a generic data object. To put the data specific to the result table in this generic object, use the field symbols.

Step 1: Assign output data object variable to a field symbol. This assignment will ensure that any changes that are done on the field symbol will be copied in the output variable DATA as well. Later in the code, the result table will be assigned to this field symbol.

```
field-symbols: <results> type standard table.
assign data->table->* to <results>.
```

If there is any problem in this field symbol assignment, a predefined exception can be raised.

```
if sy-subrc ne 0.
    raise exception type /citxd/cx_exc_class exporting textid =
        /citxd/cx_exc_class=>unassigned_fieldsymbols.
endif.
```

Step 2: Read selections supplied at runtime to prepare the ranges.

```
*Prepare range tables from Runtime Selections
if selections is not initial.
    loop at selections into selection.
        case: selection-fieldname.
            when 'BUKRS'.
                move-corresponding selection to rwa_bukrs.
                append rwa_bukrs to rt_bukrs.
                clear rwa_bukrs.
```

```

when 'GJAHR'.
    move-corresponding selection to rwa_gjahr.
    append rwa_gjahr to rt_gjahr.
    clear rwa_gjahr.
when others.
endcase.
endloop.
endif.

```

Step 3: Model the code to create the result table with Joins. This can be done in two ways.

Option 1: Using select-endselect

*Two Table Join Option 1: Select-EndSelect: Relatively faster than Option 2

```

select bukrs from bkpj into ls_bkpj up to 1000 rows where bukrs in rt_bukrs.

select belnr gjahr from bseg into corresponding fields of ls_bseg up to 1000
rows where bukrs = ls_bkpj-bukrs and gjahr in rt_gjahr.

ls_result-bukrs = ls_bkpj-bukrs.
ls_result-belnr = ls_bseg-belnr.
ls_result-gjahr = ls_bseg-gjahr.
append ls_result to lt_results.
clear: ls_result.
clear ls_bseg.
endselect.

clear ls_bkpj.
endselect.

```

Option 2: Using for all entries

*Two Table Join Option 2: Use For all entries

```

data: lt_bseg type table of ty_bseg,
      lt_bkpj type table of ty_bkpj.

select bukrs from bkpj into corresponding fields of table lt_bkpj up to 1000
rows where bukrs in rt_bukrs.

select bukrs belnr gjahr from bseg up to 1000 rows into corresponding fields
of table lt_results for all entries in lt_bkpj where bukrs = lt_bkpj-bukrs.

```

This code can be enhanced to accommodate 3 table joins as well. Example:

*Three Table join Example Option 1


```
select buhrs from bkp into ls_bkp up to 5000 rows where buhrs in rt_buhrs.

select belnr gjahr from bseg into ls_bseg up to 5000 rows where buhrs = ls_
bkpf-buhrs and gjahr in rt_gjahr.

select buzei from bsec into corresponding fields of ls_bsec up to 5000 rows
where buhrs = ls_bseg-buhrs.

and belnr = ls_bkp-belnr.

and gjahr = ls_bkp-gjahr.

ls_result-buhrs = ls_bkp-buhrs.
ls_result-belnr = ls_bseg-belnr.
ls_result-gjahr = ls_bseg-gjahr.
ls_result-buzei = ls_bsec-buzei.
append ls_result to lt_results.

clear: ls_result, ls_bsec.

endselect.

clear ls_bseg.

endselect.

clear ls_bkp.

endselect.
```

Finally return the result back, by extracting data from the cluster join tables into the `<results>` parameter in this method.

```
<results> = lt_results.
```

Implement method `/CITXD/IF_RUNTIME~CHANGE_DATA_POST_AUTH_CHECKS` of class:

If any processing needs to be done after field level authorization checks are applied, this can be done in this method. For Example: Add any additional authorizations etc.

Implement method `/CITXD/IF_RUNTIME~CHANGE_CONVERSION_EXITS` of class:

This is used to convert any data on fields before exiting the class.

Appendix B – Example Custom DataSource in EMF

The following images show a custom data source defined within EMF.

Data Source | Tables to join | Data fields | Fields for join

Name: ZCUSTOM_DS_EXAMPLE

SAP connection: APTEANTEST1

SAP System data source name: ZCUSTOM_DS_EXAMPLE Get Data Sources

Primary table name: BKPF

Data Source type: ☐ Standard ☒ Custom

Data Source | Tables to join | Data fields | Fields for join

Table to join: BSEG

Data Source | Tables to join | Data fields | Fields for join

Return field	Use in Query	Key	Table	Field	Description
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	BSEG	BUKRS	Company Code
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	BSEG	BELNR	Accounting Document Number
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	BSEG	GJAHR	Fiscal Year

Data Source | Tables to join | Data fields | Fields for join

From table	From field	To table	To field
BKPF	BUKRS	BSEG	BUKRS

JDBC Driver Configuration Details

JDBC drivers are required by various EMF components for accessing the Repository and Configuration databases, as well as for accessing other JDBC data sources. For connecting to the Repository and Configuration databases, as well as your own data sources, **you can use any JDBC drivers that are 100% JDBC 2.0 compliant.**

This topic provides configuration information for three supported JDBC drivers. For details on the configuration settings for other drivers, refer to the documentation accompanying the drivers.

Driver name	For use with the following supported database versions
Oracle JDBC driver (OCI8 and Thin)	Oracle 8i release 3, Oracle 9i release 2, Oracle 10g and Oracle 11g
jTDS JDBC driver for SQL Server	Microsoft SQL Server 7, 2000, 2005, 2008, 2012, and 2014.
Microsoft SQL Server 2005 JDBC Driver , version 1.0.809.102	SQL Server 2000 and 2005
Microsoft SQL Server 2008 JDBC Driver , version 3.0.1301.101	SQL Server 2000, 2005 and 2008

Important: The JDBC drivers "jTDS JDBC driver for SQL Server" and "Microsoft SQL Server 2005 JDBC Driver" are supplied with EMF to connect to MS SQL Server. The Oracle instant client is provided in the installation media that you can get the JDBC driver from. You can obtain the Oracle JDBC driver from your Oracle installation, or download it from the [Oracle web site](#) (you may need to register and log in to access this site). You can download the Microsoft SQL Server drivers from the [Microsoft web site](#). For any other JDBC drivers, refer to the relevant supplier's web site.

In cases of drivers that are not shipped with EMF, references to the JDBC drivers must be added to the EMF Administrator and Server environment. Refer to [How to Check EMF Configuration Details](#) for more information. The driver files can be placed anywhere, as long as they are correctly referenced from the Administrator and Server environments. References to the drivers listed in this topic are already included in the environments and expect the driver files to be located in the **auto_jar** and **Server/lib** directories of your EMF installation. The filenames included in the environments are: **classes12.zip**, **ojdbc14.jar**, **ojdbc5.jar**, **jtlds**, **msbase.jar**, **mssqlserver.jar**, and **msutil.jar**. If your driver has a different filename you will need to add it to these environments yourself.

There are several optional properties that you can use to further configure the performance of the JDBC connection. For more information, see [Pooling Options](#).

Oracle JDBC Driver

To connect to an Oracle database using the OCI8 driver, you must have the Oracle client/Instant client installed on the machine, as well as a correctly configured net service name. You must also have the shared library installed on your PC and, depending on your Oracle installation, you may or may not have this. If you do not have the shared library you can download it from the [Oracle web site](#).

Note: When you are creating the Oracle repository and configuration databases, you **must** have the Oracle client installed on your machine as the Microsoft OLE DB driver for Oracle needs a TNS name to connect. You can reconfigure the JDBC connection later so it uses the Thin driver.

To connect to an Oracle database using the Thin driver, you do not need the Oracle client installed or any additional libraries. You only need the driver file.

Remember to copy the driver file to the **auto_jar** and **Server/lib** directories of your EMF installation. The filename of the driver depends on the type of Oracle driver you are using.

Oracle OCI8 Driver Example 1:

This example shows the default settings that are configured when you create a new Oracle configuration and repository database using the Repository Wizard.

Driver name `oracle.jdbc.pool.OracleDataSource`
URL syntax `n/a`
Properties `user=%username%`
`password=%password%`
`TNSEntryName=%TNS Name%`
`driverType=oci8`
`xa_oracleFailoverCheck=true` (this is an optional property that can help the EMF Server recover in the event that the database fails but can have a negative impact on performance)

[Example of EMF Server Configuration connection](#)

[Example of EMF JDBC Data Source](#)

Oracle OCI8 Driver Example 2:

This example shows how to use the Oracle connection pooling instead of the EMF Server's internal connection pooling.

Driver name `oracle.jdbc.pool.OracleConnectionCacheImpl`
URL syntax `n/a`
Properties `user=%username%`
`password=%password%`
`TNSEntryName=%TNS Name%`
`driverType=oci8`
`xa_disableInternalPooling=true` (this disables the EMF Server's internal connection pooling)

Oracle Thin Driver Example:

This example shows the properties for connecting to an Oracle configuration and repository database using the Oracle Thin driver.

You can also disable the EMF Server's internal connection pooling and use the Oracle connection pooling by changing the driver name to [oracle.jdbc.pool.OracleConnectionCacheImpl](#) and adding the [xa_disableInternalPooling=true](#) property.

Driver name	oracle.jdbc.pool.OracleDataSource
URL syntax	n/a
Properties	user=%username% password=%password% driverType=thin databaseName=%OracleSID% serverName=%server machine% portNumber=%port number%, for example 1521 xa_oracleFailoverCheck=true (this is an optional property that can help the EMF Server recover in the event that the database fails but can have a negative impact on performance)

The following drivers have not been tested and are not supported by Apteian:

- [oracle.jdbc.pool.OracleConnectionPoolDataSource](#)
- [oracle.jdbc.xa.OracleXAConnection](#)

jTDS JDBC driver for SQL Server

You can use this driver to connect to **SQL Server 7, 2000, 2005, 2008, 2012, and 2014 databases**. This driver is shipped with EMF.

Driver name	net.sourceforge.jtds.jdbcx.JtdsDataSource
URL syntax	n/a
Properties	user=%username% password=%password% databaseName=%databaseName% serverName=%serverName%

`useLOBs=false`

`xa_verifyQuery=SELECT 1` (this property sets the query that will be run to test the connection is still valid)

`instance = %sqlservername`, or blank for the default
`instance%`

Connecting to SQL Server Using Windows Authentication

It is possible to connect to SQL Server using your Windows authentication. You can log on to SQL Server as the Windows user running the EMF administrator/server, without having to enter a user name or password in the JDBC connection properties.

To do this, add a property to the JDBC connection tab:

Name = domain

Value = <Your domain name>

Remove the username and password properties.

Note: For this feature to function correctly, the following DLL should be present:
C:\Program Files\Aptean\EMF\SupportDLL\ntlmauth.dll
This DLL is included in the recent versions.

[Example of EMF Server Configuration connection](#)

[Example of EMF JDBC Data Source](#)

Microsoft SQL Server 2005 JDBC Driver

You can use this driver to connect to SQL Server 2000 and 2005 databases.

To do this, you must copy the files named **msbase.jar**, **mssqlserver.jar** and **msutil.jar** from the driver's installation directory (by default this is **C:\Program Files\Microsoft SQL Server 2005 Driver for JDBC\lib**) to the **auto_jar** and **Server/lib** directories of your EMF installation.

Driver name	<code>com.microsoft.sqlserver.jdbc.SQLServerDataSource</code>
URL syntax	n/a
Properties	<code>user=%username%</code> <code>password=%password%</code> <code>databaseName=%databaseName%</code> <code>serverName=%serverName%</code> <code>selectMethod=cursor</code> (this mandatory property is necessary for the Synchronized EMF Processes Retriever)

to operate correctly)

`xa_verifyQuery=SELECT 1` (this property sets the query that will be run to test the connection is still valid)

Note: To connect to a SQL Server named instance, you should either enter a serverName of *machine_name\instance_name*, or add the `portNumber=%port number of named instance%` property. You can determine the port number in SQL Server Enterprise Manager.

[Example of EMF Server Configuration connection](#)

[Example of EMF JDBC Data Source](#)

Microsoft SQL Server 2008 Driver for JDBC

You can use this driver to connect to SQL Server 2000, 2005 and 2008 databases. This driver is shipped with EMF.

Driver name	<code>com.microsoft.jdbcx.sqlserver.SQLServerDataSource</code>
URL syntax	n/a
Properties	<code>user=%username%</code> <code>password=%password%</code> <code>databaseName=%databaseName%</code> <code>serverName=%serverName%</code> <code>selectMethod=cursor</code> (this mandatory property is necessary for the Synchronized EMF Processes Retriever to operate correctly) <code>xa_verifyQuery=SELECT 1</code> (this property sets the query that will be run to test the connection is still valid)

Note: To connect to a SQL Server named instance, you should either enter a serverName of *machine_name\instance_name*, or add the `portNumber=%port number of named instance%` property. You can determine the port number in SQL Server Enterprise Manager.

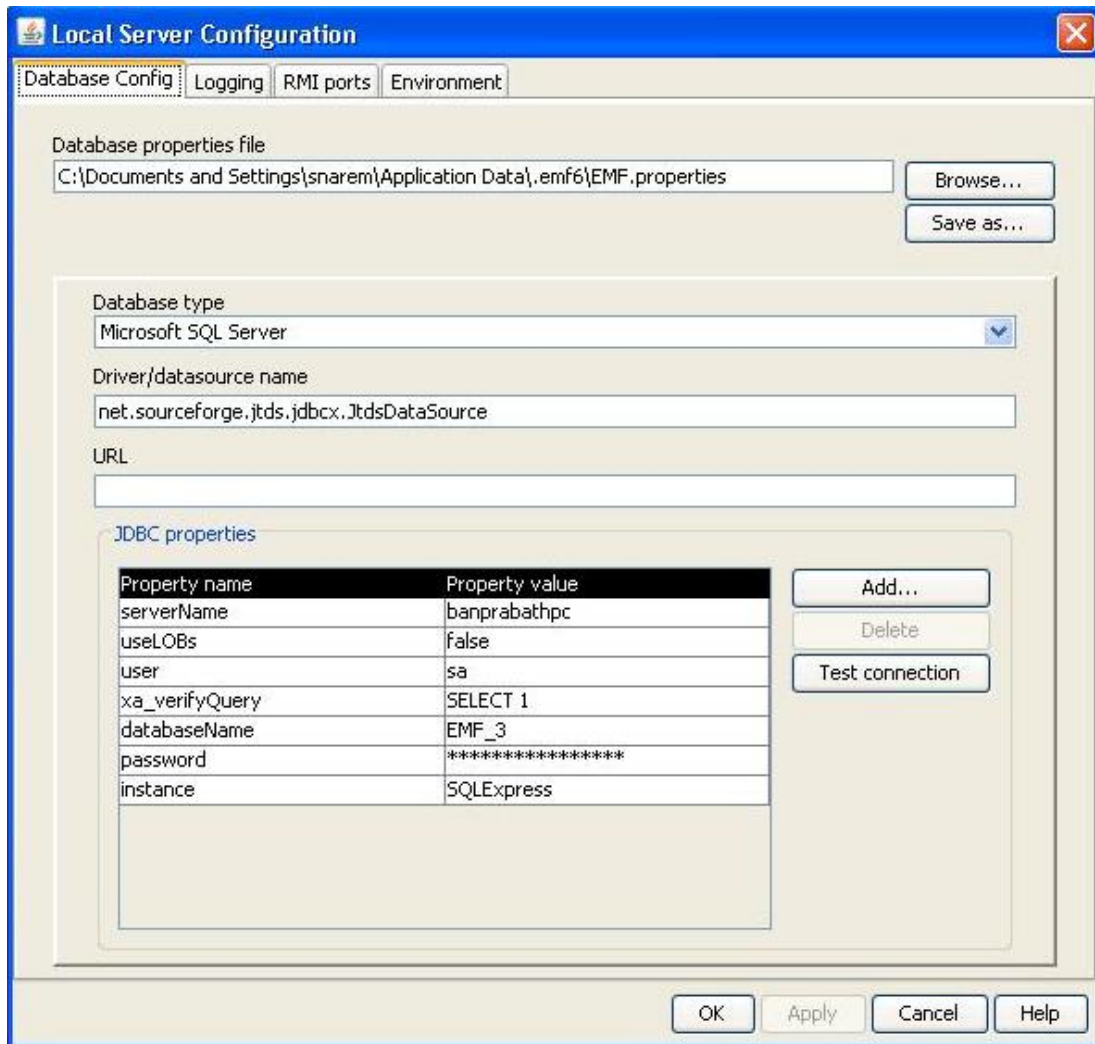
[Example of EMF Server Configuration connection](#)

[Example of EMF JDBC Data Source](#)

Note: You can continue to use these settings if you wish. However, any new databases you create using the [EMF Repository Wizard](#) will automatically be configured with the new settings as described above.

[How to Check EMF Configuration Details](#)

EMF Server Configuration Connection Example



[JDBC Driver Configuration Details](#)

Retrieving BAPI Metadata

The BAPI Metadata page of the [SAP connection](#) dialog shows information stored by the SAP Data Source. It displays the BAPIs (Business Application Programming Interface) available for a particular Business Object within the SAP R/3 system. The pane displays a tree view consisting of the SAP R/3 System, Application Areas and Nodes representing the Business Objects and the BAPIs (GETLIST, etc.)

Note: EMF no longer supports an offline mode.

To view the metadata for a connection:

1. Configure a valid connection.
2. Press the **Test** button to retrieve the metadata.

[Creating SAP Connections](#)

[Using the SAP Module](#)

Database Trigger Wizard

A database trigger is procedural code that is automatically executed in response to certain events on a particular table or view in a database.

You can use the Database Trigger Generator Wizard to create a SQL Server/Oracle trigger that fires a process when an insert, update or delete transaction occurs in a particular SQL Server/Oracle data source.

SQL Server/Oracle triggers both use the EMF API to communicate with the EMF Server.

[SQL Trigger Wizard](#)

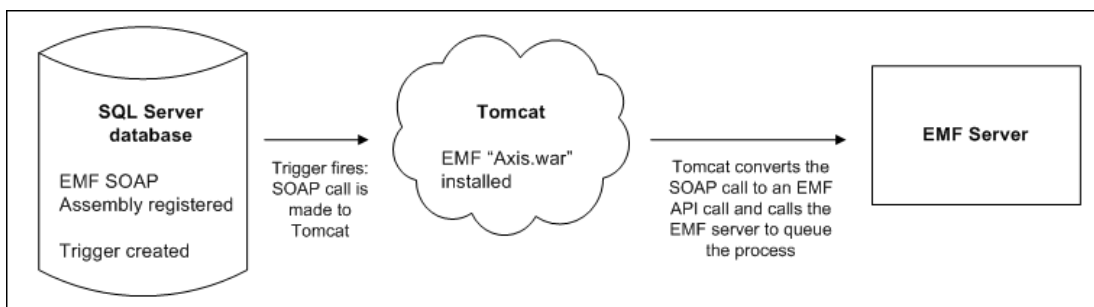
[Oracle Trigger Wizard](#)

SQL Server Trigger Wizard

Overview - How a SQL Server database trigger causes a process to run in EMF

When a database insert/update/delete causes an EMF created database trigger to fire, the following steps happen:

1. The database trigger code calls the .Net EMF SOAP assembly that has been registered in the SQL Server database.
2. This assembly code makes a SOAP call which is picked up by a webserver (typically Tomcat) that has the EMF "Axis.war" installed.
3. The Tomcat server then turns this SOAP request into a standard EMF API call that will call the EMF Server, which then causes an EMF process to be queued to run with the relevant data.

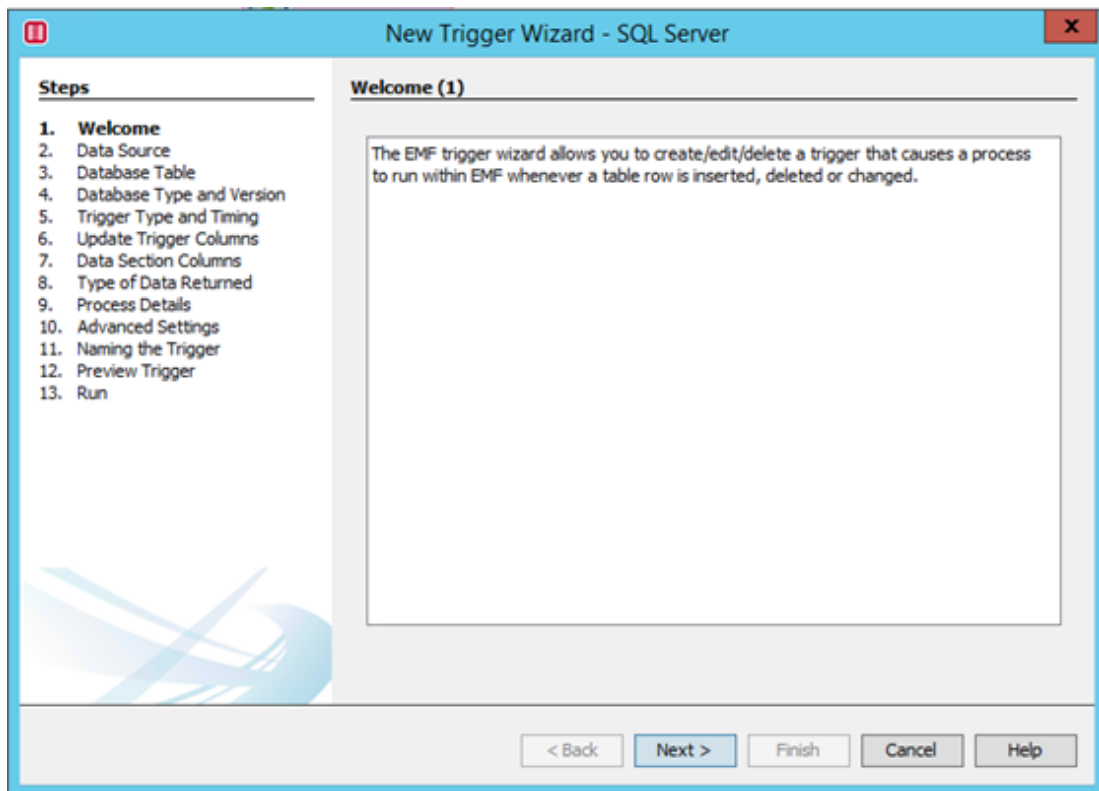


To create a trigger:

Follow the wizard's step-by-step instructions to create an SQL Server trigger.

Note: You must install the SOAP components before you can successfully run a trigger. Please see the [Configuring the EMF SOAP API](#) section for more details. You must register the EMF SOAP assembly in the SQL Server database that you are creating triggers in for the trigger to work. This can be done during installation of the SOAP components or later. Failure to correctly configure the SOAP components will cause the triggers to fail to run when executed.

1. In the tree view, click on the plus sign (+) next to the **Triggers** icon. Right-click **SQL Server** and select **New**.



2. Click **Next**.
3. Select the EMF JDBC data source from the drop-down list, and click **Next**. You can click the **Test Connection** button to check that your connection to the datasource is working.
4. Select the database table on which the trigger will run and click **Next**.

Note: Select the **Include views** check box to choose from a list of tables and views. All views have names with a suffix VIEW.

5. Select the type and version of the database and click **Next**.
6. Select the timing of the EMF process from the **Run EMF Process** section.
 - Select **BEFORE** to fire the trigger during the SQL INSERT, UPDATE or DELETE operations.

- Select **AFTER** to fire the trigger when all operations specified in the trigger have executed successfully on the table.
- Select **INSTEAD OF** to specify that the trigger is to be executed instead of the triggering SQL statement.

You must select one of the operations before you can continue in the wizard.

Note: The default for the trigger timing is **BEFORE**.

7. Select the operation on the table from the **On Operation** section. Click **Next**.
 - Select **INSERT** to fire the trigger when the specified table has rows added to it.
 - Select **UPDATE** to fire the trigger when rows are changed within the specified table.
 - Select **DELETE** to fire the trigger when rows are deleted from the specified table.

Note: Due to SQL Server restrictions, a **DELETE** trigger cannot be created at the same time as an **UPDATE** trigger if the **UPDATE** trigger only acts on some of the table's columns. If the **UPDATE** trigger only acts on some of the table's columns, then to create the **DELETE** trigger, you will need to run the wizard again for the **DELETE** trigger only. If the **UPDATE** trigger acts on all of the table's columns, then continue to the next screen where all columns are selected.

Note: The columns returned in the data section may not be in the order shown in the selected columns list. You should, therefore, access the columns in the data section in the API process, by their name, rather than index.

8. Select the columns that will activate the trigger. Click **Next**.
9. Enter the data section name. The data section name entered here will be used in the process that is run by the trigger. Select the columns to be returned in the data section.

Note: Columns which hold data of LONG, LOB, and BINARY types cannot be returned, and are not available for selection.

10. Click **Next**.
11. Choose the type of data to be returned after an update, and click **Next**.
12. Choose the EMF process that will be run when the trigger is activated. Select the Username and enter the Password of an operator who has necessary rights to initiate the process. Click **Next**.

Note: The EMF process must have an API initiator module for it to be listed and selectable as a valid process to be run.

13. Enter the machine name and RMI port number that the EMF Server is running on. The defaults are `localhost` and `50001` which will give a URL of:
`rmi://localhost:50001/com/xalert/server/api/EMFFactory`, located at port 50001 on the localhost. Please see the [Configuring the EMF SOAP API](#) section for more details. Localhost is only appropriate if the "Axis.war" installed in Tomcat is running on

the same machine as the EMF Server. If Tomcat and the EMF server are on different machines then make sure that the required ports are not blocked by firewalls.

14. Enter the machine name and port number where a suitable Web Service Description of the SOAP-enabled EMF API can be found (this is the machine and port that Tomcat is running on with the "Axis.war" installed). The defaults are `localhost` and `8080`, but they would only be appropriate if Tomcat and SQL Server were running on the same machine. With these settings, the URL `http://localhost:8080/xalertsssoap/xalertssapi/xalertapi.wsdl` obtains the description from a webserver running on port 8080 on localhost. The SQL server trigger implementation communicates with the EMF server using the SOAP-enabled API. Please see the [Configuring the EMF SOAP API](#) section for more details. If Tomcat and the SQL Server database server are on different machines then make sure that the required ports are not blocked by firewalls.
15. Click **Next**.
16. Enter a trigger name and click **Next**.
 - You can select **Overwrite any trigger of the same name** option to replace existing triggers with same name in the database where the trigger will be created/replaced. Note that this option does not overwrite any trigger definition saved in the EMF Repository, and the trigger name must be unique in the EMF Repository.
 - Select **Replicate trigger for every server** option if you want the trigger to be replicated for every server.
17. Click **Next** to create a trigger.

Note: Click **Edit** to modify the Java class/Stored procedure/Trigger script. To save these scripts to a file, click **Save to file**. Clear the **Auto Preview** check box if you do not want to preview the trigger script next time you edit the trigger in the wizard.

18. Click **Yes**, when prompted to run the trigger script.

Note: If you click **No**, the trigger is not created in the database. However, you can still save the trigger information in the EMF repository.

19. Click **Finish** to save the trigger information.

Note: If you click **Cancel**, the trigger information is not saved to the EMF repository. In this case, you have to manually delete the trigger, if needed. Clicking **Finish** will save the trigger information to the EMF repository, thereby allowing you to easily modify and delete the trigger from the database.

To edit a trigger:

1. In the tree view, click on the plus sign (+) next to the **Triggers** icon. Click on the plus sign (+) next to the **SQL Server** icon. Right-click a trigger and select **Edit**.
2. Follow [step 2](#) to [step 19](#) of the new trigger wizard procedure to edit the trigger.

To delete a trigger:

1. In the tree view, click on the plus sign (+) next to the **Triggers** icon. Click on the plus sign (+) next to the **SQL Server** icon. Right-click a trigger and select **Delete**.

2. Click **Next**. The **Preview Trigger** window displays the script used to delete the trigger/Java class/stored procedure.
3. Click **Next** to delete the trigger.

Note: Click **Edit** to modify the Java class/Stored procedure/Trigger script. To save these scripts, click **Save to file**. Clear the **Auto Preview** check box if you do not want to preview the trigger script next time you edit the trigger.

4. Click **Yes**, when prompted to run the scripts.
5. Click **Finish** to delete the trigger definition from the EMF repository.

[Troubleshooting the SQL Server Trigger Wizard](#)

Oracle Trigger Wizard

The EMF API is used to trigger EMF processes to run from sources external to EMF where a custom-made adapter does not exist. Data can be passed to the EMF process that will be instantiated by the trigger. The API can be accessed via Java, C#, and SOAP calls.

As Oracle has built-in support for Java, the EMF Java API is used for communication between Oracle and EMF. This section describes the procedure to create a database trigger in Oracle that will trigger an EMF process to run.

There are several tasks that need to be accomplished to successfully create the trigger.

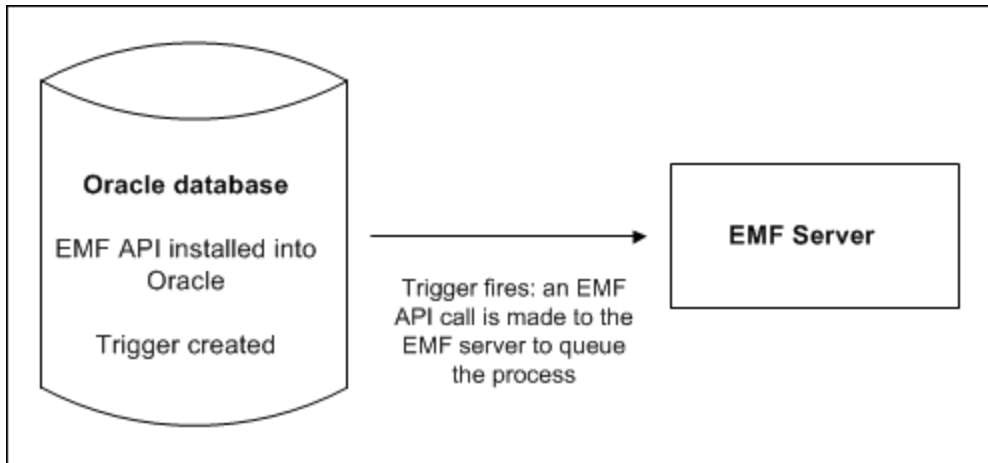
Some tasks have to be performed once per database/schema, while the other steps are for each trigger created. They are:

- Create a process that will be fired by the trigger.
- Load the EMF API into Oracle. (Once per schema)
- Load the Java routine (that uses the EMF API to initiate an EMF Process) into Oracle.
- Create an Oracle stored procedure that will wrap the Java code.
- Create the Oracle database trigger to call the Stored procedure.

The Oracle trigger wizard allows you to perform the above-mentioned tasks without the need for knowing the procedure to write the Java code or the Stored procedure.

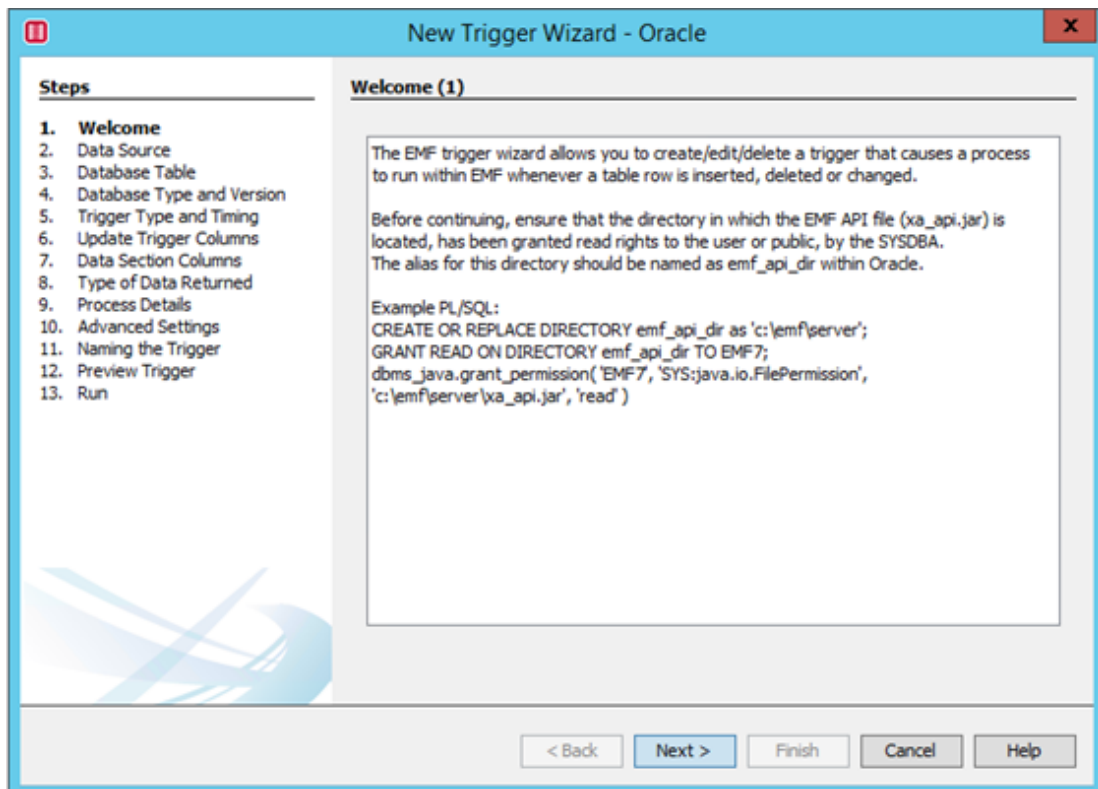
Overview - How an Oracle database trigger causes a process to run in EMF

The database trigger code calls the EMF Java API (which has been preloaded into the Oracle database). The EMF API calls the EMF Server, which then causes a process to be queued with the relevant data.



To create a trigger:

1. In the tree view, click on the plus sign (+) next to the **Triggers** icon. Right-click **Oracle** and select **New**.



Note: The EMF API resource file ('xa_api.jar') needs to be loaded into Oracle by the trigger wizard once per schema. To tell the wizard to load the EMF API then the option **Load EMF API class into Oracle** in step 15 must be selected. If the EMF API is being loaded then the Oracle user (for the JDBC connection being used) has to have Read rights to the directory where the jar is located. The alias for this directory must be named emf_api_dir within Oracle.

Example:

```
conn <SYS>/<SYS password> as sysdba
```

```
CREATE or REPLACE DIRECTORY emf_api_dir AS 'c:\emf';
```

Running this command will not create a folder at the specified location. You have to create a folder manually and copy the 'xa_api.jar' file to the folder. It is important that the directory does not contain any spaces in its name. This directory must be accessible by Oracle database server and will typically be on the same machine as the Oracle database server.

The SYSDBA needs to give Read rights for that directory to the Oracle user before creating any Oracle trigger (It only needs to be done once per Oracle user, or the DBA might decide to grant Read rights to all users - in which case, it needs to be done once per Oracle instance).

Example for giving Read rights to a user called "EMF7":

```
GRANT READ ON DIRECTORY emf_api_dir TO EMF7;
```

To load the EMF API file, the Oracle user needs Read permission.

Example PL/SQL:

```
call dbms_java.grant_permission( '<user>', 'SYS:java.io.FilePermission',  
'c:\emf\xa_api.jar', 'read' )
```

-
2. Click **Next**.
 3. Select the Oracle JDBC data source from the drop-down list, and click **Next**. You can click the **Test Connection** button to check that your connection to the datasource is working.
 4. Select the database table on which the trigger will run and click **Next**.

Note: Select the **Include views** check box to choose from a list of tables and views. All views have names with a suffix VIEW.

5. Select the type and version of the database and click **Next**.
6. Select the timing of the EMF process from the **Run EMF Process** section.
 - Select **BEFORE** to fire the trigger during the SQL INSERT, UPDATE or DELETE operations.
 - Select **AFTER** to fire the trigger when all operations specified in the trigger have executed successfully on the table.
 - Select **INSTEAD OF** to specify that the trigger is to be executed instead of the triggering SQL statement.

You must select one of the operations before you can continue in the wizard.

Note: The default for the trigger timing is **BEFORE**. You can choose Instead of only if you have selected a view in step 4 of the wizard.

7. Select the operation on the table from the **On Operation** section. Click **Next**.
 - Select **INSERT** to fire the trigger when the specified table has rows added to it.
 - Select **UPDATE** to fire the trigger when rows are changed within the specified table.
 - Select **DELETE** to fire the trigger when rows are deleted from the specified table.

Note: The columns returned in the data section may not be in the order shown in the selected columns list. You should, therefore, access the columns in the data section in the API process, by their name, rather than index.

8. Select the columns that will activate the trigger. Click **Next**.

Note: This screen is not displayed when you select **Instead of** timing in step 6 of the wizard.

9. Enter the data section name. The data section name entered here will be used in the process that is run by the trigger. Select the columns to be returned in the data section.

Note: Columns which hold data of LONG, LOB, and BINARY types cannot be returned, and are not available for selection.

10. Click **Next**.
11. Choose the type of data to be returned after an update, and click **Next**.
12. Choose the EMF process that will be run when the trigger is activated. Select the Username and enter the Password of an operator who has necessary rights to initiate the process. Click **Next**.

Note: The EMF process must have an API initiator module for it to be listed and selectable as a valid process to be run.

13. Enter the machine name and RMI port number that the EMF Server is running on. The default values are '`<name of the machine on which the wizard is running>`' and '50001' which will lead to the URL `rmi://< machine name>:50001/com/xalert/server/api/XAlertsFactory`, located at port 50001 on the machine on which the wizard is running. If the wizard cannot determine the machine name, it will default to localhost. Remember to change localhost to the machine name, otherwise Oracle will not be able to locate the RMI port. Localhost is only appropriate if the Oracle server is running on the same machine as the EMF Server. If Oracle and the EMF server are on different machines then make sure that the required ports aren't blocked by firewalls.
14. Click **Next**.

Note: The Oracle user should have permission to communicate with the EMF RMI port, in order to run the EMF process in the trigger (this can be set after the trigger is created, but before it is run).

Example:


```
call dbms_java.grant_permission( 'EMF6', 'SYS:java.net.SocketPermission',  
'192.168.0.5', 'connect,resolve' )
```

where EMF6 is the Oracle schema, 192.168.0.5 is the IP address/name of the machine running the EMF Server.

15. Enter a trigger name, Java class name, and an Oracle function name. Default names are already created. Click **Next**.
 - You can select **Overwrite any trigger of the same name** option to replace existing triggers with same name in the database where the trigger will be created/replaced. Note that this option does not overwrite any trigger definition saved in the EMF Repository, and the trigger name must be unique in the EMF Repository.
 - Select **Load EMF API class into Oracle** option to load the EMF API into the Oracle database when the trigger is created. This needs to be done only once per schema. See step 1 for a list of assumed permissions that are needed if this option has been selected.
16. Click **Next** to create a trigger.

Note: Click **Edit** to modify the Java class/Stored procedure/Trigger script. To save these scripts to a file, click **Save to file**. Clear the **Auto Preview** check box if you do not want to preview the trigger script next time you edit the trigger in the wizard.

17. Click **Yes**, when prompted to run the trigger script.

Note: If you click **No**, the trigger is not created in the Oracle database. However, you can still save the trigger information in the EMF repository.

18. Click **Finish** to save the trigger information.

Note: If you click **Cancel**, the trigger information is not saved to the EMF repository. In this case, you have to manually delete the trigger, if needed. Clicking **Finish** will save the trigger information to the EMF repository, thereby allowing you to easily modify and delete the trigger from the Oracle database.

To edit a trigger:

1. In the tree view, click on the plus sign (+) next to the **Triggers** icon. Click on the plus sign (+) next to the **Oracle** icon. Right-click a trigger and select **Edit**.
2. Follow [step 2](#) to [step 18](#) of the new trigger wizard procedure to edit the trigger.

To delete a trigger:

1. In the tree view, click on the plus sign (+) next to the **Triggers** icon. Click on the plus sign (+) next to the **Oracle** icon. Right-click a trigger and select **Delete**.
2. Click **Next**. The **Preview Trigger** window displays the script used to delete the trigger/Java class/stored procedure.
3. Click **Next** to delete the trigger.

Note: Click **Edit** to modify the Java class/Stored procedure/Trigger script. To save these scripts to a file, click **Save to file**. Clear the **Auto Preview** check box if you do not want to preview the trigger script next time you edit the trigger.

4. Click **Yes**, when prompted to run the scripts.
5. Click **Finish** to delete the trigger definition from the EMF repository.

[Troubleshooting the Oracle and SQL Server Trigger Wizard](#)

White Lists

A **white list** is a list of items that have been marked as being expected or allowable.

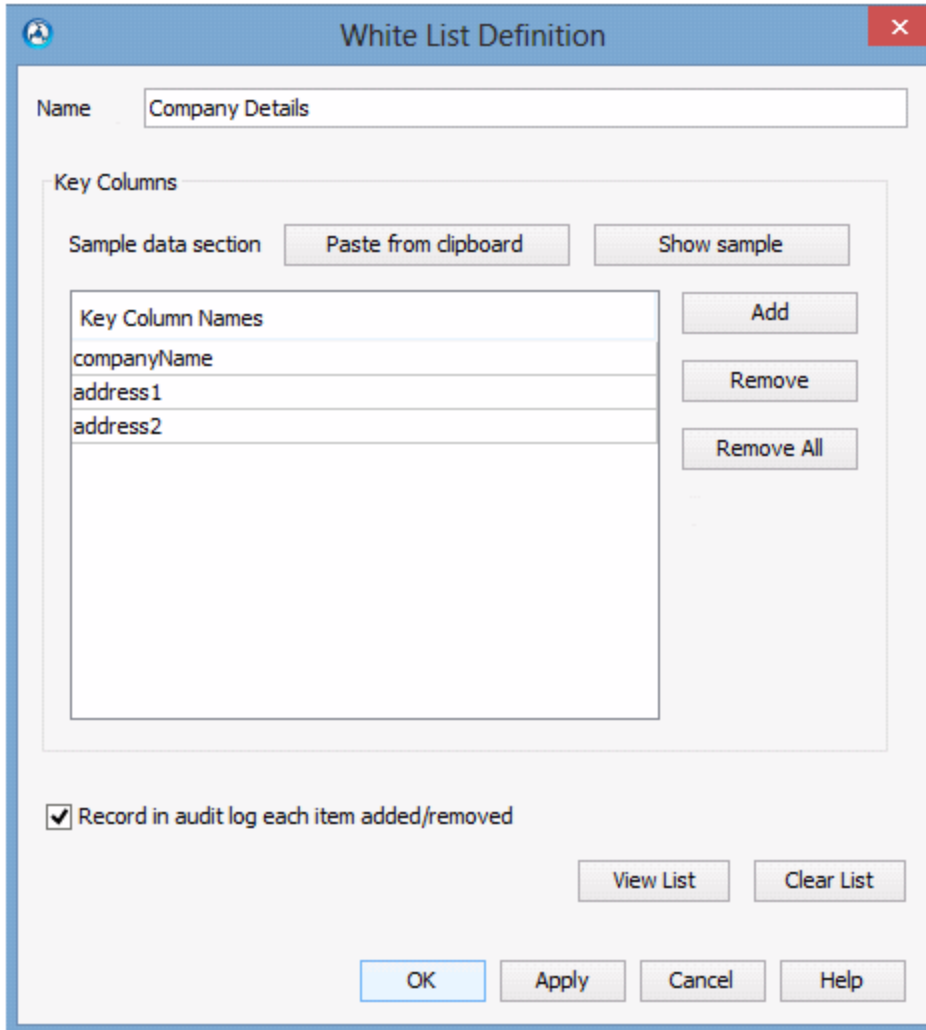
The white listing feature has been designed primarily for use with Apteian Respond to white list cases within Respond. However, it can be used with any system (or within just EMF) that can present the data in the required format.

[White List Definition](#)

White List Definition

[White lists overview](#)

A **white list definition** defines the list and the fields that are used to identify white listed items.



The dialog box is titled "White List Definition". It has a "Name" field containing "Company Details". Below this is a "Key Columns" section. Inside this section, there is a "Sample data section" with buttons for "Paste from clipboard" and "Show sample". Below the sample section is a list box labeled "Key Column Names" containing "companyName", "address1", and "address2". To the right of the list box are buttons for "Add", "Remove", and "Remove All". At the bottom of the "Key Columns" section is a checkbox labeled "Record in audit log each item added/removed" which is checked. Below the checkbox are buttons for "View List" and "Clear List". At the very bottom of the dialog are buttons for "OK", "Apply", "Cancel", and "Help".

- **Name:** The name entered here will be referenced throughout the EMF System to identify the instance of the White List.
- The **Key Column Names** list defines the field names that when combined together uniquely identify a white list event. These names typically come from the SQL database that is queried to get the events of interest, and is then passed through the white list filter module. At least one column name should be added. Use the **Add** button to add a new entry. Names can be typed in, but to make it easier, it is possible to select the column names from a drop-down list.

To get a list of available column names, first save a sample data section to the clipboard containing the columns of interest (for example, you could use an SQL module or a parser module to do this). Then, in the White List Definition dialog, click **Paste From Clipboard**. Now, when a new row is added, a drop-down combo box should be available to select from a list of available column names. To view the sample data pasted from the clipboard, press the **Show Sample** button.

Note: Adding, removing or changing a **Key Column Name** invalidates the current white list data and current events logged in the white list will need to be removed

before the dialog changes can be saved (you will be prompted to do this when trying to save changes).

- If the **Record in audit log each item added/removed** check box is selected, then every time a [White List event module](#) runs and adds (or removes) items from the white list, details of the item are logged in the audit log.
- **View List** button displays the contents of the white list, i.e., the items that are currently white listed. If a white list item has expired, then its White List until date will be displayed in red. Expired white list items are ignored when the white list filter module is used to filter a data-section. Expired events are cleared out when the next white list event module is run against this definition.
- **Clear List** button removes the contents of the white list. i.e., all items currently white listed are removed. This is useful to reset the list during testing.

[White List Overview](#)

[White List Event Module](#)

[White List Filter Module](#)

Event Definitions

The Event Definition defines the different types of Events that an [Event Plan](#) will expect, and the data fields that will be associated with the event. Event Definitions can be created within their own folder structure allowing better organisation. Before you can use [Event Plans](#), you must configure an **Event Definition** from the **Event Definitions** section of the EMF Tree View.

To specify an Event Definition

1. Double-click the **Event Definitions** section in the EMF tree view.
 - To modify an existing Event Definition, double-click the icon.
 - To create a new Event Definition, right-click the selected folder and click **New Event Definition**.

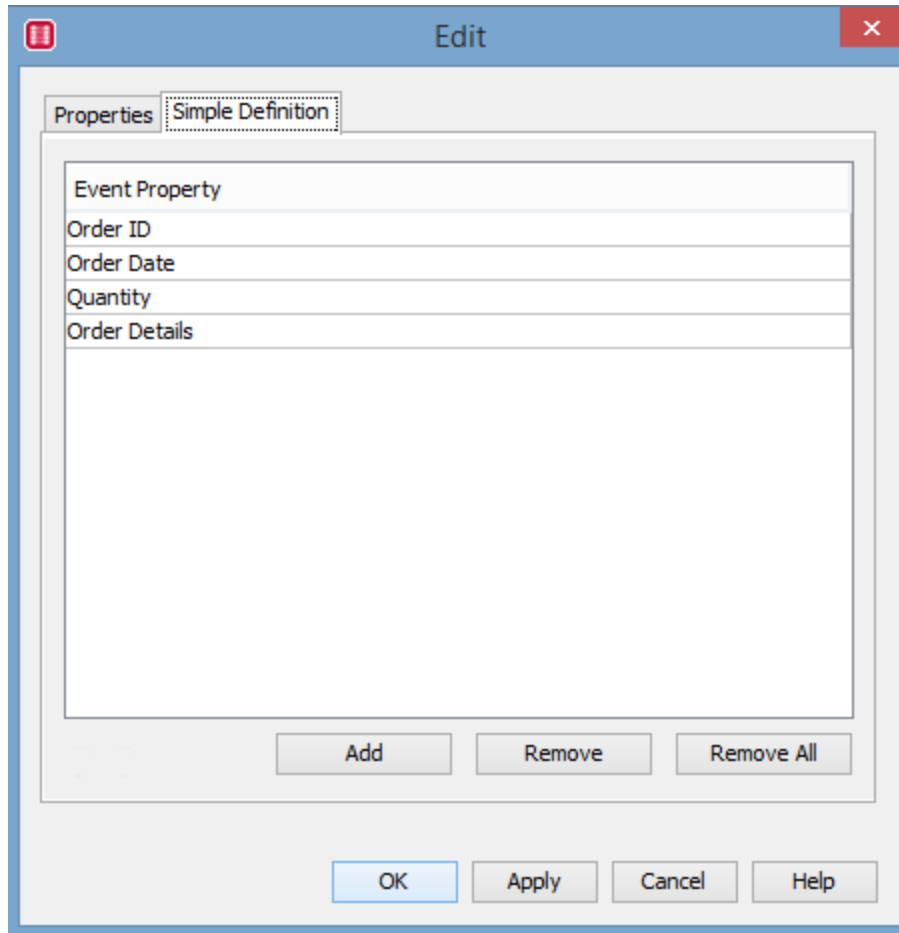
The **New Event Definition** window is displayed:

The screenshot shows a 'New Event Definition' dialog box. It has a title bar with a red close button. The dialog contains two tabs: 'Properties' and 'Simple Definition'. The 'Simple Definition' tab is active. Inside this tab, there is a 'Name' text field containing 'New Order', a 'Description' text area, and a 'Definition Type' section with two radio buttons: 'Simple' (selected) and 'Free Form'. At the bottom of the dialog are four buttons: 'OK', 'Apply', 'Cancel', and 'Help'.

2. Type a **Name** for the Event Definition.

Note: You can have Event Definitions with the same name as long as they are in different folders.

3. Type a **Description** for documentation purposes.
4. Select the **Definition Type**.
 - **Simple** - If selected, then the **Simple Definition** tab is enabled. This allows a list of properties to be entered that are the names of the expected properties of this event. Property names must be unique for this definition.



- **Free Form** - If selected, there is no need to setup any information or associated text to the **Event Definition**. The data will be formed when setting up the [Publish Event](#). This definition type allows far more complex correlation rules to be built rather than the simple matching of properties that the **Simple** format uses. The [Expect Event](#) module can then correlate on a particular item in the data using, for example, an XPATH expression.

Note:

- **Simple** event definitions perform faster and are simpler to configure. They use simple matching of properties.
 - **Free Form** event definitions allow more complex correlation rules to be built, such as order quantity < 10. They however perform slower at runtime than **Simple** event definitions.
-

[Event Plans](#)

[Event Plan Modules](#)

Managing Authentication Protocols

You can use the **Net Security Key Stores** node in the EMF tree view to install and manage information about Key stores and other authentication protocols.

- [Key stores](#) are used to authenticate sources and guarantee that information coming from them is genuine. You can also install a Key store to verify your EMF system to remote users.
- Support for other certificates and other authentication protocols will be added in future versions of EMF.

[Configuring the EMF System](#)

[Installing EMF SSL Certificate](#)

[How to import a self-signed certificate into EMF](#)

Managing Key Stores

You can use the **Key Store** screen to install and manage details of key stores, which are used to authenticate remote sites when communicating over the Internet. A key store includes specific information about the source and an expiry date, and any information received can be checked against this to ensure that it is genuine.

Note: If you wish to do client authentication, you must also have an appropriate key store in order that third parties can verify your EMF system.

To access the Key Store screen:

1. Double-click the **Key store** icon under the **Net Security Key Stores** icon in the EMF tree view, under the **System** node.
2. Enter an appropriate name for the Key store in the **Name** field.
3. Select the appropriate Key store file type from the **File type** drop-down list.
4. Select the appropriate Key store type from the **Certificate Type** drop-down list.
5. Add the key store file using either of the following two options:
 - Select **Location of key store file** and enter the path to the certificate in the **File** field, or click the [...] button to browse for it.
 - Select **Use imported key store file**. Click **Import** to import the key store file. Using this, you can import the key directly into the repository database. This has the advantage that when an EMF key store is exported, the actual key store contents are exported at the same time.

Note: When the correct password has been entered for the imported key store file, the certificate contents are displayed directly in the text area.

6. Enter the password for the certificate in the **File password** field.
7. Enter the password for the private key in the **Private key password** field.

[Net Security](#)

[Configuring the EMF System](#)

Installing EMF SSL Certificate

The following topics provide information on the setup and configuration of the SSL Certificate on the EMF Listening port. They provide instructions on how to enable a self-signed certificate and associate it with the EMF HTTP Listener.

- [Creating a Self-Signed Certificate](#)
- [Installing an SSL Certificate on EMF](#)

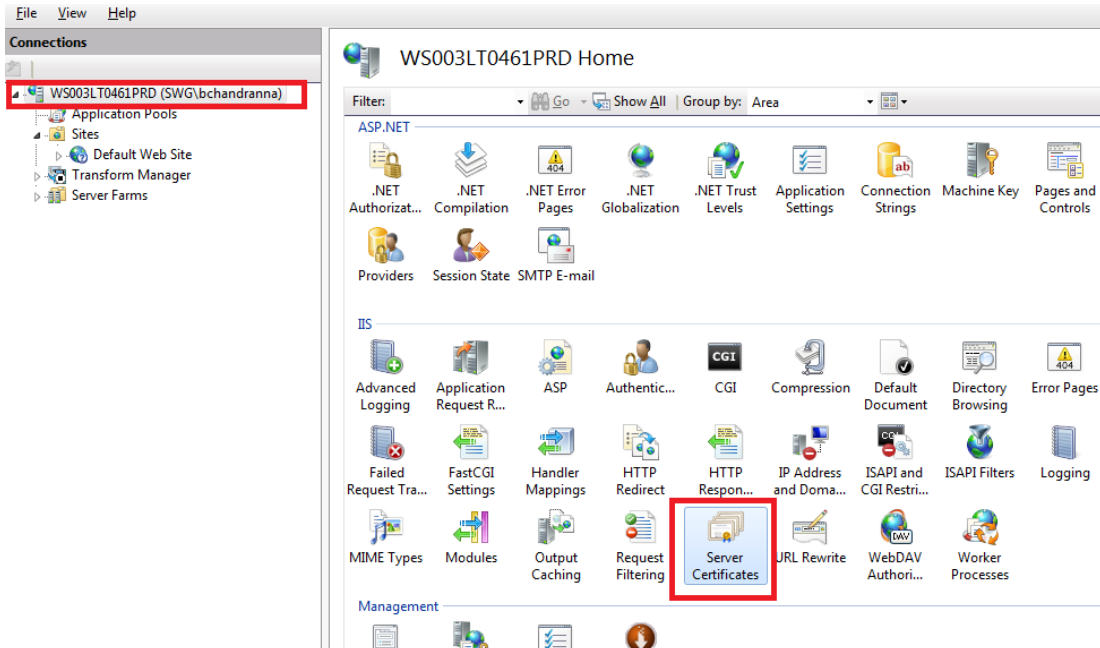
Note: The processes used in these topics can also be implemented for certificates obtained from CA authorities.

Creating a Self-Signed Certificate

Perform the following steps to create a certificate:

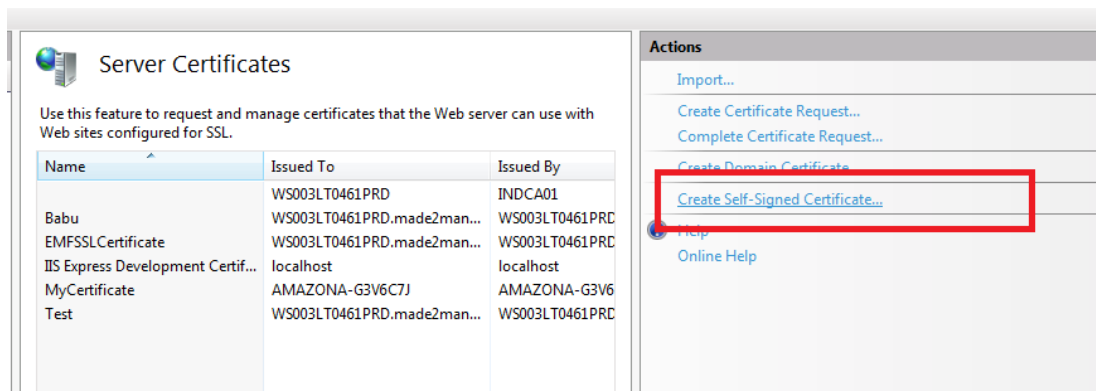
Note: To create a self-signed certificate on a Windows system, you will need Internet Information Services (IIS).

1. Click **Start > Control Panel > Administrative Tools**, and then double-click **Internet Information Services (IIS) Manager**.
2. Double-click **Server Certificates**.



A list of certificates installed on the system is displayed in the **Internet Information Services (IIS) Manager** page.

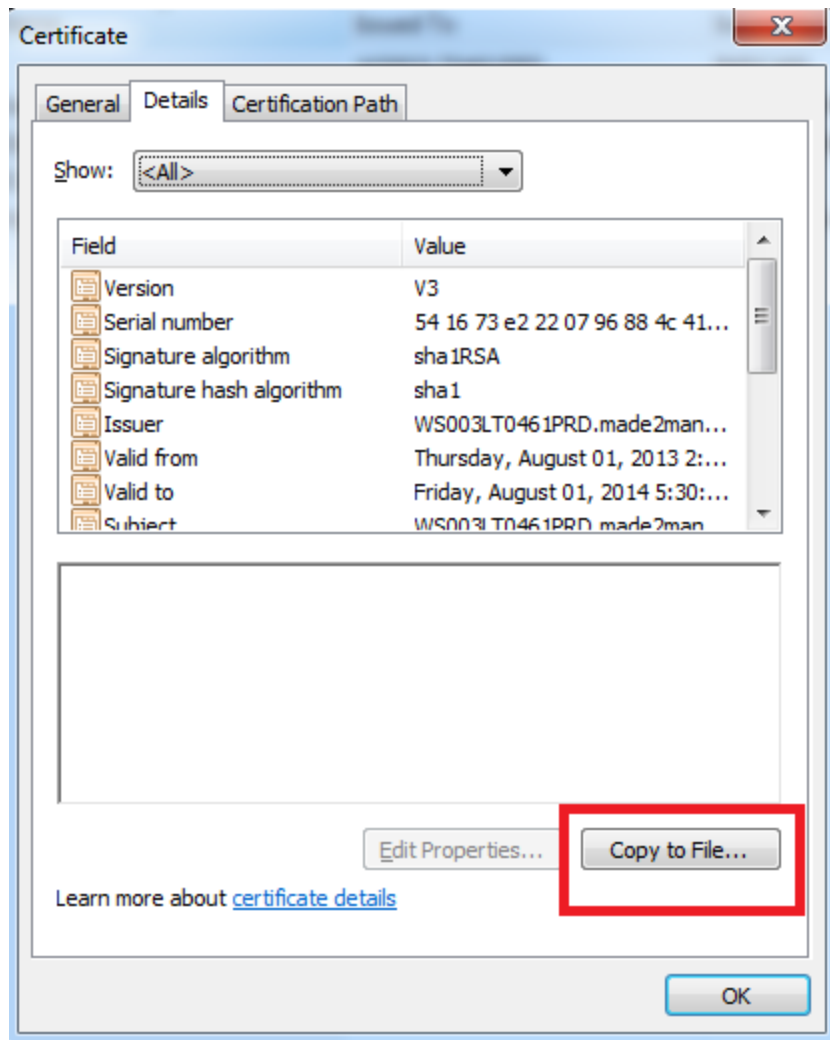
3. In the **Actions** pane, click **Create Self-Signed Certificate**.



4. On the **Create Self-Signed Certificate** page, type a friendly name for the certificate in the **Specify a friendly name for the certificate** box, and then click **OK**.

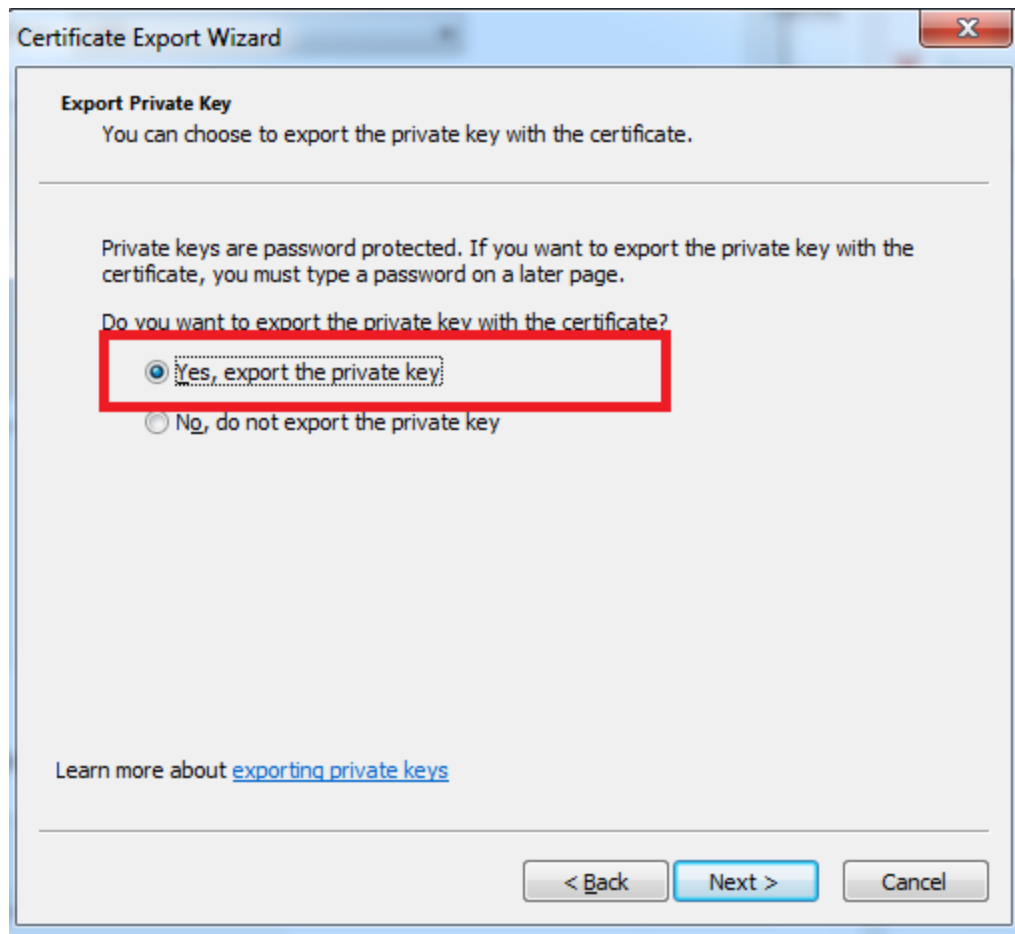
The new certificate is added to the list of certificates installed on the system.

5. Export the certificate to be used with EMF as follows:
 - a. Double-click on the newly created certificate. The **Certificate** dialog box opens.
 - b. Select the **Details** tab and then select **Copy To File**.

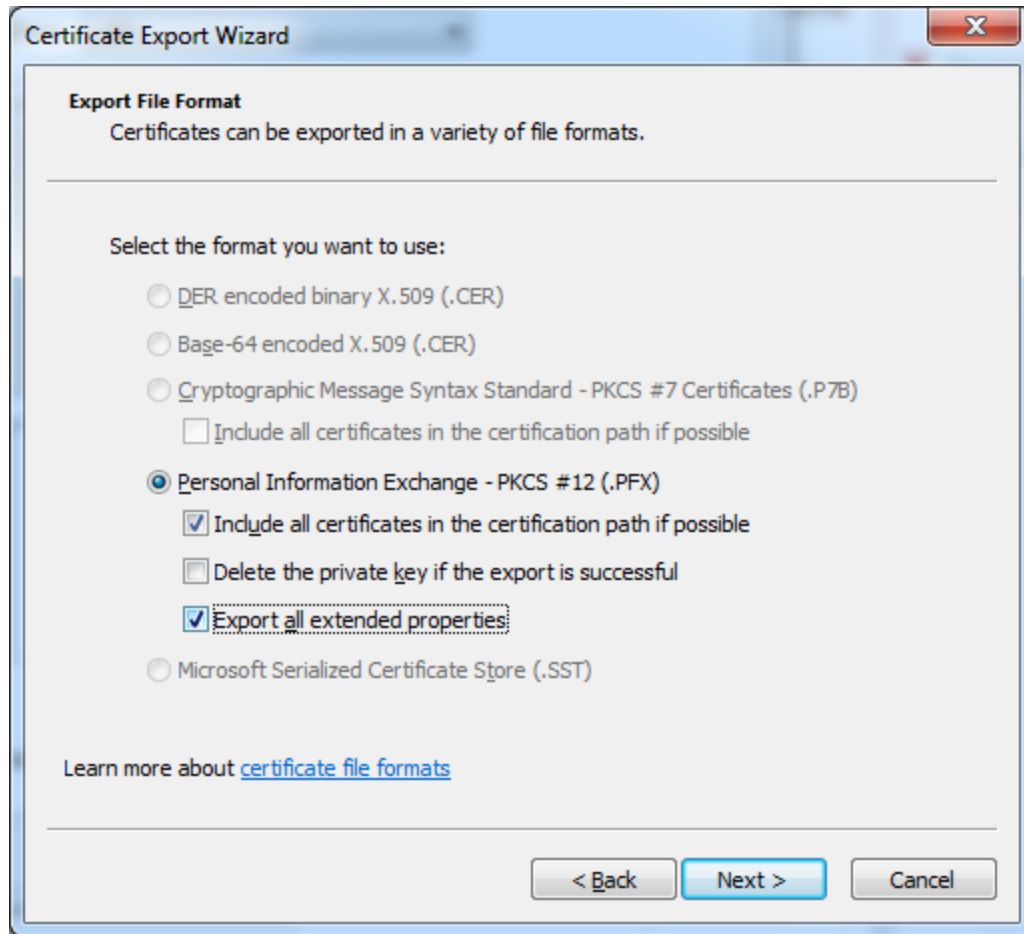


This starts the **Certificate Export Wizard**. Click **Next**.

- c. Select **Yes, export the private key** and click **Next**.



- d. Choose the options as shown in the following screen shot and click **Next**.



- e. Type and then confirm a password for the certificate file in the fields provided.
- f. Specify the name of the file you want to export. Click **Browse** if you want to use the **Save As** dialog box to set the file location and name.
- g. Click **Next** and then click **Finish**. Click **OK** after the Certificate Export Wizard confirms that the certificate was successfully exported.

[Installing an SSL Certificate on EMF](#)

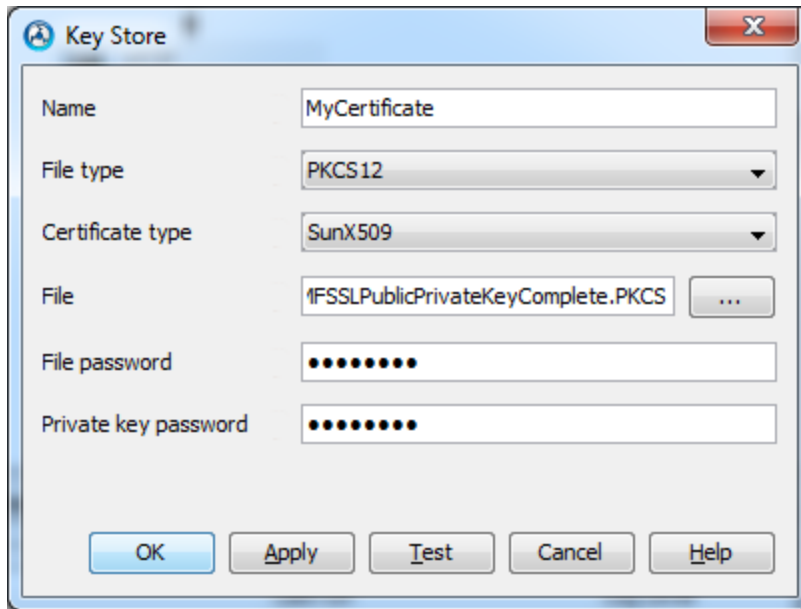
[Installing EMF SSL Certificate](#)

Installing an SSL Certificate on EMF

Perform the following steps to install SSL on the EMF HTTP Listener:

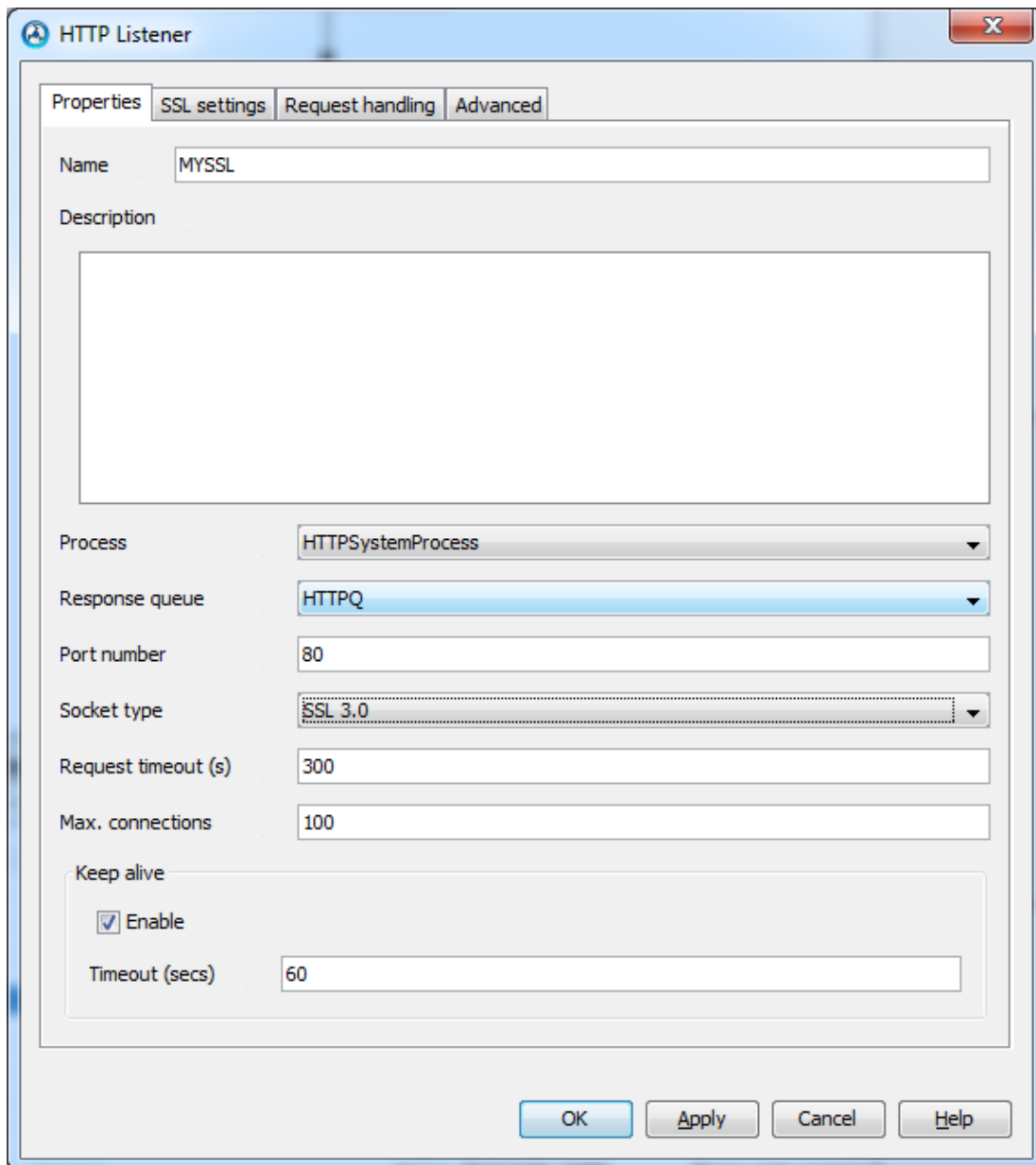
1. Log in to the EMF system.
2. In the EMF tree view, go to **System > Net Security Key Stores**.

3. Right-click and select **New**. The **Key Store** page opens.

A screenshot of the 'Key Store' dialog box. It has a title bar with a blue icon and the text 'Key Store'. The dialog contains several fields: 'Name' with the text 'MyCertificate', 'File type' with a dropdown menu showing 'PKCS12', 'Certificate type' with a dropdown menu showing 'SunX509', 'File' with the text '1FSSLPublicPrivateKeyComplete.PKCS' and a browse button '...', 'File password' with a masked password field, and 'Private key password' with a masked password field. At the bottom, there are five buttons: 'OK', 'Apply', 'Test', 'Cancel', and 'Help'.

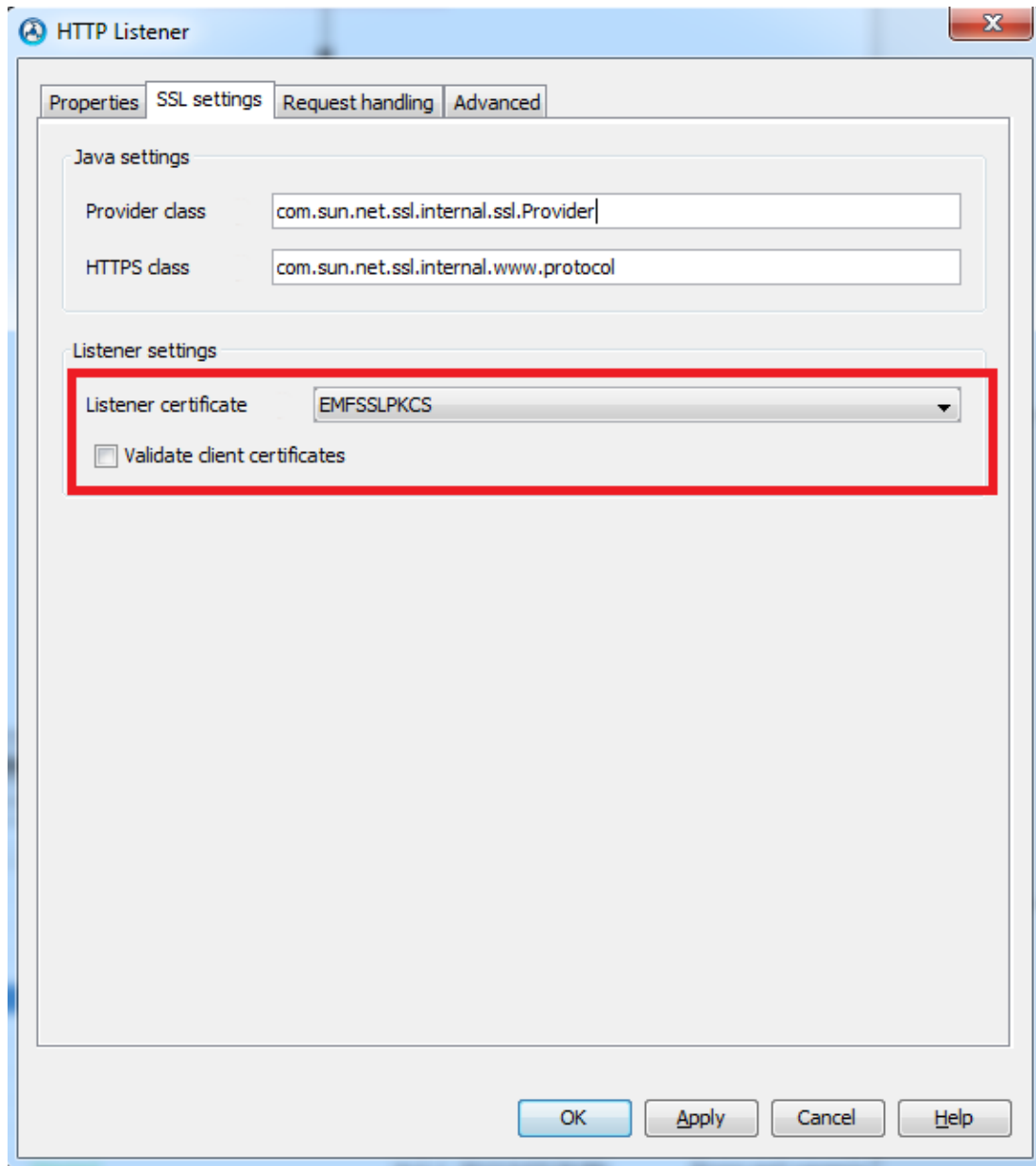
4. Provide the following details:
 - a. In the **Name** field, type an appropriate name.
 - b. From the **File type** drop-down list, select **PKCS12**.
 - c. From the **Certificate type** drop-down list, select **SunX509**
 - d. In the **File** field, browse and select the certificate file created from IIS or any other provider.
 - e. In the **File password** field, type the password that is used when exporting the certificate.
 - f. In the **Private key password** field, enter the password for the private key.
 - g. Click **Ok**
5. In the EMF tree view, go to **System > Machines > "machine name"**.

6. Right-click and select **New > HTTP Listener**.



The screenshot shows the 'HTTP Listener' configuration dialog box. It has a title bar with a close button. Below the title bar are four tabs: 'Properties', 'SSL settings', 'Request handling', and 'Advanced'. The 'Properties' tab is selected. Inside the 'Properties' tab, there is a 'Name' field with the text 'MYSSL' and a 'Description' text area. Below these are several configuration fields: 'Process' (a dropdown menu showing 'HTTPSystemProcess'), 'Response queue' (a dropdown menu showing 'HTTPQ'), 'Port number' (a text field with '80'), 'Socket type' (a dropdown menu showing 'SSL 3.0'), 'Request timeout (s)' (a text field with '300'), and 'Max. connections' (a text field with '100'). At the bottom of the 'Properties' tab is a 'Keep alive' section with a checked 'Enable' checkbox and a 'Timeout (secs)' text field with '60'. At the bottom of the dialog box are four buttons: 'OK', 'Apply', 'Cancel', and 'Help'.

7. Type the appropriate **Name** and **Port number** and select **Socket type** as **SSL 3.0**.
8. Go to the **SSL settings** tab and select the newly created key store value from the **Listener certificate** drop-down list.



9. Click **OK**. Any processes which use the http listener will respond to a secure request.

[Creating a Self-Signed Certificate](#)

[Installing EMF SSL Certificate](#)

Using Multiple Languages in EMF

EMF has a single default language (usually English) that is used when sending out EMF Processes, and another default language (usually the same as the default output language) that is used in the User Interface of the administrator (example, for dialogs, captions, labelling and so on).

However you can, if you wish, send EMF Processes that are in different languages, or in several languages at the same time.

To send EMF Processes in multiple languages:

1. Define the languages that you want to be able to use.
2. Select one of the languages as the default (this will be the language that is used if no other is specifically chosen).

See [Defining Languages for EMF Process Output](#) and [Setting a Default Language for EMF Process Output](#).

[Configuring the EMF System](#)

Defining Languages for EMF Process Output

You can use the **Language** screen to define the language variants that are available when creating multi-lingual output formatting modules. Normally English, French, German, Italian and Spanish are available (with English set as the default language).

To view the languages that are available:

- Select the **Languages** icon in the **System** folder of the EMF tree view.

The languages that are currently available are displayed in the right-hand pane.

To add a new language:

1. Right-click in the right-hand pane of the Languages view and select **New language** to display the **Language** screen.
2. Select the required language from the **Name** drop-down list. The **ISO locale**, **Default date format** and **Default time format** fields automatically change to reflect the new language.
3. If you wish, you can modify the **Default time** and **Default date** formats according to the formats that are available in your version of Windows.

Note: You cannot modify the **ISO Locale** field.

[Setting a Default Language for EMF Output](#)

[Configuring the EMF System](#)

About EMF Data Store - Sharing Data between Processes

Overview

EMF allows easy data sharing between processes and state information to be persisted between different runs of processes via the data store. This feature has numerous uses such as:

- Avoiding the user from having to create and manage the database tables to use for storing information.
- Improving performance by holding frequently used information in memory. For example, configuration information that would otherwise need to be re-read every time a process is run can be held as shared data and initialised only once.

How it works

The EMF repository has an internal "data store". Either EMF "data sections" or "message sections" can be added to the data store using the [Data store module](#). Any place in EMF that can then normally reference a data/message section can directly reference a section in the data store instead.

Example: Process A queries some data from a database and creates a data section called "Config". It then uses the data store module to place that data section in the data store. Process B can then directly access the data store section "Config" and use it the same way it would any other data section.

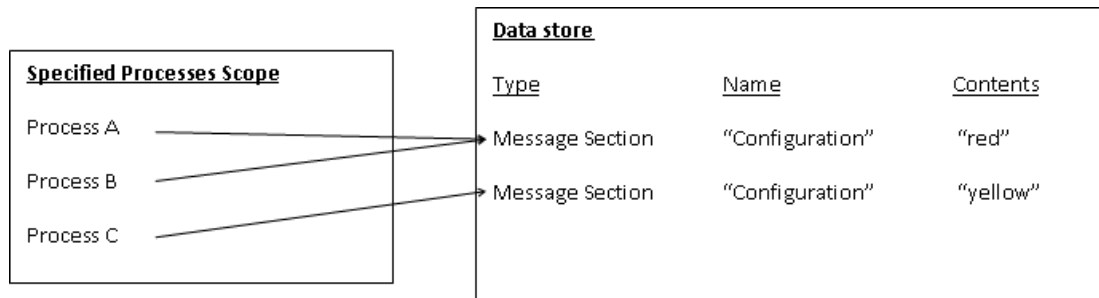
The internal "data store" is persisted, so it will be available in the event of a server restart/crash. It is also shared between multiple machines when running in a multi-machine environment.

Data Store Scope

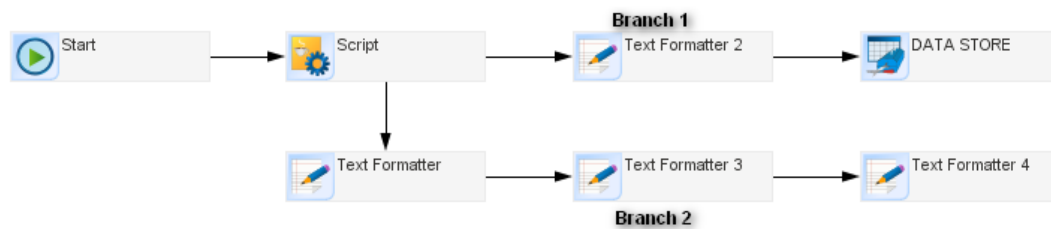
When an item is added to the data store, it is possible to define the processes to which the data should be visible (the scope). The following three different possible scope settings are available:

- **Global** – The data added to the data store will be available to all processes. If a data/message section already exists in the data store with the same name and scope then it will be overwritten.
- **Specified Processes (all instances)** - The data added to the data store will be available to all instances of the processes specified. If a data/message section already exists in the data store with the same name and for the same process set, then it will be overwritten.

Example scenario: A message section called "Configuration" with a value of "red" is added to the data store that is in scope for processes A and B. A different message section, but also called "Configuration", but with a value of "yellow", is also added to the data store but in scope for process C. When process C runs and accesses the "Configuration" message section, it will get a value of "yellow". Processes A and B will get a value of "red" when they access the message section.



- **Current Process Instance (all branches)** - The data added to the data store will be available to all branches of the current process instance. Each instance of a process will have their own set of data, and it is not shared between process instances. So, for example, in the process shown, once an item has been put in the data store, it will be available to both Branch 1 and Branch 2. Each time the process is run a new instance is created, and will have its own set of data in the data store.

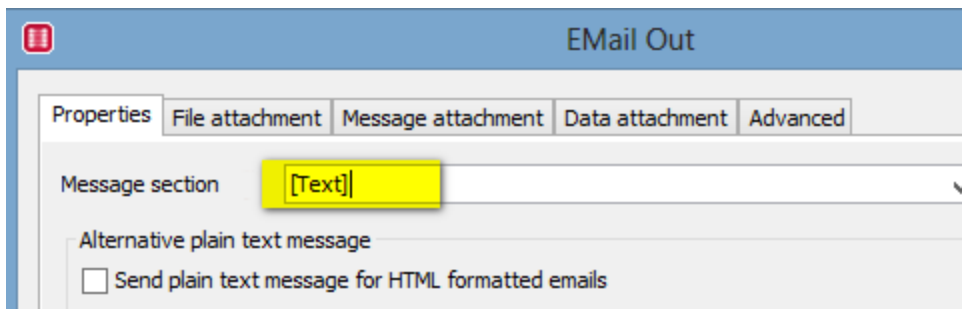
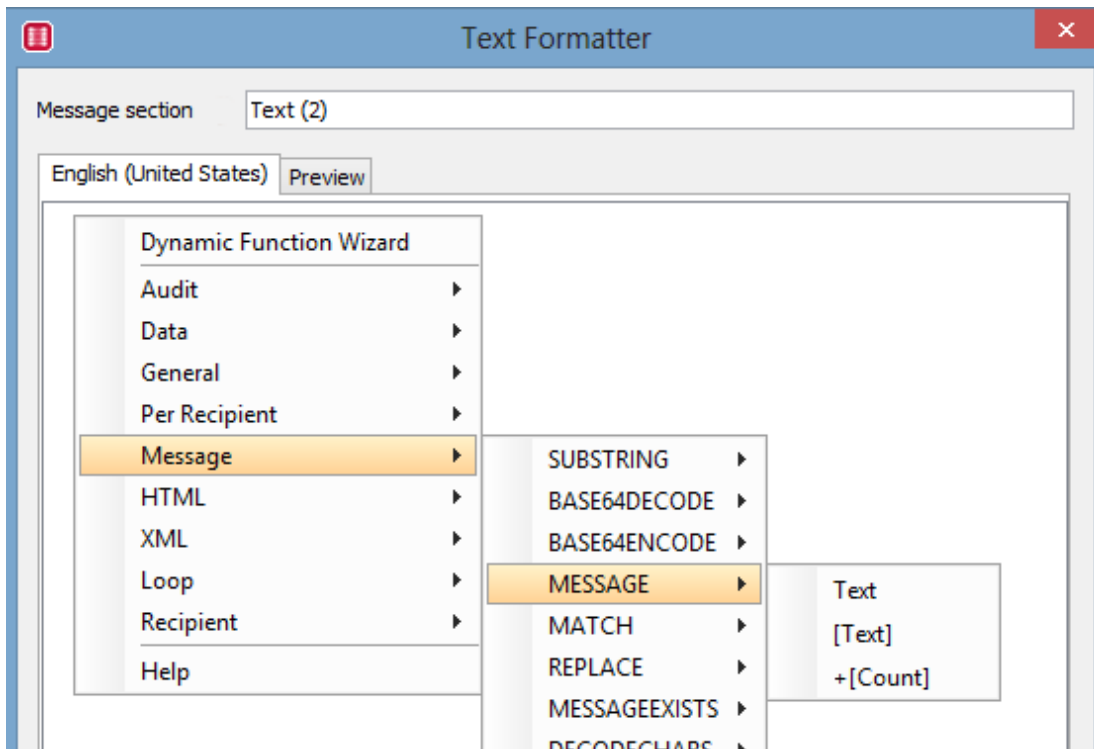


To access an item in the data store, just select the specified section as you would any other message/data section.

Accessing the data store

Items in a data store will be available as standard data/message sections in processes that are in scope. They appear in the dynamic function menus and the drop-down list in modules. To indicate that a data/message section is coming from a data store its name appears in square brackets, for example:

[Data Dir]



Information about the *scope* of the message/data section is displayed as part of the name. The following format is used:

- **Global scope** – name is prepended with a "+". For example:
+ [my section]
- **Process scope** – name doesn't have any additional identifiers, just the square brackets. For example:
[my section]
- **Process instance scope** – name is prepended with a "-". For example:
- [my section]

The preceding patterns are reserved formats when naming sections. When the data store section names appear in the module drop-down lists or dynamic function lists, they are grouped together after the usual local data sections. They are ordered by "scope" first, and then alphabetically. The scope order is as follows: local sections, data store process

instance scope sections, data store process scope sections, and data store global scope sections. For example:

SQL 1

SQL 2

-[my process instance data store item 1]

-[my process instance data store item 2]

[my process data store item 1]

[my process data store item 2]

+ [my process data store item 1]

+ [my process data store item 2]

Note: Always have a process with a data store module referencing the data store item, otherwise the data store item will be cleaned up and removed. For example, with global scope, if you add a section in a data store called "X" and reference it from another process, and then delete the original "data store" module, the data associated with "X" will be deleted and no longer be available.

See also:

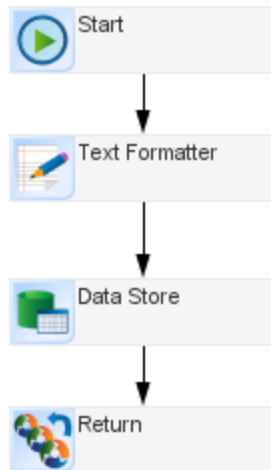
[Worked Example 1: Using the data store to hold process configuration to improve performance](#)

[Worked Example 2: Creating a log of processed files](#)

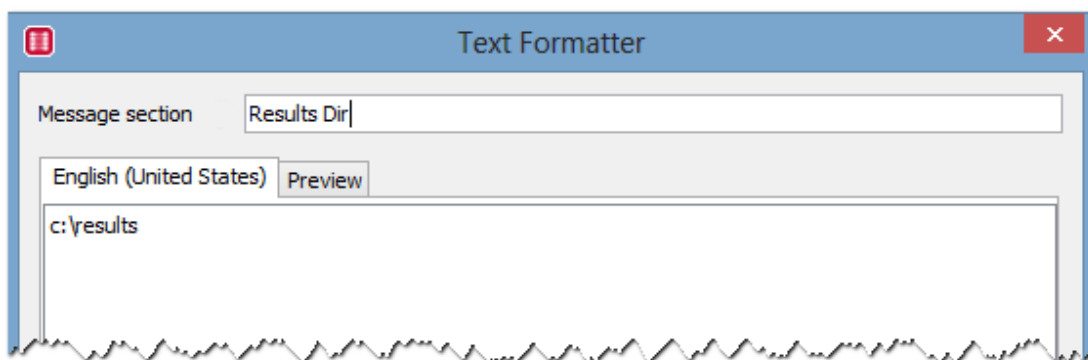
Worked Example 1: Using the data store to hold process configuration to improve performance

The following example shows how to use the data store to store configuration information for a process, and avoid having to set up configuration information with each run – thus improving performance.

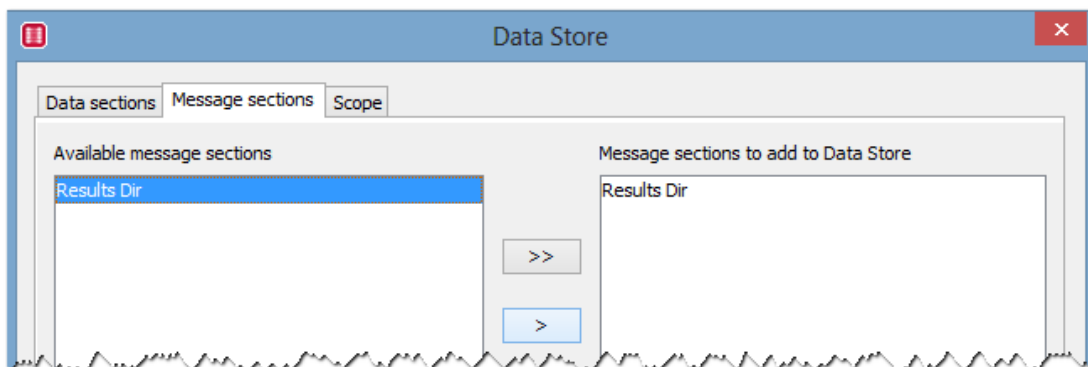
1. Create the following two processes:
 - **Main**
 - **Process Configuration**
2. Open **Process Configuration**, and add the following modules:
 - **Text formatter**
 - **Data Store**
 - **Return**



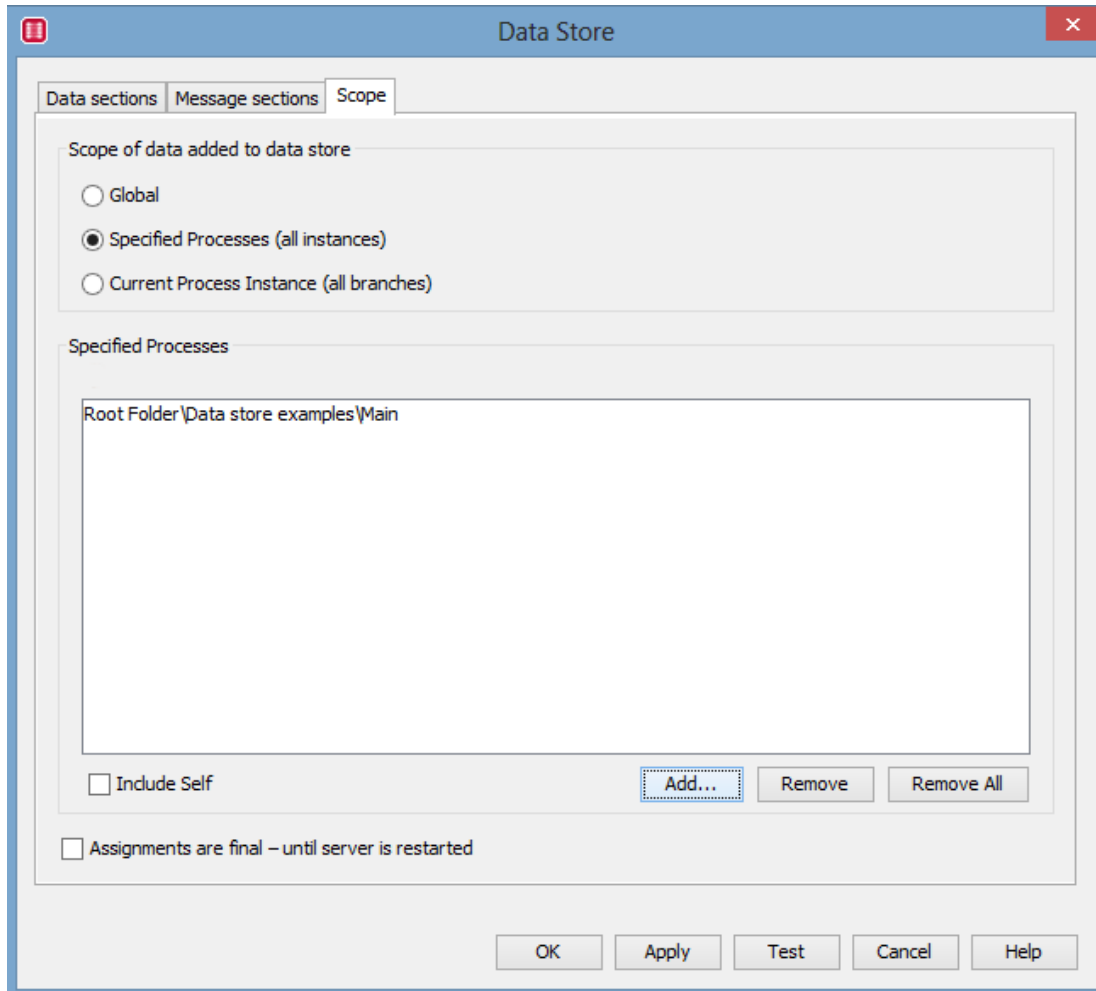
3. In the **Text Formatter** module, set the configuration information required. For example, the name of an output directory, such as `c:\results`. This information could actually be retrieved from a database using a SQL module, or read in from a Flat file using the file in module. In this example, the message section is called "**Results Dir**" and set the value to "`c:\results`".



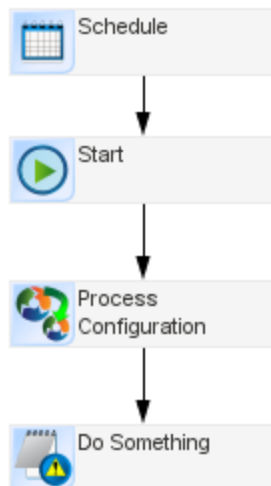
4. Open the data store module and on the **Message sections** tab copy the **Available message sections** to the **Message sections to add to Data Store** list.



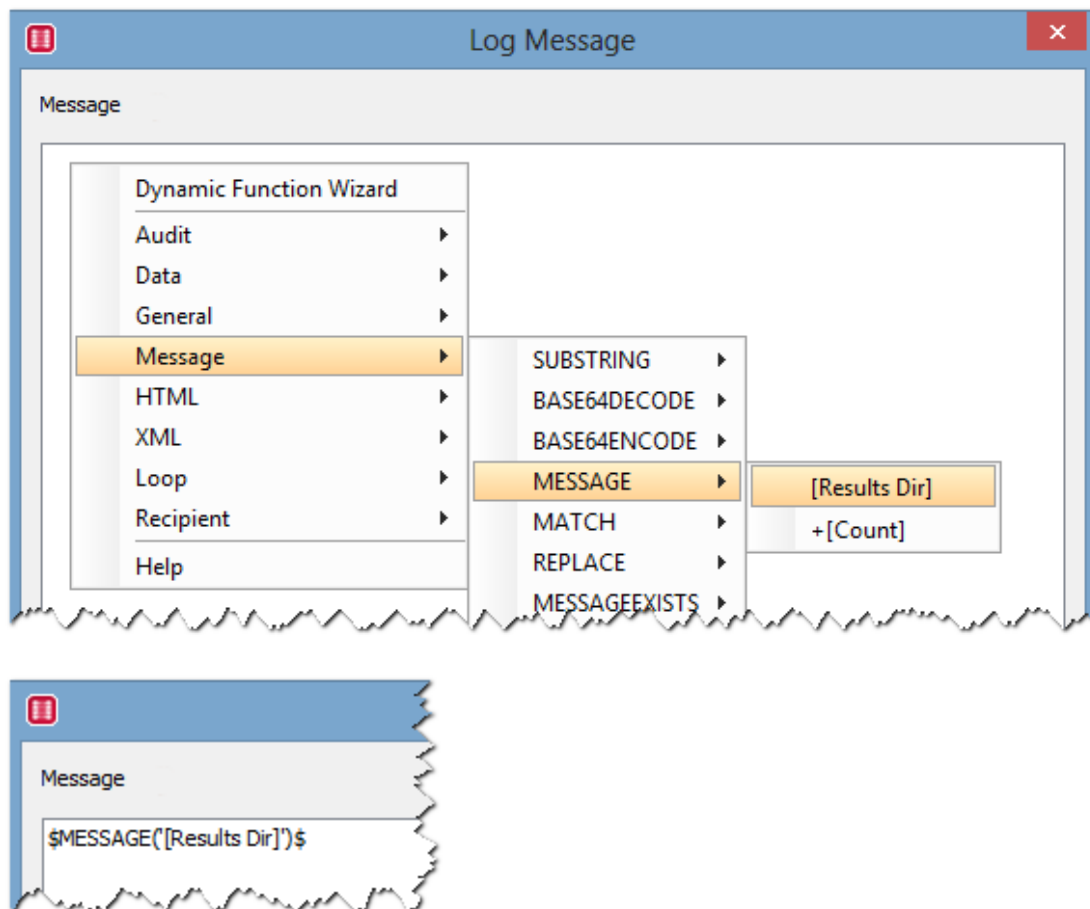
5. In the **Scope** tab, do the following:
 - a. Select **Specified Processes (all instances)**.
 - b. Click the **Add** button, and select the **Main** process created in step 1.
 - c. Clear the **Include Self** check box.
 - d. Click **Test** to save a sample value.



6. Set the process to active, and save it.
7. Create a log message module as follows:
 - a. Open the **Main** process.
 - b. Add a scheduler, and drag the **Process Configuration** process on to the process builder (this action should automatically add a **Call** module that is configured to call the **Process Configuration** process).
 - c. Add a **Log Message** module, and rename this to **Do something**. The log message module is used to simulate doing some work – this is where your normal processing modules would go.



8. Open the **Log message** module, and select the message section **[Results Dir]** that should be available.

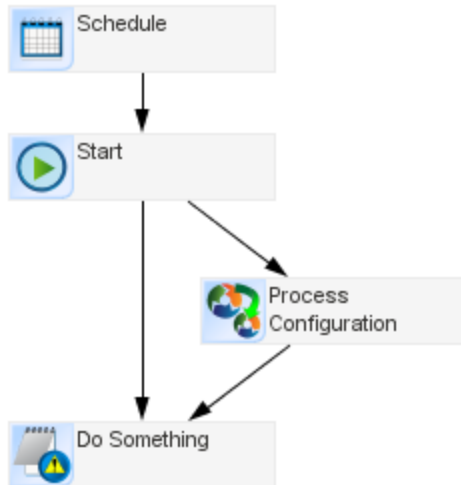


9. Set the process to active, save it, and then run it. Now, this process should output `c:\results` each time it is run.

Currently, **Process Configuration** is called every time the **Main** process is run. This can be a big overhead if the process is being called as a response to an HTTP call and a quick response is required.

To change the process so that **Process Configuration** is called only once, and improve performance, do the following:

1. Add a link as follows:



2. Edit the link between the **Start** and **Do something** and set it to **Run on condition**. Define the condition as follows:

Expression Builder

Descriptive view of condition

Run when initialised

Append AND expression Append OR expression Test

Expression details

Description

Run when initialised

Select category

STRING

Evaluate this expression (where)

\$MESSAGE([Results Dir])\$

Using operator

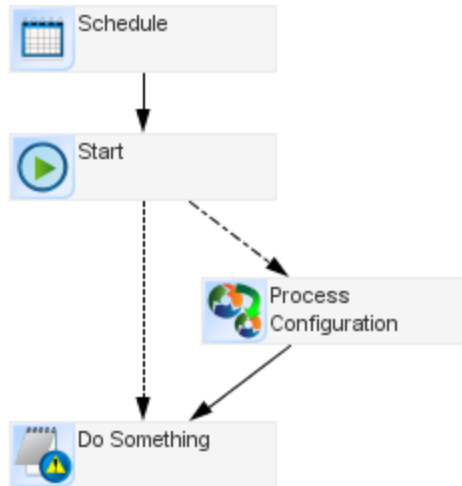
Not equal to

Against this expression (value)

Trim spaces Trim spaces

OK Cancel

3. Edit the link between **Start** and **Process Configuration** and set it to **Run when all other links using conditions have not run**.



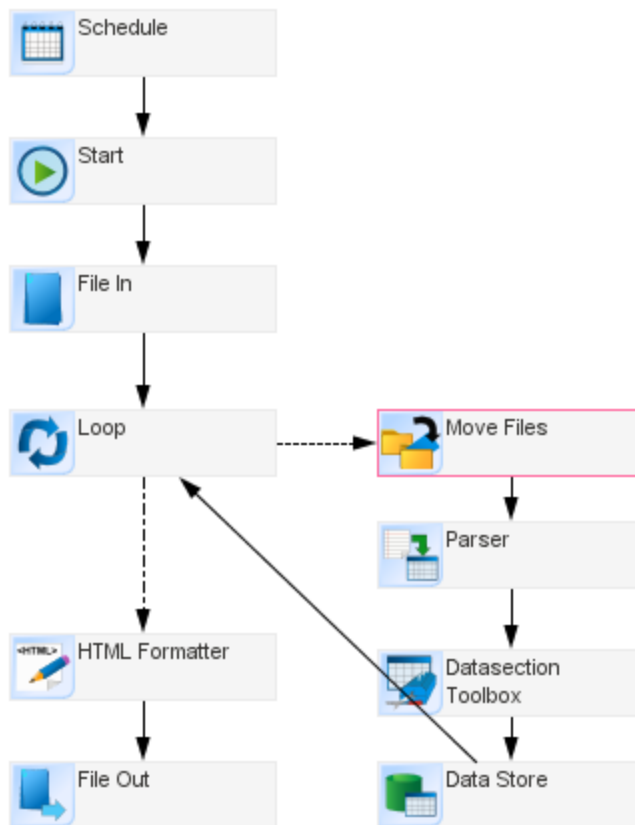
4. Save and run the process again.
5. Check the **Process status monitor** window and verify that **Process configuration** is not run and only **Main** is run.

Process configuration should never be called again until you export the processes to a new repository. If the **Process configuration** information changes, just run the **Process configuration** process manually (you will need to add a scheduler module) or press the **Test** button in the **data store** module.

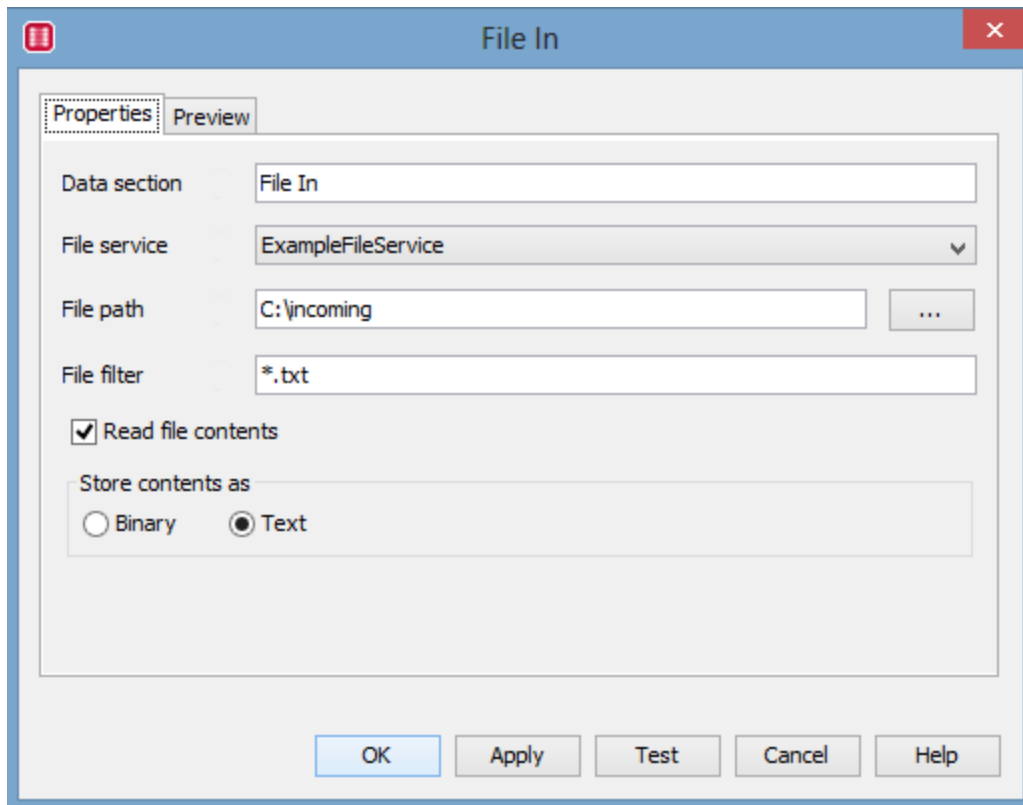
Worked Example 2: Creating a log of processed files

The following example shows a simple process that monitors a directory for incoming files. Once a file is detected, its content is read and the file is moved to a different directory. The **Data store** module is used to store a log of all files processed and their content. Without the **Data store** module, external database tables would be needed and SQL modules used to read and write the log contents to those tables. The **Data store** module greatly simplifies the process.

1. Create a new process and add the following modules as shown:



2. Open the **File In** module and set up the directory and file types you wish to monitor.
As the example process will be logging the file contents, the files should contain text information.



3. Set the **Loop** module to loop over each file read in, as follows:

Loop

Loop name: Loop

Loop type:

- ☒ Data section
- ☐ Recipients
- ☐ Count
- ☐ XPath

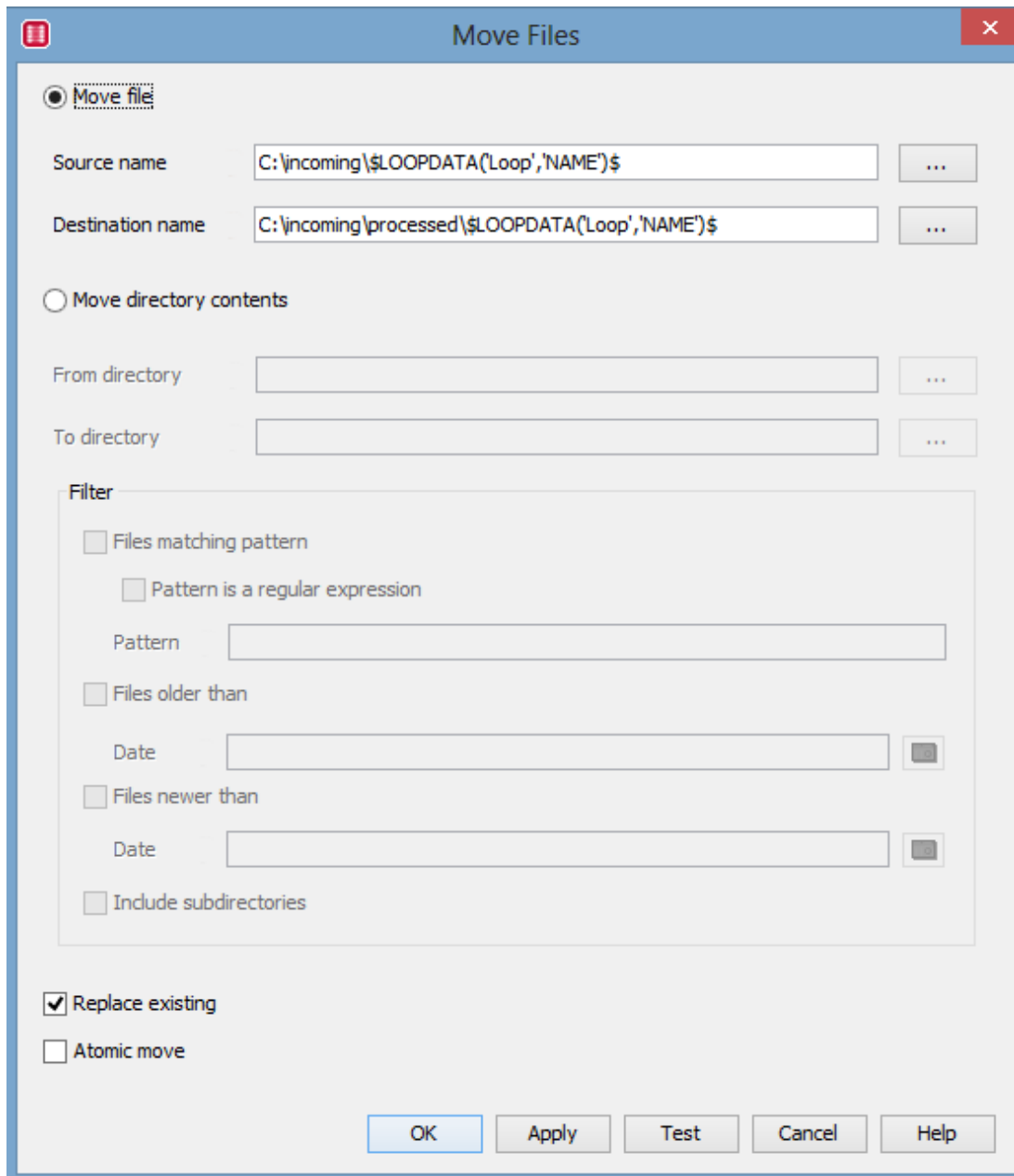
Data section Recipients Count XPath

Data section: File In

☐ Create equivalent data section with loop iteration information

OK Apply Cancel Help

4. Select the **Move Files** module to move the files that have been read in to another **archive** directory to show that they have been processed.



5. The **Parser** module is used to create a data section containing a record of the file that has been processed (the filename, date, and contents).

Note: The column delimiter and row delimiter are set as character sequences that would not be expected in the file name or file contents. This will make sure the contents are parsed correctly.

Once the parser is set up, press the **Test** button and save a sample of the data.

Parser

Properties Preview

Destination data section: FilesProcessedLog

Source text: Filename !! ProcessedDate !! Contents \$\$ \$LOOPDATA(Loop,'NAME')\$!! \$DATETIME('dd-mm-yy hh:nn:ss', 's', 0, 'now')\$!! \$LOOPDATA(Loop,'CONTENTS')\$

☒ Trim spaces Text qualifier:

☒ Use column names from first row of source

Column type: ☒ Delimited ☐ Fixed width

Delimiters: Column delimiter: !! Row delimiter: \$\$

Column Definition Column Output Order

Use	Name	Type	Format
<input checked="" type="checkbox"/>	Filename	Varchar	
<input checked="" type="checkbox"/>	ProcessedDate	Varchar	
<input checked="" type="checkbox"/>	Contents	Varchar	

Auto define
Add
Remove
Remove all
Move up
Move down

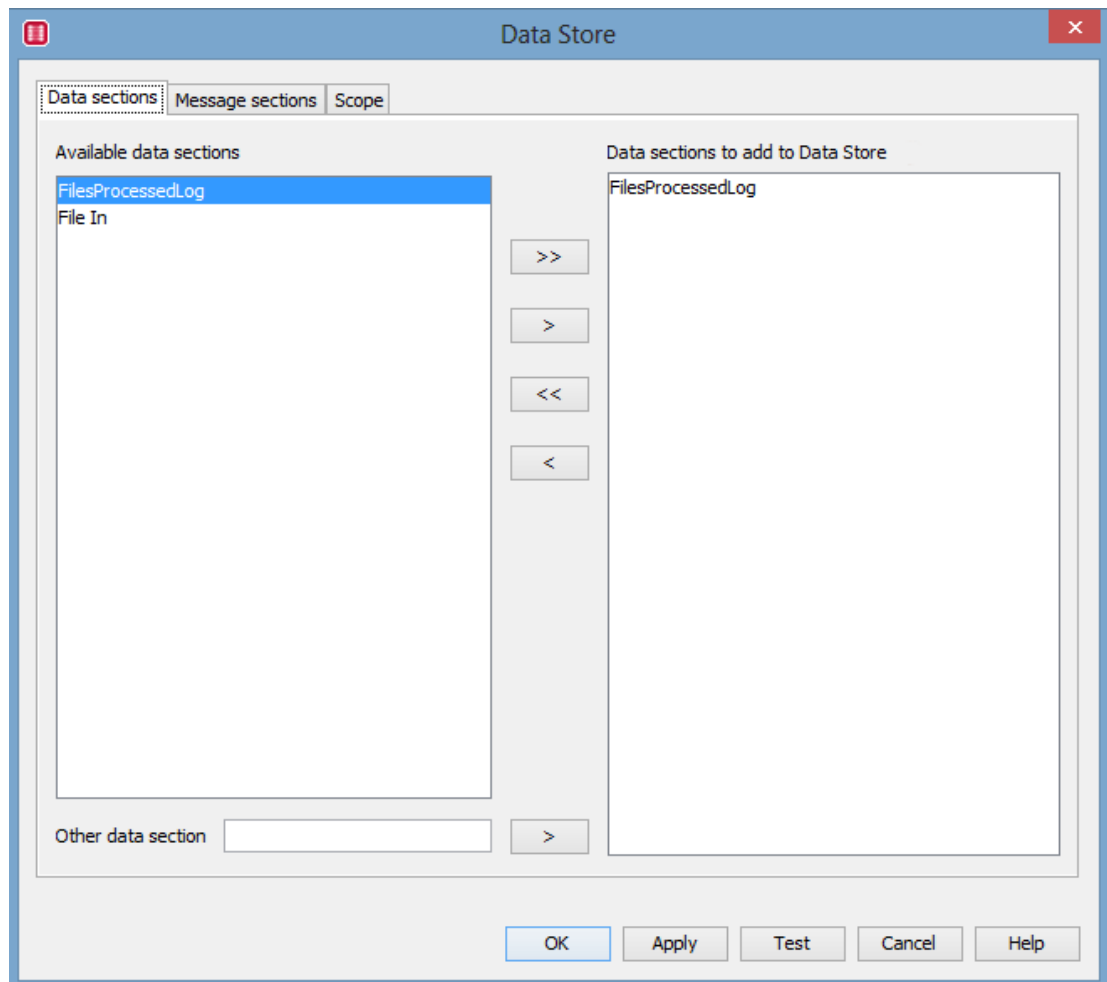
OK Apply Test Cancel Help

The source text value is defined as follows:

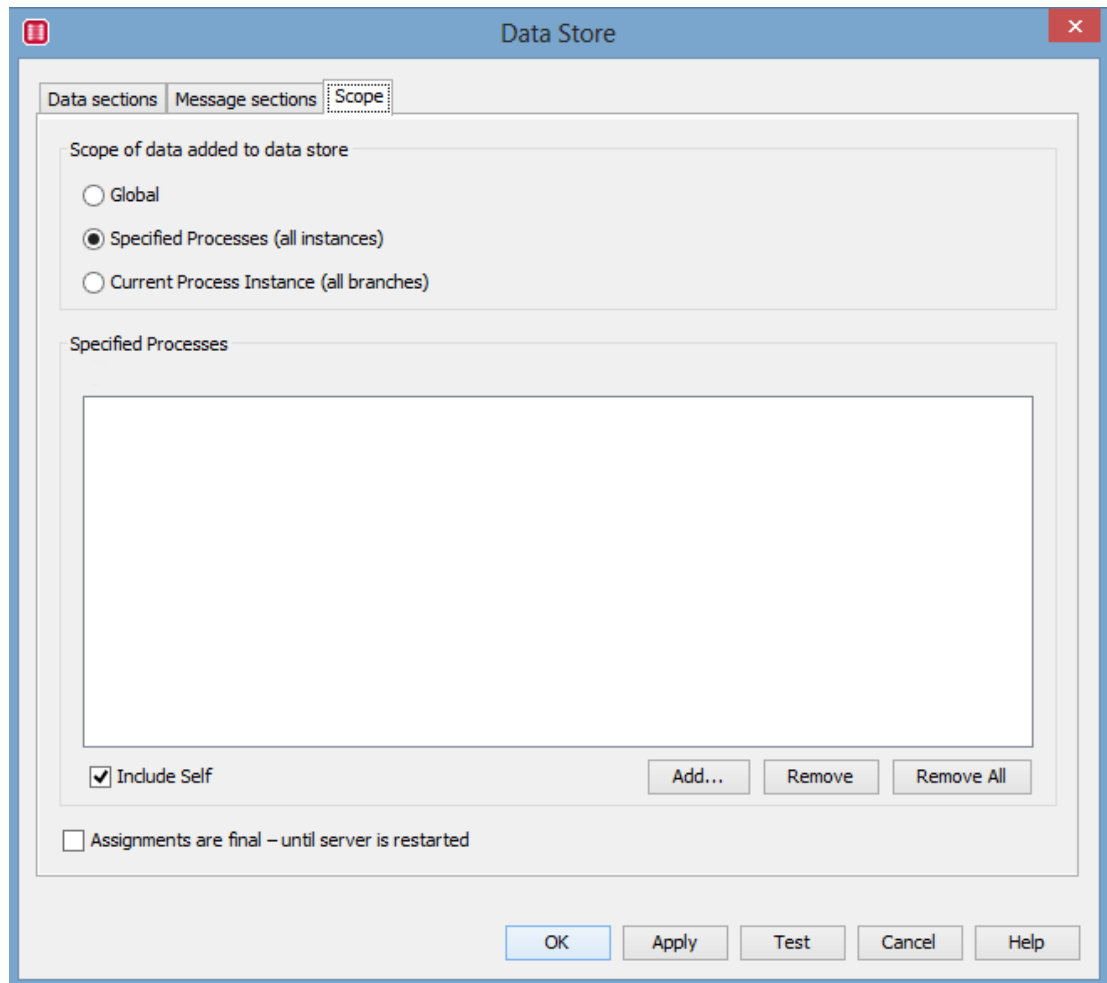
```
Filename !! ProcessedDate !! Contents $$ $LOOPDATA('Loop','NAME')$ !!
$DATETIME('dd-mm-yy hh:nn:ss', 's', 0, 'now')$ !!
$LOOPDATA('Loop','CONTENTS')$
```

6. Open the **Data store** module (not the data section toolbox). In the **Data store** module, do the following:

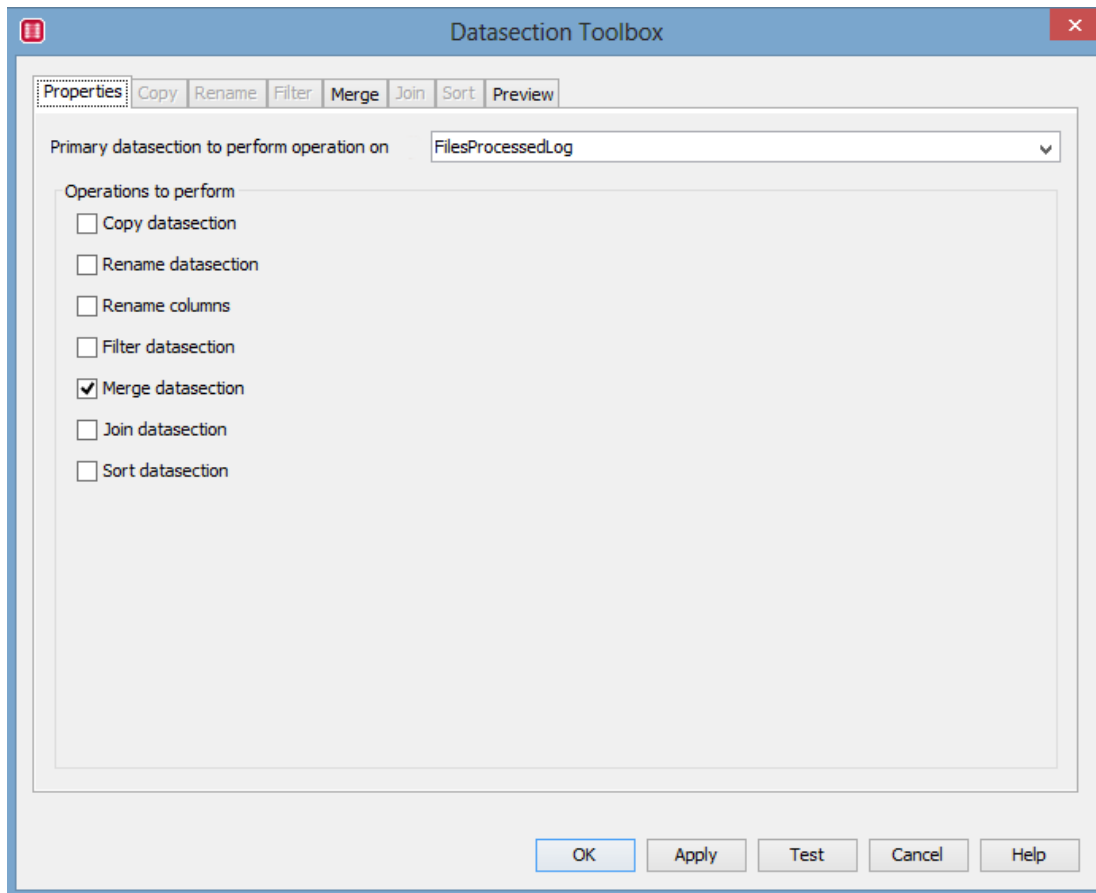
- a. Set the data section created in the **Parser** module to be added to the data store.

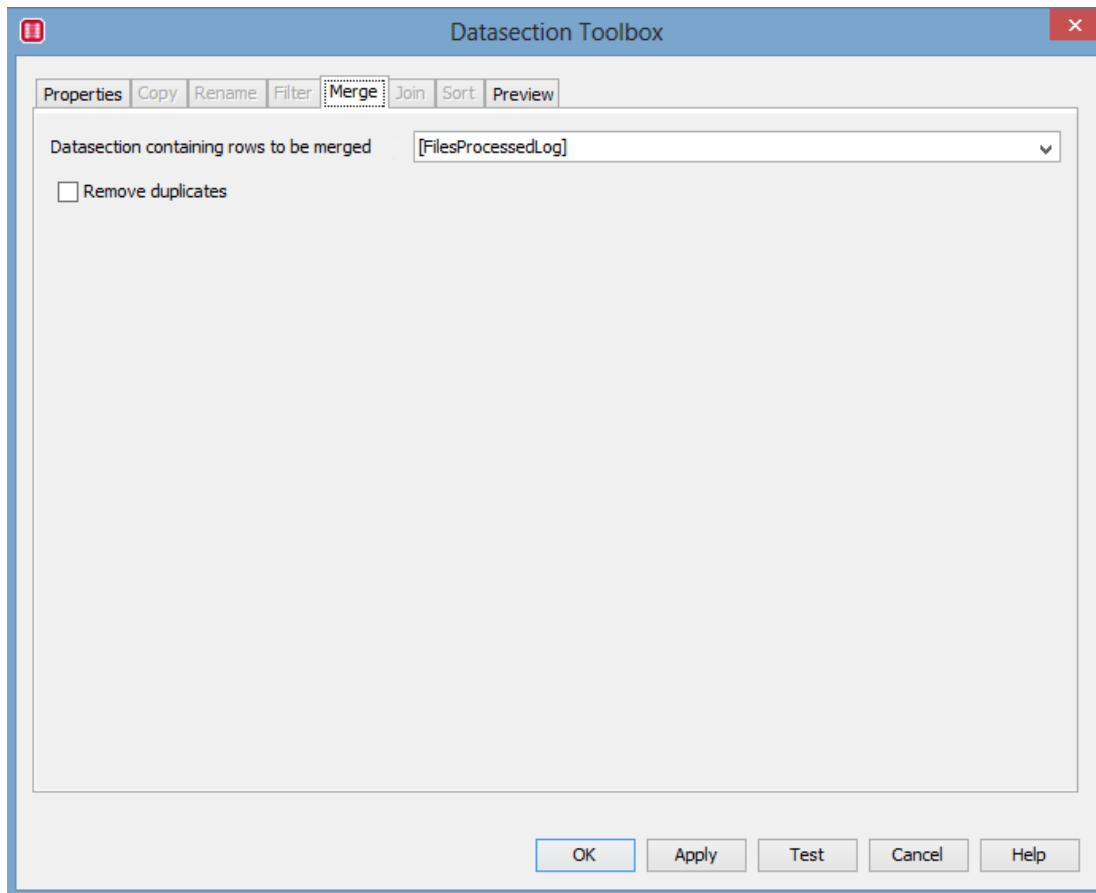


- b. Select the **Scope** as **Specified Processes** and ensure that **Include Self** remains selected. This means any instance of this process (and only this process) will be able to access that data section in the data store.



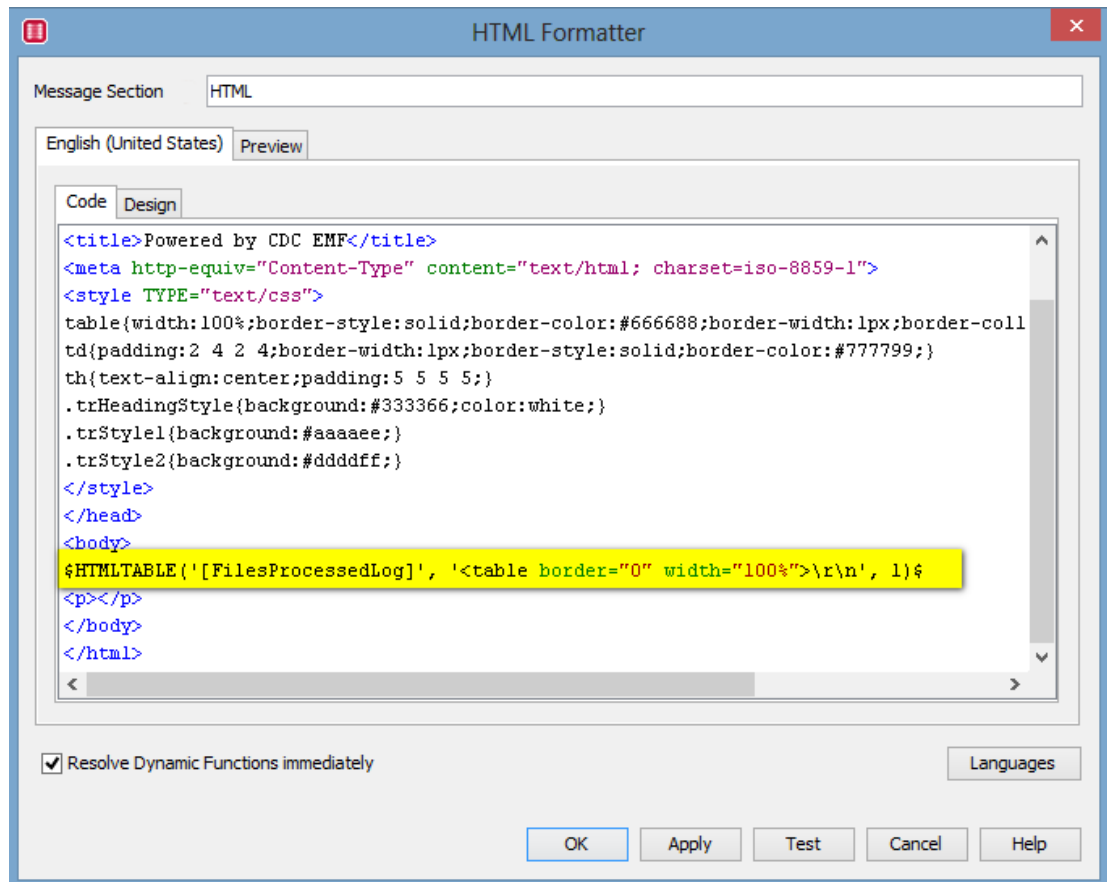
- c. Click the **Test** button. This will save the current value of the parser module to the data store.
 - d. Save the process. The newly added data store section is made available to the other modules in the process.
7. Open the **Datasection Toolbox** screen, and set the properties as shown below:





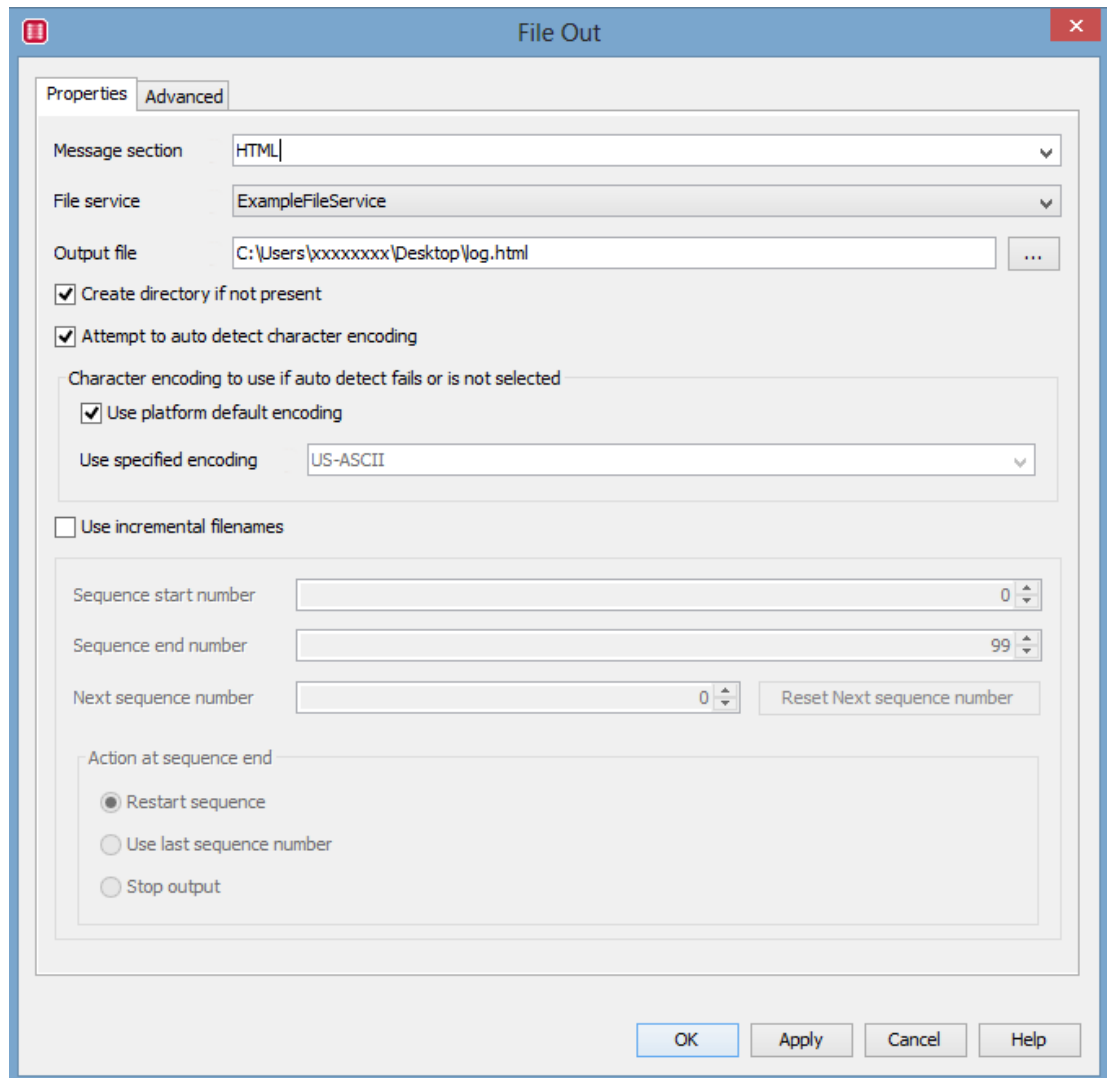
This will merge the newly created log record created in the **Parser** module with the data store version that will contain all previous records. The modified data section that then contains all log records is then placed in the **Data store** module (see **step 6**), overwriting any existing values.

8. To display the contents of the log every time the process is run, output it as an HTML file:
 - a. Open the HTML formatter and using the dynamic function menus, add the following line:



This will output the data store section **FilesProcessedLog**, which is the complete record of all processed files.

- b. Open the file out module and set it up to output the contents of the HTML formatter.



- c. Either run the process manually, or schedule to run it at a certain frequency.

Every time a file is added to the incoming directory, it should be moved to the processed folder and a log record added to the output log containing the file name, time, and contents of the file, as shown below:



Filename	ProcessedDate	Contents
t1.txt	15-08-13 16:08:25	hello World
t1 - Copy.txt	15-08-13 16:08:25	hello World, again!
test.txt	15-08-13 16:06:52	Folder: Process Folder/Module Tests/Data store, Process: test 2, Module: Gap Detection, Column: ResultsDataSection, Value: +[Gap detection] Folder: Process Folder, Process: bb, Module: Text Formatter 2, Column: MessageSection, Value: [Data Dir]
	15-08-13 16:03:47	

Using Dynamic Functions in EMF Processes

Dynamic Functions are tags used in modules that are evaluated, during the execution of an EMF Process, in order to dynamically configure module settings and personalize messages sent to recipients.

You can use Dynamic Functions to generate message content automatically and dynamically when an EMF Process is run. For instance, you might want to run an EMF Process that checks which of your clients are running low on their stocks of certain goods, and then send each a personal message with specific details of what they are short of. In order to create these personal messages automatically, you would use dynamic functions so that the same alert could create a variety of different messages.

For example: the same basic message is used in each of the following, with the text colored green inserted by dynamic functions:

DearDavid,

This is just a short message to remind you that you are running low on stocks of items 2234, 4786, 6783. Please let us know if you would like us to reorder on your behalf.

This message was sent at 12:56 on Friday July 19th to d.berkman@mycompany.com.

Roger,

This is just a short message to remind you that you are running low on stocks of items 1245, 6387. Please let us know if you would like us to reorder on your behalf.

This message was sent at 12:57 on Friday July 19th to rogwilliams@yourcompany.com.

Attn. MrSmith,

This is just a short message to remind you that you are running low on stocks of items 35, 578, 3348, 4932, 6783. Please let us know if you would like us to reorder on your behalf.

This message was sent at 12:57 on Monday July 22nd to michael.smith@myothercompany.com.

You can also [nest dynamic functions](#) inside each other, so that if the result of a dynamic function contains a further dynamic function, this will also be resolved at the same time.

There are a number of different **types of dynamic functions**:

- **Functions that display data** (for example, the codes of the relevant items, possibly retrieved using the SQL module). Data dynamic functions are evaluated once for each language.
- **Functions to display recipient information** (for example, the name of the recipient, any salutation, and their email address). Recipient dynamic functions are evaluated for each recipient, and if a message contains any recipient functions, it will be created for every recipient.
- **Functions to format and display information** (for example, to create an HTML table from a data section, or to parse XML).
- **General purpose functions** (for example, to insert the date and time in the body of a message).
- **The ESCALCODE function**, which is a special dynamic function that must be included in any EMF Processes that use [Escalation](#).

Dynamic functions are classified as **Per Recipient** or **Per EMF Process** (see [List of Dynamic Functions](#)).

- **Per Recipient** dynamic functions are executed once for each recipient in the recipient section of the EMF Process.
- **Per EMF Process** dynamic functions are executed once per EMF Process. They can return one or more values.

The following are some of the modules that can contain dynamic functions:

- The **SQL** and **HTTP** [Data modules](#)
- The **Parser**, **XML Parser** and, **XSL Transformer** [Filter and Transformation modules](#).
- **SNMP** [Data modules](#)
- All the [Output Formatter](#) modules, except the DSV Formatter
- The **File**, **Queue** and **FTP** [Output modules](#)
- The [Cascade module](#)
- The [Script module](#)

You can also use dynamic functions in the **Subject** line of the [Email Out](#) module, when defining the **Message Section**, **Target Directory** and **Output File location** for the [FTP Output](#), [File Output](#) and Fax Output modules, and when specifying Data section column names in the [Dynamic Recipients](#) module.

A note on the positioning of modules containing dynamic functions:

Dynamic functions are not necessarily evaluated within the module in which they were used; they are only evaluated at the point in the EMF Process where they are required. For example, a message section created with a Text Formatter module may contain several dynamic functions, but these will not be evaluated until an output module requires the message section. (Most dynamic functions are resolved at the output module. There are exceptions - for example, the SQL and XSLT modules - but these only resolve non-recipient based functions).

As another example, take an EMF Process using a fixed recipients module, a text formatter module that uses recipient-based dynamic functions, and an email module to send the message to the recipient. Since the message section created by the text formatter module uses recipient-based dynamic functions, you might assume that the Text Formatter module must appear after the fixed recipient module. In fact, this is not the case, because the dynamic functions are not resolved in the text formatter (as explained above)

However, if the fixed recipient module (or another, previous to it) uses the [Split recipients into blocks](#) option, it could be more efficient to put the text formatter module first because all the modules after the Fixed Recipient module are executed more than once (for example, once per block of recipients).

To insert a dynamic function:

Right-click in an appropriate field on the Properties page of one of the above modules to display a list of [all the dynamic functions available in EMF](#), and then select the required function(s).

Syntax Warnings:

- 1) Data and Message Section Names containing syntax symbols require the backslash (\) delimiter to be added when defined as a dynamic function parameter. For example, a Data Section called **My_Record's** would require entry as 'My_Record\'s'.
- 2) Section names are **case sensitive**, so you must enter them exactly as they appear in the EMF Process. The dynamic function will not get resolved if the section name is incorrect.
- 3) Dynamic function names are **case sensitive**. If you type the name of a dynamic function manually, ensure that you type it in uppercase characters, for example \$MESSAGE()\$ and not \$Message()\$.
- 4) You must put quotation marks (whether you need to use single or double quotes depends on your database server) around dynamic functions when performing string comparisons and assignments in SQL statements.

Example of using (single) quotes:

```
SELECT '$DATA('CSQL',0,0)$', '$DATA('CSQL',1,0)$'  
FROM warehousedb.dbo.warehouse_manager  
WHERE warehouse_manager.warehouse = '$DATA('CSQL',4,0)$'
```

[List of Dynamic Functions](#)

[Using Dynamic Functions in SQL Statements](#)

List of Dynamic Functions

There are two types of dynamic function that you can use in EMF, EMF Process-based (or **Per EMF Process**) and Recipient-based (**Per Recipient**).

Note: You can only use **Per Recipient** dynamic functions in [Output modules](#), and in the [Formatting modules](#) that create the message sections that they send.

Per EMF Process dynamic functions are run only once in each EMF Process instance, and therefore they only create a single output - however this output can contain **Single** or **Multiple** values.

Note: By nesting Dynamic Functions, it is possible for a "Per EMF Process function to be a "Per Recipient" function by embedding a "Per Recipient" in it - e.g. \$XMLESCAPE('\$RECIPIENT(Department)\$')\$ will escape out each and every recipient's department.

- **AUDITDATA** - returns a single cell of data from the current row in the audit module.
- **DATA** - returns a single cell of data from a specified data section in the EMF Process.
- **DATAMAP** - returns a look up a value in a data section where a column's values are mapped to another column.
- **SUBSTRING** - returns a single cell from a specified data section in the EMF Process.
- **ROWCOUNT** - returns the number of rows in a specified data section.
- **DATETIME** - returns the date and time used by the EMF Process instance, *formatted for the location and timezone of the server* (compare with the DATE and TIME functions - see below).

Note: DATETIME is system-based, and defaults to displaying the current system (e.g. local server) time, whereas **DATE** and **TIME** (see below) are recipient-based. In most cases you would use DATE and TIME, as these are automatically modified according to each recipient's personal settings (for further information see the examples for each).

- **DATETIMEDIFF** - calculates the difference between two date times, and returns the difference in units of your choice.
- **XPATH** - parses the XML in a message section using the supplied Xpath and returns a string representing an Xpath node.
- **EVALUATE** - evaluates mathematical formulae in EMF Processes.
- **FINDDATAROW** - returns the row ID, within a data section, that has a specified value in a specified column.
- **GETATTACHMENT** - used to retrieve an attachment from an email.
- **DATAFORMAT** - returns data from a specified data section in the EMF Process, formatted and delimited according to your specifications.
- **TABLE** - returns data from a specified data section formatted as a table according to your specifications.
- **HTMLTABLE** - returns data from a specified data section, formatted as a HTML table.
- **TABLEHEADER** - returns the headers (e.g. the field names of a table) from a specified data section.
- **MESSAGE** - returns the specified message section (optionally in a specified language).
- **HTMLESCAPE** - converts reserved characters that need to be escaped out of an HTML document.
- **HTMLESCAPE2** - converts reserved characters in the data section cell/message section that need to be escaped out of an HTML document.
- **HTMLUNESCAPE** - converts characters that have been HTML escaped back to their original forms.
- **HTMLUNESCAPE2** - converts characters in the data section cell/message section that have been HTML escaped back to their original forms.
- **XMLESCAPE** - converts characters that need to be escaped out of an XML document.

- [**XMLESCAPE2**](#) - converts characters in a specified message section that need to be escaped out of an XML document.
- [**XMLUNESCAPE**](#) - converts characters that have been XML escaped back to their original forms.
- [**XMLUNESCAPE2**](#) - converts characters in the data section cell/message section that have been XML escaped back to their original forms.
- [**LOWERCASE**](#) - renders data from the selected data section cell/message section in lower case letters.
- [**UPPERCASE**](#) - renders data from the selected data section cell/message section in upper case letters.
- [**URLENCODE**](#) - encodes the text into a safe format that can be passed as part of a URL.
- [**URLDECODE**](#) - decodes the text into its original format
- [**BASE64DECODE**](#) - base64 decodes some text to data.
- [**BASE64ENCODE**](#) - base64 encodes some data to text.
- [**DECODECHARS**](#) - decodes message sections that you know are encoded in a certain format (encoding).
- [**MATCH**](#) - performs a regular expression match against some text, returning the matching expressions.
- [**REPLACE**](#) - performs a regular expression match against some text, replacing the matching expression with specified text.
- [**DATAEXISTS**](#) - checks for the existence of a data section.
- [**JASONVALUE**](#) - The JASONVALUE Dynamic Function extracts data from a JSON (JavaScript Object Notation) document.
- [**LOOPDATA**](#) - returns data from the current row of the loop - see also Loop Module.
- [**LOOPITERATION**](#) - returns the count of the current loop iteration - see also Loop Module.
- [**MESSAGEEXISTS**](#) - checks for the existence of a message section.
- [**PROCESSNAME**](#) - outputs the name of the running process.
- [**PROCESSFOLDERPATH**](#) - outputs the folder structure of the running process.
- [**PROCESSAPINAME**](#) - outputs the API name of the running process.
- [**PROCESSID**](#) - outputs the process ID (AlertProfileID) of the running process.
- [**PROCESSINSTANCEID**](#) - outputs the process instance ID (AlertInstanceID) of the running process.
- [**PROCESSBRANCHINSTANCEID**](#) - outputs the process branch instance ID (alertState.getBranchInstanceID()) of the running process.
- [**MACHINENAME**](#) - outputs the EMF machine name the server is running on.
- [**REPOSITORYNAME**](#) - outputs the EMF repository database name.
- [**REPOSITORYDATABASETYPE**](#) - outputs the EMF repository database type, either "Oracle" or "SqlServer".
- [**VERSION**](#) - outputs the EMF server version number.
- [**RESOLVEENTITY**](#) - outputs the name that a virtual entity resolves to at runtime.

- **UUID** - retrieves a type 4 (pseudo randomly generated) UUID (universally unique identifier).
- **DIGEST** - calculates the message digest value of the passed message/data section, using the cryptographic hash function specified.

The following **Per Recipient** dynamic functions can run several times (once for each recipient in the recipient sections) in a single EMF Process instance.

- **DATE** - returns the system date, *formatted and (if necessary) adjusted for each recipient's time zone* (compare with DATETIME, above).
- **TIME** - returns the current time, *formatted and (if necessary) adjusted for each recipient's time zone* (compare with DATETIME, above).
- **ESCALCODE** - returns a unique escalation code that can be returned to the EMF Process by the recipient to prove receipt.
- **RECIPIENT** - returns a single specified field (e.g. name, address, email address, preferred language, aliases, etc.) from a recipient's details.
- **RECIPTABLE** - returns multiple fields from a recipient section, formatted as a table.

The following functions allow you to write a piece of code to iterate through recipients instead of data.

- **RECIPISECTIONCOUNT** - returns the number of recipient sections in the alert state at this point (has no parameters).
- **RECIPIENTCOUNT** - if no parameters are passed this function returns the total number of recipients in all recipient sections. If a quoted parameter is passed the count of recipients in the recipient section with the specified name is returned. If a number is passed as this parameter the count of recipients in the recipient section with the specified index is returned.
- **RECIPIENTDATA** - returns the value of data from the specified recipient section, column and row.

Using these three functions you can, for example, execute the following code in the [Scripting](#) module:

```
for (i = 0; i < df("$RECIPISECTIONCOUNT()$"); i++) {
  for (j = 0; j < df("$RECIPIENTCOUNT(i)$"); j++) {
    print(df("$RECIPIENTDATA(i, 'UserProfile.FirstName', " + j + ")$"));
  }
}
```

[Go to Start of Dynamic Functions](#)

AUDITDATA Dynamic Function

Dynamic Function
AUDITDATA
Description

The AUDITDATA Dynamic Function only has any relevance when being processed by the Audit Report module. It resolves to the value of the specified field for the data section row that is currently being processed by the Audit Report module. As the content of a message section that contains AUDITDATA is expected to be used by the audit reporting module, and therefore the message section will end up being displayed in a webpage - the data output is escaped so that it will display correctly in an HTML page.

Syntax

\$AUDITDATA('FieldName')\$

\$AUDITDATA(FieldIndex)\$

Parameter	Description
FieldName	The name of the field to be referenced. Must be placed in single quotes.
FieldIndex	The index of the field to be referenced. Index is zero based.

Example

(These examples were completed using a [Text Formatter](#) module.)

Information Entered	Information Returned
(Assumes the Data Section SQL Exists)	
ID - \$AUDITDATA('AddressID')\$, City - \$AUDITDATA('City')	1st row processed: ID - 1, City - Seattle 2nd row processed: ID - 2, City - Tacoma 3rd row processed: ID - 3, City - Kirkland 4th row processed: ID - 4, City - Redmond 5th row processed: ID - 5, City - London I live in Seattle
ID - \$AUDITDATA(1)\$, City - \$AUDITDATA(2)\$	1st row processed: ID - 1, City - Seattle 2nd row processed: ID - 2, City - Tacoma 3rd row processed: ID - 3, City - Kirkland 4th row processed: ID - 4, City - Redmond 5th row processed:

	ID - 5, City - London
Additional Information	
<p>FieldName and FieldIndex are alternative methods of referencing the same column in a table of data.</p> <p>All the parameters are required.</p>	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

BASE64ENCODE Dynamic Function

Dynamic Functions	
BASE64ENCODE	
Description	
<p>The BASE64ENCODE Dynamic Function converts some data into a representation using only 64 standard ASCII characters. It is a widely adopted standard. This is useful for dealing with binary data that may need to be transferred by means where strange characters would break the transfer, and also as a (poor) method of encrypting passwords so that the text isn't human readable. To reverse the operation, use the BASE64DECODE Dynamic Function</p>	
Syntax	
\$BASE64ENCODE('message_section_name')\$	
Parameter	Description
message_section_name	The name of the message section that contains the characters to encode. Must be placed in single quotes.
Example	
<p>If a message section called "Text" contains</p> <pre>Hello World~#;] [{ (*&^%\$£"! then \$BASE64ENCODE('Text')\$ will return SGVsbG8gV29ybGR+IztdW3t9KComXiUlJKMiIQ==</pre>	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

[Binary Data in EMF](#)

BASE64DECODE Dynamic Function

Dynamic Functions	
BASE64DECODE	
Description	
<p>The BASE64DECODE Dynamic Function converts a base 64 encoded text string back to its true representation. Base 64 is a standard way of representing data using only 64 standard ASCII characters. It is a widely adopted standard. This is useful for dealing with binary data that may need to be transferred by means where strange characters would break the transfer, and also as a (poor) method of encrypting passwords so that the text isn't human readable. To reverse the operation, use the BASE64ENCODE Dynamic Function. The BASE64DECODE Dynamic Function data is returned as binary data or text data.</p>	
Syntax	
\$BASE64DECODE('message_section_name', [type_of_data_to_return])\$	
Parameter	Description
message_section_name	The name of the message section that contains the characters to be decoded. Must be placed in single quotes.
type_of_data_to_return	Whether when the data is decoded it should be treated as binary data or text. A value of 1 means binary, a value of 0 means text (the default).
Example	
<p>If a message section called "Text" contains</p> <p>SGVsbG8gV29ybGR+IztdW3t9KComXiUIJKMiIQ==</p> <p>then</p> <p>\$BASE64DECODE('Text')\$</p> <p>will return</p> <p>Hello World~#;][{ }(*&^%\$£"</p>	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

[Binary Data in EMF](#)

DATA Dynamic Function

Dynamic Function
DATA

Description	
The DATA Dynamic Function returns one cell from the specified Data Section in the EMF Process.	
Syntax	
\$DATA('DataSectionName','FieldName',RowIndex)\$	
\$DATA('DataSectionName',FieldIndex,RowIndex)\$	
\$DATA('DataSectionName','FieldName',RowIndex,LogWarning)\$	
\$DATA('DataSectionName',FieldIndex,RowIndex,LogWarning)\$	
Parameter	Description
DataSectionName	The name of the Data Section in the EMF Process. Must be placed in single quotes.
FieldName	The name of the field to be referenced. Must be placed in single quotes.
FieldIndex	The index of the field to be referenced. Index is zero based.
RowIndex	The index of the row to be referenced. Index is zero based.
LogWarning	Whether to log a warning when a field/row is missing in a data section. This is useful to set when processing a field/row that may be optional and you do not want unnecessary information to be logged. 0 to log warnings and 1 to NOT log warnings. If the parameter is not set, then it will default to logging warnings.
Example	
(These examples were completed using a Text Formatter module.)	
Information Entered	Information Returned
(Assumes the Data Section SQL Exists)	
I live in \$DATA('SQL','City',0)\$	I live in Seattle
I live in \$DATA('SQL',2,0)\$	I live in Seattle

Additional Information	
<p><code>FieldName</code> and <code>FieldIndex</code> are alternative methods of referencing the same column in a table of data. They cannot be used in the same line of syntax when defining the dynamic function.</p> <p>If the row is invalid (i.e. there aren't that many rows) an empty string is returned.</p> <p>If the data section does not exist, no information is returned.</p> <p>All the parameters are required.</p>	
Properties of POP3IN_RD	Properties of POP3IN_SD

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

DATAEXISTS Dynamic Function

Dynamic Function	
DATAEXISTS	
Description	
Returns whether or not a data section exists. This dynamic function is mainly used in conditional link expressions to detect for the presence of a data section. Different links can then be followed if the data section existed, compared to if it didn't.	
Syntax	
\$DATAEXISTS('DataSectionName')\$	
Parameter	Description
DataSectionName	The name of the data section to be look for. Must be placed in single quotes.
Return value	
Returns true if the data section exists, false if it doesn't .	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

DATAFORMAT Dynamic Function

Dynamic Function	
DATAFORMAT	
Description	

The DATAFORMAT Dynamic Function returns the data in the specified format for the Data Section.

Syntax

\$DATAFORMAT('DataSectionName','FieldDelimiter','RowDelimiter',FieldNames)\$

Parameter	Description
DataSectionName	The name of the Data Section in the EMF Process. Must be placed in single quotes.
FieldDelimiter	The delimiter for the field. Must be placed in single quotes.
RowDelimiter	The delimiter for the row. Must be placed in single quotes.
FieldNames	A '0' or '1'. If setting '1' is used, the field names will be displayed using the same format as that is used for the data. The default value is '0' which does not display the field names.

Example

(These examples were completed using a [Text Formatter](#) module.)

Information Entered	Information Returned
(Assumes the Data Section SQL Exists)	
Here are the field values for the SQL Data Section: \$DATAFORMAT ('SQL', '\t', '\r\n', 0)\$	View DATAFORMAT returns
\$DATAFORMAT('SQL', '---', ';', 1)\$	View DATAFORMAT returns

Additional Information

In formatting fields `\t`, `\r`, `\n` are replaced by TAB, CR and LF respectively. e.g. `\r\n` means CRLF.

Field and Row delimiters are optional. If they are not added, then the field delimiter will default to a comma and the row delimiter will default to a carriage return and linefeed.

Data in quotes must be delimited with a backslash. (\)

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

DATAMAP Dynamic Function

Dynamic Function																	
DATAMAP																	
Description																	
The DATAMAP Dynamic Function is used to look up a value in a data section where a column's values are mapped to another column. For example, looking up an employee email address with a specified employee ID in an employee data section.																	
Syntax																	
<code>\$DATAMAP('DataSectionName', 'LookupKeyColumnName', 'LookupValueColumnName', 'LookupKey', ['DefaultValue'])\$</code>																	
<code>\$DATAMAP('DataSectionName', LookupKeyColumnId, LookupValueColumnId, 'LookupKey', ['DefaultValue'])\$</code>																	
Parameter	Description																
DataSectionName	The name of the Data Section in the EMF Process. Must be placed in single quotes.																
LookupKeyColumnName	The name of the column to be used as the lookup key. Must be placed in single quotes.																
LookupValueColumnName	The name of the column to be used to lookup the value. Must be placed in single quotes.																
LookupKey	The value of the lookup key. This value will be used to lookup the data in the specified LookupKeyColumnName. Must be placed in single quotes.																
LookupKeyColumnId	The index of the column to be used as the lookup key. Index is zero-based.																
LookupValueColumnId	The index of the column to be used to lookup the value. Index is zero-based.																
DefaultValue	Optional. This is the default value that will be used if the lookup is not found. Must be placed in single quotes.																
Example																	
(These examples were completed using a Text Formatter module.)																	
Information Entered	Information Returned																
Assume the Data Section 'data' exists and it contains the following information:																	
<table><tr><th>ID</th><th>Name</th><th>Department</th><th>Age</th></tr><tr><td>1</td><td>John</td><td>Sales</td><td>45</td></tr><tr><td>2</td><td>Ian</td><td>R&D</td><td>27</td></tr><tr><td>3</td><td>Cathy</td><td>Sales</td><td>32</td></tr></table>	ID	Name	Department	Age	1	John	Sales	45	2	Ian	R&D	27	3	Cathy	Sales	32	
ID	Name	Department	Age														
1	John	Sales	45														
2	Ian	R&D	27														
3	Cathy	Sales	32														

\$DATAMAP('data','Name','Age','Ian')\$	27
\$DATAMAP('data',1,3,'Ian')\$	27
\$DATAMAP('data',1,3,'Cathy','No age defined')\$	32
\$DATAMAP('data',1,3,'Bill','No age defined')\$	No age defined
\$DATAMAP('data',1,3,'Bill')\$	Nothing returned
Additional Information	
LookupKeyColumnName /LookupValueColumnName and LookupKeyColumnId/ LookupValueColumnId are alternative methods to reference the same column in a table of data. They cannot be used in the same line of syntax when defining the dynamic function.	
If the data or message section does not exist, no information is returned.	

DATE Dynamic Function

Dynamic Function	
DATE	
Description	
<p>The DATE Dynamic Function returns the current date, formatted according to each recipient's individual settings.</p> <p>The default value is the current EMF system date.</p> <p>Note: DATE and TIME are recipient-based, and are automatically modified according to each recipient's personal settings. If you want all recipients to receive identical data you should use the DATETIME function, which is system-based, and defaults to displaying the current system (e.g. local server) time.</p>	
Syntax	
\$DATE()\$	
Parameter	Description
None	N/A
Example	
(These examples were completed using a Text Formatter module.)	
Information Entered	Information Returned
(Assumes the current date is 20th August 2000)	
\$DATE()\$	20/08/2000
Additional Information	
N/A	

[List of Dynamic Functions](#)

Using Dynamic Functions in EMF

DATETIME Dynamic Function

Dynamic Function	
DATETIME	
Description	
<p>Returns a date and time, formatted to your requirements. The default value is the current EMF system date and time, but you can apply offset values to add or subtract a set period from these.</p> <p>Using DATETIME will ensure that all recipients receive identical data - e.g. to specify the arrival time of a flight to Helsinki to recipients worldwide, in Finnish local time. (Compare with the DATE and TIME functions, where the data is reformatted and modified for each recipient's personalized and local settings.)</p> <p>Note: In most cases, you would use the DATE and TIME functions instead of DATETIME, as these are recipient-based and automatically modified according to each recipient's personal settings. DATETIME is system-based, and defaults to displaying the current system (local server) time. For example, for U.K, it is 'dd/MM/yy hh:mm' (yy can be yyyy). The short date format is defined in the regional settings.</p> <p>It is important to specify both the date and time components in the format string. Otherwise, the format will not be recognized and resolved. Unresolved dates, by default, will be set to today.</p>	
Syntax	
\$DATETIME('Format', 'Interval', Displacement, 'DateTime')\$	
Parameter	Description
Format	<p>The output format of the date.</p> <p>The default value is the System date format.</p> <p>For more information, see Allowed characters.</p> <p>Must be placed in single quotes.</p>
Interval	<p>Used if the date add behavior is required.</p> <p>The default value is S (seconds)</p> <p>For more information, see Allowed characters.</p> <p>Must be placed in single quotes.</p>
Displacement	<p>The number to be added to the date.</p> <p>The default value is 0. (Do nothing)</p>
DateTime	The base date for the function.

<p>The DateTime component should be entered using the short date and short time format of your locale, as defined initially in the Windows Region and Language settings. It ignores any changes to the "Short date" and "Short time" formats set in the Windows Region and Language settings. So when you first select a country locale, the default formats displayed in the Windows Region and Language settings dialog are the formats the DateTime parameter will use. If the locale is changed, the server must be restarted for the changes to take effect.</p> <p>For example, in English US locale, the following format would work "5/25/2013 4:34 PM". In English UK "21/09/2013 16:30".</p> <p>Note: Setting this component as 'NOW' returns the System Date and Time.</p>	
<p style="text-align: center;">Example</p> <p style="text-align: center;">(These examples were completed using a Text Formatter module.)</p>	
Information Entered	Information Returned
(Assumes the current date and time are 20/08/2000, 12.24.26 pm)	
\$DATETIME('dd-mm-yy hh:nn:ss','s',0,'now')\$	20-08-00 12:24:26
\$DATETIME('dd/mmm/yyyy','yyyy',3,'01/01/2000 00:00')\$	01/Jan/2003
Additional Information	
Commas cannot be used as separators.	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

DATETIMEDIFF Dynamic Function

Dynamic Function
DATETIMEDIFF
Description
Calculates the difference between two date times, and returns the difference in units of your choice. The allowed characters are identical to those for DATETIME.

Syntax	
<pre>\$DATETIMEDIFF('now', 'dd-mm-yy hh:nn:ss', 'now', 'dd-mm-yy hh:nn:ss', 's')\$ \$DATETIMEDIFF('now', 'dd-mm-yy hh:nn:ss', 'now', 'dd-mm-yy hh:nn:ss')\$ or \$DATETIMEDIFF('now', 'dd-mm-yy hh:nn:ss')\$\$</pre>	
Parameter	Description
Parameter 1	A date time, can also be set to 'now' to represent the execution time.
Parameter 2	The format of the date time in Parameter 1.
Parameter 3	A date time, can also be set to 'now' to represent the execution time. Defaults to 'now'.
Parameter 4	The format of the data time in Parameter 3, can also be set to 'now' to represent the execution time. Defaults to 'dd-mm-yy hh:nn:ss'.
Parameter 5	<p>The unit of time in which to display the difference. Defaults to 'H'.</p> <p>Units supported are:</p> <p>YYYY Difference in years</p> <p>M Difference in months</p> <p>WW Difference in weeks</p> <p>D Difference in days</p> <p>H Difference in hours</p> <p>N Difference in minutes</p> <p>S Difference in seconds</p> <p>SN Difference in seconds of the minute</p> <p>NH Difference in minutes of the hour</p> <p>HD Difference</p>

	in hours of the day DW Difference in days of the week DM Difference in days of the month DY Difference in days of the year WM Difference in weeks of the month WY Difference in weeks of the year MY Difference in months of the year
Example	
Information Entered	Information Returned
\$DATETIMEDIFF('now', 'dd-mm-yy hh:nn:ss', '1-1-2002 12:00:00', 'dd-mm-yy hh:nn:ss', 'N')\$	The number of minutes since midday on the first day of the year.
\$DATETIMEDIFF('6-3-2002 04:36:23', 'dd-mm-yy hh:nn:ss', '1-1-2002 12:00:07', 'dd-mm-yy hh:nn:ss', 'SN')\$	16 seconds of the minute (23 seconds minus 7 seconds).
\$DATETIMEDIFF('6-3-2002 04:36:23', 'dd-mm-yy hh:nn:ss'\$	The time in the default unit interval (hours) from the displayed time to now.

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

DECODECHARS Dynamic Function

Dynamic Function
DECODECHARS
Description

Decodes message sections that you know are encoded in a certain format (encoding). For example, the HTTP module may read an XML document that has been encoded as 'UTF-8', or the File In module may read a file that was saved as Unicode. This dynamic function decodes the message section correctly, so that all characters are displayed correctly.

For example, to decode text in a message section "My message" that was UTF-8 encoded, create a new message section and add the following:

```
$DECODECHARS('My message', 'UTF-8')$
```

You can then use this newly created message section instead of "My message" throughout the rest of the process.

To decode a Unicode file, use:

```
$DECODECHARS('Text formatter (1)', 'UTF-16')$
```

Syntax

```
$DECODECHARS('<message section name>', 'encoding name')$
```

Parameter	Description
message_section_name	The name of the message section that contains the characters to be decoded. Must be placed in single quotes.
encoding name	The type of encoding. For example, 'UTF-8'.

[List of Dynamic Functions](#)

DIGEST Dynamic Function

Dynamic Function
DIGEST
Description
Calculates the message digest value of the passed message/data section, using the cryptographic hash function specified.

Syntax	
<pre>\$DIGEST('Algorithm','MessageSectionName')\$ \$DIGEST('Algorithm','DataSectionName','FieldName',RowIndex)\$ \$DIGEST('Algorithm','DataSectionName',FieldIndex,RowIndex)\$</pre>	
Parameter	Description
Algorithm	<p>The name of the Algorithm in the EMF Process. One of the following values:</p> <ul style="list-style-type: none"> • MD5 • SHA-1 (or SHA1) • SHA-256 (or SHA256) • SHA-384 (or SHA384) • SHA-512 (or SHA512) <p>(the "-" character is optional)</p>
MessageSectionName	<p>The name of the Message Section containing the text to calculate the digest value.</p> <p>Must be placed in single quotes.</p>
DataSectionName	<p>The name of the Data Section containing the data to calculate the digest value.</p> <p>Must be placed in single quotes.</p>
FieldName	<p>The name of the field to be referenced.</p> <p>Must be placed in single quotes.</p>
FieldIndex	<p>The index of the field to be referenced.</p> <p>Index is zero based.</p>
RowIndex	<p>The index of the row to be referenced.</p> <p>Index is zero based.</p>
<p>Example</p> <p>If the message section text 1 contained "The quick brown fox jumps over the lazy dog", then <pre>\$DIGEST('SHA-1','Text 1')\$</pre> would resolve to 2fd4e1c67a2d28fcd849ee1bb76e7391b93eb12.</p>	

ESCALCODE Dynamic Function

Dynamic Function
ESCALCODE
Description
The ESCALCODE Dynamic Function returns a unique escalation code that can be returned to the EMF Process by the recipient to prove receipt.

Syntax	
\$ESCALCODE()\$	
Parameter	Description
None	N/A
Example	
(These examples were completed using a Text Formatter module.)	
Information Entered	Information Returned
\$ESCALCODE()\$	@!1:A:1!@ The characters between the "at" signs (@) are the Escalcode.
Additional Information	
Important: To use escalation in an EMF Process, you must set Message state logging to "Full" on the Advanced Properties page of the appropriate Output module.	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

EVALUATE Dynamic Function

Dynamic Functions	
EVALUATE	
Description	
<p>The EVALUATE Dynamic Function evaluates mathematical formulae in EMF Processes. This allows you, for example, to do conditional branching in an EMF Process (e.g. "branch if $x+y>z$") without using an XPATH expression (which would require the creation of an XML message section).</p> <p>The EVALUATE Dynamic Function runs the parameter of the function as a JavaScript expression, and is similar to a Scripting module except that it allows you to embed the script anywhere that you can use a Dynamic Function. By nesting functions you can have complex statements such as <code>\$EVALUATE('\$DATA('SQL','InventoryQty',0)\$ + \$DATA('SQL','ReorderQty',0)\$')\$</code>.</p> <p>Note: You can also use the EVALUATE function to manipulate data using JavaScript code before outputting it.</p>	
Syntax	
\$EVALUATE('formula')\$	
Parameter	Description
The mathematical formula to evaluate.	Can also include further Dynamic Functions (see example below).

	Must be placed in single quotes.
Example	
Information Entered	Information Returned
<code>\$EVALUATE('5+3')\$</code>	8
<code>\$EVALUATE('\$DATA ('SQL',1,0)\$ + \$DATA ('SQL',2,0)\$')\$</code>	The result will depend on what the data section 'SQL' contains. (Would be useful in conditional branching).
<code>\$EVALUATE('("\$RECIPIENT (Department)\$" == "Stores") ? "09" : "9"\$</code>	Used in a text formatter (in the format Please dial \$EVALUATE ('("\$RECIPIENT(Department)\$" == "Stores") ? "09" : "9"\$ for an outside line.), this would return the number that must be entered in order to dial a telephone number outside the current internal exchange, depending upon what department the recipient worked in (if they worked in "Stores" it would return "09", any other department, it would return "9").
Additional Information	
It is also possible to write multiple lines of script code. If you do this, the first statement should be an empty double-quoted string (i.e. <code>""</code>) as this gets assigned internally to a variable, and without it the script will be syntactically incorrect. The value that will be displayed by the DF should be assigned to the variable "result".	
<code>\$EVALUATE('""; for (i = 0; i<10; i++) { System.out.println("Hello World " + i); } result="abc";')\$</code>	
Security risks: Malicious attack is possible while processing the Evaluate Dynamic Function. It is possible to change the behavior of Evaluate Dynamic Function, see Compatibility settings to work with or without security risks.	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

FINDDATAROW Dynamic Function

Dynamic Function	
FINDDATAROW	
Description	
The FINDDATAROW Dynamic Function returns the row ID, within a data section, that has a specified value in a specified column.	
Syntax	
<code>\$FINDDATAROW()\$</code>	
Parameter	Description
SectionName	The name of the data section to work

	with.
ColumnIndex / ColumnName	The name or index of the column within which the searching will be done.
RowValue	The value you are trying to match.
Return value	The first row number that matches RowValue in the specified column.

Example

For a data section named SQL that contains the following:

Empl oyee ID	Empl oyee name
1	Brow ning
2	Wilco x
3	Beau mont
4	Seab orn
5	Smit h

Information Entered	Information Returned
<code>\$FINDDATAROW('SQL', 'Employee name', 'Wilcox')\$</code> or <code>\$FINDDATAROW('SQL', 1, 'Wilcox')\$</code>	1 (rows and columns are zero-based in Dynamic Functions)
Additional Information	
N/A	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

GETATTACHMENT Dynamic Function

Dynamic Function
GETATTACHMENT
Description
The GETATTACHMENT Dynamic Function is used to retrieve the content of an email attachment. The email attachment is stored in the POP3IN_ATTACHMENTS data section .

Syntax	
\$GETATTACHMENT()\$	
Parameter	Description
AttachmentName	The name of the attachment that you want to retrieve.
Return value	The attachment with the specified name.
Example	
For an email that contains first quarter sales figures called 'Q1_Sales.csv':	
Information Entered	Information Returned
\$GETATTACHMENT('Q1_Sales.csv')\$	The contents of the file attachment
Additional Information	
<p>Important: GETATTACHMENT can be used with both plain text and binary attachments. Binary attachments are encoded as Base64. If you want to send attachments via email, see the Email out module.</p>	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

HTMLTABLE Dynamic Function

Dynamic Function	
HTMLTABLE	
Description	
Returns a data section as a HTML table, using either the specified ' StartOfTable ' or a default provided if this is omitted.	
<p>Changing the table appearance - when the table is written out a number of a predefined style sheet classes are written into the table. This allows a style sheet to be defined in the HTML document to change the appearance of the table. The style class names written are:</p> <ul style="list-style-type: none"> • "trStyle1" and "trStyle2" for alternative rows. • "tdStyle1" and "tdStyle2" for alternative columns. <p>The table header is written out with the <THEAD> and <TH> HTML tags which allow a style to be defined to change the appearance of the header.</p>	
Syntax	
\$HTMLTABLE('DataSectionName', 'StartOfTable', ShowColumnNames, ColumnSettings)\$	
Parameter	Description
DataSectionName	The name of the data section to be returned as a HTML Table. Must be placed in single quotes.

StartOfTable	<p>Defines the format of the HTML Table.</p> <p>Must be placed in single quotes.</p>
ShowColumnNames	<p>A '0' or '1'. If setting '1' is used, the column names will be displayed using the same format as that is used for the data. The default value is '1' which displays the column names.</p>
ColumnSettings	<p>Allows the columns to display to be specified, the column headings that should be displayed for each column, and the order the columns should appear in. If no value is specified then all columns will be displayed, and the column heading will be taken from those in the data section.</p> <p>The order that columns are displayed in is set by the order that they appear in this settings - so columns can be displayed in a different order than they appear in the data section.</p> <p>Each setting takes the format \[label="abc", index=2] or \[label="abc", name="myColumn"]. In both cases the label parameter is optional - and if not specified then the column will be labelled with the name of the column from the data section. So in the first example, the 3rd column (index is zero based) will be displayed and its heading will be "abc". In the second example the column with the name "myColumn" will be displayed and its heading will be "abc".</p> <p>Column settings must be placed in single quotes. So a full column settings example of displaying 3 columns would be '\[label="My First Column", index=0]\[label="My Second Column", index=1]\[label="My Third Column", index=2]'. See the last two examples below for further guidance.</p>
Example	
Information Entered (Assumes the Data Section SQL Exists)	Information Returned
<pre><html> <head> <title>Powered by EMF</title> <meta http- equiv="Content-Type" content="text/html; charset=iso-8859-1"> </head> <body></pre>	<p>View HTML Table</p> <p>Table with headers and no styles</p>

<p><p>HTMLTABLE Example (Including Field Names): </p></p> <pre>\$HTMLTABLE('SQL','<table border="1" width="100%">\r\n',1)\$</pre> <p></body></p> <p></html></p>	
<p><html></p> <p><head></p> <p><title>Powered by EMF</title></p> <p><meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"></p> <p></head></p> <p><body></p> <p><p>HTMLTABLE Example (Excluding Field Names): </P></p> <pre>\$HTMLTABLE('SQL','<table border="1" width="100%">\r\n',0)\$</pre> <p></body></p> <p></html></p>	<p>View HTML Table</p> <p>Table with no headers and no styles</p>
<p><html></p> <p><head></p> <p><title>Powered by EMF</title></p> <p><meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"></p> <p><STYLE TYPE="text/css"></p> <p><!--</p> <pre>.trStyle1{ background: Aquamarine }</pre>	<p>View HTML Table</p> <p>Changing the background of the row colors</p>

<pre>.trStyle2{ background: Aquamarine } --> </STYLE> </head> <body> <p>HTMLTABLE Example (Including Field Names): </P> \$HTMLTABLE('SQL','<table border="1" width="100%">\r\n',0)\$ </body> </html></pre>	
<pre><html> <head> <title>Powered by EMF</title> <meta http- equiv="Content-Type" content="text/html; charset=iso-8859-1"> <STYLE TYPE="text/css"> <!-- .trStyle1{ background: Aquamarine } .trStyle2{ background: Gainsboro } --> </STYLE> </head> <body> <p>HTMLTABLE Example (Including Field Names): </P> \$HTMLTABLE('SQL','<table border="1"</pre>	<p>View HTML Table</p> <p>Alternating the background row colors</p>

<pre>width="100%">\r\n',0)\$ </body> </html></pre>	
<pre><html> <head> <title>Powered by EMF</title> <meta http- equiv="Content-Type" content="text/html; charset=iso-8859-1"> <STYLE TYPE="text/css"> <!-- .tdStyle1{ color: #B8860B; } .tdStyle2{ color: #000000; } --> </STYLE> </head> <body> <p>HTMLTABLE Example (Including Field Names): </P> \$HTMLTABLE('SQL','<table border="1" width="100%">\r\n',0)\$ </body> </html></pre>	<p>View HTML Table</p> <p>Alternating the text color of the columns</p>
<pre><html> <head> <title>Powered by EMF</title> <meta http- equiv="Content-Type" content="text/html; charset=iso-8859-1"></pre>	<p>View HTML Table</p> <p>Alternate row background colors with alternative column text colors and a different header color</p>

```
<STYLE TYPE="text/css">
<!--
THEAD { background:
yellow; }

.trStyle1{ background:
Aquamarine }

.trStyle2{ background:
Gainsboro }

.tdStyle1{ color:
#B8860B; }

.tdStyle2{ color: #000000;
}

-->
</STYLE>
</head>
<body>
<p>HTMLTABLE Example
(Including Field
Names): </P>
$HTMLTABLE('SQL','<table
border="1"
width="100%">\r\n',0)$
</body>
</html>
```

```
<html>
<head>
<title>Powered by
EMF</title>

<meta http-
equiv="Content-Type"
content="text/html;
charset=iso-8859-1">

<STYLE TYPE="text/css">
<!--
THEAD { background:
yellow; }

.trStyle1{ background:
```

[View HTML Table](#)

Reordering the fields that are output, specifying to only output selected fields and changing the headings on those fields. Fields are referenced by 'name'

```

Aquamarine }
.trStyle2{ background:
Gainsboro }
-->
</STYLE>
</head>
<body>
<p>HTMLTABLE Example
(reordered fields with new
headings)):</P>
$HTMLTABLE('SQL','<table
border="1"
width="100%">\r\n',1,'\
[label="Surname Name",
name="HouseholdName"]\
[label="Phone Number",
name="HomePhone"]' )$
</body>
</html>

```

```

<html>
<head>
<title>Powered by
EMF</title>
<meta http-
equiv="Content-Type"
content="text/html;
charset=iso-8859-1">
<STYLE TYPE="text/css">
<!--
THEAD { background:
yellow; }
.trStyle1{ background:
Aquamarine }
.trStyle2{ background:
Gainsboro }
-->
</STYLE>
</head>

```

[View HTML Table](#)

Reordering the fields that are output, specifying to only output selected fields and changing the headings on those fields. Fields are referenced by 'index'

```
<body>
<p>HTMLTABLE Example
(reordered fields with new
headings)):</P>
$HTMLTABLE('SQL','<table
border="1"
width="100%">\r\n',1,'\
[label="Surname Name",
index=6]\[label="Phone
Number", index=5]' )$
</body>
</html>
```

Additional Information

The default '[StartOfTable](#)' value is `<table border="1" width="100%">\r\n`
Any HTML table formatting commands can be used by the [StartOfTable](#) parameter.

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

HTMLESCAPE Dynamic Function

Dynamic Functions	
HTMLESCAPE	
Description	
The HTMLESCAPE Dynamic Function converts reserved characters that need to be escaped out of an HTML document into the form &xx;	
Syntax	
\$HTMLESCAPE('Text to escape')\$	
Parameter	Description
Text to escape	The text that contains the characters to be escaped. Must be placed in single quotes.
Example	
(These examples were completed using a Text Formatter module.)	
Information Entered	Information Returned
\$HTMLESCAPE('ÖßÚ96')\$	ÖßúÚ96
Additional Information	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

HTMLESCAPE2 Dynamic Function

Dynamic Function	
HTMLESCAPE2	
Description	
The HTMLESCAPE2 Dynamic Function converts reserved characters in the data section cell/message section that need to be escaped out of an HTML document into the form &xx;.	
Syntax	
\$HTMLESCAPE2('DataSectionName',FieldIndex,RowIndex)\$	
\$HTMLESCAPE2('DataSectionName','FieldName',RowIndex)\$	
\$HTMLESCAPE2('MessageSectionName')\$	
Parameter	Description
DataSectionName	The name of the Data Section in the EMF Process. Must be placed in single quotes.
FieldIndex	The index of the field to be referenced. Index is zero-based.
RowIndex	The index of the row to be referenced. Index is zero-based.
FieldName	The name of the field in the Data Section to be referenced. Must be placed in single quotes.
MessageSectionName	The name of the Message Section in the EMF Process. Must be placed in single quotes.
Examples	
(These examples were completed using a Text Formatter module.)	
Information Entered	Information Returned
Assume the Data Section Data exists and it contains the following information: Column0 Column1 Ö 96;	
\$HTMLESCAPE2('Data',0,0)\$	Ö
\$HTMLESCAPE2('Data',1,0)\$	96;
Information Entered	Information Returned

Assume the Message Section Text exists and it contains the following information: Ö	
\$HTMLESCAPE2('Text')\$	Ö
Additional Information	
<p>FieldName and FieldIndex are alternative methods to reference the same column in a table of data. They cannot be used in the same line of syntax when defining the dynamic function.</p> <p>If the row is invalid (that is, there are not that many rows) an empty string is returned.</p> <p>If the data or message section does not exist, no information is returned.</p>	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

HTMLUNESCAPE Dynamic Function

Dynamic Functions	
HTMLUNESCAPE	
Description	
<p>The HTMLUNESCAPE dynamic function converts characters that have been HTML escaped back to their original forms. Any text passed in will be unescaped. For example, if the text contains <code>&#8482;</code> or <code>&trade;</code>, then it will be replaced with the trademark symbol "™". See http://www.escapecodes.info/ for a complete list of HTML escape characters.</p>	
Syntax	
\$HTMLUNESCAPE('Test to unescape')\$	
Parameter	Description
Text to unescape	The text that contains the characters to be unescaped. Must be placed in single quotes.
Example	
(These examples were completed using a Text Formatter module.)	
Information Entered	Information Returned
\$HTMLUNESCAPE('Aptean®')\$	Aptean®
Additional Information	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

HTMLUNESCAPE2 Dynamic Function

Dynamic Function	
HTMLUNESCAPE2	
Description	
<p>The HTMLUNESCAPE2 dynamic function converts characters that have been HTML escaped out back to their original forms. Any text in the data section cell/message section will be unescaped. For example, if the text contains <code>&#8482;</code> or <code>&trade;</code>, then it will be replaced with the trademark symbol "™". See http://www.escapecodes.info/ for a complete list of HTML escape characters.</p>	
Syntax	
\$HTMLUNESCAPE2('DataSectionName',FieldIndex,RowIndex)\$	
\$HTMLUNESCAPE2('DataSectionName','FieldName',RowIndex)\$	
\$HTMLUNESCAPE2('MessageSectionName')\$	
Parameter	Description
DataSectionName	The name of the Data Section in the EMF Process. Must be placed in single quotes.
FieldIndex	The index of the field to be referenced. Index is zero-based.
RowIndex	The index of the row to be referenced. Index is zero-based.
FieldName	The name of the field in the Data Section to be referenced. Must be placed in single quotes.
MessageSectionName	The name of the Message Section in the EMF Process. Must be placed in single quotes.
Examples	
(These examples were completed using a Text Formatter module.)	
Information Entered	Information Returned
<p>Assume the Data Section <i>Data</i> exists and it contains the following information:</p> <p>Column0 Column1</p> <p>Aptean&#174; &#60;</p>	
\$HTMLUNESCAPE2('Data',0,0)\$	Aptean®
\$HTMLUNESCAPE2('Data',1,0)\$	<
Information Entered	Information Returned
<p>Assume the Message Section <i>Text</i> exists and it contains the following information:</p> <p>Aptean&#174;</p>	

`$HTMLUNESCAPE2('Text')$`

Aptean®

Additional Information

FieldName and FieldIndex are alternative methods to reference the same column in a table of data. They cannot be used in the same line of syntax when defining the dynamic function.

If the row is invalid (that is, there are not that many rows) an empty string is returned.

If the data or message section does not exist, no information is returned.

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

JSONVALUE Dynamic Function

Dynamic Function	
JSONVALUE	
Description	
The JSONVALUE Dynamic Function extracts data from a JSON (JavaScript Object Notation) document.	
Syntax	
<code>\$JSONVALUE('MessageSectionName' , 'JSONPointerExpression')\$</code>	
Parameter	Description
MessageSectionName	The name of the Message Section containing a JSON document to be parsed.
JSONPointerExpression	<p>A JSON Pointer expression that identifies a specific value within the JSON document. Must be placed in single quotes.</p> <p>For more information on JSON Pointer expression, refer to the following:</p> <p>https://tools.ietf.org/html/rfc6901</p>
Information Entered	Information Returned
<p>(Assumes the message section "Text" exists)</p> <p>The message section "Text" contains the following:</p> <pre>{ "id": "0001", "type": "donut",</pre>	


```

    "name": "Cake",
    "ppu": 0.55,
    "topping": [
        { "id": "5001", "type": "None"
    },
        { "id": "5002", "type": "Glazed"
    },
        { "id": "5005", "type": "Sugar"
    },
        { "id": "5007", "type":
    "Powdered Sugar" }
    ]
}

```

\$JSONVALUE('Text', '/ppu')\$	0.55
\$JSONVALUE('Text', '/topping/2')\$	{"id": "5005", "type": "Sugar"}
\$JSONVALUE('Text', '/topping/1/type')\$	Glazed

Additional Information

If the JSON pointer expression attempts to reference a JSON property that is defined as "Null" or if the property does not exist at runtime, then a blank value will be returned.

For more information on JSON, refer to the following:

<http://json.org/>

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

LOOPDATA Dynamic Function

Dynamic Function
LOOPDATA
Description
<p>The LOOPDATA Dynamic Function returns the data for the specified column of the current row of the specified loop. This is only used if loops are looping over datasections or recipients. If iterating over recipients then the predefined values for the column names can be found here.</p> <p>When XPath is used, the FieldName parameter is ignored and can be set to any value. The dynamic function will return the result of the XPath query for that iteration.</p> <p>When JSON is used, the second parameter is a JSON pointer expression. The dynamic function will return the JSON fragment referenced by the JSON pointer expression evaluated against the JSON document of that iteration.</p>
Syntax
\$LOOPDATA('LoopName', 'FieldName')\$

\$LOOPDATA('LoopName',FieldIndex)\$	
\$LOOPDATA('LoopName', 'JSON Pointer expression')\$	
Parameter	Description
LoopName	The name of the loop that the data is required from. Must be placed in single quotes.
FieldName	The name of the field to be referenced. Must be placed in single quotes.
FieldIndex	The index of the field to be referenced. Index is zero based.
JSON Pointer expression	A JSON Pointer expression that identifies a specific value within the JSON document. Must be placed in single quotes. If the expression is empty, then the whole JSON document for that iteration is returned. For more information on JSON Pointer expression, refer to the following: https://tools.ietf.org/html/rfc6901
Example	
(These examples were completed using a Text Formatter module.)	
Information Entered	Information Returned
(Assumes the Data Section SQL Exists and a loop called "loop" has been set-up with the loop module to iterate over the datasection)	
I live in \$LOOPDATA('loop','City')\$	<p>The first iteration around the loop, the message section would resolve to I live in Seattle</p> <p>The second iteration I live in Tacoma</p> <p>The third iteration I live in Kirkland</p> <p>The fourth iteration I live in Redmond</p> <p>and finally the fifth iteration I live in Tacoma</p>

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

LOOPITERATION Dynamic Function

Dynamic Function	
LOOPITERATION	
Description	
The LOOPITERATION Dynamic Function returns the current iteration number of the specified loop. See the Loop module for more information on Loops. The first iteration round the loop will return 1, the second iteration will return 2, and so on.	
Syntax	
\$LOOPITERATION('LoopName')\$	
Parameter	Description
LoopName	The name of the loop that the iteration number is required for. Must be placed in single quotes.

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

LOWERCASE Dynamic Function

Dynamic Function	
LOWERCASE	
Description	
The LOWERCASE dynamic function renders data from the selected data section cell/message section in lower case letters.	
Syntax	
\$LOWERCASE('DataSectionName',FieldIndex,RowIndex)\$	
\$LOWERCASE('DataSectionName','FieldName',RowIndex)\$	
\$LOWERCASE('MessageSectionName')\$	
Parameter	Description
DataSectionName	The name of the Data Section in the EMF Process. Must be placed in single quotes.
FieldIndex	The index of the field to be referenced. Index is zero-based.
RowIndex	The index of the row to be referenced. Index is zero-based.
FieldName	The name of the field in the EMF Process.

	Must be placed in single quotes.
MessageSectionName	The name of the Message Section in the EMF Process. Must be placed in single quotes.
Examples (These examples were completed using a Text Formatter module.)	
Information Entered Assume the Data Section <i>Data</i> exists and it contains the following information: Column0 Column1 abc 1bc3	Information Returned
\$UPPERCASE('Data',0,0)\$	abc
\$UPPERCASE('Data',1,0)\$	1bc3
\$UPPERCASE('Data','Column1',0)\$	1bc3
Information Entered Assume the Message Section <i>Text</i> exists and it contains the following information: Test Data 1	Information Returned
\$UPPERCASE('Text')\$	test data 1
Additional Information FieldName and FieldIndex are alternative methods to reference the same column in a table of data. They cannot be used in the same line of syntax when defining the dynamic function. If the row is invalid (that is, there are not that many rows) an empty string is returned. If the data or message section does not exist, no information is returned.	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

MACHINENAME Dynamic Function

Dynamic Function	
MACHINENAME	
Description	
Outputs the EMF machine name the server is running on.	
Syntax	
\$MACHINENAME()\$	
Parameter	Description

None	N/A
<p>Example</p> <p>\$MACHINENAME()\$ would output "IAN-LAPTOP3" if that was the machine name. The name should match what the name of the machine defined under "System->Machines->..." that server instance is running on.</p>	

MATCH Dynamic Function

Dynamic Function	
MATCH	
Description	
<p>The MATCH Dynamic Function is a useful way of searching some text for particular pieces of data. It runs a regular expression (a powerful way of specifying a search pattern) against some text - returning the matched text. See http://www.regular-expressions.info for details on writing regular expressions.</p> <p>IMPORTANT: because dynamic functions use the backslash character to escape characters, if a regular expression contains a backslash character then it must be represented as a double backslash. The basic rule is, wherever your regular expression has a single backslash, replace it for two backslash characters.</p>	
Syntax	
\$MATCH('MessageSectionName', 'RegEx', ['Separator'], ['SubSequenceSeparator']))\$	
\$MATCH('DataSectionName', 'FieldName', RowIndex, 'RegEx', ['Separator'], ['SubSequenceSeparator']))\$	
\$MATCH('DataSectionName', FieldIndex, RowIndex, 'RegEx', ['Separator'], ['SubSequenceSeparator']))\$	
Parameter	Description
MessageSectionName	<p>The name of the message section that contains the text to match against.</p> <p>Must be placed in single quotes.</p>
RegEx	<p>The regular expression that should be used to perform the match. For the exact syntax supported by the regular expression processor in EMF please refer to https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html</p> <p>Must be placed in single quotes.</p>
Separator	<p>The separator characters that is placed between each match. This is optional, and if not present then only the first match will be returned.</p> <p>Must be placed in single quotes.</p>
SubSequenceSeparator	If the regular expression contains group captures, symbolised

	by brackets e.g. (.*?)\\.(.*?) then this is the separator characters used to separate each group returned. Must be placed in single quotes.
DataSectionName	The name of the Data Section that contains the text to match against. Must be placed in single quotes
FieldName	The name of the field in the Data Section to be referenced. Must be placed in single quotes.
FieldIndex	The index of the field to be referenced. Index is zero-based.
RowIndex	The index of the row to be referenced. Index is zero-based.

Examples

**Mess
age
secti
on
text**

In the examples the message section is assumed to be called 'Text'

The price is £15.95.	\$MATCH('Text', '.*£(\\d*?\\.\\d*?)\\.')\$	15.95
Item "Chair". Item "Door". Item "Table"	\$MATCH('Text', '.*?"(.*?)"', ',')\$	Chair,Door,Table

The price of item "Chair" is £15.95. The price of item "Door" is £55.95.	\$MATCH('Text', '.*?\"(.*)\".*?£(\\d*?\\.\\d*?)\\.\', '#', '-')\$	Chair-15.95##Door-55.95	
Additional Information			
In the RegEx, Separator and SubSequenceSeparator fields <code>\t</code> , <code>\r</code> , <code>\n</code> are replaced by TAB, CR and LF respectively. e.g. <code>\r\n</code> means CRLF.			

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

MESSAGE Dynamic Function

Dynamic Function	
MESSAGE	
Description	
Returns the message section name specified in the language specified (optional).	
Syntax	
\$MESSAGE('MessageSectionName', 'LanguageID')\$	
Parameter	Description
MessageSectionName	The name of the message section to be used. Must be placed in single quotes.
LanguageID	The language ID of the language for the message to be changed to. Must be placed in single quotes.
Examples	
(These examples were completed using a Text Formatter module.)	
Information Entered	Information Returned

(Assumes the Message Sections TextFormatter and POP3IN)	
Here is the content of the first text formatter module in this EMF Process: \$MESSAGE('TextFormatter')\$	Here is the content of the first text formatter module in this EMF Process: I live in \$DATA('SQL','City',0)\$ This is an Example from the DATA Dynamic Function help.
Here is the content of the email that fired this EMF Process: \$MESSAGE('POP3IN')\$	Here is the content of the email that fired this EMF Process: Testing POP3 Initiator Module
Additional Information	
The Language parameter is optional. If not defined, the EMF System default language will be selected.	
POP3IN	
Appears under the MESSAGE heading in the Dynamic Functions menu when a POP3IN initiator has been added to the EMF Process. It contains the message section of the email that has fired the EMF Process. (If applicable)	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

MESSAGEEXISTS Dynamic Function

Dynamic Function	
MESSAGEEXISTS	
Description	
Returns whether or not a message section exists. This dynamic function is mainly used in conditional link expressions to detect for the presence of a message section. Different links can then be followed if the message section existed, compared to if it didn't.	
Syntax	
\$MESSAGEEXISTS('MessageSectionName')\$	
Parameter	Description
MessageSectionName	The name of the message section to be look for. Must be placed in single quotes.
Return value	

Returns **true** if the message section exists, **false** if it doesn't .

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

PROCESSAPINAME Dynamic Function

Dynamic Function	
PROCESSAPINAME	
Description	
Outputs the API name of the running process.	
Syntax	
\$PROCESSAPINAME()\$	
Parameter	Description
None	N/A
Example	
\$PROCESSAPINAME()\$ would resolve to "Test_1393494672312" if the process running name was "Test_1393494672312".	

PROCESSBRANCHINSTANCEID Dynamic Function

Dynamic Function	
PROCESSBRANCHINSTANCEID	
Description	
Outputs the process branch instance ID (alertState.getBranchInstanceID()) of the running process.	
Syntax	
\$PROCESSBRANCHINSTANCEID()\$	
Parameter	Description
None	N/A
Example	
\$PROCESSBRANCHINSTANCEID()\$ would resolve to "413214" if the process running had a process branch instance id of "413214".	

PROCESSFOLDERPATH Dynamic Function

Dynamic Function	
PROCESSFOLDERPATH	
Description	
Outputs the folder structure of the running process.	
Syntax	
\$PROCESSFOLDERPATH()	
Parameter	Description
None	N/A
Example	
\$PROCESSFOLDERPATH()\$ would resolve to "/MyFolder1/Folder2" if the process running was called "Test" was in the appropriate folder structure.	

PROCESSID Dynamic Function

Dynamic Function	
PROCESSID	
Description	
Outputs the process ID (AlertProfileID) of the running process.	
Syntax	
\$PROCESSID()	
Parameter	Description
None	N/A
Example	
\$PROCESSID()\$ would resolve to "1004" if the process running had a process id of "1004".	

PROCESSINSTANCEID Dynamic Function

Dynamic Function	
PROCESSINSTANCEID	
Description	
Outputs the process instance ID (AlertInstanceID) of the running process.	
Syntax	
\$PROCESSINSTANCEID()	
Parameter	Description

None	N/A
Example \$PROCESSINSTANCEID()\$ would resolve to "13214" if the process running had a process instance id of "13214".	

PROCESSNAME Dynamic Function

Dynamic Function	
PROCESSNAME	
Description	
Outputs the name of the running process.	
Syntax	
\$PROCESSNAME()\$	
Parameter	Description
None	N/A
Example \$PROCESSNAME()\$ would resolve to "Test" if the process running was called "Test".	

RECIPIENT Dynamic Function

Dynamic Function	
RECIPIENT EMF Process	
Description	
<p>The RECIPIENT Dynamic Function returns a single specified property from a recipient's details. The parameter passed should be the property name as found within the appropriate Recipient Section in the EMF Process. If this isn't found, then the information will be retrieved from the database. Properties that are not available from the drop-down list of functions (e.g. alias values) can be included by adding the field name in single quotes.</p>	
Syntax	
\$RECIPIENT(Property)\$	
Parameter	Description
Property	<p>The database name of the property for which a value is required.</p> <p>Predefined values</p> <p>When using in conjunction with recipient sections created with an Aliased Recipients module, other fields can be referenced by the RECIPIENT dynamic functions using the field name enclosed in single</p>

	quotes. The field names can be established using Debug Mode to view the recipient section created.
Example	
(These examples were completed using a Text Formatter module.)	
Example	Information Returned
\$RECIPIENT(Email address)\$	EMFSupport@cdcsoftware.com
\$RECIPIENT ('UserProfile.EMailAddress')\$	EMFSupport@cdcsoftware.com
Additional Information	
If gaining recipient information from a table other than EMF System Recipients, placing the column name of the relevant data in single quotes allows the values held within to be brought back. For Example, from Aliased or Dynamic Recipients.	
Note: The RECIPIENT function differs from the RECIPTABLE function in that it only returns the first match found. RECIPTABLE can retrieve <i>multiple</i> values (e.g. where a single recipient might have multiple matches against an Alias value) and return all of these in the form of a table.	
RECIPIENT and RECIPTABLE are comparable to the TABLE and DATA functions, but they query a Recipient section instead of a Data section.	
Note: You cannot use the RECIPIENT dynamic function in a WHERE clause in a SQL module, because these are only evaluated in the output module.	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

RECIPIENTCOUNT Dynamic Function

Dynamic Function	
RECIPIENTCOUNT	
Description	
The RECIPIENTCOUNT Dynamic Function returns the number of recipients in one or more Recipient sections. If no parameters are passed, this function returns the total number of recipients in all recipient sections.	
Syntax	
\$RECIPIENTCOUNT(['recipientSectionName' or recipientSectionIndex])\$	
Parameter	Description
'recipientSectionName'	If a <i>quoted</i> parameter is passed, the number of recipients in the recipient section with the specified name is returned.

recipientSectionIndex	If a <i>number</i> is passed, the number of recipients in the recipient section with the specified index number is returned.
Example	
Information Entered	Information Returned
Recipient section 3 contains \$RECIPIENTCOUNT(2)\$ recipients out of a total of \$RECIPIENTCOUNT()\$ recipients.	Recipient section 3 contains 5 recipients out of a total of 15 recipients.
Additional Information	
This Dynamic Function is particularly suitable for use in the Scripting module.	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

RECIPIENTDATA Dynamic Function

Dynamic Function	
RECIPIENTDATA	
Description	
The RECIPIENTDATA Dynamic Function returns the value of data from the specified recipient section, column, and row.	
Syntax	
\$RECIPIENTDATA('recipientSectionName' or recipientSectionIndex, 'columnName' or columnIndex, rowIndex)\$	
Parameter	Description
'recipientSectionName' or recipientSectionIndex	Returns the value of data from the specified recipient section
'columnName' or columnIndex.	Returns the value of data from the specified column. If column names are used, then the standard predefined recipient values can be used to access standard recipient properties.
rowIndex	Returns the value of data from the specified row
Examples	
Information Entered	Information Returned
The third recipient in the only recipient section has the first name \$RECIPIENTDATA(0, 'UserProfile.FirstName', 2)\$.	The third recipient in the only recipient section has the first name John.
The tenth recipient in the recipient	The tenth recipient in the recipient section

section MySection has the first name
\$RECIPIENTDATA('MySection',
'UserProfile.FirstName', 9)\$.

MySection has the first name Daisy.

Additional Information

This Dynamic Function is particularly suitable for use in the [Scripting](#) module.

Multiple recipient sections in an EMF Process are not necessarily numbered in the order in which they were added to the EMF Process. Therefore, when there is more than one recipient section in the EMF Process, you should reference the sections by name rather than index.

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

RECIPSECTIONCOUNT Dynamic Function

Dynamic Function	
RECIPSECTIONCOUNT	
Description	
The RECIPSECTIONCOUNT Dynamic Function returns the number of recipient sections in the alert state at this point.	
Syntax	
\$RECIPSECTIONCOUNT()\$	
Parameter	Description
N/A	
Example	
Information Entered	Information Returned
The EMF Process contains \$RECIPSECTIONCOUNT()\$ recipient sections.	The EMF Process contains 3 recipient sections.
Additional Information	
This Dynamic Function is particularly suitable for use in the Scripting module.	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

RECIPTABLE Dynamic Function

Dynamic Function	
RECIPTABLE	
Description	
<p>RECIPTABLE is a per recipient dynamic function that returns a table of the data in the specified fields for a recipient in a Recipient Section. You can use RECIPTABLE to extract and present recipient information from any Recipient Sections, including those created by Aliased Recipients and Dynamic Recipients modules.</p> <p>The parameter you pass to RECIPTABLE is a string containing table formatting tags and references to the fields that you want to display. The field names are either the predefined Recipient Section properties, such as UserProfile.LastName, or additional fields included in the Recipient Section from a Data Section, for example when you use an Aliased Recipients module to generate a Recipient Section. It is strongly recommended that you always reference fields by name, not by column index as the order of the columns is not known in advance.</p>	
Syntax	
\$RECIPTABLE('FormatString')\$	
Parameter	Description
FormatString	<p>The format string containing references to the fields to be displayed and the format commands to be used to define how the table appears.</p> <p>Index is zero based.</p> <p>Predefined values for referencing by [name="----"]</p> <p>Must be placed in single quotes.</p> <p>When using in conjunction with recipient sections created with an Aliased Recipients module other fields can be referenced by the RECIPTABLE dynamic function. These field names can be established using Debug Mode to view the recipient section created.</p>
Examples	
(These examples were completed using a Text Formatter module.)	
Information Entered	Information Returned
(Assumes this Recipient Section exists)	
\$RECIPTABLE('\[index=0]\t\[index=1]\r\n')\$	2 User
\$RECIPTABLE('\	2 User

<code>[name="UserProfile.UserProfileID"]\t\</code> <code>[name="UserProfile.LastName"]\r\n')\$</code>	
(Assumes a Recipient Section exists with these additional data fields, added by an Aliased Recipients module)	
<code>\$RECIPTABLE (' \ [name="LOGIN"] \t\</code> <code>[name="SURNAME"] \r\n') \$</code>	<code>jsmith Smith</code>

Additional Information

Note: the RECIPTABLE function differs from the [RECIPIENT](#) function in that it can retrieve *multiple* values (e.g. where a single recipient might have multiple matches against an [Alias value](#)) and return all of these in the form of a table. The RECIPIENT function only returns the first match.

RECIPIENT and RECIPTABLE are comparable to the [TABLE](#) and [DATA](#) functions, but they query a Recipient section instead of a Data section.

\r This function will be replaced by a carriage return.

\n This function will be replaced by a linefeed.

\r\n This function will be replaced by a carriage return followed by a linefeed.

\t This function will be replaced by a tab character.

Comma's cannot be used as separators.

When referencing fields by the index it is important to remember the index is zero based. e.g. the first column is referenced by 0, the second by 1 and so on. The order in which the fields are placed into the recipient section cannot be pre-determined; thus it is recommended that fields are referenced by name.

When referencing fields by name it is important to remember to enclose the field name required in double quotes to distinguish what information should be taken as the field name.

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

REPLACE Dynamic Function

Dynamic Function
REPLACE
Description
The REPLACE Dynamic Function is a useful way of searching some text for particular pieces of data and replacing it with other values. It runs a regular expression (a powerful way of specifying a search pattern) against a message or data section - and replaces all matching instances with the specified replacement string. See http://www.regular-expressions.info for details on writing regular expressions.

IMPORTANT: because dynamic functions use the backslash character to escape characters, if a regular expression contains a backslash character then it must be represented as a double backslash. The basic rule is, wherever your regular expression has a single backslash, replace it for two backslash characters.

Syntax

```
$REPLACE('MessageSectionName','RegEx','ReplacementText',[ReplaceAllInstances])$
```

```
$REPLACE('DataSectionName','FieldName',RowIndex,'RegEx','ReplacementText',  
['SubSequenceSeparator'])$
```

```
$REPLACE('DataSectionName',FieldIndex,RowIndex,'RegEx','ReplacementText',  
[ReplaceAllInstances])$
```

Parameter	Description
MessageSectionName	The name of the message section that contains the text that values should be replaced in. Must be placed in single quotes.
RegEx	The regular expression that should be used to search for the terms to replace. For the exact syntax supported by the regular expression processor in EMF please refer to https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html Must be placed in single quotes.
ReplacementText	The text that should replace any matching expressions. Must be placed in single quotes.
SubSequenceSeparator	If the regular expression contains group captures, symbolised by brackets e.g. (.*)\.\.(.*) then this is the separator characters used to separate each group returned. Must be placed in single quotes.
ReplaceAllInstances	A '0' or '1'. If setting '1' is used, then all matching instances are replaced, otherwise only the first instance is replaced. The default value is '0' which does not replace all values. This value is optional and should not be quoted.
DataSectionName	The name of the Data Section that contains the text that values should be replaced in. Must be placed in single quotes
FieldName	The name of the field in the Data Section to be referenced. Must be placed in single quotes.
FieldIndex	The index of the field to be referenced. Index is zero-based.
RowIndex	The index of the row to be referenced. Index is zero-based.

Examples

Message section text In the examples the message section is assumed to be called 'Text'	Dynamic Function Entered	Information Returned
The price is £15.95.	<code>\$REPLACE('Text formatter', '£(\\d*?\\. \\d*?)\\.', '£1.99.')</code>	The price is £1.99.
Item "Chair". Item "Door". Item "Table"	<code>\$REPLACE('Text', "(.*)", '---')</code>	Item ---. Item "Door". Item "Table"
Item "Chair". Item "Door". Item "Table"	<code>\$REPLACE('Text', "(.*)", '---', 0)</code>	Item ---. Item "Door". Item "Table"
Item "Chair". Item "Door". Item "Table"	<code>\$REPLACE('Text', "(.*)", '---', 1)</code>	Item ---. Item ---. Item ---
Additional Information		
In the RegEx and ReplacementText fields <code>\t</code> , <code>\r</code> , <code>\n</code> are replaced by TAB, CR and LF respectively. e.g. <code>\r\n</code> means CRLF.		

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

REPOSITORYDATABASETYPE Dynamic Function

Dynamic Function	
REPOSITORYDATABASETYPE	
Description	
Outputs the EMF repository database type, either "Oracle" or "SqlServer".	
Syntax	
\$REPOSITORYDATABASETYPE()\$	
Parameter	Description
None	N/A
Example	
\$REPOSITORYDATABASETYPE()\$ would output "Oracle" if the repository database was running on an Oracle database server.	

REPOSITORYNAME Dynamic Function

Dynamic Function	
REPOSITORYNAME	
Description	
Outputs the EMF repository database name.	
Syntax	
\$REPOSITORYNAME()\$	
Parameter	Description
None	N/A
Example	
\$REPOSITORYNAME()\$ would output "EMF_63" if that was the EMF repository database name.	

RESOLVEENTITY Dynamic Function

Dynamic Function	
RESOLVEENTITY	
Description	
Outputs the name that a virtual entity resolves to at runtime. For more information on virtual entities, see Process Re-use with Process Interface and Call module .	

Syntax	
\$RESOLVEENTITY('entity type', 'virtual entity name')\$	
Parameter	Description
Entity type	<p>The name of the entity type, possible values are:</p> <ul style="list-style-type: none"> • JDBCDataSource • SAPDataSource • FTPService • ApteanSMSService • FileService • ExecutableService • SMTPService • JNDIService • HTTPService • SNMPService • RespondService • SAPSystemDataSource
Virtual entity name	The name of the virtual entity as defined in the start module.
Example \$RESOLVEENTITY('JDBCDataSource', 'database')\$ would resolve to "Repository" if the call module passed the "Repository" JDBC datasource to the process and mapped on to a virtual entity called "database" in the start module.	

ROWCOUNT Dynamic Function

Dynamic Function	
ROWCOUNT	
Description	
Returns the number of rows in the Data Section specified.	
Syntax	
\$ROWCOUNT('DataSectionName')\$	
Parameter	Description
DataSectionName	<p>The name of the data section to be used by ROWCOUNT.</p> <p>Must be placed in single quotes.</p>
Example (These examples were completed using a Text Formatter module.)	

Information Entered	Information Returned
(Assumes the Data Section SQL Exists)	
There are \$ROWCOUNT('SQL')\$ family address details in your address book	There are 5 family address details in your address book.
Additional Information	
N/A	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

SUBSTRING Dynamic Function

Dynamic Function	
SUBSTRING	
Description	
Returns the first n characters from the selected Data Section cell starting from the offset specified.	
Syntax	
\$SUBSTRING('DataSectionName','FieldName',RowIndex,StartPos,[NoOfCharacters])\$	
\$SUBSTRING('DataSectionName',FieldIndex,RowIndex,StartPos,[NoOfCharacters])\$	
\$SUBSTRING('MessageSectionName',StartPos,[NoOfCharacters])\$	
Parameter	Description
DataSectionName	The name of the Data Section in the EMF Process. Must be placed in single quotes.
MessageSectionName	The name of the Message Section in the EMF Process. Must be placed in single quotes.
FieldName	The name of the Field Name in the EMF Process. Must be placed in single quotes.
FieldIndex	The index of the field to be referenced. Index is zero based.
RowIndex	The index of the row to be referenced. Index is zero based.
StartPos	The initial position in the text to start returning characters. If the StartPos is past the length of the text, nothing will be returned. StartPos is zero based.
NoOfCharacters	Optional Parameter. This is the Number of characters to return. If omitted, the entire string starting from StartPos will be returned.

Example	
(These examples were completed using a Text Formatter module.)	
Information Entered (Assumes the Message Section MSG Exists) Assume MSG contains the following information: The quick brown fox jumps over the Lazy Dog	Information Returned
\$SUBSTRING('MSG',10)\$	brown fox jumps over the Lazy Dog
\$SUBSTRING('MSG',4,6)\$	quick (Space included at end of word)
\$SUBSTRING('MSG',0)\$	The quick brown fox jumps over the Lazy Dog
Information Entered (Assumes the Data Section SQL Exists) Assume SQL contains the following information: Column0 Column1 ABCDE 1234 FGHIJ @:Lki o 90()&* 998124	Information Returned
\$SUBSTRING('SQL','Column0',0,2)\$	CDE
\$SUBSTRING('SQL',0,0,2,1)\$	C
\$SUBSTRING('SQL',1,2,3,2)\$	12
\$SUBSTRING('SQL',0,0,0)\$ Equivalent to: \$DATA('SQL',0,0)\$	ABCDE
Additional Information	
<p>FieldName and FieldIndex are alternative methods of referencing the same column in a table of data. They cannot be used in the same line of syntax when defining the dynamic function.</p> <p>If the row is invalid (i.e. there aren't that many rows) an empty string is returned. If the data or message section does not exist, no information is returned.</p>	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

TABLE Dynamic Function

Dynamic Function	
TABLE	
Description	
Returns a table of data values as specified in the format string	
Syntax	
\$TABLE('DataSectionName','FormatString',FieldNames)\$	
Parameter	Description
DataSectionName	The name of a Data Section in the EMF Process. Must be placed in single quotes.
FormatString	The output format string Must be placed in single quotes.
FieldNames	A '0' or '1'. If setting '1' is used, the field names will be displayed using the same format as that is used for the data. The default value is '0' which does not display the field names.
Example	
(These examples were completed using a Text Formatter module.)	
Information Entered	Information Returned
(Assumes the Data Section SQL Exists)	
\$TABLE('SQL','\[index=0]\t\[index=1]\r\n',0)\$	View Table
\$TABLE('SQL','\[index=0]\t\[index=1]\t\[name="HouseholdName"]\r\n',1)\$	View Table
Additional Information	
Data in quoted fields must delimit backslashes with a backslash. e.g. \\	
Data in quoted fields must delimit quotes with a backslash. e.g. \'	
Data in quoted fields must delimit dollars with a backslash. e.g. \\$	
In formatting fields \t, \r, \n are replaced by TAB, CR and LF respectively. e.g. \r\n means CRLF.	
In the format string Index references by the column number which is zero based. e.g. \[Index=0] returns the values for the first column in the table.	
In the format string Name references by the column name. e.g. \[Name="FieldName"] returns the values for the column named FieldName in the table.	
White space within quoted parameters is replicated. e.g. \t can be used to denote a tab, or	

a tab character within quotes may be placed in the field.

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

TABLEHEADER Dynamic Function

Dynamic Function	
TABLEHEADER	
Description	
Returns a field names for the fields in a Data Section as specified in the format string.	
Syntax	
\$TABLEHEADER('DataSectionName', 'FormatString')\$	
Parameter	Description
DataSectionName	The name of the Data Section to be used. Must be placed in single quotes.
FormatString	The output format string. Must be placed in single quotes.
Example	
(These examples were completed using a Text Formatter module.)	
Information Entered	Information Returned
(Assumes the Data Section SQL Exists)	
\$TABLEHEADER('SQL','\[index=0]\t\[index=1]\r\n')\$	Address AddressID
\$TABLEHEADER('SQL','\[index=0]\t\[index=1]\t\[index=4]\r\n')\$	Address AddressID FaxNumber

Additional Information

In formatting fields `\t`, `\r`, `\n` are replaced by TAB, CR and LF respectively. e.g. `\r\n` means CRLF.

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

TIME Dynamic Function

Dynamic Functions	
TIME	
Description	
The TIME Dynamic Function returns the current system time, formatted according to each recipient's individual settings.	
Note: TIME and DATE are recipient-based, and are automatically modified according to each recipient's personal settings. If you want all recipients to receive identical data you should use the DATETIME function, which is system-based, and defaults to displaying the current system (e.g. local server) time.	
Syntax	
\$TIME()\$	
Parameter	Description
None	N/A
Example	
(These examples were completed using a Text Formatter module.)	
Information Entered	Information Returned
(Assumes the current Time is 2.16.58 pm.)	
\$TIME()\$	14:16:58
Additional Information	
N/A	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

UPPERCASE Dynamic Function

Dynamic Function	
UPPERCASE	
Description	
The UPPERCASE dynamic function renders data from the selected data section cell/message section in upper case letters.	
Syntax	
\$UPPERCASE('DataSectionName',FieldIndex,RowIndex)\$	
\$UPPERCASE('DataSectionName','FieldName',RowIndex)\$	
\$UPPERCASE('MessageSectionName')\$	
Parameter	Description
DataSectionName	The name of the Data Section in the EMF Process. Must be placed in single quotes.
FieldIndex	The index of the field to be referenced. Index is zero-based.
RowIndex	The index of the row to be referenced. Index is zero-based.
FieldName	The name of the field in the EMF Process. Must be placed in single quotes.
MessageSectionName	The name of the Message Section in the EMF Process. Must be placed in single quotes.
Examples	
(These examples were completed using a Text Formatter module.)	
Information Entered	Information Returned
Assume the Data Section <i>Data</i> exists and it contains the following information: Column0 Column1 abc 1bc3	
\$UPPERCASE('Data',0,0)\$	ABC
\$UPPERCASE('Data',1,0)\$	1BC3
\$UPPERCASE('Data','Column1',0)\$	1BC3
Information Entered	Information Returned
Assume the Message Section <i>Text</i> exists and it contains the following information: Test Data 1	
\$UPPERCASE('Text')\$	TEST DATA 1

Additional Information

FieldName and FieldIndex are alternative methods to reference the same column in a table of data. They cannot be used in the same line of syntax when defining the dynamic function.

If the row is invalid (that is, there are not that many rows) an empty string is returned.

If the data or message section does not exist, no information is returned.

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

URLENCODE Dynamic Function

Dynamic Functions	
URLENCODE	
Description	
The URLENCODE Dynamic Function encodes the URL text into a safe format that can be passed as part of a URL. You would not encode a whole URL (as this would escape out the : and / symbols), but typically just the parameter values.	
Syntax	
\$URLENCODE('MessageSectionName')\$	
\$URLENCODE('DataSectionName', 'FieldName', RowIndex, 'FieldName', RowIndex)\$	
\$URLENCODE('DataSectionName', FieldIndex, RowIndex)\$	
Parameter	Description
MessageSectionName	The name of the message section that contains the URL text to encode. Must be placed in single quotes.
DataSectionName	The name of the data section that contains the URL text to encode. Must be placed in single quotes.
FieldName	The name of the field in the Data Section to be referenced. Must be placed in single quotes.
FieldIndex	The index of the field to be referenced. Index is zero-based.
RowIndex	The index of the row to be referenced. Index is zero-based.
Example	
Information Entered	Information Returned
Assume the message section "Text" contains "Data Ñ":	

http://www.test.com/myPage?param1=\$URLDECODE('Text')\$	http://www.test.com/myPage?param1=Data%20%C3%91
---	---

Additional Information

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

URLDECODE Dynamic Function

Dynamic Functions	
URLDECODE	
Description	
The URLDECODE Dynamic Function decodes the URL text into its original format.	
Syntax	
\$URLDECODE('MessageSectionName')\$	
\$URLDECODE('DataSectionName', 'FieldName', RowIndex)\$	
\$URLDECODE ('DataSectionName', FieldIndex, RowIndex)\$	
Parameter	Description
MessageSectionName	The name of the message section that contains the URL text to decode. Must be placed in single quotes.
DataSectionName	The name of the data section that contains the URL text to decode. Must be placed in single quotes.
FieldName	The name of the field in the Data Section to be referenced. Must be placed in single quotes.
FieldIndex	The index of the field to be referenced. Index is zero-based.
RowIndex	The index of the row to be referenced. Index is zero-based.
Example	
Information Entered	Information Returned
Assume the message section "Text" contains http://www.test.com/myPage?param1=Data%20%C3%91	
\$URLDECODE('Text')\$	http://www.test.com/myPage?param1

	=Data Ñ
Additional Information	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

UUID Dynamic Function

Dynamic Function	
UUID	
Description	
Retrieves a type 4 (pseudo randomly generated) UUID (universally unique identifier).	
Syntax	
\$UUID()\$	
Parameter	Description
None	N/A
Example	
\$UUID()\$ would resolve to a string, such as f47ac10b-58cc-4372-a567-0e02b2c3d479.	

VERSION Dynamic Function

Dynamic Function	
VERSION	
Description	
Outputs the EMF server version number.	
Syntax	
\$VERSION()\$	
Parameter	Description
None	N/A
Example	
\$VERSION()\$ would resolve to "GA-6-3 49" on EMF 6.3.	

XMLESCAPE Dynamic Function

Dynamic Functions	
XMLESCAPE	
Description	
The XMLESCAPE Dynamic Function converts reserved characters that need to be escaped out of an XML document (that is < > " ' and &) into the form &#xx; (where xx is the numeric equivalent).	
Syntax	
\$XMLESCAPE('Text to escape')\$	
Parameter	Description
Text to escape.	The text that contains the characters to be escaped. Must be placed in single quotes.
Example	
Information Entered	Information Returned
\$XMLESCAPE('The characters at the end of the string will be escaped <>&"\'')\$	<p>The characters at the end of the string will be escaped &#60;&#62;&#38;&#34;&#39;</p> <p>Note that the backslash in this example escapes out the first of the two apostrophes, which would otherwise represent the end of the string.</p>
Additional Information	
Note: You cannot include characters with a value of less than 32 in the range of 55296 - 57344 and 65534-65535 (except 9, 10, and 13), whether escaped out or not, because these cannot exist in an XML document and so will cause the XMLESCAPE Dynamic Function to fail.	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

XMLESCAPE2 Dynamic Function

Dynamic Function
XMLESCAPE2
Description
The XMLESCAPE2 Dynamic Function converts reserved characters in data section cell/message that need to be escaped out of an XML document (i.e. < > " ' and &) into the form &#xx; (where xx is the numeric equivalent).
Syntax

\$XMLEScape2('DataSectionName',FieldIndex,RowIndex)\$	
\$XMLEScape2('DataSectionName','FieldName',RowIndex)\$	
\$XMLEScape2('MessageSectionName')\$	
Parameter	Description
DataSectionName	The name of the Data Section in the EMF Process. Must be placed in single quotes.
FieldIndex	The index of the field to be referenced. Index is zero-based.
RowIndex	The index of the row to be referenced. Index is zero-based.
FieldName	The name of the field in the Data Section to be referenced. Must be placed in single quotes.
MessageSectionName	The name of the Message Section in the EMF Process. Must be placed in single quotes.
Examples	
(These examples were completed using a Text Formatter module.)	
Information Entered	Information Returned
Assume the Data Section <i>Data</i> exists and it contains the following information: Column0 Column1 > @	
\$XMLEScape2('Data',0,0)\$	>
\$XMLEScape2('Data',1,0)\$	@
Information Entered	Information Returned
Assume the Message Section <i>Text</i> exists and it contains the following information: &	
\$XMLEScape2('Text')\$	&
Additional Information	
Note: You cannot include characters with a value of less than 32 in the range of 55296 - 57344 and 65534-65535 (except 9, 10, and 13), whether escaped out or not, because these cannot exist in an XML document and so will cause the XMLEScape2 Dynamic Function to fail.	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

XMLUNESCAPE Dynamic Function

Dynamic Functions	
XMLUNESCAPE	
Description	
The XMLUNESCAPE dynamic function converts characters that have been XML escaped back to their original forms. Any text passed in will be unescaped. For example, if the text contains >, then it would be replaced with the greater than symbol ">".	
Syntax	
\$XMLUNESCAPE('Test to unescape')\$	
Parameter	Description
Text to unescape.	The text that contains the characters to be unescaped. Must be placed in single quotes.
Example	
Information Entered	Information Returned
\$XMLUNESCAPE('>')\$	>
Additional Information	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

XMLUNESCAPE2 Dynamic Function

Dynamic Function	
XMLUNESCAPE2	
Description	
The XMLUNESCAPE2 dynamic function converts characters that have been XML escaped out back to their original forms. Any text in the data section cell/message section will be unescaped. For example, if the text contains >, then it will be replaced with the greater than symbol ">".	
Syntax	
\$XMLUNESCAPE2('DataSectionName',FieldIndex,RowIndex)\$	
\$XMLUNESCAPE2('DataSectionName','FieldName',RowIndex)\$	
\$XMLUNESCAPE2('MessageSectionName')\$	
Parameter	Description
DataSectionName	The name of the Data Section in the EMF

	Process. Must be placed in single quotes.
FieldIndex	The index of the field to be referenced. Index is zero-based.
RowIndex	The index of the row to be referenced. Index is zero-based.
FieldName	The name of the field in the Data Section to be referenced. Must be placed in single quotes.
MessageSectionName	The name of the Message Section in the EMF Process. Must be placed in single quotes.

Examples

(These examples were completed using a [Text Formatter](#) module.)

Information Entered	Information Returned
Assume the Data Section Data exists and it contains the following information: Column0 Column1 > @;	

\$XMLUNESCAPE2('Data',0,0)\$	>
\$XMLUNESCAPE2('Data',1,0)\$	@;

Information Entered	Information Returned
Assume the Message Section Text exists and it contains the following information: >	
\$XMLUNESCAPE2('Text')\$	>

Additional Information

FieldName and FieldIndex are alternative methods to reference the same column in a table of data. They cannot be used in the same line of syntax when defining the dynamic function.

If the row is invalid (that is, there are not that many rows) an empty string is returned.

If the data or message section does not exist, no information is returned.

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

XPATH Dynamic Function

Dynamic Function
XPATH

Description	
Parses the XML within a message or data section using the supplied Xpath and returns a string representing an Xpath node. Optionally, also associates a namespace prefix with a namespace used in the XML document.	
Syntax	
\$XPATH('MessageSectionName' , 'Xpath' , LanguageID)\$	
\$XPATH('MessageSectionName' , 'Xpath')\$	
\$XPATH('MessageSectionName' , 'XPath' , LanguageID, 'Namespace')\$	
\$XPATH(SectionType,'DataSectionName', 'FieldName', RowIndex, 'XPath', ['Namespace'])\$	
\$XPATH(SectionType,'DataSectionName', FieldIndex, RowIndex, 'XPath', ['Namespace'])\$	
Parameter	Description
MessageSectionName	The name of the Message Section containing XML to be parsed. Must be placed in single quotes.
Xpath	Xpath to use to navigate the XML. Must be placed in single quotes.
LanguageID	Optional parameter. The language ID of the language for the message to be changed to.
Namespace	Optional parameter. Must be placed in single quotes. Use double quotes within the single quotes for the namespace URI (s). For example, 'xmlns:d="http://schemas.company.com/docs"'. If you do not specify this parameter, it is assumed that any prefixes being used in the XPath expression are present in the XML in the message section.
DataSectionName	The name of the Data Section containing XML to be parsed. Must be placed in single quotes
FieldName	The name of the field in the Data Section to be referenced. Must be placed in single quotes.
FieldIndex	The index of the field to be referenced. Index is zero-based.
RowIndex	The index of the row to be referenced. Index is zero-based.
SectionType	The SectionType at the start indicates a "message section" (0) or a "data section" (1).

	<p>If the syntax is \$XPath(0, 'gggg', 'ggg', 1, 'ggg')\$, it indicates a message section.</p> <p>If the syntax is \$XPath(1, 'gggg', 'ggg', 1, 'ggg')\$, it indicates a data section</p> <p>SectionType is optional. If the parameter is not present, then it is assumed that the section is a "message section", not a "data section".</p> <p>For a data section expression, there must be a 1 at the start.</p>
<p style="text-align: center;">Example</p> <p style="text-align: center;">(These examples were completed using a Text Formatter module.)</p>	
<p>Information Entered</p> <p>(Assumes the message section MSG exists)</p> <p>The message section MSG contains the XML:</p> <pre><?xml version="1.0"?> <DOCELEM> <AAA ddd="d" ccc="c"> <CCC>C</CCC> <DDD>D</DDD> <CCC>C</CCC> </AAA> <BBB ccc="c" ddd="d" eee="e"> <CCC>C</CCC> <DDD>D</DDD> <EEE>E</EEE> </BBB> </DOCELEM></pre>	<p>Information Returned</p>
\$XPath('MSG','//AAA/DDD')\$	D
\$XPath('MSG','/DOCELEM/BBB/CCC')\$	C
\$XPath('MSG','//AAA/@ddd')\$	d
<p>Assuming a message section XDOCDATA exists with the following XML:</p> <pre><?xml version="1.0"?> <DOCELEM xmlns:d="http://www.mycompany.com /namespace/d" xmlns:dfs="http://www.mycompany.com/names</pre>	

<pre>pace/dfs"> <dfs:myFields><dfs:dataFields> <d:Customers Fax="555-1234"/> </dfs:dataFields></dfs:myFields> </DOCELEM></pre>	
<pre>\$XPATH('XDOCDATA', '/DOCELEM/abc:myFields/abc:dataFields /a:Customers/@Fax',1, 'xmlns:a="http://www.mycompany.com/ namespace/d" xmlns:abc= "http://www.mycompany.com/namespace/dfs")\$</pre>	555-1234
Additional Information	
<p>The message section must contain XML in the specified language.</p> <p>For more information on Xpath, refer to the following: Xpath specification: http://www.w3.org/TR/xpath Xpath tutorial: http://www.zvon.org/xxl/XPathTutorial/General/examples.html Xpath demonstration: http://www.vbxml.com/xpathvisualizer/default.asp</p>	

[List of Dynamic Functions](#)

[Using Dynamic Functions in EMF](#)

Character Description

d	Displays the day of month in the number format (1 - 31).
D	Displays the day of the year in the number format (1 - 365).
dd	Displays the day of month in the number format with a minimum of two digits, i.e., a leading zero will be appended in the case of a single-digit date (01 - 31).
DD	Displays the day of the year in the number format, with a minimum of two digits (01 - 365).
DDD	Displays the day of the year in the number format, with a minimum of three digits (001 - 365).
dddd, EEEE	Displays the day in week, in full text (Sunday - Saturday).
E	Displays the day in week, as an abbreviation (Sun - Sat).
w	Displays the week of the year as a number. Example: \$DATETIME('w','w',0,'now')\$
	Note: Leading zeros will be appended if the number of characters used is more than one. For example, www returns the value 009.
W	Displays the week of the month as a number.

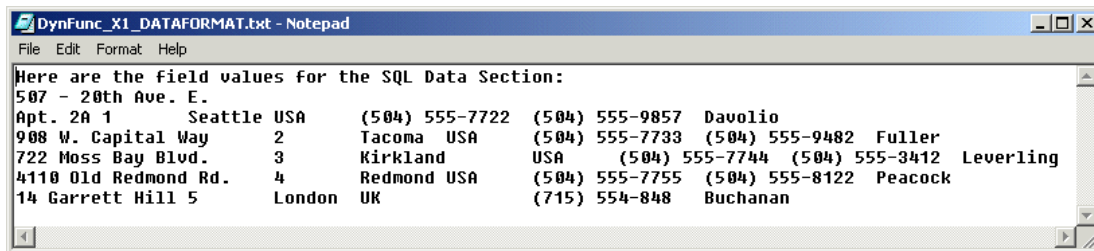
	Note: Leading zeros will be appended if the number of characters used is more than one.
F	Displays the number of occurrence of the day of week in the month. For example, 2 for second Wednesday in June. Note: Leading zeros will be appended if the number of characters used is more than one.
M	Displays the month as a number without a leading zero (1 - 12).
MM, mm	Displays the month as a number with a leading zero (01 - 12) in the case of a single-digit.
MMM, mmm	Displays the month as an abbreviation (Jan - Dec).
MMMM, mmmm	Displays the month in full text (January - December).
y	Displays the year in the two-digit (00 - 99) number format respectively.
yyyy	Displays the year in the four-digit number format respectively.
K	Displays the hour in the day, in the 12-hour format (0 - 11). Note: For numeric values, leading zeros will be appended if the number of characters used is more than one.
k	Displays the hour in the day, in the 24 hour format (1 - 24).
h	Displays the hour in the day, in the 12-hour format (1 - 12).
H	Displays the hour in the day in the 24 hour format (0 - 23).
m	Displays the minute (0 - 59).
nn	Displays the minute (00 - 59) with a leading zero in the case of single-digit.
s	Displays the second in the minute (0 - 59).
S	Displays the second in the minute (0 - 59).
AM/PM, a	Displays an uppercase AM or PM, based on the 12-hour clock. Example: \$DATETIME('dddd-mm-yy hh:nn:ss a','s',0,'now')\$.
G	This is the era designator, and displays AD in the output.
z	Displays the general time zone as an abbreviation. For example, PST for Pacific Standard Time.
zzzz	Displays the general time zone name. For example, Pacific Standard Time.
Z	Displays the time zone as per the RFC 822 standard.

Character	Description
-----------	-------------

(Not Case-sensitive)

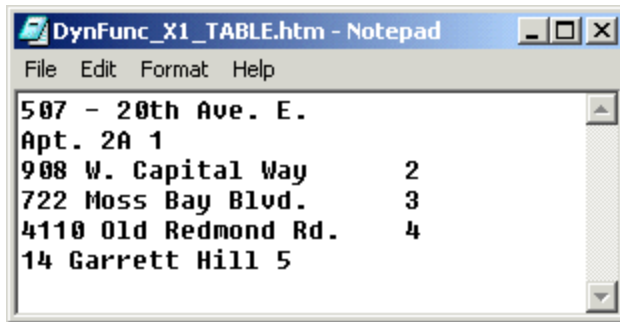
YYYY	Year
Q	Quarter
M	Month
Y	Day of year
D	Day of month
W	Day of week
WW	Week of year
H	Hour
N	Minute
S	Second

Note: The default interval value is Year if no interval character is specified.



Recipient Section 0					
UserProfile.UserID	UserProfile.LastName	UserProfile.FirstName	UserProfile.UserName	UserProfile.Salutation	UserProfile.TelephoneNumber
2	User	Server	Server	Hello	
1	User	Administrator	Admin	Hello	1234
3	User	Script	Script	Hello	

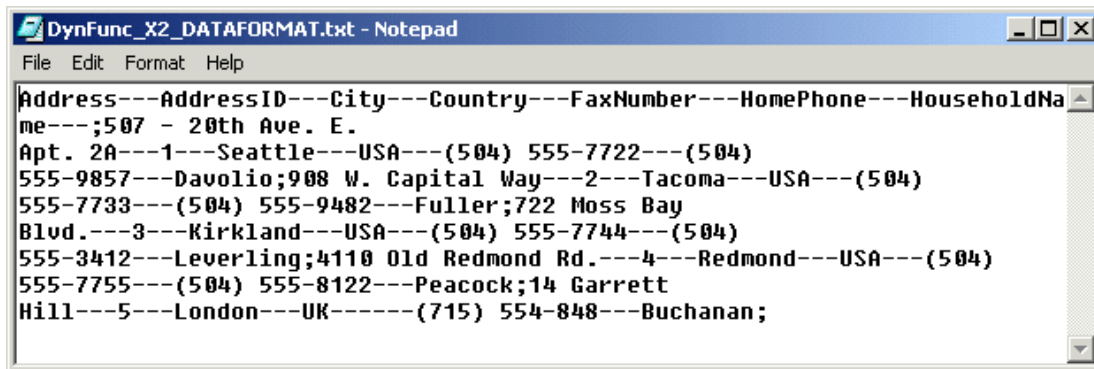
Results - 5 rows						
Address	AddressID	City	Country	FaxNumber	HomePhone	HouseholdName
507 - 20th Ave. E.	1	Seattle	USA	(504) 555-7722	(504) 555-9857	Davolio
908 W. Capital Way	2	Tacoma	USA	(504) 555-7733	(504) 555-9482	Fuller
722 Moss Bay Blvd.	3	Kirkland	USA	(504) 555-7744	(504) 555-3412	Leverling
4110 Old Redmond Rd.	4	Redmond	USA	(504) 555-7755	(504) 555-8122	Peacock
14 Garrett Hill	5	London	UK		(715) 554-848	Buchanan



```

DynFunc_X1_TABLE.htm - Notepad
File Edit Format Help
507 - 20th Ave. E.
Apt. 2A 1
908 W. Capital Way      2
722 Moss Bay Blvd.     3
4110 Old Redmond Rd.   4
14 Garrett Hill 5

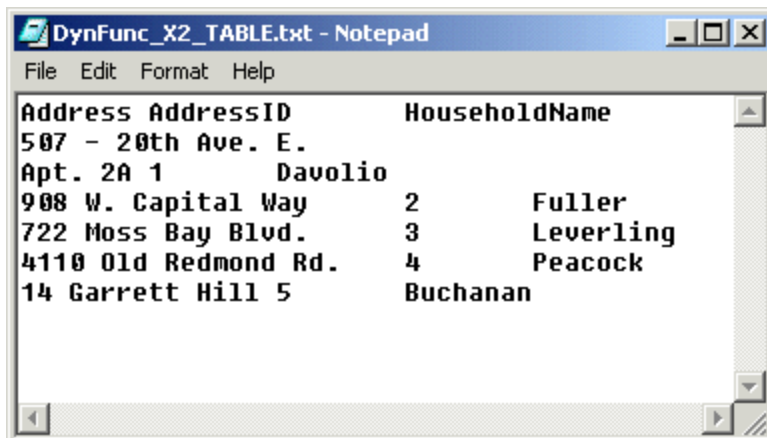
```



```

DynFunc_X2_DATAFORMAT.txt - Notepad
File Edit Format Help
Address---AddressID---City---Country---FaxNumber---HomePhone---HouseholdName---;507 - 20th Ave. E.
Apt. 2A---1---Seattle---USA---(504) 555-7722---(504)
555-9857---Davolio;908 W. Capital Way---2---Tacoma---USA---(504)
555-7733---(504) 555-9482---Fuller;722 Moss Bay
Blvd.---3---Kirkland---USA---(504) 555-7744---(504)
555-3412---Leverling;4110 Old Redmond Rd.---4---Redmond---USA---(504)
555-7755---(504) 555-8122---Peacock;14 Garrett
Hill---5---London---UK------(715) 554-848---Buchanan;

```



```

DynFunc_X2_TABLE.txt - Notepad
File Edit Format Help
Address AddressID      HouseholdName
507 - 20th Ave. E.
Apt. 2A 1      Davolio
908 W. Capital Way      2      Fuller
722 Moss Bay Blvd.     3      Leverling
4110 Old Redmond Rd.   4      Peacock
14 Garrett Hill 5      Buchanan

```

Example Output

HTMLTABLE Example (Including Field Names):

Address	AddressID	City	Country	FaxNumber	HomePhone	HouseholdName
14 Garrett Hill	5	London	UK		(715) 554-848	Buchanan
4110 Old Redmond Rd.	4	Redmond	USA	(504) 555-7755	(504) 555-8122	Peacock
507 - 20th Ave. E. Apt. 2A	1	Seattle	USA	(504) 555-7722	(504) 555-9857	Davolio
722 Moss Bay Blvd.	3	Kirkland	USA	(504) 555-7744	(504) 555-3412	Leverling
908 W. Capital Way	2	Tacoma	USA	(504) 555-7733	(504) 555-9482	Fuller

Example Output

HTMLTABLE Example (Excluding Field Names):

14 Garrett Hill	5	London	UK		(715) 554-848	Buchanan
4110 Old Redmond Rd.	4	Redmond	USA	(504) 555-7755	(504) 555-8122	Peacock
507 - 20th Ave. E. Apt. 2A	1	Seattle	USA	(504) 555-7722	(504) 555-9857	Davolio
722 Moss Bay Blvd.	3	Kirkland	USA	(504) 555-7744	(504) 555-3412	Leverling
908 W. Capital Way	2	Tacoma	USA	(504) 555-7733	(504) 555-9482	Fuller

Example Output

HTMLTABLE Example (Including Field Names):

Address	AddressID	City	Country	FaxNumber	HomePhone	HouseholdName
14 Garrett Hill	5	London	UK		(715) 554-848	Buchanan
4110 Old Redmond Rd.	4	Redmond	USA	(504) 555-7755	(504) 555-8122	Peacock
507 - 20th Ave. E. Apt. 2A	1	Seattle	USA	(504) 555-7722	(504) 555-9857	Davolio
722 Moss Bay Blvd.	3	Kirkland	USA	(504) 555-7744	(504) 555-3412	Leverling
908 W. Capital Way	2	Tacoma	USA	(504) 555-7733	(504) 555-9482	Fuller

Example Output

HTMLTABLE Example (Including Field Names):

Address	AddressID	City	Country	FaxNumber	HomePhone	HouseholdName
14 Garrett Hill	5	London	UK		(715) 554-848	Buchanan
4110 Old Redmond Rd.	4	Redmond	USA	(504) 555-7755	(504) 555-8122	Peacock
507 - 20th Ave. E. Apt. 2A	1	Seattle	USA	(504) 555-7722	(504) 555-9857	Davolio
722 Moss Bay Blvd.	3	Kirkland	USA	(504) 555-7744	(504) 555-3412	Leverling
908 W. Capital Way	2	Tacoma	USA	(504) 555-7733	(504) 555-9482	Fuller

Example Output

HTMLTABLE Example (Including Field Names):

Address	AddressID	City	Country	FaxNumber	HomePhone	HouseholdName
14 Garrett Hill	5	London	UK		(715) 554-848	Buchanan
4110 Old Redmond Rd.	4	Redmond	USA	(504) 555-7755	(504) 555-8122	Peacock
507 - 20th Ave. E. Apt. 2A	1	Seattle	USA	(504) 555-7722	(504) 555-9857	Davolio
722 Moss Bay Blvd.	3	Kirkland	USA	(504) 555-7744	(504) 555-3412	Leverling
908 W. Capital Way	2	Tacoma	USA	(504) 555-7733	(504) 555-9482	Fuller

Example Output

HTMLTABLE Example (Including Field Names):

Address	AddressID	City	Country	FaxNumber	HomePhone	HouseholdName
14 Garrett Hill	5	London	UK		(715) 554-848	Buchanan
4110 Old Redmond Rd.	4	Redmond	USA	(504) 555-7755	(504) 555-8122	Peacock
507 - 20th Ave. E. Apt. 2A	1	Seattle	USA	(504) 555-7722	(504) 555-9857	Davolio
722 Moss Bay Blvd.	3	Kirkland	USA	(504) 555-7744	(504) 555-3412	Leverling
908 W. Capital Way	2	Tacoma	USA	(504) 555-7733	(504) 555-9482	Fuller

Example Output

HTMLTABLE Example (reordered fields with new headings):

Surname Name	Phone Number
Buchanan	(715) 554-848
Peacock	(504) 555-8122
Davolio	(504) 555-9857
Leverling	(504) 555-3412
Fuller	(504) 555-9482

Example Output

HTMLTABLE Example (reordered fields with new headings):

Surname Name	Phone Number
Buchanan	(715) 554-848
Peacock	(504) 555-8122
Davolio	(504) 555-9857
Leverling	(504) 555-3412
Fuller	(504) 555-9482

Display text (only for use with RECIPIENT)	Fully qualified field name (for use with RECIPIENT, RECIPTABLE, LOOPDATA and RECIPIENTDATA)	Description
Address line 1	UserAddress.AddressLine1	The first line of the recipient's address.
Address line 2	UserAddress.AddressLine2	The second line of the recipient's address.
City	UserAddress.AddressCity	The recipient's city.
Country	UserAddress.Country	The recipient's country.
Date format	Language.DefaultDateFormat	The recipient's preferred date format (for example, dd/mm/yyyy).
Department	UserProfile.Department	Name of the department in the organization that the recipient is associated with.

Email address	DeliveryAddress.EMailAddress	The recipient's email address.
Fax number	DeliveryAddress.FaxNumber	The recipient's fax number, including the international dialing code.
First name	UserProfile.FirstName	The recipient's first name.
HTTP URL	HTTP.URL	The URL to which HTTP requests should be sent.
IP address	DeliveryAddress.IPAddress	IP address or machine name for the recipient's computer.
Language	UserProfile.LanguageID	The recipient's preferred language.
Last name	UserProfile.LastName	The recipient's surname.
Organization	UserProfile.Organisation	Name of the company the recipient is associated with.
Pager country code	DeliveryAddress.PgrCountryCode	International dialing code for the recipient's pager.
Pager number	DeliveryAddress.PagerNumber	The recipient's pager number.
Pager provider	PagerProvider.Name	Name of the provider for the recipient's pager.
Queue lookup name	ExtQueue.LookUpName	The lookup name of the recipient's logical queue.
Queue provider name	ExtQProvider.FactoryName	The name of the recipient's queue provider.
Salutation	UserProfile.Salutation	How the recipient prefers to be addressed. For example, Hello. Dear.
SMS country code	DeliveryAddress.SMSCountryCode	International dialing code for the recipient's cellphone.
SMS number	DeliveryAddress.SMSNumber	The recipient's cellphone number.
SMS provider	SMSProvider.Name	Name of the provider for the recipient's cellphone.
State	UserAddress.AddressState	The recipient's state or

		county.
Telephone number	UserProfile.TelephoneNumber	The recipient's telephone number, including international dialling code.
Time format	Language.DefaultTimeFormat	The recipient's preferred date format (for example, dd/mm/yyyy).
Title	UserProfile.Title	(for example, Mr, Mrs, Ms).
UMT offset	UserProfile.UMTOffsetMins	The number of minutes that the recipient's time zone is earlier than (-) or later than (+) UMT.
User name	UserProfile.UserName	The name the recipient enters when accessing the EMF system.
User profile ID	UserProfile.UserProfileID	The ID of the recipient's profile.
Voice country code	DeliveryAddress.VoiceCtryCode	The country code of the device specified for Voice output.
Voice number	DeliveryAddress2.VoiceNumber	The number of the device specified for Voice output.
WAP address	DeliveryAddress2.WAPAddress	The address of the device specified for WAP Push output.
WAP address type	WAPAddressType.Name	The address type of the device specified for WAP Push output.
Zip code	UserAddress.AddressZipCode	The recipient's zip code or post code.

Nesting Dynamic Functions

You can nest dynamic functions inside each other in an EMF Process, so that if the result of a dynamic function contains a further dynamic function, this will also be resolved at the same time. Nesting allows you to resolve dynamic functions that are:

- Within dynamic functions.
- Created as a result of evaluating other dynamic functions.

You can resolve functions down to a preset recursive limit in a nested hierarchy - the default depth is 20 (you can change this by modifying the value of the DF_RECURSION_DEPTH server property in the EMF.properties file). This is in order to prevent the generation of a set of dynamic functions that will never finish processing, i.e. where the result of one function results in another, which is the same as the first. This is demonstrated in [example 1a](#).

An example of using nested dynamic functions

A data section may specify, in the first column of each row, the column that contains the desired data. In the following example, we are examining row 5 of data section SQL, and column 0 contains the name of the column in which the data is stored:

```
$DATA('SQL', $DATA('SQL', 0, 5)$, 5)$
```

The result of a dynamic function may also contain further dynamic functions to be resolved. For example, in the example above, if the inner function returned:

```
$DATA('SQL', 1, 5)$
```

This would be resolved before the outer function could be completed.

Note: You can only nest dynamic functions within the body of other dynamic functions.

Important notes about nesting dynamic functions

- Normally, when you are editing text that accepts dynamic functions, some pre-processing work is done when saving to and reading from the database as follows:
 - If the text within a RECIPIENT function that specifies a recipient or delivery property is *not* included within quotation marks, it is converted to a delivery property ID.
 - If the text *is* included within quotation marks, it is left as it is, and assumed to be a fully qualified field name.

However, this functionality may not work as intended within nested functions. It is therefore important that when using nested dynamic functions, you must **either** specify the delivery property ID **or** fully qualify the field name.

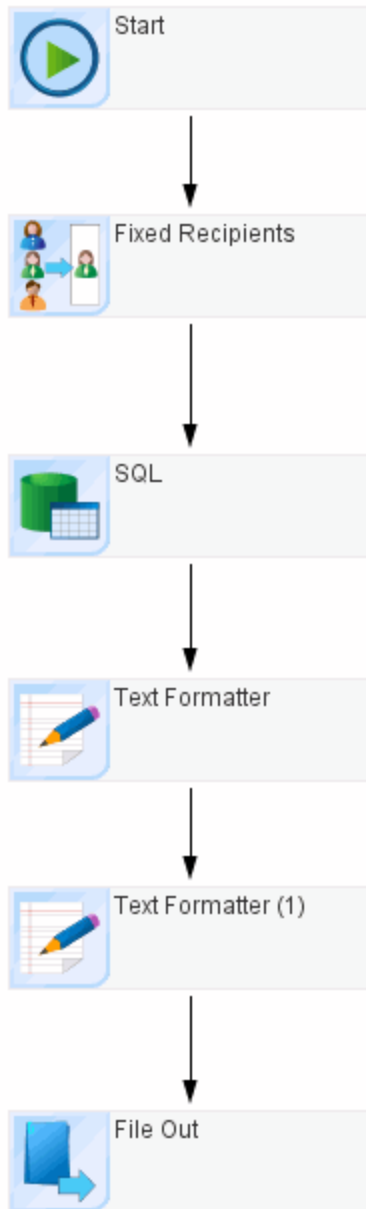
- Also, you are normally warned against using unavailable message/data/recipient sections, and of not using the same when they are available in the EMF Process builder. However, if you want to dynamically select a section from the result of a dynamic function, the builder cannot be aware of this and so cannot warn you.
- At present, the dynamic function parser within the EMF Process builder cannot parse nested functions and will not be able to detect the use of sections when used within nested functions.

[Examples of Using Nested Dynamic Functions](#)

[Dynamic Functions in EMF Processes](#)

Examples of Using Nested Dynamic Functions

Example 1

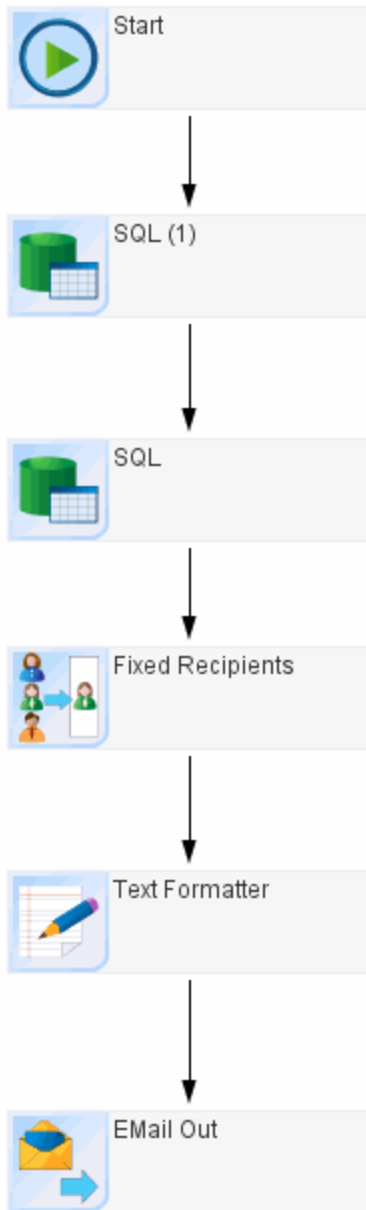


This example creates a recipient section with one recipient, a data section containing rows from the UserProfile table and two text formatters.

The text contained in the first text formatter is:

The resolved data formatting function:
\$DATAFORMAT('SQL','\t','\r\n',0)\$
\$RECIPIENT(First name)\$

Example 2



This example creates two data sections, and a single recipient section containing one recipient. The text formatter creates a message section containing the following:

Contents of the specified data section (\$DATA('SQL1',1,0)\$) are:
 \$DATAFORMAT('\$DATA('SQL1',1,0)\$','\\t','\\r\\n',0)\$

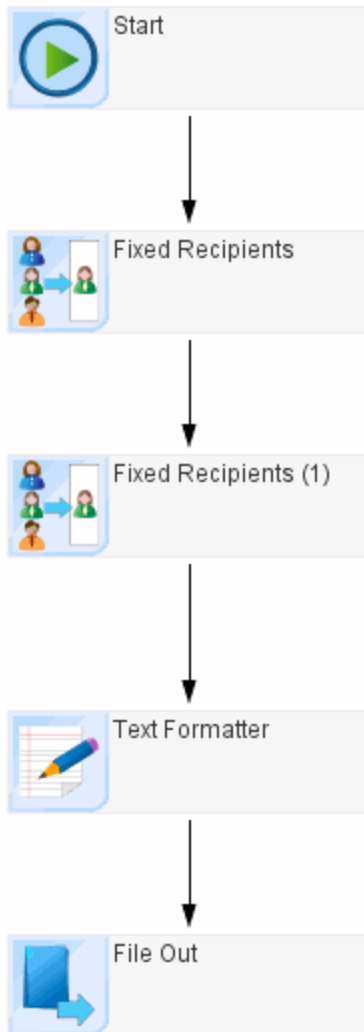
This will select the data section to use in the DATAFORMAT function from the result of the function \$DATA('SQL1',1,0)\$ - i.e. the value of the second column on the first row of data section SQL1. This contains the text SQL. Therefore the result of this function is an email containing the following:

Contents of the specified data section (SQL) are:

```
1
2
3
1001  0  0      jsmith@company.com
```

This is the content of the data section created by the SQL module SQL (1).

Example 3



This EMF Process creates two recipient sections (**Note:** 'Append to existing recipients' has been selected on the Advanced tab of Fixed recipients (2) in order to prevent the first being overridden by the second) with one (the same) recipient in each. The text formatter contains the following:

There are \$RECIPISECTIONCOUNT()\$ recipient sections.
 There are \$RECIPIENTCOUNT()\$ recipients in total.
 There are \$RECIPIENTCOUNT(0)\$ recipients in the first recipient section.
 There are \$RECIPIENTCOUNT(1)\$ recipients in the second recipient section.
 The last name of the first recipient is \$RECIPIENTDATA(0, 'UserProfile.LastName', 0)\$.
 The last name of the second recipient is \$RECIPIENTDATA(1, 'UserProfile.LastName', 0)\$.

This demonstrates the use of the [RECIPIENTCOUNT](#), [RECIPIENTDATA](#) and [RECIPISECTIONCOUNT](#) dynamic functions by returning a count of the number of recipient sections (should be 2), the number of recipients in total (again, should be 2) and the

number of recipients in each section (1 in each). It then uses the RECIPIENTDATA function to return the last name of the single recipient in each section.

The result of running this EMF Process is the creation of a file containing the following:

There are 2 recipient sections.
There are 2 recipients in total.
There are 1 recipients in the first recipient section.
There are 1 recipients in the second recipient section.
The last name of the first recipient is Smith.
The last name of the second recipient is Smith.

[Nesting Dynamic Functions](#)

[Dynamic Functions in EMF Processes](#)

Java API Guide

EMF can be embedded in your own applications, using the EMF application programming interfaces (APIs).

Using the Java APIs you can write Java code to do any of the following:

- Queue EMF Processes on the EMF Server over RMI (EMF Runtime API)
- Queue EMF Processes on the EMF Server using SOAP and the EMF Runtime API (SOAP API)
- Create and change entities in the EMF Repository. For example, create a recipient, update SMTP service settings (EMF Entity API)

You can also write JavaScript code in the Scripting module to further process the information contained within an EMF Process ([Scripting API](#)).

To queue EMF Processes with the EMF Runtime API, the user must have Create rights in Operators.

Where to find more information

Details for using the APIs, including prerequisites and configuration information:

- [API Documentation](#)

For information on how to use Entity APIs, refer [EMF Entity API](#).

Click on the **Description** link at the top of these pages to navigate quickly to the configuration information.

For information on configuring EMF for SOAP, refer to readme.htm in the SOAP directory on your EMF Installation media. Alternatively, view the SOAP help file, soap.chm, located in the Bin\Help directory of your EMF Administrator installation.

For Java examples of using the EMF Runtime_API and JavaScript examples of using Scripting API, see [Java API Examples](#)

API Documentation

- [Click here to access API Documentation](#)
- [Click here to access Entity API Documentation](#)

EMF Entity API

The EMF Entity API is used to edit items in the EMF repository. It provides a mechanism, so that settings can be changed programmatically without needing to manually do it through the EMF UI. Using the EMF Entity APIs, it is currently possible to:

- Perform CRUD operations on all EMF service components
- Perform CRUD operations on key stores
- Perform CRUD operations on Data Sources, i.e., JDBC connections & SAP Connections
- Perform CRUD operations on Recipients/Groups/Aliases
- Perform CRUD operations on SAP Domains
- Perform read/update operations on POP3/IMAP/HTTP listeners
- Perform limited operations on Process entities:
 - Change the Active/Hold status of a process
 - Change the logging level of a process
 - Change the schedule of the process runs

How to use the EMF Entity APIs

The EMF entity API can either be called from a Java program or from the scripting module to create/read/update/delete EMF entities such as services/recipient, etc. The general steps to using the API are:

1. Obtain a Session object that gives you access to the Repository.
2. Create a data access object (DAO) for the entity type you want to perform operations on (e.g. a JDBC Data Source).
3. Use the DAO to create a new entity, find an existing entity or remove an existing entity.
4. If creating/updating an entity – make the required changes to the entity, and then use the DAO to save the changes.
5. If using the session from a Java program, close the session when finished with. If using the default session from the Script module, do NOT close the session (or subsequent operations will fail).

Pre-requisites for including the API from a Java Program

The `<EMF installation directory>/entity_api` directory contains the `EmfEntityApi.jar` that needs to be included in any Java program to use the API. Also all JAR files in the `<EMF`

installation directory>/entity_api/lib directory are also required. The API has only been tested against the version of Java that EMF ships with.

Obtaining a Session object

Before being able to perform any operation, you must obtain a session object via the API that controls the access to the Repository. From a Java Program, this is done by calling: [com.aptean.emf.SessionFactory.createSession\(...\)](#).

When using the API via script module, a default session object is automatically created. It is recommended for performance reasons to always use this one, as creating your own for each execution in the script module has a performance impact. The default session object is called "session", and the "Script" operator is used to control the access rights, i.e. a session object has already been created with the rights of the "Script" operator. So, if the "Script" operator doesn't have the privilege to create an entity, then the API operation to create the entity will fail.

Creating the Data Access Object (DAO) and working with entities

You use the session object to get a Data Access Object (DAO) of the entity type you want to manipulate.

Example:

For JDBC Connection entity from a script module

```
var jdbcConnectionDAO =  
com.aptean.emf.entities.datasources.JDBCConnectionDAO.getInstance(session);
```

It creates an instance of the JDBC Connection Data Access Object and stores it in the javascript variable (jdbcConnectionDAO)

For JDBC Connection entity from a Java program

```
com.aptean.emf.framework.Session session = com.aptean.emf.SessionFactory.createSession  
("operator name", "password", "repository properties file name");  
  
com.aptean.emf.entities.datasources.JDBCConnectionDAO dao =  
com.aptean.emf.entities.datasources.JDBCConnectionDAO.getInstance(session);
```

Use the above Data Access Object (DAO) to Create/Read/Update/Delete the entities.

To create a new entity (Javascript)

```
var jdbcConnectionEntity = jdbcConnectionDAO.create();  
  
// Set values on the entity, for example to set the name  
jdbcConnectionEntity.setName( "Test" );  
  
// Save the new entity  
jdbcConnectionDAO.save(jdbcConnectionEntity );
```

To update an entity (Javascript)

```
var jdbcConnectionEntity = jdbcConnectionDAO.findByName("name of entity to update");

// Update values on the entity, for example to set the name
jdbcConnectionEntity.setName( "Test" );

// Save the changes
jdbcConnectionDAO.save(jdbcConnectionEntity );
```

To delete an entity (Javascript)

```
var jdbcConnectionEntity = jdbcConnectionDAO.findByName("name of entity to delete");

jdbcConnectionDAO.remove(jdbcConnectionEntity);
```

For further information on the Entity API, consult the [Entity API Documentation](#).

Java API Examples

The following examples demonstrate common usage of the Java APIs:

- [Scripting API examples](#)
- [EMF Runtime API examples \(including SOAP\)](#)

Java EMF Runtime API Examples

Click on the links below to view Java examples of using the EMF Runtime_API.

To use the examples:

Click on a link below to display the example. Then copy and paste the example text to a file, ensuring that you give the filename a .java extension.

Example Script	Description
AddAlert	The simplest usage of the API shows how to queue a single EMF Process.
AddAlertWithDataSection	Demonstrates how to create a data section and then queue an EMF Process with the new data section.
AddAlertWithMessageSection	Demonstrates how to create a message section and add some text in a specified language. an EMF Process is then queued with the new message section.
AddMultipleCopiesOfAlert	Demonstrates how to queue multiple instances of the same EMF Process.

TestAllDataSectionMethods	Tests calling all the methods available on a data section.
TestAllMessageSectionMethods	Tests calling all the methods available on a message section.
EMFApiSoapExample	Shows running an EMF Process using EMF SOAP client for Java
TestDataSectionExample	Extends XalertsApiSoapExample to add a populated data section, then run the EMF Process.
TestMessageSectionExample	Extends XalertsApiSoapExample to add a populated message section, then run the EMF Process.

Java API classes

Java Scripting API Examples

Click on the following links to view JavaScript examples of using the Scripting API.

To use the examples:

Click on a link below to display the example. Then copy and paste the example text to a file, ensuring that you give the filename a .js extension.

Example Script	Description
changeloglevel	Demonstrates how to change the logging level for the current EMF Process state.
datasections	Demonstrates how to handle, find, create, and manage EMF Process state data sections. This example is very comprehensive and should be studied as it covers many of the issues that will need to be addressed when writing JavaScript in EMF.
deletesections	Demonstrates how to delete sections from an EMF Process state. For completeness, this example will delete all sections from the current EMF Process state.
displayrecipientsections	Echoes the contents of any existing recipient sections to the console window of the server, including any data associated with the recipients.
enumsections	Demonstrates how to enumerate all sections in the current EMF Process state.
escalationreference	Demonstrates how to get the escalation

	reference for the current EMF Process state.
exception	Demonstrates how to handle an XalertsException exception in JavaScript. These exceptions are raised when an error condition occurs within the scripting API. This example shows how to make use of the exception handling capabilities of JavaScript.
getloglevel	Demonstrates how to get the set log level for the current EMF Process state.
haltalert	Demonstrates how to halt the current EMF Process state.
logmessage	Demonstrates how to use the logmessage method that allows a user to write their own messages to the debug log.
mergedatasections	Shows how to merge two data sections from two different data sources within the same EMF Process (useful for a heterogeneous database environment)
messagesection	Demonstrates how to handle find, create and manage EMF Process state message sections. A comprehensive example.
pop3attachments	Demonstrates how to iterate through all attachments in the POP3IN_ATTACHMENTS data section and include them in an XML-formatted message section.
recipientsections	Demonstrates how to handle find, create and manage EMF Process state recipient sections. This should be studied as RecipientSections are similar to DataSections with some small differences.
splittingrecipients	Demonstrates how to split recipient sections contained in the current EMF Process state. The splitting of recipients in a script module rather than at the recipient read module may be useful under some circumstances.

[Java API classes](#)

```
/* $Id: AddAlert.java,v 1.6 2001/07/27 09:23:39 ian Exp $  
 * AddAlert.java Created on 15 November 2000, 12:20  
 * Copyright Categoric Software 2000  
 */  
import java.rmi.*;
```

```

import com.xalert.server.api.open.xalert.*;
class AddAlert extends Object
{
    public static void main (String args[])
    {
        XalertsFactory xalertsFactory = null;
        Xalerts xalerts = null;

        try
        {
            // Get the factory object to get a new Xalert object from.
            // You will need to change the 'localhost' to point to the machine where
            // your Xalerts server is installed. The port (50001) needs to match
            // the value defined for the RMIREGISTRY_PORT.
            xalertsFactory = (XalertsFactory)Naming.lookup
("rmi://194.223.2.236:50001/com/xalert/server/api/XAlertsFactory");

            // Get a copy of the xalerts object from the factory.
            xalerts = xalertsFactory.getXalertsInstance();

            Response response;

            // Queue an alert. The message section will be added to any alert that is now queued.
            // 1 is the user id, no password, Demo is the repository and 1004 is the Id of the alert to
            queue.
            response = xalerts.queue( 1, "", "Win0282", 1020 );
            // Finished with the xalerts object now, so release it.
            // The number of xalerts objects is capped by the value of MAX_XALERT_API_
INSTANCES
            // defined in the servers Xalerts.properties so we must remember to release them.
            xalertsFactory.releaseXalertsInstance( xalerts );
            xalerts = null;

            // Check the response to see that the alert was successfully queued.

```

```
    if (response.getSuccess())
    {
        System.out.println("Alert queued successfully.");
    }
    else
    {
        System.out.println("Alert failed to queue successfully.");
        System.out.println("Error message returned - " + response.getErrorText() );
    }
}
catch(Exception e)
{
    e.printStackTrace();
}
finally
{
    if (xalerts != null)
    {
        // Release the xalerts resource if we haven't already done so.
        try
        {
            xalertsFactory.releaseXalertsInstance( xalerts );
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}

// Exit program cleanly
System.exit(0);
}
```

```
}

/* $Id: AddAlertWithDataSection.java,v 1.5 2001/07/27 09:23:39 ian Exp $
 * AddAlertWithDataSection.java Created on 16 November 2000, 11:38
 * Copyright Categoric Software 2000
 */
import java.rmi.*;
import com.xalert.server.api.open.xalert.*;
class AddAlertWithDataSection extends Object
{
    public static void main (String args[])
    {
        XalertsFactory xalertsFactory = null;
        Xalerts xalerts = null;

        try
        {
            // Get the factory object to get a new Xalert object from.
            // You will need to change the 'localhost' to point to the machine where
            // your Xalerts server is installed. The port (50001) needs to match
            // the value defined for the RMIREGISTRY_PORT.
            xalertsFactory = (XalertsFactory)Naming.lookup
("rmi://localhost:50001/com/xalert/server/api/XAlertsFactory");

            // Get a copy of the xalerts object from the factory.
            xalerts = xalertsFactory.getXalertsInstance();

            // Define the columns that will be used to create the data section.
            // This example defines 3 columns, first name, last name and age. A persons age may
            // not be known so the column
            // is being defined as nullable.
            Column[] columns = new Column[3];
            columns[0] = new Column( "First_Name", java.sql.Types.VARCHAR, false, false );
        }
    }
}
```

```
columns[1] = new Column( "Last_Name", java.sql.Types.VARCHAR, false, false );
columns[2] = new Column( "Age", java.sql.Types.VARCHAR, true, false );
// Add a data section
DataSection dataSection = xalerts.addDataSection( "My Data Section", columns );
// Add some data into the data section
Object[] fields = new Object[3];
fields[0] = new String( "Fred" );
fields[1] = new String( "Bloggs" );
fields[2] = new Integer( 25 );
dataSection.appendRecord( fields );
// Add another record
fields[0] = new String( "Joe" );
fields[1] = new String( "Bloggs" );
fields[2] = null;
dataSection.appendRecord( fields );

// Queue an alert. The message section will be added to any alert that is now queued.
// 1 is the user id, no password, Demo is the repository and 1004 is the Id of the alert to
queue.
Response response = xalerts.queue( 1, "", "Demo", 1004 );
// Finished with the xalerts object now, so release it.
// The number of xalerts objects is capped by the value of MAX_XALERT_API_
INSTANCES
// defined in the servers Xalerts.properties so we must remember to release them.
xalertsFactory.releaseXalertsInstance( xalerts );
xalerts = null;

// Check the response to see that the alert was successfully queued.
if (response.getSuccess())
{
    System.out.println("Alert queued successfully.");
}
else
{

```

```

        System.out.println("Alert failed to queue successfully.");
        System.out.println("Error message returned - " + response.getErrorText() );
    }
}
catch(Exception e)
{
    e.printStackTrace();
}
finally
{
    if (xalerts != null)
    {
        // Release the xalerts resource if we haven't already done so.
        try
        {
            xalertsFactory.releaseXalertsInstance( xalerts );
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
// Exit program cleanly
System.exit(0);
}
}
/* $Id: AddAlertWithMessageSection.java,v 1.6 2001/08/06 10:24:29 ian Exp $
 * AddAlertWithMessageSection.java Created on 15 November 2000, 15:09
 * Copyright Categorical Software 2000
 */
import java.rmi.*;
import com.xalert.server.api.open.xalert.*;

```

```

class AddAlertWithMessageSection extends Object
{
    public static void main (String args[])
    {
        XalertsFactory xalertsFactory = null;
        Xalerts xalerts = null;

        try
        {
            // Get the factory object to get a new Xalert object from.
            // You will need to change the 'localhost' to point to the machine where
            // your Xalerts server is installed. The port (50001) needs to match
            // the value defined for the RMIREGISTRY_PORT.
            xalertsFactory = (XalertsFactory)Naming.lookup
("rmi://localhost:50001/com/xalert/server/api/XAlertsFactory");

            // Get a copy of the xalerts object from the factory.
            xalerts = xalertsFactory.getXalertsInstance();
            // Add a message section
            MessageSection messageSection = xalerts.addMessageSection( "My Message Section"
);
            // Set the default language id.
            messageSection.setDefaultLanguageID( 1 );

            // Add some text to the message section in the required language
            messageSection.addLanguageText(1, "Hello,\nThis is a test message section." );

            // Queue an alert. The message section will be added to any alert that is now queued.
            // 1 is the user id, no password, XAlertsBeta2 is the repository and 1014 is the Id of the
            alert to queue.
            Response response = xalerts.queue( 1, "", "Demo", 1032 );
            // Finished with the xalerts object now, so release it.
            // The number of xalerts objects is capped by the value of MAX_XALERT_API_
INSTANCES
            // defined in the servers Xalerts.properties so we must remember to release them.

```

```
xalertsFactory.releaseXalertsInstance( xalerts );
xalerts = null;
// Check the response to see that the alert was successfully queued.
if (response.getSuccess())
{
    System.out.println("Alert queued successfully.");
}
else
{
    System.out.println("Alert failed to queue successfully.");
    System.out.println("Error message returned - " + response.getErrorText() );
}
}
catch (Exception e)
{
    e.printStackTrace();
}
finally
{
    if (xalerts != null)
    {
        // Release the xalerts resource if we haven't already done so.
        try
        {
            xalertsFactory.releaseXalertsInstance( xalerts );
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
// Exit program cleanly
```



```

        System.exit(0);
    }
}

/* $Id: AddMultipleCopiesOfAlert.java,v 1.5 2001/07/27 09:23:39 ian Exp $
 * AddMultipleCopiesOfAlert.java  Created on 16 November 2000, 14:15
 * Copyright Categorical Software 2000
 */

import java.rmi.*;
import com.xalert.server.api.open.xalert.*;
class AddMultipleCopiesOfAlert extends Object
{
    public static void main(String args[])
    {
        XalertsFactory xalertsFactory = null;
        Xalerts xalerts = null;

        try
        {
            // Get the factory object to get a new Xalert object from.
            // You will need to change the 'localhost' to point to the machine where
            // your Xalerts server is installed. The port (50001) needs to match
            // the value defined for the RMIREGISTRY_PORT.
            xalertsFactory = (XalertsFactory)Naming.lookup
("rmi://localhost:50001/com/xalert/server/api/XAlertsFactory");

            // Get a copy of the xalerts object from the factory.
            xalerts = xalertsFactory.getXalertsInstance();

            // Set the number of copies of the alert to add.
            xalerts.setCopiesOfAlertToQueue( 5 );

            // Queue an alert. The message section will be added to any alert that is now queued.
            // 1 is the user id, no password, XAlertsBeta2 is the repository and 1014 is the Id of the
            alert to queue.

```

```
Response response = xalerts.queue( 1, "", "SerengetiXSP1", 1001 );

// Finished with the xalerts object now, so release it.
// The number of xalerts objects is capped by the value of MAX_XALERT_API_
INSTANCES
// defined in the servers Xalerts.properties so we must remember to release them.
xalertsFactory.releaseXalertsInstance( xalerts );
xalerts = null;

// Check the response to see that the alert was successfully queued.
if (response.getSuccess())
{
    System.out.println("Alert queued successfully.");
    System.out.println("Alert queued successfully.");
    int [] alertInstanceIDs = response.getAlertInstanceIDs();
    for (int i = 0; i < alertInstanceIDs.length; i++)
    {
        System.out.println("Queued alert with id " + alertInstanceIDs[i] );
    }
}
else
{
    System.out.println("Alert failed to queue successfully.");
    System.out.println("Error message returned - " + response.getErrorText() );
}
}
catch(Exception e)
{
    e.printStackTrace();
}
finally
{
    if (xalerts != null)
```

```

    {
        // Release the xalerts resource if we haven't already done so.
        try
        {
            xalertsFactory.releaseXalertsInstance( xalerts );
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
// Exit program cleanly
System.exit(0);
}
}

```

Changing Alert State Log Levels

```

//
// Demonstrates how to change the logging level for the current alert state.
//
alertState.setLogLevel(Packages.com.xalert.server.api.open.scripting.AlertState.LOG_
LEVEL_VERBOSE);
print("Alert State (ID:"+alertState.getAlertInstanceID()+") log level is now VERBOSE!");

```

Data Sections

```

//      =====
//
// Demonstrates how to handle find, create and manage alert state data sections.
//
// The example will get a join of the UserProfile and Delivery profile tables from
// the Xalerts repository database and produce two new data sections. One for all
// users who have an email address specified and one for all user who have an SMS

```

```
// SMS number specified. This is to demonstrate how to extract, navigate and create
// data sections.
//
// The alert that runs this will need to run an SQL module with the following
// SQL prior to this script being executed. The data section must be named "UserDelivery"
//
// "SELECT * FROM UserProfile A, DeliveryAddress B WHERE B.UserProfileID =
// A.UserProfileID."
//
print("Looking for Data Section 'UserDelivery'.");
var userDelData = alertState.getDataSection("UserDelivery");
if(userDelData!=null)
{

    var emailData = createEmailDataSection();
    print("Created Data Section: "+emailData.getName());
    var smsData = createSmsDataSection();
    print("Created Data Section: "+smsData.getName());

    var recCount = userDelData.getRecordCount()
    print("Number of record to Iterate Through: "+recCount);
    for(i=1; i<=recCount; i++)
    {
        print("Processing Record: "+i);

        var emailAddress = userDelData.getCellAsString(i,"EmailAddress");
        if(emailAddress!=null && emailAddress!="")
        {
            print("Adding Email Address!");
            var values = new Array();
            values[0] = userDelData.getCellAsString(i,"UserProfileID");
            values[1] = userDelData.getCellAsString(i,"FirstName");
            values[2] = userDelData.getCellAsString(i,"LastName");
```

```

        values[3] = emailAddress;
        emailData.appendRecord(values);
        print("Add Row to Data Section: "+emailData.getName());
    }
    var smsNumber = userDelData.getCellAsString(i,"SMSNumber");
    if(smsNumber!=null && smsNumber!="")
    {
        print("Adding SMS Number!");
        var values = new Array();
        values[0] = userDelData.getCellAsString(i,"UserProfileID");
        values[1] = userDelData.getCellAsString(i,"FirstName");
        values[2] = userDelData.getCellAsString(i,"LastName");
        values[3] = smsNumber;
        smsData.appendRecord(values);
        print("Add Row to Data Section: "+smsData.getName());
    }

}

}

print("All Done!");
function createEmailDataSection()
{
    var columns = new Array();
    columns[0] = new Packages.com.xalert.server.api.open.common.Column("UserProfileID",
    Packages.com.xalert.server.api.open.common.Column.TYPE_INTEGER,false,false);
    columns[1] = new Packages.com.xalert.server.api.open.common.Column("FirstName",
    Packages.com.xalert.server.api.open.common.Column.TYPE_VARCHAR,true,false)
    columns[2] = new Packages.com.xalert.server.api.open.common.Column("LastName",
    Packages.com.xalert.server.api.open.common.Column.TYPE_VARCHAR,true,false)
    columns[3] = new Packages.com.xalert.server.api.open.common.Column("EmailAddress",
    Packages.com.xalert.server.api.open.common.Column.TYPE_VARCHAR,true,false);
    var emailData = alertState.addDataSection("EmailUsers",columns);
    return emailData;
}

function createSmsDataSection()

```

```

{
    var columns = new Array();

    columns[0] = new Packages.com.xalert.server.api.open.common.Column("UserProfileID",
    Packages.com.xalert.server.api.open.common.Column.TYPE_INTEGER,false,false);

    columns[1] = new Packages.com.xalert.server.api.open.common.Column("FirstName",
    Packages.com.xalert.server.api.open.common.Column.TYPE_VARCHAR,true,false)

    columns[2] = new Packages.com.xalert.server.api.open.common.Column("LastName",
    Packages.com.xalert.server.api.open.common.Column.TYPE_VARCHAR,true,false)

    columns[3] = new Packages.com.xalert.server.api.open.common.Column("SmsNumber",
    Packages.com.xalert.server.api.open.common.Column.TYPE_VARCHAR,true,false);

    var smsData = alertState.addDataSection("SmsUsers",columns);

    return smsData;
}

```

Deleting Sections

Demonstrates how to delete sections from an alert state. For completeness, this example will delete all sections from the current alert state.

```

print("");
print("Deleteing Existing Data Sections");
print("=====");
var dsNames = alertState.getDataSectionList();
var dsCount = dsNames.length;
for(i=0; i<dsCount; i++)
{
    alertState.removeDataSection(dsNames[i]);
    print("Deleted Data Section named: "+dsNames[i]);
}
print("");
print("Deleteing Existing Message Sections");
print("=====");
var msNames = alertState.getMessageSectionList();
var msCount = msNames.length;
for(i=0; i<msCount; i++)
{

```

```

    alertState.removeMessageSection(msNames[i]);
    print("Deleted Message Section named: "+msNames[i]);
}
print("");
print("Deleteing Existing Recipient Sections");
print("=====");
var rsNames = alertState.getRecipientSectionList();
var rsCount = rsNames.length;
for(i=0; i<rsCount; i++)
{
    alertState.removeRecipientSection(rsNames[i]);
    print("Deleted Recipient Section named: "+rsNames[i]);
}

```

Display Recipient Sections

Prints out all the information about a recipient section to the console.

```

    var rsNames = alertState.getRecipientSectionList();
    var rsCount = rsNames.length;
    print("Number of recipient section is : " + rsCount );
    for( sectionNumber = 0; sectionNumber < rsCount; sectionNumber++)
    {
        print("Recipient Section " + (sectionNumber+1) + " is called " + rsNames
[sectionNumber] + " ");
        var rs = alertState.getRecipientSection(rsNames[sectionNumber]);

        print("Found Recipient Section named : " + rs.getName());
        print("Number of recipients : " + rs.getRecipientCount());
        print("Attribute Count : " + rs.getAttriuteCount());

        for (recipientNumber = 1; recipientNumber <= rs.getRecipientCount();
recipientNumber++ )
        {
            var rsAttCount = rs.getAttributeCount();

```

```

var attributes;
attributes = rs.getAttributes();
for (attributeNumber=1; attributeNumber <= rsAttCount; attributeNumber++)
{
    var type;
    switch (attributes[attributeNumber-1].getType())
    {
        case java.sql.Types.JAVA_OBJECT : type = "JAVA_OBJECT"; break;
        case java.sql.Types.INTEGER : type = "INTEGER"; break;
        case java.sql.Types.BIGINT : type = "BIGINT"; break;
        case java.sql.Types.SMALLINT : type = "SMALLINT"; break;
        case java.sql.Types.REAL : type = "REAL"; break;
        case java.sql.Types.DOUBLE : type = "DOUBLE"; break;
        case java.sql.Types.NUMERIC : type = "NUMERIC"; break;
        case java.sql.Types.DECIMAL : type = "DECIMAL"; break;
        case java.sql.Types.CHAR : type = "CHAR"; break;
        case java.sql.Types.VARCHAR : type = "VARCHAR"; break;
        case java.sql.Types.LONGVARCHAR : type = "LONGVARCHAR"; break;
        case java.sql.Types.DATE : type = "DATE"; break;
        case java.sql.Types.TIME : type = "TIME"; break;
        case java.sql.Types.TIMESTAMP : type = "TIMESTAMP"; break;
        case java.sql.Types.VARBINARY : type = "VARBINARY"; break;
        case java.sql.Types.BINARY : type = "BINARY"; break;
        case java.sql.Types.LONGVARBINARY : type = "LONGVARBINARY"; break;
        default:
            type = "Some other type";
    }
    print("Attribute :" + attributes[attributeNumber-1].getName() + " Type : " +
type );
    print("Value :" + rs.getAttribute( recipientNumber, rs.getAttributeName(
attributeNumber ) ) );
}

// Display any associated data

```



```

var associatedData = rs.getAssociatedRecipientData( recipientNumber );
print("");
print("ASSOCIATED DATA");
if (associatedData == null)
{
    print("None");
}
else
{
    print( "Record count " + associatedData.getRecordCount() );
    var rsAssociatedAttCount = associatedData.getAttributeCount();
    print( "Attribute Count " + rsAssociatedAttCount );
    var associatedAttributes = associatedData.getAttributes();
    for (j = 1; j <= rsAssociatedAttCount ; j++)
    {
        print("Attribute :" + associatedAttributes[j-1].getName() );
        for (k = 1; k <= associatedData.getRecordCount(); k++)
        {
            print("Value " + k + " :" + associatedData .getAttribute( k, associatedData
.getAttributeName( j ) ) );
        }
    }
}
print("");
}
print("");
}

```

EMFApiSoapExample.java

```

import java.io.*;
import java.net.*;
import java.util.*;

```

```
import com.xalert.server.soap.client.*;

/**
 *
 * @author Lee Hepplewhite
 * @version
 */

public class XalertsApiSoapExample {
    // Location of the SOAP service router
    protected String rpcUrl = "http://localhost:8080/xalertsssoap/servlet/rpcrouter";
    // URI of the Xalerts SOAP API
    protected String targetURI = "http://www.categoric.com/xalerts/xalertsapi";
    // Name of SOAP method to invoke on service
    protected String methodName = "queueXalert";
    // Name of name-based SOAP method to invoke on service
    protected String nameBasedMethodName = "queueXalertByName";
    // RMI Lookup URL for the Xalerts Server
    protected String lookupURL =
"rmi://localhost:50001/com/xalert/server/api/XAlertsFactory";
    // UserProfile ID of user with right to run the alert
    protected int userID = 1001;
    // UserProfile ID of user with right to run the alert
    protected String userProfileName = "Admin";
    // Password of above user
    protected String password = "password";
    // Name of the Repository containing the alert to be run
    protected String repositoryName = "MyRepository";
    // Xalert Profile ID of the Xalert to run
    protected int alertID = 1008;
    // Xalert Profile ID of the Xalert to run
    protected String alertProfileApiName = "My Xalert API Name";

    /** Creates new XalertsApiSoapClient */
    public XalertsApiSoapExample() {
    }
}
```

```
protected void initialiseState(Xalerts xalert) throws Exception
{
    //do nothing in this version
    // subclass this to build different tests
}

protected void printResponse(Response resp) throws Exception
{
    if (resp == null)
    {
        System.out.println("Error - returned NULL response.");
        return;
    }
    if (resp.getSuccess() == true)
    {
        // if success then print out the instance id of alert that was run
        System.out.println("Success - Ran alert instance "+resp.getAlertInstanceID
());
    }
    else
    {
        // if failure then print out the error message
        System.out.println("Failure - "+resp.getErrorText());
    }
}

protected void runTest() throws Exception
{
    // Create a instance of the Xalerts Factory object
    XalertsApiSoapClient sc = new XalertsApiSoapClient();
    // Using the factory obtain a new Xalerts instance
    Xalerts xalert = sc.getXalertsInstance(lookupURL, rpcUrl, targetURI, methodName);
    // Now initialise the xalert object with relevant data sections etc.
    initialiseState(xalert);
    // Now do the SOAP call
    Response resp = xalert.queue(userID, password, repositoryName, alertID);
}
```

```

        //Finally print out the response
        printResponse(resp);
    }

    protected void runByNameTest() throws Exception
    {
        // Create a instance of the Xalerts Factory object
        XalertsApiSoapClient sc = new XalertsApiSoapClient();
        // Using the factory obtain a new Xalerts instance
        Xalerts xalert = sc.getXalertsInstance(lookupURL, rpcUrl, targetURI,
nameBasedMethodName);
        // Now initialise the xalert object with relevant data sections etc.
        initialiseState(xalert);
        // Now do the SOAP call
        Response resp = xalert.queue(userProfileName, password, repositoryName,
alertProfileApiName);
        //Finally print out the response
        printResponse(resp);
    }

    public static void main (String args[])
    {
        try
        {
            XalertsApiSoapExample ex = new XalertsApiSoapExample();
            ex.runTest();
        }
        catch (Exception e)
        {
            System.out.println(e.getMessage());
            e.printStackTrace();
        }
    }
}

```

Enumerating Data Sections

Demonstrates how to enumerate all sections in the current alert state.

```
//
print("");
print("Outputting Existing Data Sections");
print("=====");
var dsNames = alertState.getDataSectionList();
var dsCount = dsNames.length;
for(i=0; i<dsCount; i++)
{
    var ds = alertState.getDataSection(dsNames[i]);
    print("Found Data Section named: "+ds.getName());
}
print("");
print("Outputting Existing Message Sections");
print("=====");
var msNames = alertState.getMessageSectionList();
var msCount = msNames.length;
for(i=0; i<msCount; i++)
{
    var ms = alertState.getMessageSection(msNames[i]);
    print("Found Message Section named: "+ms.getName());
}
print("");
print("Outputting Existing Recipient Sections");
print("=====");
var rsNames = alertState.getRecipientSectionList();
var rsCount = rsNames.length;
for(i=0; i<rsCount; i++)
{
    var rs = alertState.getRecipientSection(rsNames[i]);
    print("Found Recipient Section named: "+rs.getName());
}
```

```
}
```

Escalation Reference

```
//  
// Demonstrates how to get the escalation reference for the current alert state.  
//  
var escalRef = alertState.getEscalationReference();  
print("Escalation Reference for Alert Instance (ID="+alertState.getAlertInstanceID()+") is:  
"+escalRef);
```

Handling Exceptions

Demonstrates how to handle an `XalertsException` exception in JavaScript. These exceptions are raised

```
// when an error condition occurs within the scripting API.  
//  
//This example will trap a forced error. In this case the error (exception) will be caught and  
handled.  
//The script module and thus the alert will continue as if everything was find.  
try {  
    //Force an error - can't have a negative log level.  
    alertState.setLogLevel(-1);  
}catch(excep) {  
    //Catch the exception.  
    print("API Error Occured: "+excep);// excep.getMessage()  
}  
//Here the error is forced and not trapped and so the script module will error, halting the  
parent alert.  
//alertState.setLogLevel(-1);
```

Getting Log Levels

Demonstrates how to get the set log level for the current alert state.

```
//  
var logLevel = alertState.getLogLevel();
```

```
switch(logLevel) {  
    case Packages.com.xalert.server.api.open.scripting.AlertState.LOG_LEVEL_ERROR:  
        print("Current Log Level: ERRORS ONLY!");  
        break;  
    case Packages.com.xalert.server.api.open.scripting.AlertState.LOG_LEVEL_WARNING:  
        print("Current Log Level: ERRORS and WARNINGS!");  
        break;  
    case Packages.com.xalert.server.api.open.scripting.AlertState.LOG_LEVEL_DETAILED:  
        print("Current Log Level: ERRORS, WARNINGS and DETAILED!");  
        break;  
    case Packages.com.xalert.server.api.open.scripting.AlertState.LOG_LEVEL_VERBOSE:  
        print("Current Log Level: ERRORS, WARNINGS, DETAILED and VERBOSE!");  
        break;  
    default:  
        print("*** ERROR: Invalid Log Level ***");  
        break;  
}
```

Halting Alerts

Demonstrates how to halt the current alert state.

```
alertState.halt(true);  
print("The Alert State Instance (ID: "+alertState.getAlertInstanceID()+") has been halted!");
```

Log Message

```
logMessage( Packages.com.xalert.server.api.open.scripting.AlertState.LOG_LEVEL_VERBOSE,  
"This message is logged at a verbose logging level" );  
  
logMessage( Packages.com.xalert.server.api.open.scripting.AlertState.LOG_LEVEL_DETAILED,  
"This message is logged at a detailed logging level" );  
  
logMessage( Packages.com.xalert.server.api.open.scripting.AlertState.LOG_LEVEL_WARNING,  
"This message is logged at a warning logging level" );  
  
logMessage( Packages.com.xalert.server.api.open.scripting.AlertState.LOG_LEVEL_ERROR,  
"This message is logged at a error logging level" );
```

Merge Data Sections

This example shows how to use the Script module to merge data sections extracted from different types of databases within the same event process. The alert must have two SQL modules each connecting to a separate database. The data sections from both SQL modules - DS1 and DS2 - must have a column in common, for example EmployeeID, so they can be linked.

The merged data section is called DS3. Get datasection with name DS1 and iterate through to list all records.

```
print("Looking for Data Section 'DS1'.");
var ds1Data = alertState.getDataSection("DS1");
if(ds1Data!=null)
{

    var ds1RecCount = ds1Data.getRecordCount()
    print("Number of record to Iterate Through: "+ds1RecCount);
    for(i=1; i<=ds1RecCount; i++)
    {
        // print("Processing Record: "+i + ' ' +ds1Data.getCellAsString(i,"col1"));
    }
}

print("Looking for Data Section 'DS2'.");
var ds2Data = alertState.getDataSection("DS2");
if(ds2Data!=null)
{

    var ds2RecCount = ds2Data.getRecordCount()
    print("Number of record to Iterate Through: "+ds2RecCount);
    for(i=1; i<=ds2RecCount; i++)
    {
        // print("Processing Record: "+i + ' ' +ds2Data.getCellAsString(i,"col1"));
    }
}
```



```

var mergedData = createMergedDataSection();
    for(i=1; i<=ds1RecCount; i++)
    {
        for(j=1; j<=ds2RecCount; j++)
        {
            if ( parseInt(ds1Data.getCellAsString(i,"col1")) == parseInt(ds2Data.getCellAsString(j,"col1")))
            {
                // print("matched "+i+" "+j);
                // print( ds1Data.getCellAsString(i,"col1") + "==="+ ds2Data.getCellAsString(j,"col1") );
                var values = new Array();
                values[0] = ds1Data.getCellAsString(i,"col1");
                values[1] = ds1Data.getCellAsString(i,"col2");
                values[2] = ds1Data.getCellAsString(i,"col3");
                values[3] = ds2Data.getCellAsString(j,"col2");
                values[4] = ds2Data.getCellAsString(j,"col3");
                mergedData.appendRecord(values);
            }
        }
    }
print("Done!");
function createMergedDataSection()
{
    var columns = new Array();
    columns[0] = new Packages.com.xalert.server.api.open.common.Column("col1",
Packages.com.xalert.server.api.open.common.Column.TYPE_VARCHAR,true,false);
    columns[1] = new Packages.com.xalert.server.api.open.common.Column
("col2",Packages.com.xalert.server.api.open.common.Column.TYPE_VARCHAR,true,false)
    columns[2] = new Packages.com.xalert.server.api.open.common.Column("col3",
Packages.com.xalert.server.api.open.common.Column.TYPE_VARCHAR,true,false)
    columns[3] = new Packages.com.xalert.server.api.open.common.Column("col4",
Packages.com.xalert.server.api.open.common.Column.TYPE_VARCHAR,true,false);
    columns[4] = new Packages.com.xalert.server.api.open.common.Column("col5",
Packages.com.xalert.server.api.open.common.Column.TYPE_VARCHAR,true,false);
    var mergedData = alertState.addDataSection("DS3",columns);

```

```
    return mergedData;
}
```

Message Sections

Demonstrates how to handle find, create and manage alert state message sections. This example will look for a command to exist to know if it should continue. In this case the command is received from an email. Then a new message section will be created - after checking that it does not already exist - containing an English and German version of the message to be sent. The new message section is then added to the current alert state.

```
print("Looking for Message Section 'POP3IN'.");
var msg = alertState.getMessageSection("POP3IN");
if(msg != null)
{
    print("Checking to see the 'GET_TIME' command is contained in the recieved email");
    lang = msg.getDefaultLanguageText();
    if(lang != null && lang.getText().startsWith("GET_TIME"))
    {

        print("Looking for Message Section 'Current Time'.");
        msg = alertState.getMessageSection("Current Time");
        if(msg == null)
        {
            print("Message Section 'Current Time' not found... Creating.");
            msg = alertState.addMessageSection("Current Time");
            var langID = getLanguageID("en-GB");
            //An exception maybe thrown if the ISO code (e.g. "en-GB") does not exist,
            //as the lang ID returned will be -1 in such a case.
            msg.addLanguageText(langID,"The time is $TIME()$.");
            langID = getLanguageID("de-DE");
            msg.addLanguageText(langID,"Die ziete ist $TIME()$.");
            print("Added Message Section 'Current Time' to the current Alert State.");
        }
    }
}
```

```
    }  
}  
print("All Done!");  
function getLanguageID(isoLocaleCode)  
{  
    //Find and return the language ID for the specified ISO locale code. This will return  
    //-1 if there is no ID for the specified ISO code.  
    var lang = null;  
    var sp = new Packages.com.xalert.admin.framework.SearchParameter  
(Packages.com.xalert.admin.LanguageAdmin.ISOLOCALE,Packages.com.xalert.admin.frame  
work.SearchParameter.EQUALS,isoLocaleCode);  
    var sc = new Packages.com.xalert.admin.framework.SearchCriteria(sp);  
    var langs = Packages.com.xalert.admin.LanguageAdminManager.findList(sc);  
    while(langs.hasMoreElements())  
    {  
        lang = langs.nextElement();  
        break;  
    }  
    if(lang!=null)  
    {  
        return lang.getLanguageAdminID();  
    }else{  
        return -1;  
    }  
}
```

POP3IN_ATTACHMENTS

Demonstrates how to iterate through all the attachments in POP3IN_ATTACHMENTS

```
//and save the details including content to an XML-formatted message section  
//called "MyPOP3Attachments". This message section can then be used later.  
var attachmentsData = alertState.getDataSection("POP3IN_ATTACHMENTS");  
var buf = new java.lang.StringBuffer();
```

```

if(attachmentsData!=null)
{
    //Defining the XML nodes
    var attachmentXML1 = "<Attachments>";
    var attachmentXML2 = "</Attachments>";
    var filenameXML1 = "<File>";
    var filenameXML2 = "</File>";
    var contentTypeXML1 = "<contentType>";
    var contentTypeXML2 = "</contentType>";
    var contentXML1 = "<content>";
    var contentXML2 = "</content>";
    var recCount = attachmentsData.getRecordCount();
    print("Number of record to Iterate Through: "+ recCount);

    buf.append(attachmentXML1 + "\n");
    for(i=1; i<=recCount; i++)
    {
        print("Processing Record: "+i);

        //For each attachment, write out the filename, content type and content (if the
        attachment
        //is binary, then the content will be written out Base64-encoded).

        var fileName = attachmentsData.getCellAsString(i,"FileName");
        var content = attachmentsData.getCellAsString(i,"Content");
        var contentType = attachmentsData.getCellAsString(i,"ContentType");
        buf.append(filenameXML1 + "\n" + fileName + "\n" + filenameXML2 + "\n");
        buf.append(contentTypeXML1 + "\n" + contentType + "\n" + contentTypeXML2 + "\n");
        buf.append(contentXML1 + "\n" + content + "\n" + contentXML2 + "\n");
    }
    buf.append(attachmentXML2 + "\n");
    msg = alertState.addMessageSection("MyPOP3Attachments");
    var langID = getLanguageID("en-US");

```

```

//An exception maybe thrown if the ISO code (e.g. "en-US") does not exist,
//as the lang ID returned will be -1 in such a case.
msg.addLanguageText(langID, buf.toString());
}
function getLanguageID(isoLocaleCode)
{
    //Find and return the language ID for the specified ISO locale code. This will return
    //-1 if there is no ID for the specified ISO code.
    var lang = null;
    var sp = new Packages.com.xalert.admin.framework.SearchParameter
(Packages.com.xalert.admin.LanguageAdmin.ISOLOCALE, Packages.com.xalert.admin.frame
work.SearchParameter.EQUALS, isoLocaleCode);
    var sc = new Packages.com.xalert.admin.framework.SearchCriteria(sp);
    var langs = Packages.com.xalert.admin.LanguageAdminManager.findList(sc);
    while(langs.hasMoreElements())
    {
        lang = langs.nextElement();
        break;
    }
    if(lang!=null)
    {
        return lang.getLanguageAdminID();
    }else{
        return -1;
    }
}
}

```

Recipient Sections

Demonstrates how to handle find, create and manage alert state recipient sections. This example attempts to first find a recipient section named "Example Recipient Section". If the section is not found then it is created from scratch, otherwise the found one is utilised.

If the check to see if the recipient section of that name was not made, then an existing recipient section of that name would be overwritten with the new one of the same name.

```
var rsName = "Example Recipient Section";
//Look for a specific recipient section.
var rs = alertState.getRecipientSection(rsName);
if(rs==null)
{
    //If no found then create a new one.
    var rs = createRecipientSection(rsName);
}
//Add some new recipients.
addRecipient(rs,"Mario","Andretti","mario@sideimpact.com");
addRecipient(rs,"Tim","Hendman","tim@doublefault.co.uk");
addRecipient(rs,"Duke","Nukem","daduke@comegetsome.org");
addRecipient(rs,"John","Occam","johnny@ouch!.co.uk");
addRecipient(rs,"Austim","Powers","austin@doubleohbehave.co.uk");
function createRecipientSection(name)
{
    var attributes = new Array();
    attributes[0] = new Packages.com.xalert.server.api.open.common.Attribute("FirstName",
    Packages.com.xalert.server.api.open.common.Attribute.TYPE_STRING)
    attributes[1] = new Packages.com.xalert.server.api.open.common.Attribute
    ("LastName",Packages.com.xalert.server.api.open.common.Attribute.TYPE_STRING)
    attributes[2] = new Packages.com.xalert.server.api.open.common.Attribute
    ("EmailAddress", Packages.com.xalert.server.api.open.common.Attribute.TYPE_STRING);
    var rs = alertState.addRecipientSection(name,attributes);
    return rs;
}
function addRecipient(rs, firstName, lastName, emailAddress)
{
    var values = new Array();
    values[0] = firstName;
    values[1] = lastName;
    values[2] = emailAddress;
    rs.appendRecipient(values);
}
```

Splitting Recipients

Demonstrates how to split recipient sections contained in the current alert state. This ends the current alert state, clones it and runs it as a number of new alert states, each with a close to 10 recipients as possible. The aim of this is to optimise recipient throughput. Each new alert state created is queued. To illustrate this clearly, this example ideally requires an alert state with a recipient section named "Recipients To Split" which holds 99 recipients.

```
alertState.defineRecipientSectionSplit("Recipients To Split",10);  
print("Alert State (ID: "+alertState.getAlertInstanceID()+") has been split!");
```

```
/* $Id: TestAllDataSectionMethods.java,v 1.5 2001/07/27 09:23:39 ian Exp $  
 * TestAllDataSectionMethods.java Created on 16 November 2000, 14:24  
 * Copyright Categorical Software 2000  
 */  
  
import java.rmi.*;  
import com.xalert.server.api.open.xalert.*;  
class TestAllDataSectionMethods extends Object  
{  
    public static void main(String args[])  
    {  
        XalertsFactory xalertsFactory = null;  
        Xalerts xalerts = null;  
  
        try  
        {  
            // Get the factory object to get a new Xalert object from.  
            // You will need to change the 'localhost' to point to the machine where  
            // your Xalerts server is installed. The port (50001) needs to match  
            // the value defined for the RMIREGISTRY_PORT.  
            xalertsFactory = (XalertsFactory)Naming.lookup  
("rmi://localhost:50001/com/xalert/server/api/XAlertsFactory");  
  
            // Get a copy of the xalerts object from the factory.
```

```
xalerts = xalertsFactory.getXalertsInstance();

// Define the columns that will be used to create the data section.
// This example defines 3 columns, first name, last name and age. A persons age may
not be known so the column
// is being defined as nullable.
Column[] columns = new Column[3];
columns[0] = new Column( "First_Name", Column.TYPE_VARCHAR, false, false );
columns[1] = new Column( "Last_Name", Column.TYPE_VARCHAR, false, false );
columns[2] = new Column( "Age", Column.TYPE_INTEGER, true, false );

// Add a data section
DataSection dataSection = xalerts.addDataSection( "My Data Section", columns );

// Display the columns added.
Column[] columnsAdded = dataSection.getColumns();
for (int i = 0; i < columnsAdded.length; i++)
{
    System.out.println("COLUMN " + i);
    System.out.println("name " + columnsAdded[i].getName());
    System.out.println("nullable " + columnsAdded[i].isNullable());
    System.out.println("signed " + columnsAdded[i].isSigned());
}

// Add some data into the data section
Object[] fields = new Object[3];
fields[0] = new String( "Fred" );
fields[1] = new String( "Bloggs" );
fields[2] = null;

dataSection.appendRecord( fields );

// Add another record
```



```
fields[0] = new String( "Joe" );
fields[1] = new String( "Bloggs" );
fields[2] = new Integer( 25 );

dataSection.appendRecord( fields );

// Get the 3rd column (age) of the second record.
Integer age = (Integer)dataSection.getCellContents( 2, 3 );
if (age.equals( new Integer( 25 )))
{
    System.out.println( "Correct field retrieved" );
}
else
{
    System.out.println( "Error retrieving field" );
}

// Change the value of one of the cells
dataSection.setCellContents( 2, 3, new Integer(26) );
age = (Integer)dataSection.getCellContents( 2, 3 );
if (age.equals( new Integer( 26 )))
{
    System.out.println( "Correct field retrieved" );
}
else
{
    System.out.println( "Error retrieving field" );
}

// Check the field count
if (dataSection.getFieldCount() == 3)
{
    System.out.println( "Correct field count retrieved" );
}
```

```
    }
    else
    {
        System.out.println( "Error retrieving field count, got " + dataSection.getFieldCount()
+ " expected 3" );
    }

    // Check the record count
    if (dataSection.getRecordCount() == 2)
    {
        System.out.println( "Correct record count retrieved" );
    }
    else
    {
        System.out.println( "Error retrieving record count, got " +
dataSection.getRecordCount() + " expected 2" );
    }

    // Check we can get a field name
    if (dataSection.getFieldName(2).equals("Last_Name"))
    {
        System.out.println( "Correct field name retrieved" );
    }
    else
    {
        System.out.println( "Error retrieving field name" );
    }

    // Check we can get the data section name
    if (dataSection.getName().equals("My Data Section"))
    {
        System.out.println( "Correct data section name retrieved" );
    }
    else
```

```
{
    System.out.println( "Error retrieving data section name" );
}

// Finished with the xalerts object now, so release it.
// The number of xalerts objects is capped by the value of MAX_XALERT_API_
INSTANCES
// defined in the servers Xalerts.properties so we must remember to release them.
xalertsFactory.releaseXalertsInstance( xalerts );
xalerts = null;

}
catch(Exception e)
{
    // Error adding data section, or adding data to the data section
    e.printStackTrace();
}
finally
{
    if (xalerts != null)
    {
        // Release the xalerts resource if we haven't already done so.
        try
        {
            xalertsFactory.releaseXalertsInstance( xalerts );
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

```

        // Exit program cleanly
        System.exit(0);
    }
}

/* $Id: TestAllMessageSectionMethods.java,v 1.5 2001/07/27 09:23:39 ian Exp $
 * TestAllMessageSectionMethods.java Created on 16 November 2000, 14:52
 * Copyright Categoric Software 2000
 */
import java.rmi.*;
import com.xalert.server.api.open.xalert.*;
import java.util.*;
class TestAllMessageSectionMethods extends Object
{
    public static void main (String args[])
    {
        XalertsFactory xalertsFactory = null;
        Xalerts xalerts = null;

        try
        {
            // Get the factory object to get a new Xalert object from.
            // You will need to change the 'localhost' to point to the machine where
            // your Xalerts server is installed. The port (50001) needs to match
            // the value defined for the RMIREGISTRY_PORT.
            xalertsFactory = (XalertsFactory)Naming.lookup
("rmi://localhost:50001/com/xalert/server/api/XAlertsFactory");

            // Get a copy of the xalerts object from the factory.
            xalerts = xalertsFactory.getXalertsInstance();

            // Add a message section
            MessageSection messageSection = xalerts.addMessageSection( "My Message Section"
);

```

```
// Set the default language id.
messageSection.setDefaultLanguageID( 1 );

// Add some text to the message section in the required language
messageSection.addLanguageText(1, "Text for lanaguage id 1." );
messageSection.addLanguageText(0, "Text for lanaguage id 0." );
LanguageText languageText = messageSection.getDefaultLanguageText();
if (languageText.getText().equals("Text for lanaguage id 1."))
{
    System.out.println( "Correct default language text retrieved" );
}
else
{
    System.out.println( "Error retrieving default language text" );
}
languageText = messageSection.getLanguageText(1);
if (languageText.getText().equals("Text for lanaguage id 1."))
{
    System.out.println( "Correct language text retrieved" );
}
else
{
    System.out.println( "Error retrieving language text" );
}
if (languageText.getLangaugeID() == 1)
{
    System.out.println( "Correct language id retrieved" );
}
else
{
    System.out.println( "Error retrieving language id" );
}
languageText.setText("Changed Text.");
```

```
    if (languageText.getText().equals("Changed Text."))
    {
        System.out.println( "Correct language text retrieved" );
    }
    else
    {
        System.out.println( "Error retrieving language text" );
    }
    // Check we can get the message section name
    if (messageSection.getName().equals("My Message Section"))
    {
        System.out.println( "Correct message section name retrieved" );
    }
    else
    {
        System.out.println( "Error retrieving message section name" );
    }

    // Finished with the xalerts object now, so release it.
    // The number of xalerts objects is capped by the value of MAX_XALERT_API_
INSTANCES
    // defined in the servers Xalerts.properties so we must remember to release them.
    xalertsFactory.releaseXalertsInstance( xalerts );
    xalerts = null;
}
catch(Exception e)
{
    e.printStackTrace();
}
finally
{
    if (xalerts != null)
    {
```

```

        // Release the xalerts resource if we haven't already done so.
        try
        {
            xalertsFactory.releaseXalertsInstance( xalerts );
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}

// Exit program cleanly
System.exit(0);
}
}

/*
 * TestDataSectionExample.java
 *
 * Created on 18 September 2001, 13:30
 */
import java.io.*;
import java.net.*;
import java.util.*;
import com.xalert.server.soap.client.*;
/**
 *
 * @author Lee Hepplewhite
 * @version
 */
public class TestDataSectionExample extends XalertsApiSoapExample {
    /** Creates new TestDataSectionExample */
    public TestDataSectionExample() {

```

```

}
protected void initialiseState(Xalerts xalert) throws Exception
{
    // Define the columns that will be used to create the data section.
    // This example defines 3 columns, first name, last name and age. A persons age may
    not be known so the column
    // is being defined as nullable.
    Column[] columns = new Column[3];
    columns[0] = xalert.createColumn( "First_Name", java.sql.Types.VARCHAR, false, false
);
    columns[1] = xalert.createColumn( "Last_Name", java.sql.Types.VARCHAR, false, false
);
    columns[2] = xalert.createColumn( "Age", java.sql.Types.VARCHAR, true, false );
    // Add a data section
    DataSection dataSection = xalert.addDataSection( "Users", columns );
    // Add some data into the data section
    Object[] fields = new Object[3];
    fields[0] = new String( "Fred" );
    fields[1] = new String( "Bloggs" );
    fields[2] = new Integer( 25 );
    dataSection.appendRecord( fields );
    // Add another record
    fields[0] = new String( "Joe" );
    fields[1] = new String( "Bloggs" );
    fields[2] = null;
    dataSection.appendRecord( fields );

    columns = new Column[2];
    columns[0] = xalert.createColumn( "When", java.sql.Types.VARCHAR, false, false );
    columns[1] = xalert.createColumn( "Time", java.sql.Types.DATE, false, false );
    // Add a data section
    dataSection = xalert.addDataSection( "Dates", columns );
    // Add some data into the data section
    fields = new Object[2];

```



```
        fields[0] = new String( "Now" );
        fields[1] = new java.sql.Date(System.currentTimeMillis());
        dataSection.appendRecord( fields );
    }

    public static void main (String args[])
    {
        try
        {
            TestDataSectionExample ex = new TestDataSectionExample();
            ex.runTest();
        }
        catch (Exception e)
        {
            System.out.println(e.getMessage());
            e.printStackTrace();
        }
    }
}

/*
 * TestMessageSectionExample.java
 *
 * Created on 18 September 2001, 15:30
 */
import java.io.*;
import java.net.*;
import java.util.*;
import com.xalert.server.soap.client.*;
/**
 *
 * @author Lee Hepplewhite
```

```
* @version
*/
public class TestMessageSectionExample extends XalertsApiSoapExample {
    /** Creates new TestMessageSectionExample */
    public TestMessageSectionExample() {
    }
    protected void initialiseState(Xalerts xalert) throws Exception
    {
        // Add a message section
        MessageSection messageSection = xalert.addMessageSection( "English" );
        // Set the default language id.
        messageSection.setDefaultLanguageID( 1 );
        // Add some text to the message section in the required language
        messageSection.addLanguageText(1, "Hello,\nThis is a test message section." );
        // Add a message section
        messageSection = xalert.addMessageSection( "English+" );
        // Set the default language id.
        messageSection.setDefaultLanguageID( 2 );
        // Add some text to the message section in the required language
        messageSection.addLanguageText(1, "Hello,\nThis is english." );
        // Add some text to the message section in the required language
        messageSection.addLanguageText(2, "Hello,\nThis is something else." );
    }

    public static void main (String args[])
    {
        try
        {
            TestMessageSectionExample ex = new TestMessageSectionExample();
            ex.runTest();
        }
        catch (Exception e)
        {
        }
    }
}
```

```
        System.out.println(e.getMessage());
        e.printStackTrace();
    }

}

}
```

SOAP

Simple Object Access Protocol (SOAP) is a simple and extensible XML-based communication protocol for exchanging application data, for example, via remote procedure calls.

The SOAP specification defines how application data should be encoded and represented in XML format. SOAP is based on XML, and so uses XML namespaces. SOAP implementations can also use XML schemas and the Web Services Description Language (WSDL). WSDL allows you to describe how other applications should access your Web service.

Although SOAP is transport-agnostic, the majority of SOAP implementations support HTTP (in addition to any other transports they support).

Advantages of SOAP:

- Interoperability, allowing different programs running in disparate environments to exchange application data
- Relatively simple to implement
- Uses existing well-known vendor-agnostic protocols, such as XML and HTTP
- Widely supported open standard

For more information on SOAP and SOAP implementations, see the W3C Web resource.

EMF and SOAP

SOAP support in EMF allows you to send commands from .Net and Java applications to the EMF server over HTTP. This means you have access to all the functionality of the Java EMF Runtime_API, but you can also initiate the EMF Processes from anywhere on the Internet.

Note: SOAP support is needed by the SQL Server trigger wizard, but it uses older technology. If you wish to make webservice calls to EMF from 3rd party components (such as Microsoft WCF) it is recommend to use the newer EMF API Web Services.

Important: SOAP requires the 'Axis.war' web application that is distributed with EMF to be deployed on a webserver. Jakarta Tomcat 6 that is shipped with EMF can be used for this purpose.

A .Net or Java client application sends a request to the SOAP client, using the Java object model. The SOAP client formulates the EMF request as a SOAP message and then sends this to the SOAP server (e.g. Tomcat) via HTTP. The SOAP server transforms this request

into a normal EMF API call, and sends it to the EMF Server via RMI. The EMF server response is returned to the client in the same way.

A WSDL file is provided so that the requests can be made using standard webservice calls.

If you are using SOAP to run SQL Server triggers, the SQL Server trigger is acting as the client application and sends requests to the SOAP client.

See the SOAP diagram in the [EMF SOAP FAQ](#) topic for details on how SOAP works with EMF.

Configuring the EMF SOAP API

This document describes how set up the SOAP API and generate SQL triggers:

- [Installing the EMF SOAP Components](#)
- [Installing the server component](#)
- [Installing the .Net and COM client component](#)
- [Testing the EMF SOAP API](#)
- [Creating a SQL Server Trigger](#)

Installing the EMF SOAP Components

The SOAP functionality in EMF consists of server and client components. You will need to install both the SOAP client and server components before you can use EMF SOAP. Both the server and client components are included in the standalone SOAP installation on the EMF installation media.

The server and client components `setup.msi` and `axis.war` are available in the `SupportFiles\SOAP` directory of your EMF installation media.

Note: None of the components are installed as part of the EMF installation; they are included as separate files on the installation media.

Installing the server component

The SOAP Server is packaged as a Web archive (`axis.war`) file and should be deployed in a suitable web server. This release has been tested with Jakarta Apache Tomcat version 6. The following should be performed to deploy the server:

1. Install the Sun Java Development Kit (JDK) 1.5 or later. You will need this in order for the JSP pages to compile. If you do not already have this installed, you can install version 1.6.0 (`jre-6-windows-i586.exe`) from the `SupportFiles` directory on the EMF installation media.

2. Install Tomcat 6 from the `SupportFiles/Tomcat` directory.

Important: You must install Tomcat in a directory structure that contains NO spaces for the SOAP API to work correct e.g. `c:\tomcat6`. Do NOT accept the default installation directory.

3. If `axis.war` is not already in the `<TOMCAT_INSTALLDIR>\webapps` directory, copy it from the SOAP directory of your EMF installation media. It should be in the `SupportFiles\SOAP\server` directory.
4. Run `startup.bat` or start/restart the Apache Tomcat Service (Windows) or `startup.sh` (Solaris) in the `<TOMCAT_INSTALLDIR>\bin` directory. An `axis` directory should have been created in the `<TOMCAT_INSTALLDIR>\webapps` directory.
5. To check that the SOAP Server has been correctly deployed, open a Web browser and enter the following URL: `http://localhost:8080/axis/services/xalertsport`

Installing the .Net and COM client component

The SOAP components must be installed on the same machine as SQL Server instance.

Run the `setup.msi` installer from the EMF installation media `SupportFiles\SOAP` directory. If you are installing on Windows Vista or later operating systems, please refer to the `emf6_soap_installer.bat` batch file for installation instructions. This will install the client component. This requires version 2.0 of the .Net Framework - but the installer will detect if this is not present and offer to download it. If you are planning to use the SOAP API to fire database triggers in SQL Server 2005/2008 then during this install you will get the option to install the necessary components into SQL Server 2005/2008.

Important: The SOAP installer must be run on the same machine as the SQL Server instance that will fire the database triggers.

It is possible to install the components into the database after the installation if you don't wish to do it now. Please read the section "How do I setup the SQL Server 2005/2008 Server to be able to register the EMF .NET assembly and run the EMF .NET API" on the trouble shooting page if you are having problems. Ignore the database step if you are planning to run triggers against SQL Server 2000.

This install provides API documentation and examples.

Important: If you are installing the components into SQL Server 2005/2008 then the database user you are using to install the components with must have permissions to create 'logins' in the master database.

Installing the MSoleError stored procedure (only needed for SQL Server 2000)

Before running the generated SQL Server 2000 trigger the stored procedure contained within the `msoleerror.sql` file needs to be added to the SQL Server 2000. Load the `msoleerror.sql` file (from the `SupportFiles\SOAP` directory of your EMF installation media),

into the SQL Server Query Analyzer or any query tool and execute it into your database that contains the trigger.

Testing the EMF SOAP API

A test program has been included in the client install to test the functionality of the SOAP API.

Important: You must install the SOAP client components before you can test the EMF SOAP API.

To test the SOAP API:

1. On the machine where you installed the SOAP client - from the programs menu, select **Aptean->EMF.NET API->EMF.NET API Test Client**.
2. Change the "localhost" on the SOAP connection end point to be the machine running Tomcat.
3. Change the "localhost" on the RMI lookup URL to be the machine running the EMF server (probably the same machine as above). This is the location of the RMI EMF Runtime_API. This will be located on the machine which is running the EMF server. (see the EMF Runtime_API documentation for details).
4. Make sure your EMF server is up and running and then click on the **Test** button - the client will try to connect and you will get an error that the EMF does not exist or does not have an API initiator on it. If this is the case, then create an EMF with an API initiator on it - and enter its alert profile id in the box on the entry field. Testing again should make the alert run.

Other fields in the test client

- **EMF Process Profile ID** - The ID of the EMF Process you wish to run.

Note: The EMF Process MUST be active and have an API in module.

- **User Profile ID** - The ID of the operator with rights to run the EMF Process. This operator must belong to a EMF role that has rights (at least Create) to the Operators component, for example, the Default role.
- **Password** - Password of this operator.
- **Repository Name** - Name of the repository from which the EMF Process will run. (i.e. the name it has inside EMF)

Note: The default values in the URL text boxes assume all components are installed locally.

Creating a SQL Server Trigger

You can create a SQL Server trigger using the EMF Administrator SQL trigger utility. For

more information, refer to [SQL Server Trigger Wizard](#) help.

Important: If you are using SOAP in a SQL Server 7 environment, ensure that you are using a named account (**not** the local system account), with administrator rights, and that the password is not blank.

EMF SOAP FAQ

Q. What does the RMI URL do?

A. This URL allows the SOAP Server to locate the RMI version of the EMF Runtime_API and thus the EMF Server. (i.e. this is where the EMF Server is running)

Q. What does the WSDL URL do?

A. This URL allows the SOAP Client to discover how to talk to the EMF SOAP Server. (i.e. this is where the EMF SOAP server is installed on the web server)

Q. What is in the WSDL file?

A. The WSDL file contains a description of how to use the SOAP EMF Runtime_API including:

- The datatypes supported by EMF.
- The messages used to talk to the SOAP API.
- The location of the EMF SOAP API. (i.e. the URL where messages can be sent so that the SOAP Server is activated).

Q. What happens when a call is made to the SOAP API?

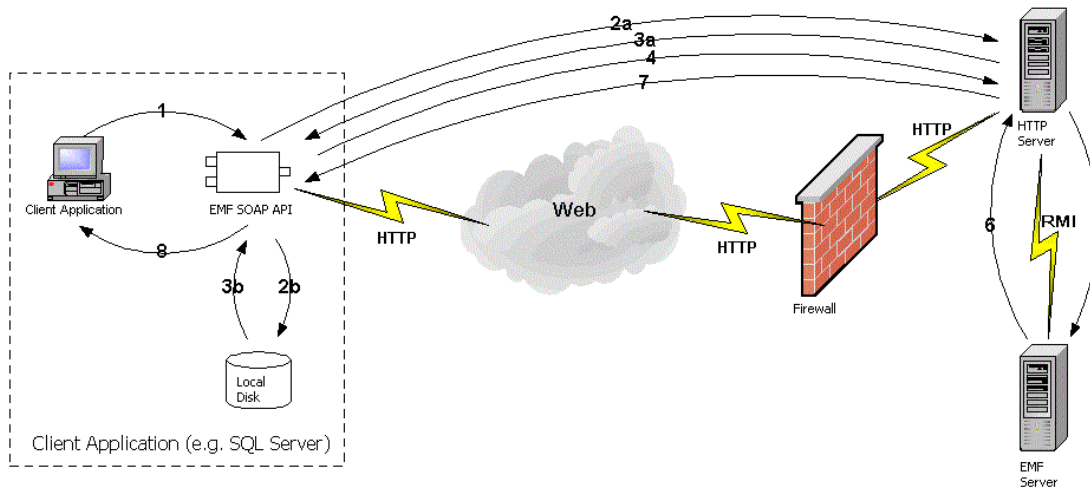
A. See the [SOAP Service Context Diagram](#).

Q. How do I set a different location for the EMF Server?

A. This is done using the RMI Lookup URL parameter passed into the API call.

SOAP Service Context Diagram

The following diagram shows a graphical representation of the various processes that take place when a call is made to the SOAP API.

SOAP Service Context Diagram

Following are the details of the process flow in the SOAP Service Context diagram:

1. Client Application constructs EMF object using COM/Java Object model.
- 2a. SOAP Client requests Web Service Description (WSDL) from webserver.
- 3a. WSDL for EMF is returned by webserver.
- 2b. Alternative flow, WSDL is requested from local filesystem.
- 3b. Alternative flow, WSDL is returned by local filesystem.
4. SOAP Client complies with WSDL and sends SOAP request to webserver.
5. SOAP request is converted into EMF API calls and executed.
6. EMF API returns status of EMF calls.
7. API returns are packaged as SOAP response and reply sent to SOAP client.
8. EMF SOAP client returns status of EMF call to Client Application.

EMF SOAP Troubleshooting Guide

Q. Tomcat will not start - I get an error in the tomcat/log directory saying "Failed creating java jvm.dll."

A. Download and install the latest Visual C++ 6.0 run-time components from <http://support.microsoft.com/kb/259403/en-us>.

Q. I get an error "Could not find stored procedure 'sp_displayoerrorinfo'" when the EMF trigger gets executed.

A. An error has occurred in your trigger and the trigger needs this stored procedure 'sp_displayoerrorinfo' to display the error. Load the msOLEerror.sql file (from the SupportFiles\SOAP directory of your EMF installation media), into the SQL Server Query Analyzer and execute it into your database that contains the trigger. This stored procedure

can also be created in the master database but will need to have execute permissions set on it.

Q. I have just changed something in the trigger by hand and on re-creating the trigger, it now says "Cannot add rows to sysdepends for the current stored procedure because it depends on the missing object 'sp_displayoaerrorinfo' ..."

A. Load the msOLEerror.sql file (from the SupportFiles\SOAP directory of your EMF installation media), into the SQL Server Query Analyzer and execute it into your database (as above).

Q. I get an error "EXECUTE permission denied on object 'sp_OACreate'" when the EMF trigger is triggered.

A. Grant Execute permission on the object (in the above case, the object can be found under the Extended Procedures node in the master database) for the SQL Server user who is accessing the object. You may need to refer to your Database Administrator to grant these permissions.

Q. I have restored a backed-up SQL Server EMF database, but cannot add the EMF.NET assembly to this database.

A. This happens when the owner associated with the backed-up database does not exist when the same database is restored. This can be resolved with the following:

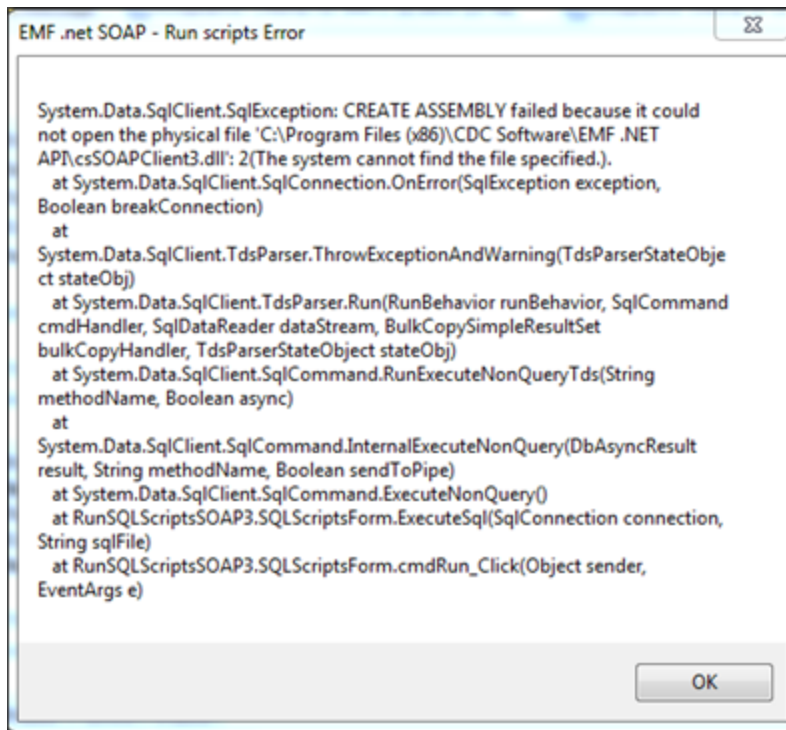
Under database properties of the restored Database, go to the Files page. If the owner is blank, specify an owner so that proper authorization is given for file access.

Q. How do I set up the SQL Server 2005 Server to be able to register the EMF.NET assembly and run the EMF.NET API ?

A.

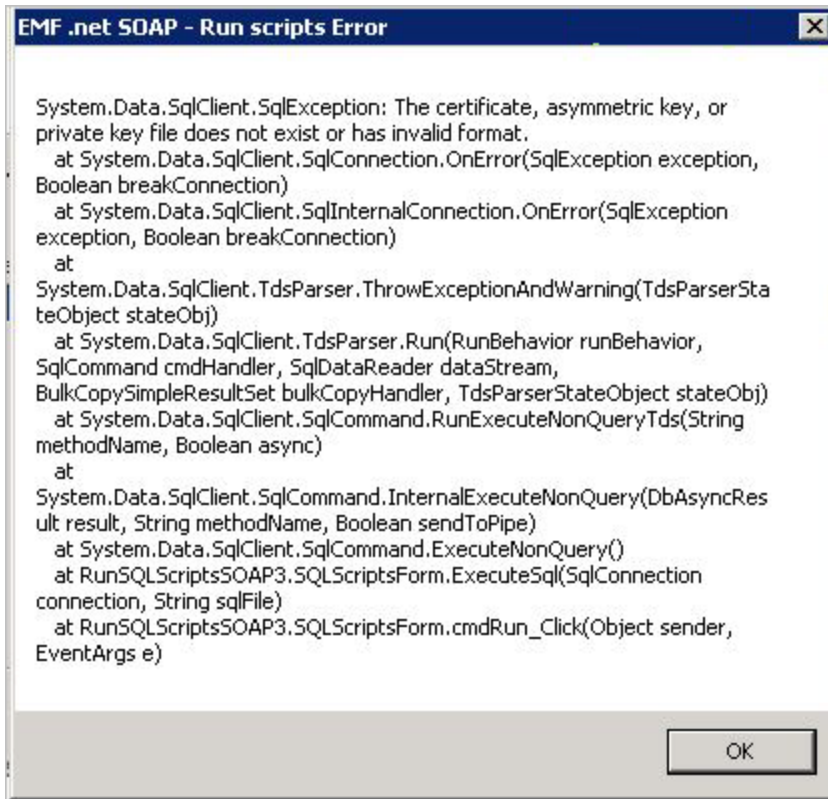
1. In the SQL Server Surface Area Configuration tool that is shipped with the SQL Server 2005, select the Surface Area Configuration for Features option. Under the Database Engine component of the SQL Server instance that is to be changed, select the CLR Integration feature and enable it.
2. Set the Trustworthy property of the Database that will contain the .NET assembly to ON. This can be achieved by running the following query from within the SQL Server Management Studio or any query tool that is being used to connect to the SQL Server:
ALTER database <EMFDatabaseName> set trustworthy on

Q. I get the following error while registering the assemblies:



A. This happens when the .Net installer is not running on the same machine as SQL server. This can be resolved by installing the .Net installer on the same machine as SQL server.

Q. I get the following error while registering the assemblies:



A. This happens when the .Net installer is not running on the same machine as SQL server. This can be resolved by installing the .Net installer on the same machine as SQL server.

Editing the Configuration Database Properties File

As a rule you should set up the connection to the configuration database using the EMF Processes Server Configuration Utility, which saves its settings in the **database properties file**. However, if you prefer, you can define the JDBC connection details manually by using a text editor to modify the contents of this file.

Note: The database properties file is **not** installed with the EMF server, it is created when you modify and save your server settings. The default name and location are **<Repository Name>.properties** in the EMF **\Server** folder, but you can change this using the **DB Config** tab of the Server Configuration Utility or by changing the **<Repository Name>.properties** entry in the database properties file.

If you wish to configure your JDBC connection by editing the configuration properties file:

1. Confirm the name and location of the configuration file (see above).
2. Enter the following **essential** settings:
 - **DriverName=** the name of the JDBC driver class or JDBC datasource class (e.g. com.inet.tds.TdsDriver)
 - **ConnectionString=** the connection string passed to the driver (e.g.

jdbc:inetdae7:databaseServerName:1433). For further information, refer to the JDBC driver documentation. Note that this may not be necessary for datasources.

- **DatabaseType=** identifies the database vendor, which will affect the syntax of the sql command that is sent to the database server. Valid entries are **0** (the default, uses Microsoft SQL Server syntax), **1** (uses Microsoft SQL Server syntax) or **2** (uses Oracle syntax).

Depending on the database that you are using, you may also need to enter the following:

- **user=** the database user name.
- **password=** the user's password.
- **database=** the name of the configuration database.

3. If required, enter the following **optional** settings:

- **xa_minPoolSize=** the number of connections that will initially created in the pool, and held in the pool at all times. The default value is **-1**, which means an unlimited number.
- **xa_maxPoolSize=** the maximum number of connections that the pool will give out. The default value is **-1**, which means an unlimited number. If this value is set to 0, attempts to gain further connections after the minimum number of connections have been retrieved will fail. If this value is set to any number greater than 1, when the maximum number of connections has been retrieved the pool will wait for a connection to become available. If no connection becomes available within the time specified by xa_maxWait (see below) the request will fail.
- **xa_maxIdleTime=** the maximum amount of time (in seconds) that a connection can sit in the pool unused. Any connections that are still in the pool and unused after this time will be released. The default value is **600** (10 minutes).
- **xa_propertyCycle=** how frequently (in seconds) to check for connections in the pool that have been idle for the time specified in xa_maxIdleTime and should be removed. The default value is **600** (10 minutes).
- **xa_oracleFailoverCheck=** allows you to specify whether a check should be made against every Oracle connection to check that the connection is still valid. Enabling this option will allow the EMF server to recover in the event that the network or database fails, but will have a detrimental effect on performance. The default setting for this is **True**.
- **xa_disableInternalPooling=** prevents the server from using its own internal pooling. This allows you to use a third-party datasource that has its own pooling (e.g. oracle.jdbc.pool.OracleConnectionCacheImpl). The default setting for this is **False**.
- **xa_maxWait=** how long (in seconds) to wait for a connection to become available (see xa_maxPoolSize). If set to 0 or less it will wait indefinitely as long as xa_maxPoolSize is set to greater than 0. The default value is **0**.

[Server Manager Overview](#)

Dead Letter Queue Browser

The Dead Letter Queue (DLQ) Browser in EMF allows you to view the messages on the Dead Letter Queue. Any process or process branch that fails to complete because of an error in the process is placed on the Dead Letter Queue. The messages on the Dead Letter Queue is selected to retry or remove permanently.

Note: You must be logged into a repository and the queues must be started for the DLQ Browser to function.

To view the Dead Letter Queue Browser

- Click **Window > Dead Letter Queue Browser**.

The **Dead Letter Queue Browse table** is displayed with the following information about the messages on the Dead Letter Queue:

- **Id:** The JMS message ID in the DLQ.
- **Process Id:** The process ID that the message belongs to.
- **Process Instance Id:** The instance identifier of the processed EMF Process. If an EMF Process is fired on 10 separate occasions, then there must be 10 EMF Process Instances with a unique ID.
- **Process name:** The name of the process.
- **Queue:** The queue in which the message is processed when it fails before being placed in the Dead Letter Queue.
- **Date:** The date the message is created.

The following option is available on the right-click menu for each entry in the table view:

- **View:** Displays the details of the selected entry.

JMS Data Entry

Property	Value
Id	ID:ukjrafiq-61486-1418221398202-5:4748:1:1:1
Date	10/12/14 15:42:26
csEMFVersion	6.2
csRepositoryId	1
csQueueName	csPIQ
csAlertId	1036
csAlertInstanceId	1138

```

<?xml version="1.0" encoding="UTF-8" ?>
<DEBUG_OUTPUT>
  <ALERT_STATE AbortFlag="false" AlertInstanceId="1138"
    AlertName="runexe" AlertProfileID="1036" BranchComplete="false"
    BranchInstanceId="1" BranchSynchronized="false"
    CacheGroupKey="null" CacheGroupName="null"
    ClassVersion="1.18/27" ClearDataStoreCache="false"
    CurrentRetry="0" DebugBreakPoint="-1" DebugLogLevel="2"
    DebugMode="false" ErrorAlertProfileID="0"
    ErrorHandlingEnabled="false" EscalationRefBranchHashCode="0"
    EscalationRefModuleID="0" EventArrived="false"
    EventArrivedEarly="false" EventArrivedLate="false"
    EventArrivedOnTime="false" EventCancelled="false"
    EventExpectedFirstRun="false" EventOutOfSequence="false"
    EventOverdue="false" ExpectedRunTimeInterval="10"
    ExpectedRunTimePeriod="1" HaltFlag="false" LastModuleID="0"
    LogBilling="false" LogStartComplete="true" LogStatistics="true"
    LogUserMessages="true" PersistAlertWhenQueued="true"
    Priority="4" RepositoryID="1" RetryDelay="0" RetryThreshold="0"
    SyncOnly="false" UseDefaultDLQExpiry="true">
    <SYNC_JOIN_INFO_LIST/>
    <SYNC_SPLIT_INFO_LIST/>
    <ALERT_STATE_HANDLER_MODULE_LIST/>
    <LOOP_STACK/>
    <ALL_MODULE_TYPES_BILLING_OPTIONS/>
    <ALERT_DATA ClassVersion="1.10/14">
      <MESSAGE_SECTIONS NumberMessageSections="0"/>
      <DATA_SECTIONS NumberDataSections="0"/>
      <RECIPIENT_SECTIONS NumberRecipientSections="0"/>
      <ENTITY_SECTIONS NumberEntitySections="0"/>
    </ALERT_DATA>
    <NEXT_MODULE_DESCRIPTION/>
  </ALERT_STATE>
</DEBUG_OUTPUT>

```

< >

Close

The following buttons are available in the DLQ Browser:

- **Connect / Disconnect:** Connects to or Disconnects from a System Queue Provider defined within EMF.
- **Refresh:** Refresh the table view.
- **Delete Selected:** Deletes the selected entry from the Dead Letter Queue.
- **Delete All:** Deletes all the entries from the Dead Letter Queue.
- **Retry Selected:** Removes the selected entry from the Dead Letter Queue and retries running the process by returning it to the original queue.

Note: The retry returns to the start of the initiator or the start of the branch it failed on.

Exclusive Queue - Single Thread Operation/Message Order Dependency

There are instances when the sequence of operations in a process are order dependant or that a process branch must be run mutually exclusive. This can be achieved by creating queues that are known as **EXCLUSIVE**. This will pick up processes in the order they are sent and will not receive another process until the **Exclusive Queue** has finished processing the current process branch. In order for this to work in EMF, the queue must be set to be **Exclusive**.

To set the queue to Exclusive

1. Select the **Exclusive Queue** check box on the **System > Queues > Queue** (for more information, see [Creating Queues](#)).
2. Create a server manager that listens to the **Exclusive Queue** (for more information, see [Server Manager](#)).

Note: The server manager must have only one worker thread defined (as it will process only a single process at a time).

3. Change the **ActiveMQ JNDI** properties and restart the EMF server.

Place the following entries in the `jndi.properties` file in the `<EMF install folder>\auto_jar\activemq_jndi_properties` and `<EMF install folder>\server` directories.

- `queue.<QUEUE_NAME> = <QUEUE_NAME>`
- `queue.<QUEUE_NAME>.exclusive = <QUEUE_NAME>?consumer.exclusive=true`

Where, `<QUEUE_NAME>` is the lookup name of the queue defined in EMF.

External Queues

External queue can also defined as **Exclusive**, ensuring messages are picked up in

sequence. However, once the **Queue Listener** has picked up the message and queued the process to run – if the process is not being run on an **Exclusive** system queue, then the processes may be run in any order simultaneously.

Production Mode

Production Mode is a mode in EMF designed to improve server performance. In normal, or default mode, various setting for a process are re-read from the repository database every time the process is run. If the repository is not changed, and all the design work is finished, then this is a wasteful overhead.

Production mode allows the EMF administrator to flag that no changes will be made to the repository (no processes edited, JDBC settings changed, recipients added, service settings changed and so on) so that the server can intelligently optimise performance, and avoid redundant database queries.

In tests run, it was shown that using this setting gave an average 20% performance improvement. This however will vary from scenario to scenario depending on the process design and different hardware/database speeds.

Important: Once the server is running in Production Mode, any changes to the EMF repository may be ignored and/or can lead to unpredictable results/errors in the server. It is recommended not to edit any items in the repository while the server is running in this mode.

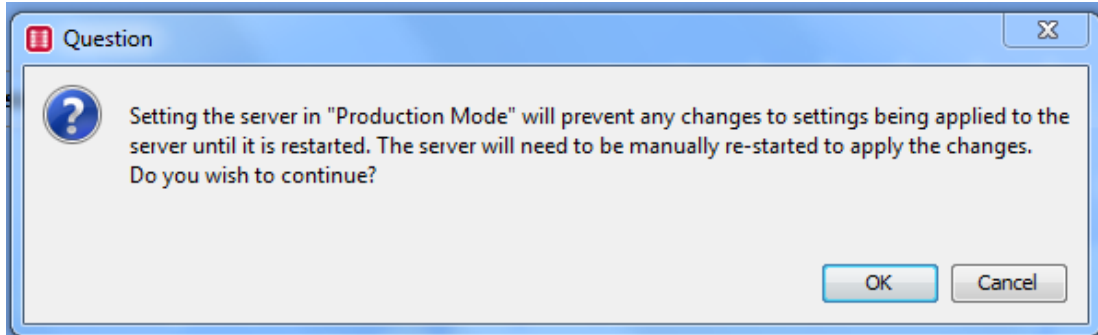
To turn on Production Mode:

1. Right-click the repository in the EMF tree view for which you want to change the setting and select **Set Server In Production Mode**.

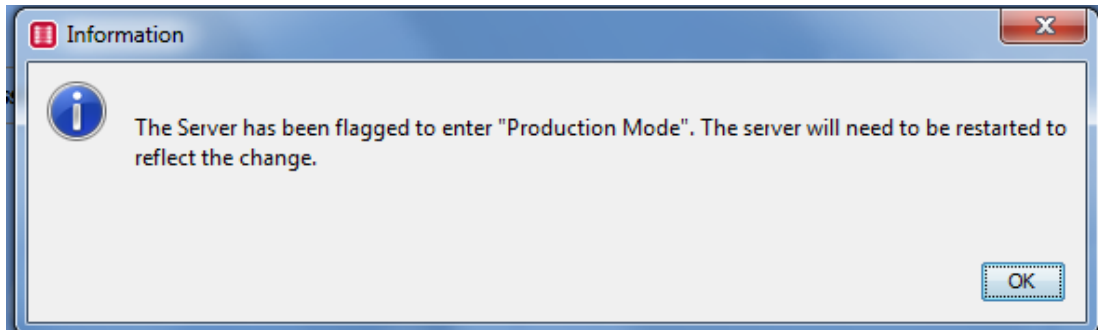


Note: You must restart the server for the change to take effect.

2. A message box appears warning you that the module settings cannot be changed while in production, click **OK**



3. Another message box will appear informing you to restart the server for the change to take effect, click **OK**.



4. Restart the server.

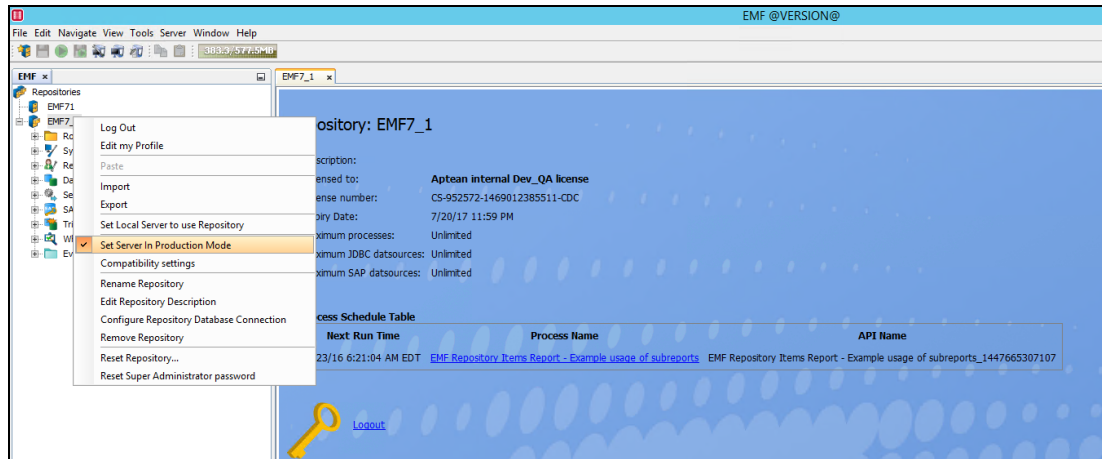
The server should now be in production mode.



To turn off Production Mode:

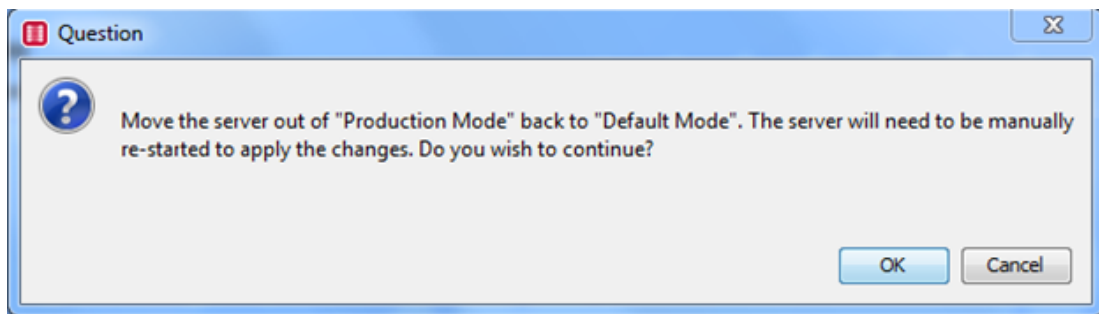
1. Right-click the repository in the EMF tree view for which you want to change the setting and select **Set Server In Production Mode**.

The selection will be cleared when the change takes effect.

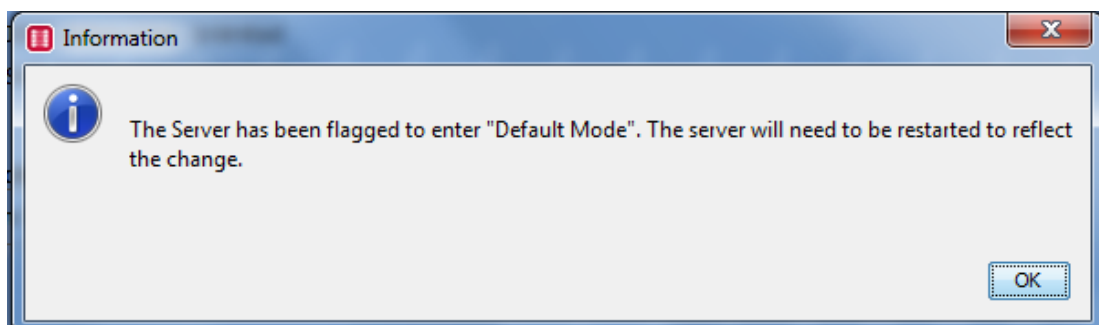


Note: You must restart the server for the change to take effect.

2. A message box will appear confirming the change, click **OK**.



3. Another message box will appear informing you that the Default Mode has been set and to restart the server for the change to take effect, click **OK**.



4. Restart the server.

The server should now be in Default Mode.



Event Plans

The Event Plan Concept

An "Event plan" allows related "events" to be tracked through a business process. An event can be created from anything that can be detected by a system (for example, SMS message arrived, database record changed, file appearing on file system), and will typically have data associated with it (for example, order number).

Actions are defined in the Event plan for what happens as each event does, or does not arrive. Typically, an event type is set to instantiate an instant of an event plan (for example, *New Order Received*). Further, events are then waited upon in that event plan instance that correlate directly back to the initiating event (for example, have the same order number).

Note: Multiple event plan instances of the same event plan run at the same time.

Example

A business process can be modelled to track the despatch and delivery of a package. Events such as *New Order Received*, *Order Picked*, *Item Despatch*, and *Item Delivered* are modelled in the **Event Plan**. The plan controls the whole sequencing of operations.

1. When an email arrives with a *New Order*, a new Event Plan instance is started.
2. This kicks off a process to cause the order to be picked.
3. Once the order is picked, the user updates the system to "picked".
4. The update to the picking system is detected and an *Order Picked* event is fired.
5. The event plan receives the ordered picked event, and starts the despatch item process.
6. The warehouse worker receives the instruction to load the item onto the truck.
7. The truck driver sends an SMS message to say the truck is leaving.
8. The SMS message is received, and an *Item Despatch* event is fired.

9. The Event Plan receives the *Item Despatch* event and sends an email to notify the customer that the order is on the way.

Some characteristics of Event Plans are:

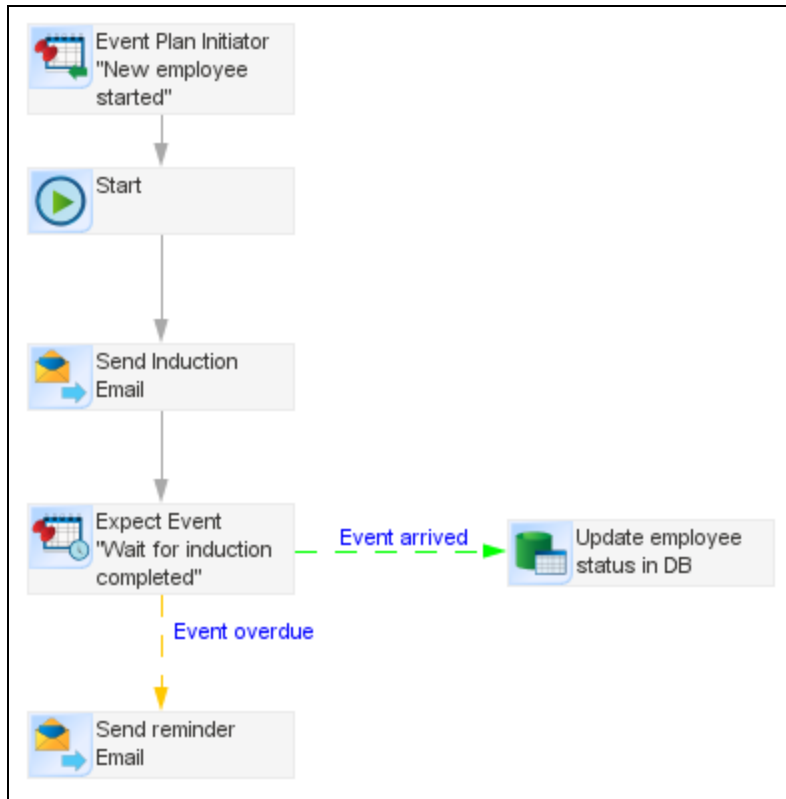
- Tie together related events within a single event plan instance, and have the visibility as to what stage of processing a particular plan instance is at.
- Can be long running, taking weeks, months or even years to complete. For example, an event plan that models a customer ordering a new car, through its manufacture, to eventual delivery could take months to complete.
- Allow different actions to be taken when events are overdue, arrive early or late.
- Multiple identifier mappings may be used across a plan. For example, in an instance of a plan initially tied to an order id, the delivery company may use the delivery id in event notifications. The system must be able to correlate the order id to the delivery id in the plan instance.
- Different plans may act upon the same event. For example, an event plan to track an order and an event plan to manage a stock system may both take action on the same "new order" event. Multiple processes can be waiting for the same event, or multiple event plans can be started from a single event.
- Separating out the detection and creation of events, from the processing of events to build more flexible and maintainable systems.

Implementation of Event Plans in EMF

In EMF, an Event Plan is an EMF Process that has some Event handling modules in it. The following modules are available to allow Events to be created and processed:

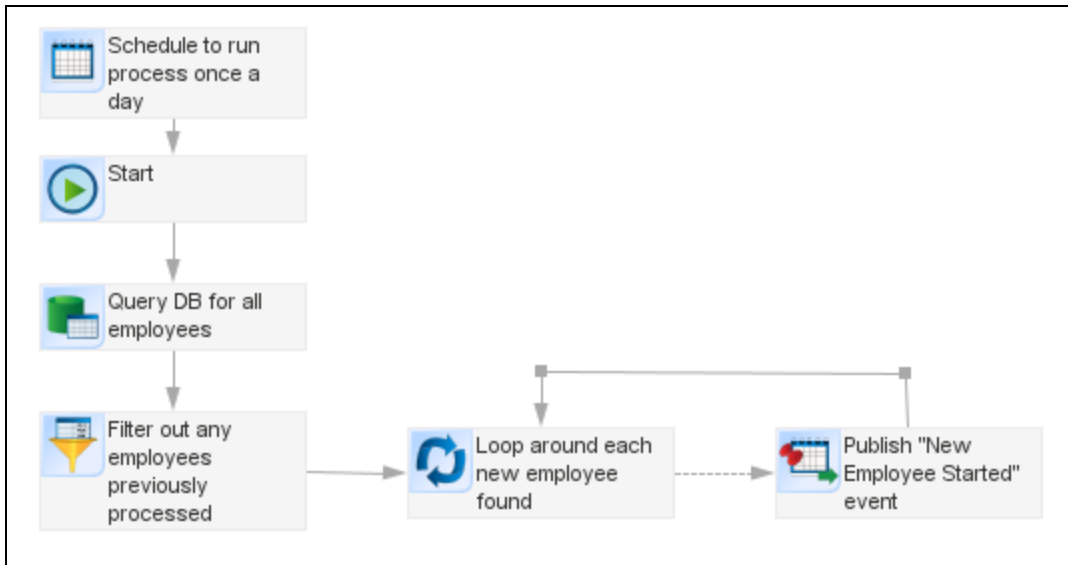
- [Publish Event module](#) – allows events to be created and published in the system.
- [Event Plan Initiator module](#) – starts a new process instance (event plan instance) when a particular event type is published.
- [Expect Event module](#) – waits for an event(s) in a process to be published. Different actions can be taken if the event is overdue/early/late.

Typically, an event plan will contain an [Event Plan Initiator](#) module and zero or more [Expect Event](#) modules. The following process demonstrates how an event plan may look to deal with a new employee starting (the email message text/recipient modules have been left off for the sake of simplicity).



In the example, it is irrelevant how we detected that a "new employee has started", or that their "induction is completed". These events could have been created in the system in different ways (for a company that has different "employee" systems for different departments can still ensure all employees follow the same induction process). Different "Event Plans" can also be built against the same "New employee started" event (one to initiate the induction process, another to update the wage system and so on).

To publish events into the system that the Event Plans can operate against, separate processes are typically built that will detect the new data/changed data and change that data into an event using the [Publish Event](#) module. Continuing the previous example, the following shows how new employees could be detected from the company employee database and published as events.



Any process instance that is initialised (started) by an [Event Plan Initiator](#) module becomes an Event Plan instance. Processes that are not started by an [Event Plan Initiator](#), but contain an [Expect Event](#) module, become Event Plan instances when the first [Expect Event](#) module within the process runs.

Every event that is published is associated with an [Event Definition](#) (Event type). The Event definition identifies the type of event (for example, "New Employee Started") and the type of data that is associated with the event.

To build a new Event Plan Process, the following sequence is typically followed:

1. Create the new Event Definitions required.
2. Build the Event Plan process using [Expect Event/Event Plan Initiator](#) modules.
3. Build the processes to publish the events needed by the Event Plan process.

See [Example steps](#) for a set of instructions on how to build a sample event plan.

Monitoring of Events and Event Plans

At runtime, there are two tools that can be used to monitor events and event plans:

1. The state of any event plan instance can be monitored in the system by using the [Event Plan Monitor](#). It shows the events that have been correlated to that plan and those that each instance is still waiting on.
2. The [Published Event Store](#) is used to view all events that the system has published, and whether or not they have been associated with an Event Plan instance. Events can be stored in the system until they are needed. Events can also be manually published instantly without the need to create a separate publishing process and running it.

Additional Event Plan System Components

There are a number of additional server managers that control the smooth running of

Event Plans. These are set up by default and usually do not require adjustment.

- [Running Event Plan Manager](#)
- [Missed Events Manager](#)
- [Event/Event Plan Clean-up Manager](#)

[Event Definition](#)

[Publish Event Module](#)

[Event Plan Initiator Module](#)

[Expect Event Module](#)

[Event Plan Monitor](#)

[Published Event Store](#)

[Running Event Plan Manager](#)

[Missed Events Manager](#)

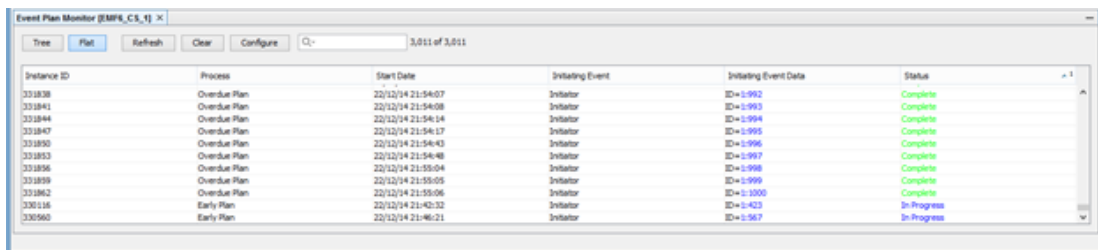
[Event/Event Plan clean-up Manager](#)

Event Plan Monitor

The **Event Plan Monitor** provides the status of each event plan that has been started. A plan is a running process instance that was either initiated by an [Event Plan Initiator](#) module, or a process instance that has processed at least one [Expect Event](#) module. It displays each event plan's details within a table that gives the user access to details about which events have been processed, and which are being waited upon.

To view the Event Plan Monitor:

- Click **Window** and select **Event Plan Monitor** to display the **Event Plan Monitor** window.



The screenshot shows the 'Event Plan Monitor (EMF6_C5_1)' window. It contains a table with the following columns: Instance ID, Process, Start Date, Initiating Event, Initiating Event Date, and Status. The table lists several instances, mostly 'Overdue Plan' and 'Early Plan', with their respective start dates and statuses. Most are 'Complete', while the last two are 'In Progress'.

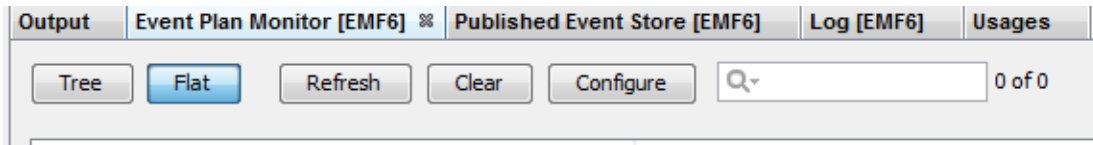
Instance ID	Process	Start Date	Initiating Event	Initiating Event Date	Status
331838	Overdue Plan	22/12/14 21:54:07	Initiator	ID=1:992	Complete
331841	Overdue Plan	22/12/14 21:54:08	Initiator	ID=1:993	Complete
331844	Overdue Plan	22/12/14 21:54:14	Initiator	ID=1:994	Complete
331847	Overdue Plan	22/12/14 21:54:17	Initiator	ID=1:995	Complete
331850	Overdue Plan	22/12/14 21:54:43	Initiator	ID=1:996	Complete
331853	Overdue Plan	22/12/14 21:54:48	Initiator	ID=1:997	Complete
331856	Overdue Plan	22/12/14 21:55:04	Initiator	ID=1:998	Complete
331859	Overdue Plan	22/12/14 21:55:05	Initiator	ID=1:999	Complete
331862	Overdue Plan	22/12/14 21:55:06	Initiator	ID=1:1000	Complete
330116	Early Plan	22/12/14 21:42:32	Initiator	ID=1:423	In Progress
330560	Early Plan	22/12/14 21:46:21	Initiator	ID=1:567	In Progress

The table displays all the event plan instances currently in the system. The following information is displayed for each in this view:

- **Instance ID** - The **Instance ID** of the event plan (this is also the process instance id).
- **Process** - The name of the process that represents this plan.
- **Start Date** - The date when the plan was started.
- **Initiating Event** - The name of the event definition that initiated the event plan.

- **Initiating Event Data** - The event data contained within the initiating event.
- **Status**: The status of the event plan, Complete or In Progress.

The following operations are available:



- **Tree** - Displays the event plans grouped by the **Process** column.
- **Flat** - Displays the event plans in a flat view with no grouping.
- **Refresh** - Re-queries the repository and updates the table.
- **Clear** - Deletes all event plans from the repository that have a Complete status. All events associated with the event plan are also removed from the [Published Event Store](#) if no other event plan is associated with those events.
- **Configure** - Determines how completed event plans are removed from the repository by the [Event Plan Clean-up Manager](#). See [Completed Event Plan Clean-up Configuration](#).
- **Search** - The search bar allows you to filter the results displayed in the Event Plan Monitor table.

The following operations are available when you right-click an entry:

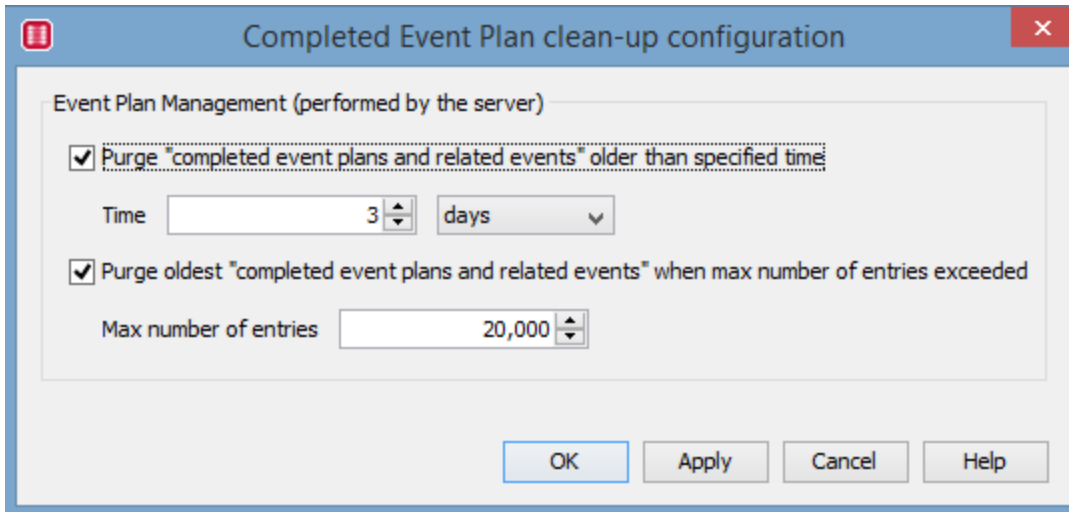
- **View** - Displays the details of an entry in the [Event Plan Entry](#) dialog.
- **Delete** - Deletes an event plan. The event plan to delete can be in any status (complete/in progress). All events associated with the event plan are also removed from the [Published Event Store](#) if no other event plan is associated with those events.
- **Select in explorer** - Opens the EMF Tree View and selects the event plan process.

[Event Plans](#)

[Event Plan Modules](#)

Completed Event Plan Clean-up Configuration

The **Completed Event Plan clean-up configuration** determines how completed event plans are removed from the repository by the [Event Plan Clean-up Manager](#).



There are two different clean-up options that can be selected:

1. **Purge “completed event plans and related events” older than specified time**
- allows all completed event plans and related events to be cleaned from the repository if it is older than the time specified.
2. **Purge “completed event plans and related events” when max number of entries exceeded** - allows all completed event plans and related events to be cleaned from the repository when a certain number of entries has been exceeded. The cut off level is configured in this dialog.

Note: It is possible to select both check boxes. If both are selected, the clean-up will happen in both the cases of time expiry and entry exceeded.

Note: If any changes are made to the configuration, you must restart the server for the changes to be applied.

Event Plan Entry

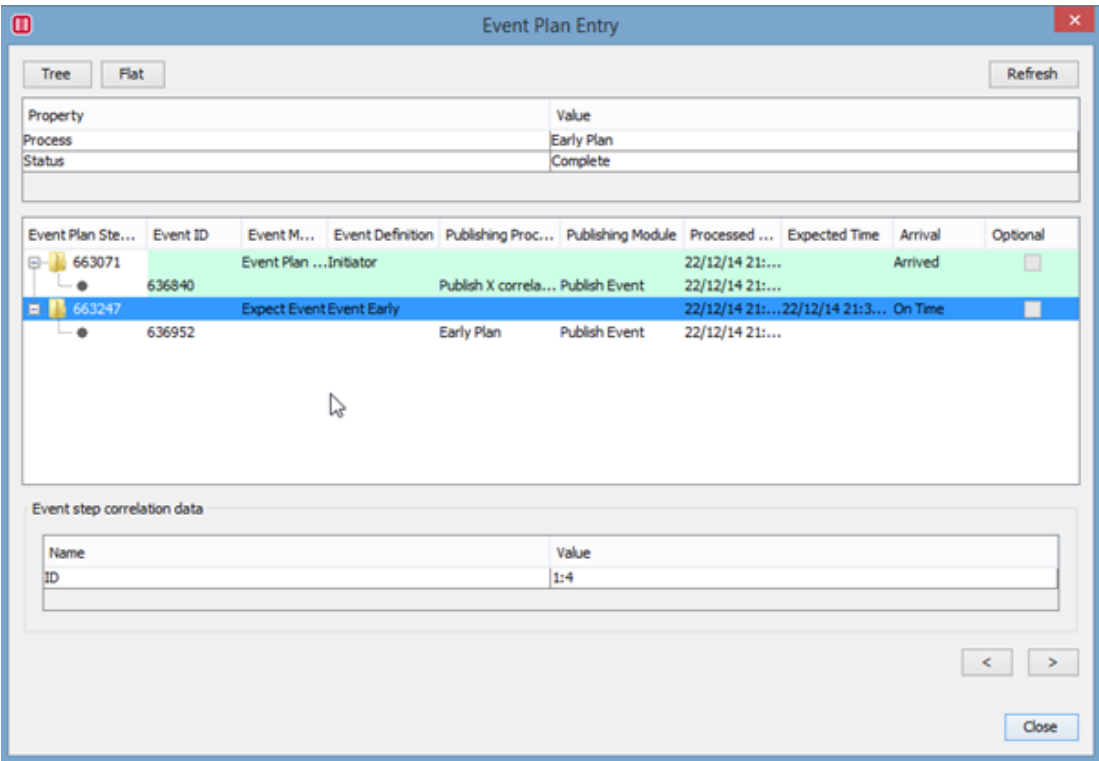
The **Event Plan Entry** dialog box displays a detailed view about the selected Event Plan instance. It will show all the events being waited on, and those that have arrived.

To view the Event Plan Entry

- Double-click an entry in the [Event Plan Monitor](#), or right-click and select **Open**.

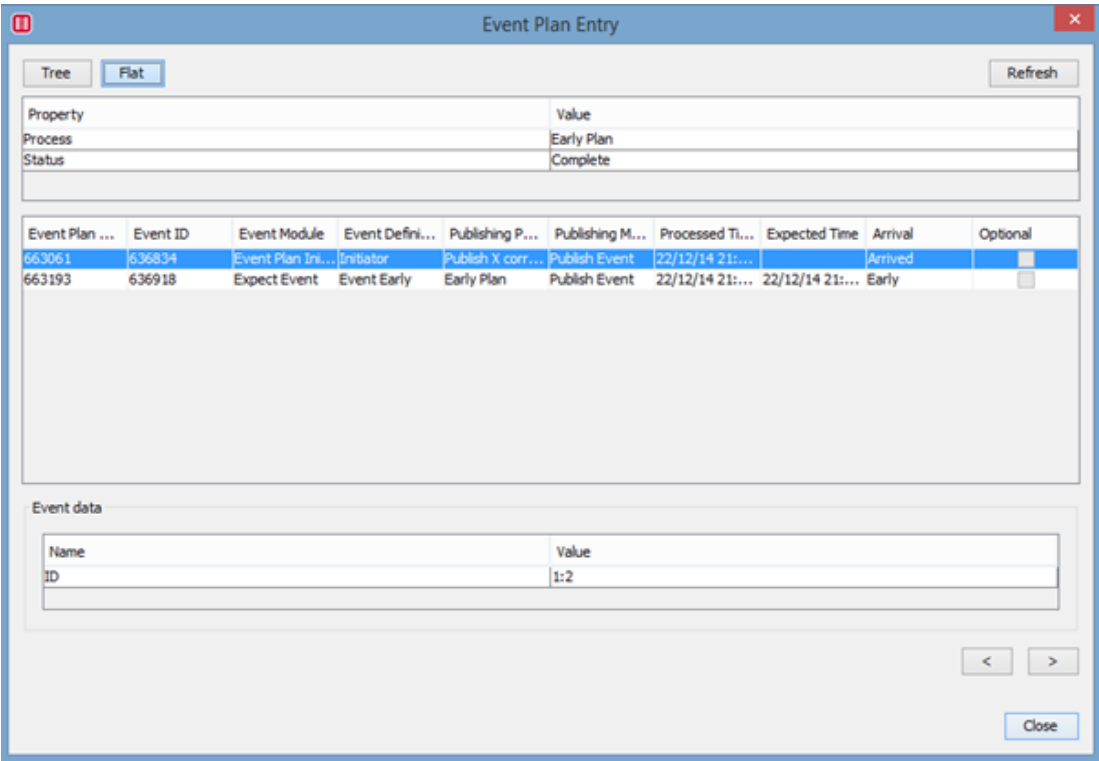
Tree View

The tree hierarchy structure shows the [Event Plan Initiator](#) (in cyan) and the different [Expect Event](#) modules that have run in the process (any expect event modules that are yet to run will not be shown). These rows are known as the "Event Plan Steps". Each step can then be expanded which displays any event that has arrived for that step. A single step can correlate with multiple events, if the event count on the [Expect Event](#) module is set as more than one. An [Event Plan Initiator](#) step will only ever have one event associated with it, and that event will have always arrived.



Flat View

The flat view structure shows all the Events as individual entries (the [Event Plan Initiator](#) shown in cyan).



The first table displays the event plan details:

- **Process** – The name of the process that started the Event Plan.
- **Status** – The Event Plan progress.

The tree/flat view tables display the following information:

- **Event Plan Step ID** – An identifier for the specific [Event Plan Initiator](#) or [Expect Event](#) module.
- **Event ID** – The identifier of the Event.
- **Event Module** – The name of the [Event Plan Initiator](#) or [Expect Event](#) module that created this step.
- **Event Definition** – The **Event Definition** associated to this entry.
- **Publishing Process** – The name of the process that published the Event (or empty if manually published).
- **Publishing Module** – The name of the module that published the Event (or empty if manually published).
- **Processed Time** – The date/time when the step was completed and the required number of events were processed. The value will be empty if the step is still waiting on more events.
- **Expected Time** – The date/time when the Event is supposed to be processed. It will be in red if the events arrived late, or are overdue.
- **Arrival** – The arrival status (*Early, On Time, Late, Overdue*) of this event. If the event cannot be determined if it is early/on time/late (for example, because early/late options are not specified in the expect event module), then the status will just be *Arrived*.
- **Optional** – Indicates if this Event is optional for the completion of the Event Plan. See [Expect Event](#) module for more information on optional events.

The following operations are available when you right-click on an Event Plan Entry step:

- **Manually Publish Event** – If the event step is yet to complete, this option will be available to generate the event needed to match to this step:
 - For simple event definitions, the correlation data will be automatically filled in and read-only. Values for properties in the event definition that are not correlated on can be edited.
 - For free form event definition, the data has to be entered manually.
- **Cancel** – If the individual event step entry has not been processed (for example, No Arrival), then it may be cancelled and the Event Plan will not need the event step to complete.

Published Event Store

The Published Event Store contains all the Published Events. Events are published through either the [Publish Event](#) module or manually published through the Published Event Store UI or the Event Monitor UI. Events in the store can be in one of the following two states:

- **Processed** – The event has been correlated with at least one event plan (either an [Event Plan Initiator](#) or an [Expect Event](#) module). No further processing will happen to

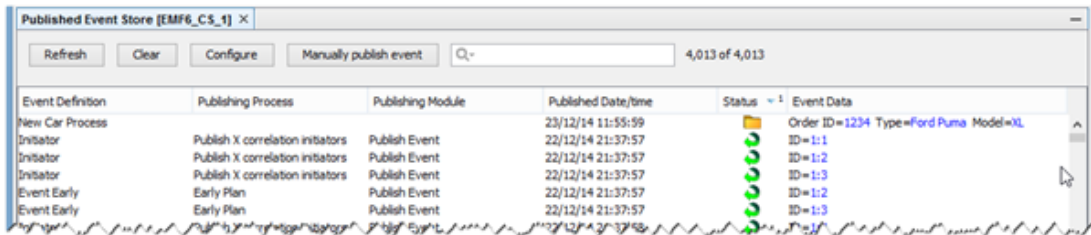
this event.

- **Unprocessed** – Nothing has yet correlated to this event. It remains in the store and will be used to match against [Expect Event](#) modules as they run to try and gain a positive match.

Note: When an event is published, if the option is not selected to store the event if nothing is waiting for it, and nothing matches with the event when it is published, then the event is discarded and will not appear in the Published Event Store.

To view the Published Event Store

Click **Window > Published Event Store** to display the **Published Event Store** window.



The following columns of information are displayed:

- **Event Definition** – The name of the **Event Definition**.
- **Publishing Process** – The process which published the Event (blank if the event was manually published).
- **Publishing Module** – The name of the module in the publishing process that published the Event (blank if the event was manually published).
- **Published Date/time** – The date/time when the event was published.
- **Status** – Processed (green arrow) / Unprocessed (yellow folder).
- **Event Data** – The data assigned to the event.
- **Missed Event Retry Counter** – (column hidden by default) The number of retry attempts made by the [Missed Events Manager](#) to match this event. For more information, see [Missed Events Manager](#).

To view the details of an individual Event, double-click on any entry and use the previous and next arrows to scroll through the events (for more information, see [Published Event Store entry detail](#)).

The following operations are available in the Published Event Store:

- **Refresh** – Refreshes the view.
- **Clear** – Removes all Unprocessed (Status is yellow folder) Events.
- **Configure** – Determines how old unprocessed events are cleared from the store by the [Event Plan Clean-up Manager](#). For more information, see [Published Event Store clean-up configuration](#).
- **Manually Publish Event** – Publish an Event without the need of running a Process. For more information, see [Manually publish event](#).

The following operations are available when you right-click on one or more entries in the table:

- **Open** – Opens the Published Event entry detail for that event.
- **Republish Event** – Publishes another event with the same data.
- **Republish Similar Event** – Publishes another event using the same Event Definition but will allow for the Event Data to be modified.
- **Select the publishing process in explorer** – Opens the EMF Tree View to the associated Publishing Process.
- **Show in Event Plan Monitor** – Opens the [Event Plan Monitor](#) showing this event only in the monitor.
- **Delete** – Deletes the event if unprocessed.
- **Reset Missed Event Retry Counter** – If the event is unprocessed, it will allow reset of the retry counter used by the [Missed Events Manager](#).

[Event Plans](#)

[Event Plan Modules](#)

Manually Publish an Event

Manually publishing an event allows events to be published without the need to build a process to run them. You can do this to test Event Plans that have been built, or to publish events that have failed to turn up (for example, a network failure may prevent the arrival of an SMS message that would have been turned into an event).

Manually publish event

Event Definition *

Simple Mapping

Event Property	Value

Free Form Mapping

Event Data

Action if nothing waiting for event

☐ Do nothing ☒ Store Event until required

Manually publish event Close

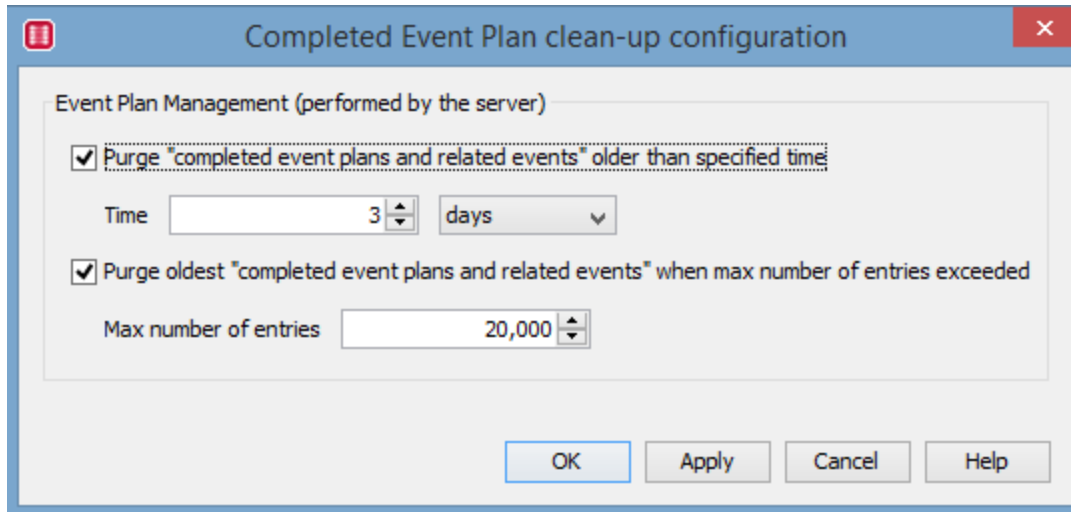
To manually publish an event

1. Select the **Event Definition** to publish.
2. Depending on the definition type selected (free form or simple), either a table of event definition properties or a text area will be presented. Enter the data required for the event.
3. Select the **Action if nothing waiting for event**:
 - a. **Do nothing** – If nothing correlates to the event, then the event will be discarded.
 - b. **Store Event until required** – If nothing correlates to the event, then it will be put into the published event store and used in future correlation attempts.
4. Click **Manually publish event** to publish the event manually.

Note: If the publishing of the event will cause an event plan to start/continue, then the EMF queue provider must be running (in a default installation this means that the EMF server will need to be running). If the EMF UI cannot connect to the queue provider, then the publishing of the event will fail.

Completed Event Plan Clean-up Configuration

The **Completed Event Plan clean-up configuration** determines how completed event plans are removed from the repository by the [Event Plan Clean-up Manager](#).



There are two different clean-up options that can be selected:

1. **Purge “completed event plans and related events” older than specified time**
- allows all completed event plans and related events to be cleaned from the repository if it is older than the time specified.
2. **Purge “completed event plans and related events” when max number of entries exceeded**
- allows all completed event plans and related events to be cleaned from the repository when a certain number of entries has been exceeded. The cut off level is configured in this dialog.

Note: It is possible to select both check boxes. If both are selected, the clean-up will happen in both the cases of time expiry and entry exceeded.

Note: If any changes are made to the configuration, you must restart the server for the changes to be applied.

Published Event Store Entry Detail

The **Published Event Store Entry** screen provides a detailed view of all the information about a single entry in the [Published Event Store](#).

To view the Entry

- Double-click an entry in the [Published Event Store](#), or right-click and select **Open**.

Published Event Store entry detail

Property	Value
Event Definition	Event Early
Publishing Process	Early Plan
Publishing Module	Publish Event
Published Date/time	22/12/14 21:42:36
Status	Processed
Missed Event Retry Counter	3

Event Data

Event Property	Value
ID	1:295

Triggered Processes

Triggered Process	Triggered Module
Early Plan	Expect Event

< >

Republish Event Republish Similar Event Delete Close

The first table in the dialog box gives a summary of the event information. The rows are:

- **Event Definition** – The name of the Event Definition.
- **Publishing Process** – The process which published the Event (blank if the event is manually published).
- **Publishing Module** – The name of the module in the publishing process that published the Event (blank if the event is manually published).
- **Published Date/time** – The date/time when the event is published.
- **Status** – Processed / Unprocessed.
- **Missed Event Retry Counter** – (column hidden by default) The number of retry attempts made by the [Missed Events Manager](#) to match this event. For more details, see [Missed Events Manager](#).

The event data is displayed after the first table. The format of this will depend on the event definition type (simple or free form). The table after the event data displays the processes (event plans) that were triggered by this event (that is, the modules that a positive match occurred against).

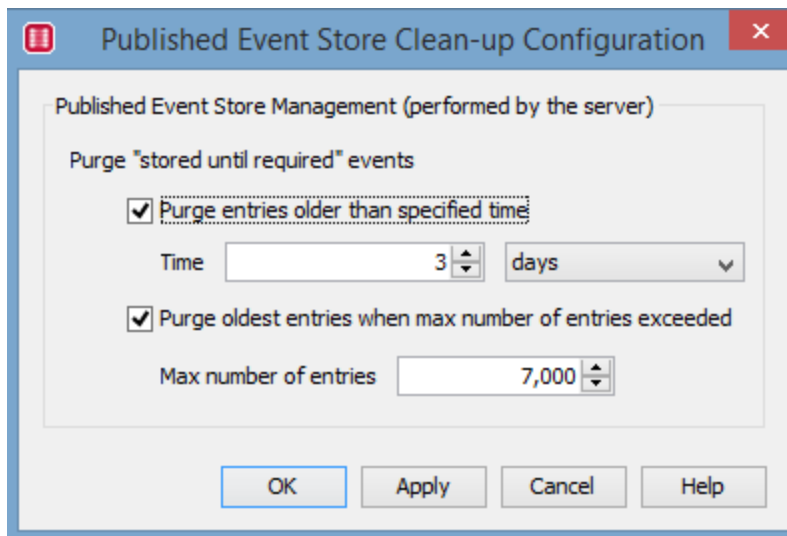
The following operations are available in the detail entry:

- **Republish Event** – Causes an exact copy of the same event to be published again.
- **Republish Similar Event** – Similar to **Republish Event** operation, except the event data can be changed before the event is published again.

- **Delete** – Removes the event from the published event store. Only unprocessed events can be removed.

Published Event Store Clean-up Configuration

The published event store clean-up configuration determines how old unprocessed events are removed from the [Published Event Store](#) by the [Event Plan Clean-up Manager](#).



There are two different clean-up options that can be selected:

1. **Purge “stored until required” events older than specified time** - When selected, unprocessed events older than the specified time will be deleted from the published event store.
2. **Purge “oldest entries” when max number of entries exceeded** - When selected, if more than the specified number of unprocessed events exists, then the oldest unprocessed events will be removed until the number of entries is reduced to the configured amount.

Note: It is possible to select both check boxes. If both are selected, the clean-up will happen in both the cases of time expiry and entry exceeded.

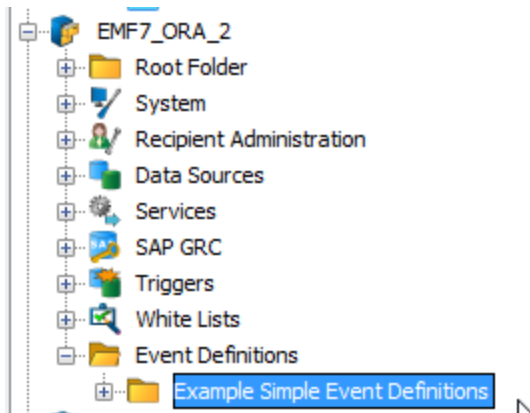
Note: If any changes are made to the configuration, you must restart the server for the changes to be applied.

Example steps to create a simple Event Plan

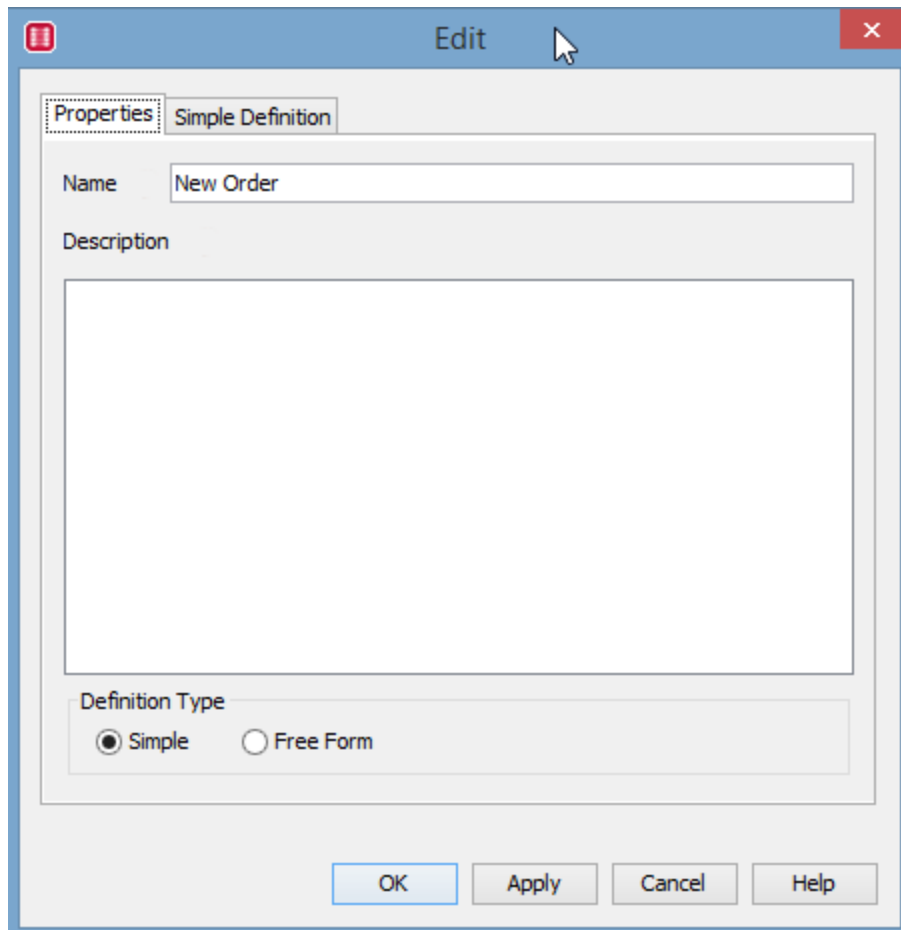
The following steps show how to create an example event plan that is initiated by one type of event (new order event), and then waits on another event (order fulfilled event) before completing. Publish event processes are not shown instead all events are published manually. The exported event plan and definitions can be downloaded from this [link](#) and imported into your repository.

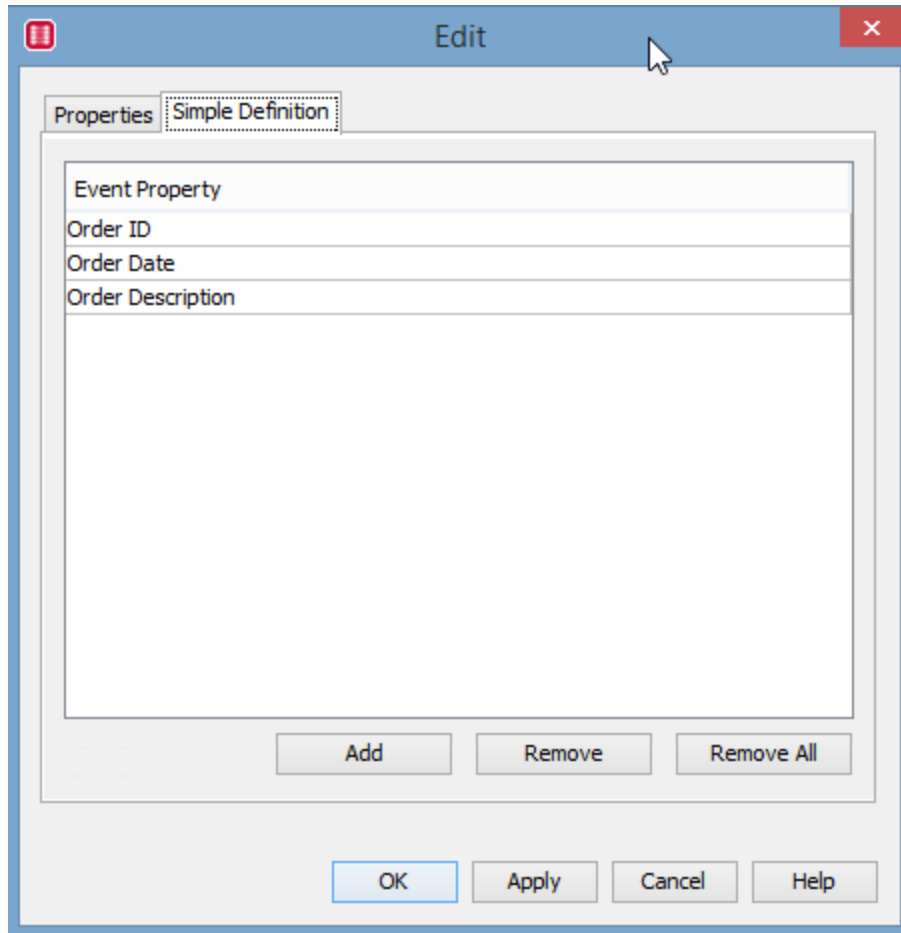
Stage 1 – create the event definitions

1. Create a new "Event Definition" folder called "Example Simple Event Definitions"



2. Create a new "Event Definition" with the settings illustrated in the following images:





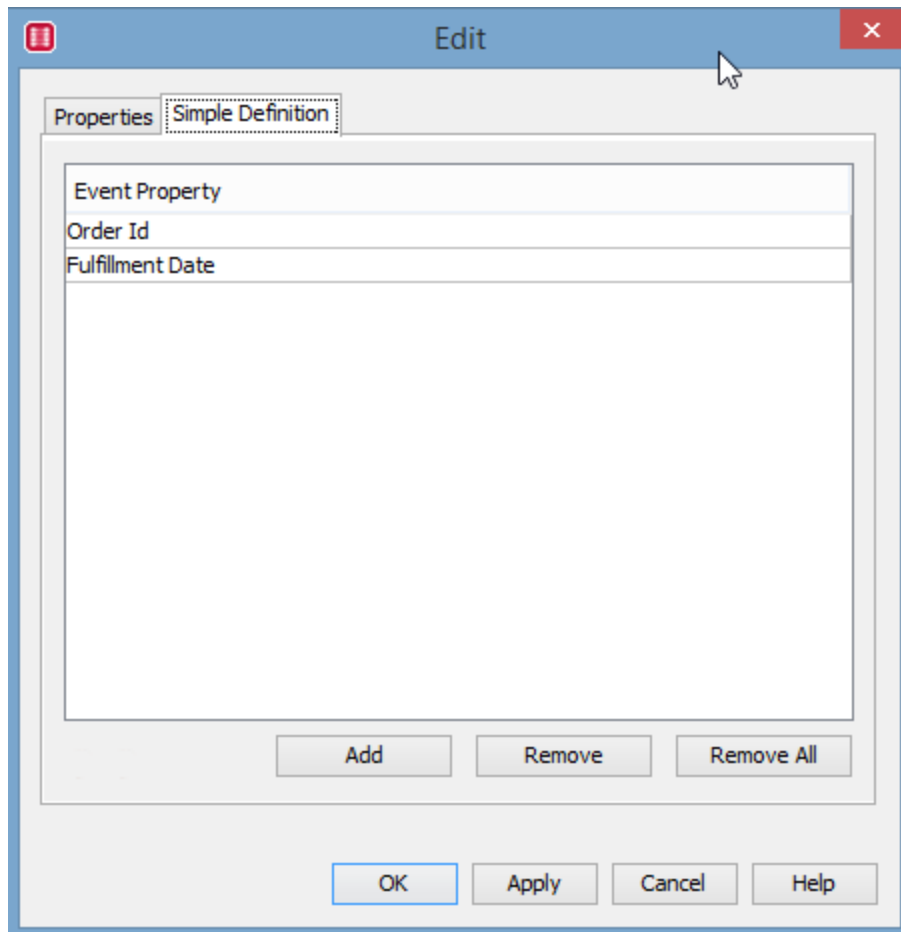
3. Create a second new "Event Definition" with the settings illustrated in the following images:

The image shows a software window titled "Edit" with a red close button in the top right corner. Inside the window, there are two tabs: "Properties" (which is selected and has a dashed border) and "Simple Definition".

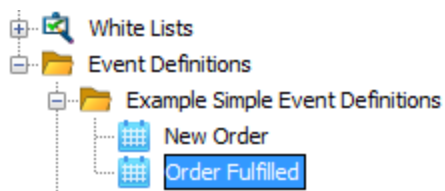
Under the "Simple Definition" tab, there is a form with the following elements:

- A "Name" label followed by a text input field containing the text "Order Fulfilled".
- A "Description" label followed by a large, empty rectangular text area.
- A "Definition Type" section containing two radio buttons: "Simple" (which is selected) and "Free Form".

At the bottom of the dialog, there are four buttons: "OK", "Apply", "Cancel", and "Help".

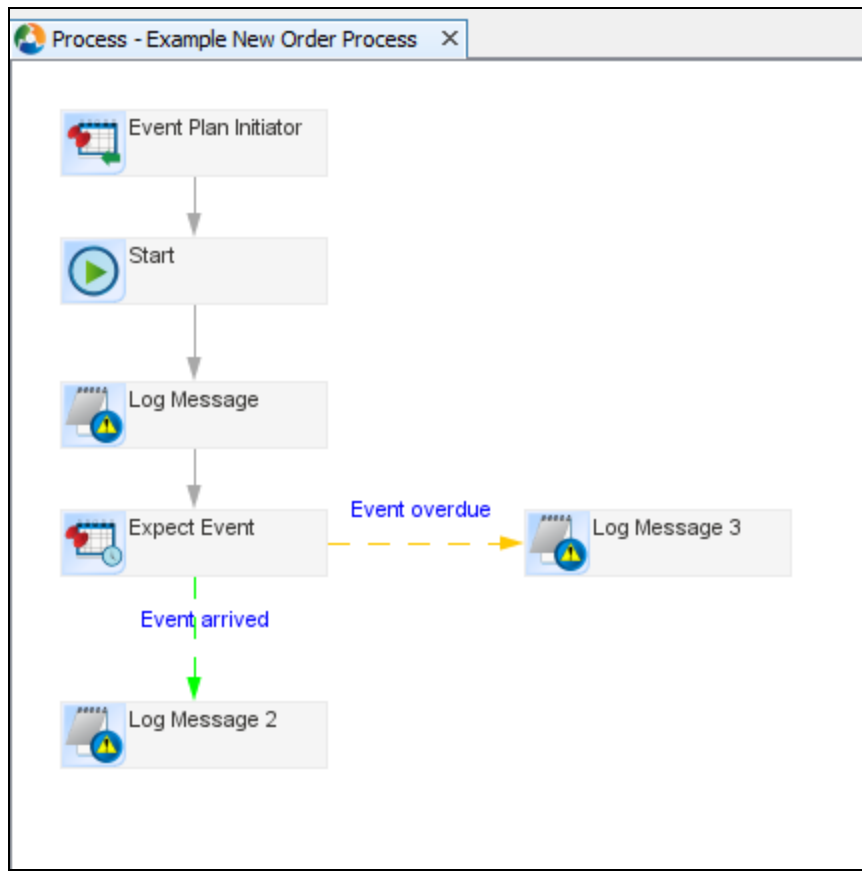


You should now have two event definitions defined:



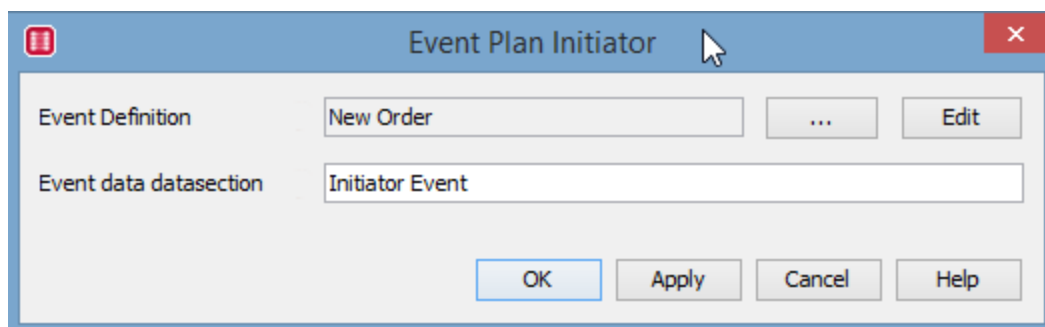
Stage 2 – create the event plan

1. Create a new process with the modules illustrated in the following image:



When adding links between the “Expect Event” module and the “log message” modules, ensure you select the correct link type.

2. In the “Event Plan Initiator” module select the “New Order” event definition.



3. In the “Expect Event” module select the following properties. The correlation expression to enter for Order ID is:

`$DATA('-[Initiator Event]', 'Order ID', 0)$`

This can be selected from the Dynamic function right-click pop-up menu. Ensure the “Fulfilment Date” property is not selected for being correlated on.

The top screenshot shows the 'Expected Event' tab of the 'Expect Event' dialog. The 'Event Definition' is 'Order Fulfilled' and the 'Event data dataset' is 'Event Data'. The 'Correlation' table is as follows:

Event Property	Value	Correlate
Order Id	\$DATA("-[Initiator Event]','Order ID', 0)\$	<input checked="" type="checkbox"/>
Fulfilment Date		<input type="checkbox"/>

The bottom screenshot shows the 'Schedule' tab. The 'Early time' section has 'Enable' unchecked. The 'Expected time' section has 'Relative' selected, 'Relative to' set to '<Itself>', and 'Time' set to 0. The 'Late time' section has 'Enable' checked and 'Offset from expected time' set to 1. All time offsets are in minutes.

The Expected Event Count page should be left with default values.

4. The log message modules are configured so we can see what is happening in the output log at run time. In the first "Log Message" enter:

Event Plan Started:

```
$DATAFORMAT('-[Initiator Event]', ',', '\r\n', 1)$
```

In "Log Message 2", enter:

```
Order Fulfilment complete
```

```
$DATAFORMAT('-[Event Data]', ',', '\r\n', 1)$
```

In "Log Message 3", enter:

Event Plan Overdue for Order ID:

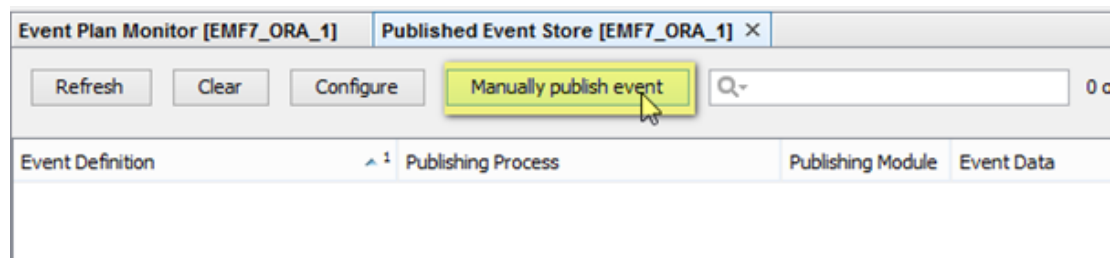
```
$DATA('-[Initiator Event]','Order ID', 0)$
```

5. Set the process state to "Active" and save the process.

Stage 3 – publishing events to run the event plan

The final stage now is to publish some events to run the event plan. Normally other processes would do this by using the "Publish Event" module when they detect changes in the system. In this example we are going to publish the events manually.

1. Ensure your EMF server is running.
2. In the Publish Event Store view, select **Manually publish event**.



- a. Enter the information as illustrated in the following image:

Manually publish event

Event Definition:

Simple Mapping

Event Property	Value
Order ID	123
Order Date	2015-01-19
Order Description	Test Order 1

Action if nothing waiting for event

☐ Do nothing ☒ Store Event until required

- b. Click **Manually publish event**.
3. Repeat step 2, but this time enter different details for the Order ID/Description.
4. Click **Refresh** to refresh the published event store, and it should show the two published orders. The "status" of both should be a green arrow, indicating the event has been matched against something waiting for the event.

Event Definition	Publishing Process	Publishing Module	Event Data	Published Date/time	Status
New Order			Order ID=123 Order Date=2015-01-19 Order Description=Test Order 1	19/01/15 16:23:22	➤
New Order			Order ID=456 Order Date=2015-01-19 Order Description=2nd Order	19/01/15 16:24:38	➤

5. Switch to the **Event Plan Monitor** view, and click **Refresh**. It should display the two event plans that have been started. One for each new order event.

Instance ID	Process	Start Date	Initiating Event	Initiating Event Data	Status
210576	Example New Order Process	19/01/15 16:23:22	New Order	Order ID=123 Order Date=2015-01-19 Order Description=Test Order 1	In Progress
210577	Example New Order Process	19/01/15 16:24:38	New Order	Order ID=456 Order Date=2015-01-19 Order Description=2nd Order	In Progress

6. Double-click on an event to bring up details for that event, showing the status. If more than one minute has passed since the event plan started, then the “overdue” branch on the process should have run, and the “Expected time” will turn Red.

Property	Value
Process	Example New Order Process
Status	In Progress

Event Plan Step ID	Event ID	Event Module	Event Definition	Publishing Proc...	Publishing Module	Processed Time	Expected Time	Arrival	Optional
421120		Event Plan In... New Order				19/01/15 16:2...		Arrived	<input type="checkbox"/>
	421104			<<Not Set>>	<<Not Set>>	19/01/15 16:2...			<input type="checkbox"/>
	421106	Expect Event	Order Fulfilled				19/01/15 16:23:23		<input type="checkbox"/>

Name	Value
Order ID	123
Order Date	2015-01-19
Order Description	Test Order 1

7. To get one of the event plans to complete, the order fulfilled event for that event plan must be published. From the **Published Event Store**, select **Manually Publish Event** again, and this time enter the information illustrated in the following image:

Manually publish event

Event Definition: Order Fulfilled

Simple Mapping

Event Property	Value
Order Id	123
Fulfillment Date	2015-01-21

Action if nothing waiting for event

☐ Do nothing ☒ Store Event until required

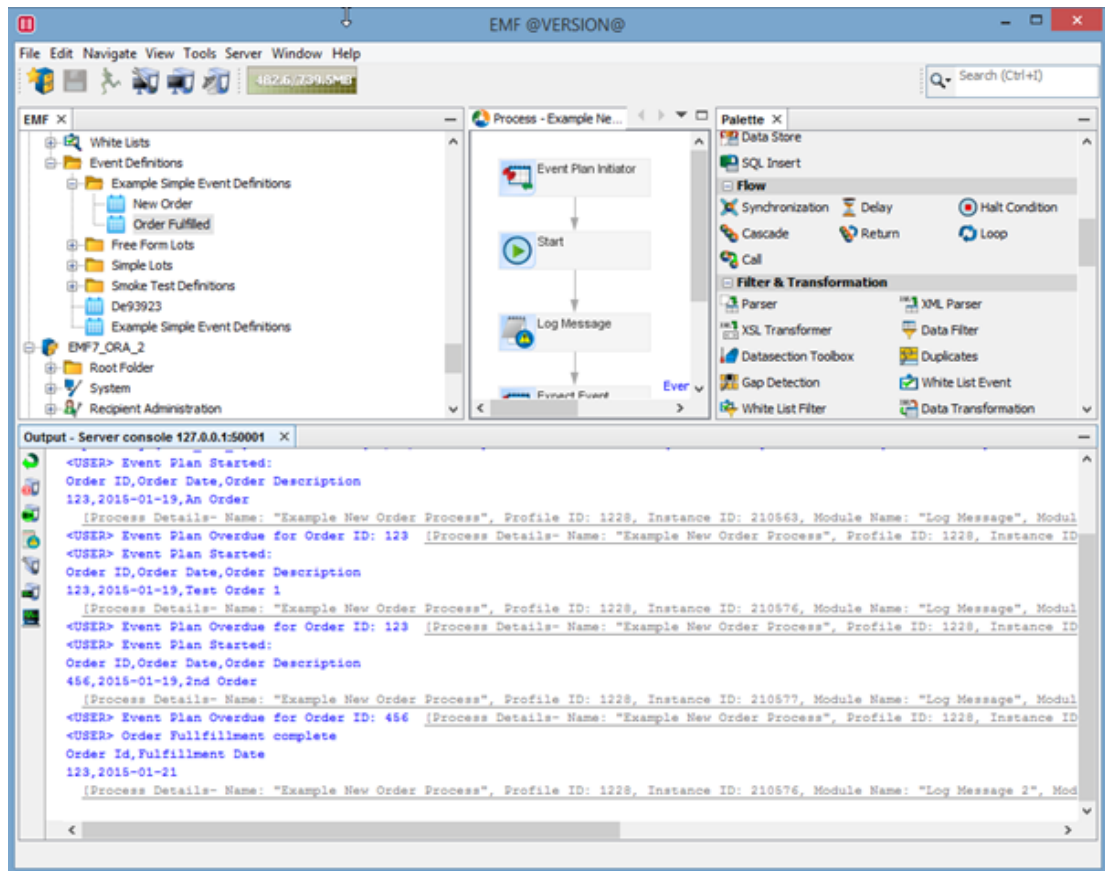
Manually publish event Close Help

The "Order ID" to enter must match the order ID for the event plan you want to complete.

- Checking the "event plan monitor" will now show the status of one of the event plans as complete.

Instance ID	Process	Start Date	Initiating Event	Initiating Event Data	Status
210576	Example New Order Process	19/01/15 16:23:22	New Order	Order ID=123 Order Date=2015-01-19 Order Description=Test Order 1	Complete
210577	Example New Order Process	19/01/15 16:24:38	New Order	Order ID=456 Order Date=2015-01-19 Order Description=2nd Order	In Progress

- Checking the "output window" (assuming you had it open), will display the user messages showing the processing of events in the two event plan instances that were started.





5

Advanced

Multi Residency in EMF

The Multi Residency feature in EMF allows you to setup multiple EMF instances on a single machine. When multiple products use EMF on the same machine using different EMF versions, it becomes necessary to allow the use of multiple EMF instances on the same machine to separate the different uses. Each instance must be licensed separately.

Setting up a new instance is a two-step process:

1. Create a copy of the existing installation
2. Configure the new installation

Creating a New EMF Instance

To create a new EMF instance, run the `instance_create.cmd` file found in the root folder of the original EMF installation. The syntax of the command is as follows:

```
instance_create <EMF instance path> <EMF instance name>
```

Where,

- `<EMF instance path>` is the full path name (where the new EMF instance has to be created)
- `<EMF instance name>` is the name of the new EMF instance.

Note: If the path or name contains spaces, then the parameter passed into the `instance_create` command must be surrounded by double quotes.

Example: `instance_create "C:\EMF Instance1" InstanceOne`

Note: EMF instance name must contain valid characters that can be used in file names.

Note: You must have relevant Operating System permissions to create folders, copy files and create files.

The `instance_create` command creates a copy of the EMF instance based on the location of the command.

For example, if the command is run from an existing instance, then it will create a new copy based on that instance.

In addition:

- In all cases, the new instance will be named according to the name passed into the `instance_create` command.
- The folder structure is retained and all the files in the current instance are copied.
- Some files are renamed with the new instance name and a file called `INSTANCE` is created in the root folder of the new EMF instance. This `INSTANCE` file contains the name of the EMF instance and is required by the EMF to determine the EMF instance that it is working with.

The path used for storing EMF related run time logs and files is `Users\Public\emf\6`. EMF instances use `Users\Public\emf\6\instance\<EMF instance name>`, where `<EMF instance name>` refers to the name of the newly created EMF instance, as specified in the `instance_create` command.

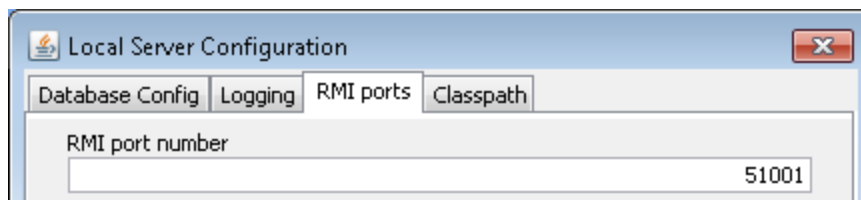
Note: Multi residency will not work on any Windows Operating System prior to Windows Vista. Refer the installation guide for a list of supported platforms for operating systems of Windows Vista and above .

Configuring the new EMF Instance

Important: Some files must be changed to access the new EMF instance for it to run as detailed in the following steps. It is important to follow the steps in the same order, else the EMF instance will not be setup properly.

To configure the new EMF instance:

1. Run the EMF UI.
 - Create a new shortcut to the EMF instance UI exe file or run the `<EMF instance path>\bin\emf70_<EMF Instance name>.exe` file.
2. Configure the Server.
 - a. Start the EMF UI and configure the server from **Server > Configure Local Server**.
 - b. In the **RMI ports** tab, change the RMI port number to a value other than 50001.



Note: 50001 is the standard EMF RMI port number, instances must not use this number . It is recommend to use 51001 for the next instance.

- c. In the **Classpath** tab, enter any additional JARs that will be used in the EMF instance server.
 - d. Click **Apply** and then click **OK** to save the information and exit the window.
3. Create a new EMF repository.
 - Create a new EMF repository or add an existing EMF repository to the **Repositories** node in the EMF Instance UI. To add an existing repository, right-click the repository and select **Set local server to use Repository**.

Important: Exit the EMF UI after creating/adding a repository.

4. Change the EMF service.

- Change the file contents as specified in the following table (changes are highlighted in italics).

File to Edit

Before Change

After Change

<EMF instance path>\server\ServiceInstall.cmd	
if exist "%CURRENT_DIR%EMFSer- vice.exe"	if exist "%CURRENT_DIR%EMFSer- vice_<EMF instance name>.exe"
%SERVICE_DIR%EMFService //IS//EMFSer- vice --DisplayName="EMF Service	%SERVICE_DIR%EMFService_<EMF instance name> //IS//EMFSer- vice_<EMF instance name> -- DisplayName="EMF Service <EMF Instance name>"
<EMF instance path>\server\ServiceUninstall.cmd	
EMFService //DS//EMFService	EMFService_<EMF Instance name> //DS//EMFService_<EMF Instance name>
<EMF instance path>\server\ServiceConfigure.cmd	
//ES//EMFService	//ES//EMFService_<EMF instance name>
<EMF instance path>\server\smRestartServer2.cmd	
net stop EMFService	net stop EMFService_<EMF instance name>
net start EMFService	net start EMFService_<EMF instance name>
Users\Public\emf\6\instance\<instance name>\server\EMF.Properties	
SM_START_WIN=net start EMFService	SM_START_WIN=net start EMFSer- vice_<EMF instance name>
SM_START_WIN_VISTA_OR_BEYOND- D=../server/Elevate.exe-wait net start EMFService	SM_START_WIN_VISTA_OR_BEYOND- D=../server/Elevate.exe -wait net start EMFService_<EMF instance name>
SM_STOP_WIN=net stop EMFService	SM_STOP_WIN=net stop EMFService_<EMF instance name>

5. Change the Run Exe files.

- Change the file contents as specified in the following table (changes are

highlighted in italics).

File to Edit

Before Change

After Change

```
<EMF instance path>\RunExe\ServiceInstall.cmd
```

```
if exist "%CURRENT_
DIR%EMFRunExeClient.exe"
%SERVICE_DIR%EMFRunExeClient
//IS//EMFRunExeClient --Dis-
playName="EMF RunExe Client"
```

```
if exist "%CURRENT_DIR%EMFRun-
ExeClient_<EMF instance name>.exe"
%SERVICE_DIR%EMFRunExeClient_<EMF
instance name> //IS//EMFRunExeClient_
<EMF instance name> --Dis-
playName="EMF RunExe Client <EMF
instance name>"
```

```
<EMF instance path>\RunExe\ServiceUninstall.cmd
```

```
EMFRunExeClient
//DS//EMFRunExeClient
```

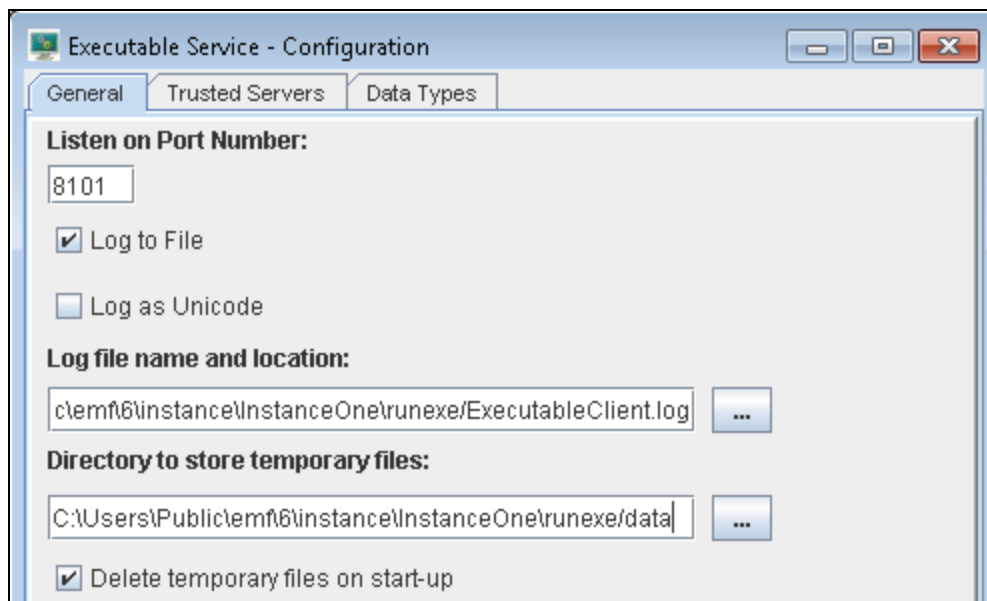
```
EMFRunExeClient_<EMF instance name>
//DS//EMFRunExeClient_<EMF instance
name>
```

```
<EMF instance path>\RunExe\ServiceConfigure.cmd
```

```
//ES//EMFRunExeClient
//ES//EMFRunExeClient_<EMF instance
name>
```

Note: If the EMF instance `RunExe` service is installed, double-click the `ServiceConfigure` file in the `<EMF instance path>\RunExe` folder and check if the display name, `RunExe` executable file, and log path are correct.

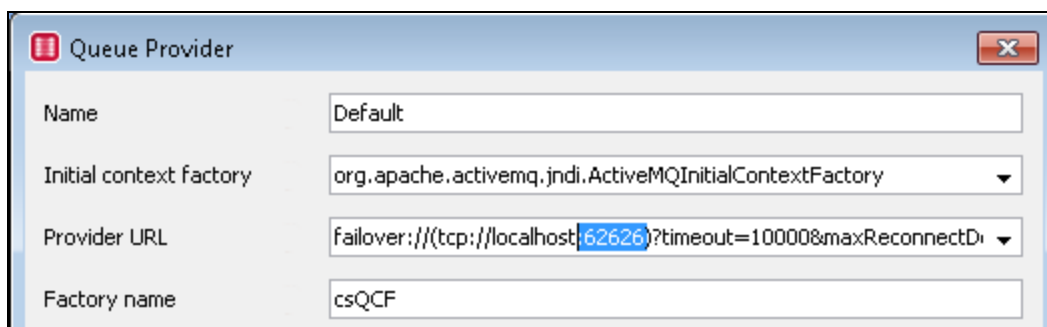
- b. Run the `ExeClient.bat` file to open the **Executable Service** window.
- c. Click **Stop** and select **Options**.
- d. Change the value of the **Listen on Port Number** from the default "8100" to a port number that is not in use.
- e. Check if the log file and temporary file locations are using the correct EMF instance location to access the `Users\Public\emf\6\instance\<EMF instance name>\runexe` folder.



6. Change the Queues.

- a. Login to the repository in the EMF instance UI.
- b. Navigate to the `System\Queue Providers` node.
- c. Edit the Active **Queue Provider** and change the port number in the **Provider URL**.

Note: Ensure that the new port number is not used by any other port. For example: `failover://(tcp://localhost:61616)` will change to `failover://(tcp://localhost:<new port number>)`.



- d. Edit the `ActiveMQ.xml` file and change as specified in the following table (changes are highlighted in *italics*).

Note: Ensure that the port number in the file matches the port number specified in the **Provider URL**.

File to Edit

Before Change

After Change

```
Users\Public\emf\6\instance\<EMF instance name>\activemq.xml|
uri="tcp://0.0.0.0:61616"          uri="tcp://0.0.0.0:
                                   <new port number>"
```

7. Update the Ports.

- Change the ports if there are any listeners that need to point to different ports for different EMF instances.

8. Install the EMF Services.

The new EMF instance services must be installed to use the new EMF instance server.

- Navigate to the <EMF instance path>\server folder and double-click on the ServiceInstall.cmd file. This will install the EMF Service _<EMF instance name> service.
- If you are planning to use the RunExe service, then navigate to the <EMF instance path>\RunExe folder and double-click the ServiceInstall.cmd file. This will install the RunExe_<EMF instance name> service.

Note: If you do not have the permissions to install the services, then the cmd files must be **Run as administrator**.

Note: The EMF user and cache directories for the new EMF instance will be different from the original EMF instance. These will be created automatically when the new EMF instance UI is launched.

For information on upgrading EMF in Multi Residency, see: [Upgrading EMF in Multi Residency](#)

Upgrading EMF in Multi Residency

Overview

When multiple instances of EMF are running on the same machine, upgrading any EMF instance can only happen if the main install is upgraded. After upgrading a main instance, the separate instances are not automatically upgraded and the following instructions should be followed to upgrade them.

Note: It is acceptable (but not advisable) to run the main instance as a newer version of EMF, and leave other instances running older EMF versions.

Before upgrading an existing EMF instance

1. Upgrade the main instance to the version you wish to upgrade your instance to.
2. Stop the EMF Instance service and the EMF Instance `RunExe` service.
3. Uninstall these services.

For the EMF Instance service, use: `<EMF instance path>\server\ServiceUninstall.cmd`

For the RunExe service, use: `<EMF instance path>\RunExe\ServiceUninstall.cmd`

Upgrading an existing EMF instance

The existing EMF instance can be upgraded by running the `instance_upgrade.cmd` file found in the root folder of the main EMF installation. The syntax of the command is:

`instance_upgrade <EMF instance path> <EMF instance name> <EMF backup folder>`

`<EMF instance path>` - this is the full path name where the EMF instance is to be upgraded

`<EMF instance name>` - this is the name of the existing EMF instance

`<EMF backup folder>` - Optional: This is the fully qualified folder name where the existing EMF instance will be backed up. If this is not specified, the upgrade appends `_backup` to the existing `<EMF instance path>` to create a new folder name.

Note: If the path or name contains spaces, then the parameter passed into the `instance_upgrade` command must be surrounded by double quotes.

Example: `instance_upgrade "C:\EMF Instance1" InstanceOne`

Note: You must have relevant Operating System permissions to create folders, copy files and create files.

The `instance_upgrade` command creates a backup copy of the existing EMF Instance folder. The upgrade will not remove any files from the Server/lib folder, so that any added JAR files required by the server need not be added again after the upgrade. Please ensure that any other folders/files added into the existing EMF Instance folder before the upgrade are added again after the upgrade.

Configuring the upgraded EMF Instance

Important: Some files must be changed to allow the upgraded EMF instance to run as detailed in the following steps. It is important to follow the steps in the same order, else the EMF instance will not be setup properly.

1. Run the EMF UI
Ensure that the existing shortcut to the EMF instance UI exe file is still valid. The EMF instance can also be run directly using the <EMF instance path>\bin\emf72_<EMF Instance name>.exe.
2. Configure EMF Server
No reconfiguration is required. The Server will have retained its pre-upgrade configuration.
3. EMF repository
The repositories should show automatically in the UI after the upgrade.
4. EMF Service
Change the file contents as specified in the following table (changes are highlighted in italics).

File to edit

Before Change

After Change

<pre><EMF instance path>\server\ServiceInstall.cmd if exist "%CURRENT_ DIR%EMFService.exe" %SERVICE_DIR%EMFService //IS//EMFService --DisplayName="EMF Service</pre>	<pre>if exist "%CURRENT_DIR%EMFService_ <EMF instance name>.exe" %SERVICE_DIR%EMFService_<EMF instance name> //IS//EMFService_ <EMF instance name> -- DisplayName="EMF Service <EMF instance name>"</pre>
<pre><EMF instance path>\server\ServiceUninstall.cmd EMFService //DS//EMFService</pre>	<pre>EMFService_<EMF Instance name> //DS//EMFService_<EMF instance name></pre>
<pre><EMF instance path>\server\ServiceConfigure.cmd //ES//EMFService</pre>	<pre>//ES//EMFService_<EMF instance name></pre>
<pre><EMF instance path>\server\smRestartServer2.cmd net stop EMFService</pre>	<pre>net stop EMFService_<EMF instance</pre>

<pre>net start EMFService</pre>	<pre>name> net start EMFService_<EMF instance name></pre>
---------------------------------	--

5. Run Exe

Make changes to the files specified below.

File to edit

Before Change

```
<EMF instance path>\RunExe\ServiceInstall.cmd
```

```
if exist "%CURRENT_
DIR%EMFRunExeClient.exe"
```

```
%SERVICE_DIR%EMFRunExeClient
//IS//EMFRunExeClient --
DisplayName="EMF RunExe Client"
```

```
<EMF instance path>\RunExe\ServiceUninstall.cmd
```

```
EMFRunExeClient //DS//EMFRunExeClient EMFRunExeClient_<EMF instance
name> //DS//EMFRunExeClient_<EMF
instance name>
```

```
<EMF instance path>\RunExe\ServiceConfigure.cmd
```

```
//ES//EMFRunExeClient
```

After Change

```
if exist "%CURRENT_
DIR%EMFRunExeClient_<EMF instance
name>.exe"
```

```
%SERVICE_DIR%EMFRunExeClient_<EMF
instance name>
//IS//EMFRunExeClient_<EMF
instance name> --DisplayName="EMF
RunExe Client <EMF instance name>"
```

```
//ES//EMFRunExeClient_<EMF
instance name>
```

Note: If the EMF instance RunExe service is installed, double-click the ServiceConfigure file in the <EMF instance path>\RunExe folder and check if the display name, RunExe executable file, and log path are correct.

The Executable Service does not need reconfiguring. It will use the pre-upgrade settings.

6. Installing the EMF Services

Having made all the above changes, the upgraded EMF instance services need to be installed so that the upgraded EMF instance server can be used.

Navigate to the <EMF instance path>\server folder and double click on the ServiceInstall.cmd file. This will install the "EMF Service_<EMF instance name>" service.

If you are planning on using the RunExe service, then navigate to the <EMF instance path>\RunExe folder and double click on the ServiceInstall.cmd file. This will install the "RunExe_<EMF instance name>" service.

Note: The cmd files above may need to be "Run as administrator" if you do not have the correct permissions to install the services.

The EMF instance server can now be started in the normal way.



6

Troubleshooting

Overview

The information contained within this section is designed to deal with any known problems that could occur while using EMF.

Checking your EMF system configuration

If you are having trouble with your EMF system you should [refer to this list](#) of all the places where you can check your configuration details in case of problems.

Known issues, problems and error messages

The remaining topics in the Troubleshooting section of EMF Help address known issues that you may encounter.

If possible, always ensure that you are using the appropriate and/or most recent versions of third-party drivers and other files.

Email support at EMF.Support@aptean.com

Specific Issues

The following help topics are intended to assist you in troubleshooting specific problems within EMF:

- [Troubleshooting Escalations](#)
- [Troubleshooting JNDI Queries](#)
- [Troubleshooting Synchronizations](#)
- [Troubleshooting the Oracle and SQL Server Trigger Wizard](#)

Troubleshooting Escalation

If Escalation does not work as expected in your EMF Process:

- Check that [Message State Logging](#) is set to **Full** in the **Advanced** Properties of the [Output module](#) that immediately precedes the **Escalation** module.
- Check that the [Output formatting](#) module that creates the content for the Output module contains the [ESCALCODE Dynamic function](#).

Email support at EMF.Support@aptean.com

Troubleshooting JNDI Queries

If no results are returned when you test the JNDI query in the EMF Process Builder:

- Check that the details in the [JNDI service screen](#) are correct. Click the **Test connection** button in the JNDI service screen to check that the initial context exists.
- If the JNDI connection requires authentication, check that the **Username** and **Password** details in the JNDI service screen have been entered and are correct. Depending on the JNDI service provider, the **Test connection** button may not check the authentication details.
- If you entered the attribute manually in the **Attributes** tab of the [JNDI query module](#), the attribute may not exist in the JNDI service provider.

If results are returned when you test the query in the EMF Process Builder, but you cannot extract information from the data section when the EMF Process is run:

- The [JNDI query module](#) can return the data back in a different order to the order you used for the selected attributes when you run the EMF Process. Therefore, when you use the data in the resulting data section, you should reference by the column name rather than the column index.
- Attributes may be returned with a different name to the name originally shown in the **Available** list. To check the actual column names that will be returned, click the **Test** button to display the test results.

Email support at EMF.Support@aptean.com

Troubleshooting Synchronizations

Missing or unexpected information

When you [synchronize](#) two or more branches of an EMF Process, normally the output from the synchronization combines all the information from all of them. If you find that some expected input has not been used, it may be for one of the following reasons:

- One of the branches that you selected for input may have been prevented from running (check the **Link options** on the **Link Conditions** Properties page for the relevant link) and so was not included in the synchronization. To correct this, ensure that all the branch links are set to "Run always".
- The specified primary branch may not have been available. If the primary branch is stopped before the synchronization, another branch (normally the first branch to deliver the required information into the synchronization) will be chosen as the primary branch. To correct this, ensure that all the links on the primary branch are set to "Run always".
- One or more of the sections in a secondary module has the same name as the equivalent section in the primary module and has been discarded. To correct this, rename the relevant section if you want to include it.

You can also check the [Debug](#) log to see whether all the modules delivered the required information on time.

Synchronization affects the performance of your system

Synchronization can be very processor and database intensive, and can make considerable demands on your system. If you notice a deterioration in overall performance you should consider some of the following:

- Reduce the number of synchronizations that you have in an EMF Process
- Reduce the number of branches that feed into a synchronization
- Remove some of the sections included from the secondary branches

Reduce the frequency with which the Synchronizer checks for completed branches (see [Synchronized EMF Processes Retriever Listener](#)).

Email support at EMF.Support@aptean.com

Troubleshooting the Oracle and SQL Server Trigger Wizard

Troubleshooting the Oracle Server Trigger Wizard

Q1. I have successfully created the Oracle trigger within EMF. When I update (or change) the table which has the trigger on it, I get the following error:

```
ORA-29541: class <schema name>.<table name>_U_CLASS could not be resolved
ORA-06512: at "<schema name>.<table name>_U_SP", line 1
ORA-06512: at "<schema name>.<table name>_U_TRIGGER", line 39
ORA-04088: error during execution of trigger '<schema name>.<table name>_U_TRIGGER'
```

A1. This error occurs because the java class cannot be compiled successfully in the Oracle environment. Check that the EMF API class (**xa_api.jar**) has been loaded into the Oracle environment. This needs to be loaded per schema or loaded into the public schema, so it is accessible to all schemas.

To load per schema, select "Load EMF API class into Oracle" at least once, per schema. This loads the **xa_api.jar** file into Oracle, so that the methods that are needed by the trigger class are available.

If you have multiple triggers defined per schema, then the first time that an oracle trigger is created for that schema, select "Load EMF API class into Oracle". Once the EMF API is loaded into the schema, then subsequent oracle triggers defined for that schema need not load the EMF API.

Q2. When I try to load the EMF API class into Oracle using the check box in the EMF Oracle trigger wizard, I get an error, "EMF API class not created".

A2. Check that the **xa_api.jar** file is in the same place as defined by the **emf_api_dir** directory alias. For example, in **c:\emf** on the Oracle server. (For details on how to set this up, see [Oracle Trigger Wizard](#)).

Note: Ensure that the **xa_api.jar** file is not "read-only".

Q3. I have created the trigger successfully and I can change the table on which the trigger is defined, so that it runs the trigger, but the API process does not run.

A3. Ensure that socket permission has been granted to the schema where the trigger is defined, so that it can communicate with the RMI port.

Example PL/SQL: call `dbms_java.grant_permission('<schema name>', 'SYS:java.net.SocketPermission', '<machine>:50001', 'connect,resolve')` where `machine` is the name or IP of the EMF server, `50001` is the RMI port.

Q4. I have followed all the steps to create the Oracle trigger, but the API process does not run.

A4. Ensure that the machine name is defined correctly in the EMF trigger wizard. For example, instead of using `localhost`, use the machine name or the IP address (if static).

Troubleshooting the SQL Server Trigger Wizard

Q1. I have restored a backed up SQL Server EMF database, but cannot add the EMF .NET assembly to this database.

A1. This happens when the owner associated with the backed up database does not exist when the same database is restored. This can be resolved with the following:

Under database properties of the restored Database, go to the Files page. If the owner is blank, specify an owner so that proper authorization is given for file access.

General Problems

The following help topics are intended to assist you in troubleshooting general problems within EMF:

- [How to Check Your User and Database Names and Details](#)
- [Licensing Restrictions](#)

How to Check Your User and Database Names and Details

Several aspects of EMF configuration require you to enter information that is specific to your own installation of the product, and this topic lists the places where you can determine your Repository name, Repository database name, User details, etc.

- **Repository name:** This is the **name** that you give the repository when running the [EMF Repository Wizard](#). It also appears in the right-hand (**Repositories**) pane of the [EMF tree view](#).
- **Repository database name:** This is the **database name** that you give the repository when running the [EMF Repository Wizard](#).
- **Database User details:** The user name is displayed, and can be changed when you right-click a repository and select **Configure Database Connection** in the EMF

Administrator screen. The password is not visible as it is encrypted, but can also be set using this screen.

- **EMF Operator details:** The details of all the known operators of EMF, including the default Admin, Server, and Script operators, are displayed on their individual Operator screens, accessible from the Operators view in the System node of the [EMF tree view](#).

[Configuring the EMF System](#)

[Troubleshooting](#)

Licensing Restrictions

Certain features and/or functionality in EMF may be missing or unavailable in your installation, depending on the licensing agreement with Aptean (or your supplier). If you wish, you can obtain an upgraded license although each installation is designed to be self-contained and not restricted for its intended purpose.

EMF can be installed and used in a number of different configurations that reflect the requirements of end users. This is achieved by offering a variety of licensing options so that customers can select the features and functionality that they require and control access to, and use of, the software.

Each licensing option offers a different set of modules and/or functionality, and the available functionality and features are dependent on the license that is currently active. Hence, certain items may be missing and/or unavailable although they are documented in the online Help.

The following aspects of EMF may be limited by the license:

- The length of time that you can use EMF for (e.g. for evaluation purposes)
- The modules that are available in the EMF Process builder
- The icons that are available in the EMF tree view
- The ability to import and/or export EMF Processes and other items
- The maximum number of operators that can log in to the system
- The ability to copy and/or move EMF Processes between folders

In addition to the above, certain functionality is also restricted by an operator's [security](#) rights.

[Licensing and Modularity in EMF](#)

Error Message

The following topics are intended to assist you in troubleshooting error messages. This includes describing the possible causes and suggested actions to resolve the problem

- Error Messages in EMF Server Console
- [Error Starting the Queue Provider](#)

Error Starting the Queue Provider

If you receive the message "Error starting the queue provider" when starting the EMF server, and your queue provider is not on the same machine as the EMF Server, it may be because EMF is trying to start the queue provider automatically along with the server. This option, which is enabled by default whenever you [create a new database](#), only functions when the queue provider and server are on the same physical machine or a mapped network drive.

To disable this option:

Click the **Machines** icon in the **System** section of the EMF tree view, then double-click the appropriate machine and deselect **Auto-start Queue Provider** (see [Managing the Machines That EMF Runs On](#)).

Note that you if you disable this option you will have to start your queueing software manually.

[Troubleshooting](#)

Email support at EMF.Support@aptean.com