INTEGRATION ENDPOINTS
USER GUIDE

STEP Trailblazer

Table of Contents

Integration Endpoints for Data Exchange

Integration endpoints are communication channels that define how STEP data is exchanged and integrated with external systems, such as web servers.

Three types of integration endpoints are available:

- Inbound Integration Endpoints
- Outbound Integration Endpoints
- Gateway Integration Endpoints

Setting up an endpoint involves specifying the location that receives the data, the transport protocol, the data to be exchanged, and a number of other configuration parameters. Both inbound and outbound data can be delivered in a variety of formats, using a number of different delivery methods. The gateway integration endpoint is more simple and enables users to request data from another source synchronously (for example, from JavaScript code), thus no background process is attached to it.

When an integration endpoint has been defined, the endpoint monitors the delivery so that you will always know if data has been delivered successfully or if an error has occurred (and action is required).

For more information on exporting / importing data manually or on a schedule, see the **Exporting Data and Images** documentation or the **Importing Data and Images** documentation.

Exporting Integration Endpoint Definitions as Comments

Integration endpoint definitions can be exported as comments using Advanced STEPXML. These exports are intended to be used for submission to external source control systems for comparison purposes. Users can import them into source code repository systems where they can be compared from version to version. Editing and/or importing of these files is not supported (e.g., users may not export, edit the comments, and re-import in STEP).

To export an integration endpoint definition for external comparison, use Advanced STEPXML and set the DefinitionsAsComments tag to 'true', as shown in the following sample template:

```
<?xml version='1.0'?>
<STEP-ProductInformation DefinitionsAsComments="true">
<OutBoundIntegrationEndpoint ExportSize="All"/>
</STEP-ProductInformation>
```

An example of the output for an integration endpoint definition as comments is shown below:

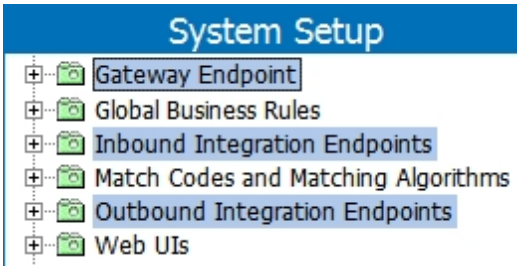
```
<OutBoundIntegrationEndpoint...>
<!--Definition: [IEP definition goes here. Removed for brevity.-->]
[Remaining configuration would populate here]
</OutBoundIntegrationEndpoint>
```

Note: The content of the comment field is not part of the STEPXML XSD and therefore Stibo Systems reserves the right to change the format of the output content at any time.

Integration Endpoint Setup

Before creating an integration endpoint, one or more setup groups must be created that are allowed to hold integration endpoints. You must also specify the setup group(s) in which integration endpoints can be created. This setup only needs to be performed once, and most systems will already have it completed.

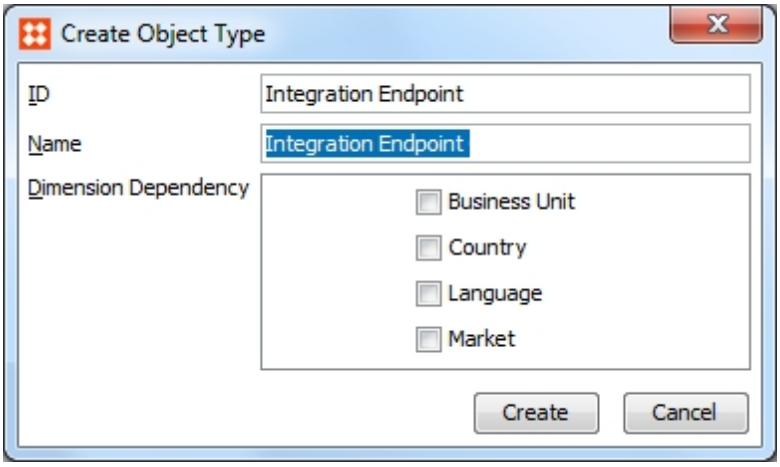
If one or more integration endpoint nodes exist on the System Setup tab, the setup has been completed and the steps in this section only need to be carried out if additional levels of organization are desired.



Note: Only users with the relevant privileges can view or maintain integration endpoints. For detailed information, see the **Action Sets** section and the **Users and Groups** section in the **System Setup / STEP Super User** documentation.

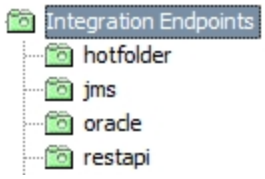
Create the Setup Group

1. In System Setup, expand **Object Types & Structures**.
2. Right-click Setup Group type root, and choose **New Object Type**.
3. Enter an **ID** and a **Name**, select any required Dimension Dependencies, and click **Create**.



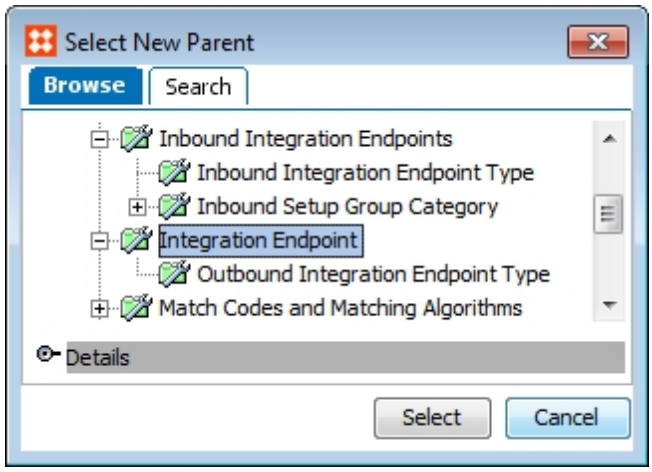
4. The **Setup Group** appears as a child in the **Setup Group type root**. Repeat steps 2 - 3 to create additional object types.

The following example illustrates a hierarchy of setup group object types. A setup group named **Integration Endpoints** has been created and contains four setup group object types: hotfolder, jms, oracle, and restapi.



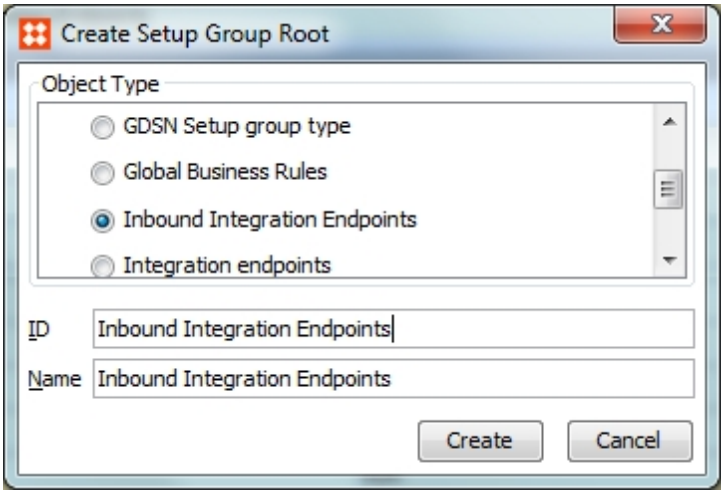
Link Object Type to Setup Group

1. In Object Types & Structures > expand Basic Object Types > select **Inbound Integration Endpoint**, or **Outbound Integration Endpoint**, or **Gateway Integration Endpoint**.
2. On the References tab, click the **Add Parent** link.
3. In the Select New Parent dialog, select the setup group you created, and click **Select** to make it a valid parent.



Create an Instance of the Integration Endpoint Object

1. On the System Setup tab, select any object in the **System Setup** hierarchy.
2. Click the Maintain menu, point to Insert, and select **Setup Group Root**.
3. In the Create Setup Group Root dialog, select the relevant object type.



4. Enter an **ID** and a **Name**, then click **Create**.

A setup group is created as a node in the System Setup hierarchy. When the Setup Group hierarchy is expanded, the color of the Integration Endpoint icons indicate the status of the endpoints. From this setup group node you can right-click to view the menu items for creating endpoints. For more information on the icons, see the **Integration Endpoint Status** section.

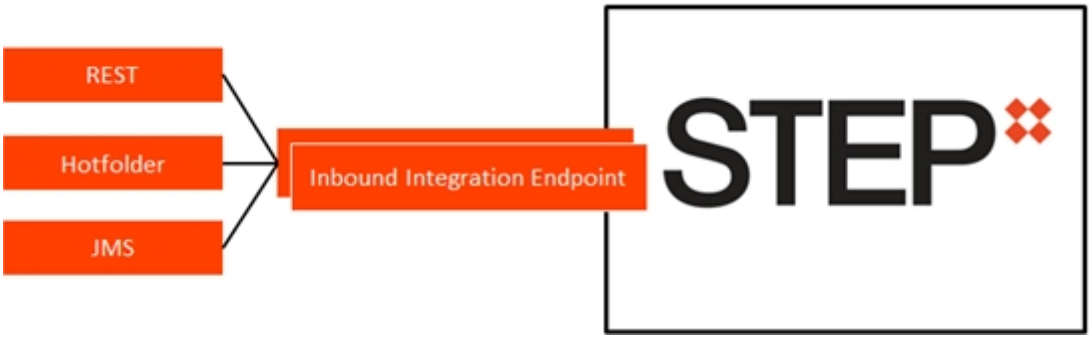
Inbound Integration Endpoints

Inbound Integration Endpoints (IIEPs) offer a centralized interface for monitoring and maintaining integrations with systems that send data to STEP in the form of files or messages.

Data being imported into STEP via an IIEP using the basic STEP Information Server license can originate as:

- A file delivered to a hotfolder
- A file delivered via REST
- A text message on a JMS server queue

IIEPs can poll the data sources for new files / messages to import on scheduled intervals–the minimum interval being one minute. As such, IIEPs offer functionality for 'near real-time' integrations. However, when the data source is REST, a POST will invoke the endpoint. For alternative real-time integrations with STEP, see the SOAP and REST APIs.

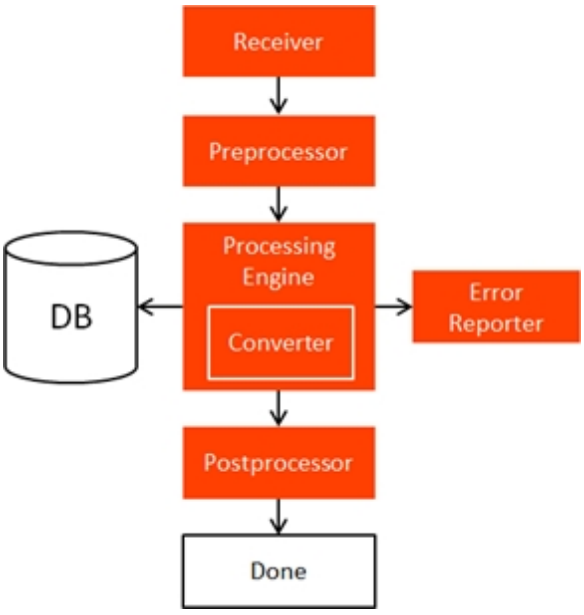


By default, IIEPs can be seen as wrapper functionality for the standard STEP import functionality. Without any extensions, files, or messages being imported, data will be handled by the STEP Importer and the same options for mapping and manipulation are available as when importing manually. The data source selection restricts the legal data formats. The table below shows the format options for core data source options.

Data Source	Supported Formats
Hotfolder	All formats supported by the standard import functionality
JMS	STEP XML, Generic XML, CSV (/ text messages)
REST	STEP XML, Generic XML (/ XML files)

Inbound Integration Endpoint Structure

The IIEP functionality has been created to allow for easy extensions / customizations and can be seen as a plugin framework with interchangeable parts. The graphic below illustrates the structure and the colored boxes represent the interchangeable parts, which are described below.



Receiver

The receiver plugin is responsible for retrieving files / messages and passing them on to the succeeding plugins. The available receiver options are described in the **Step 2: Choose Receiver** section of the **Creating Inbound Integration Endpoints** documentation.

Pre-Processor

A pre-processor plugin has access to the file / message delivered by the receiver plugin and can manipulate or discard it. STEP does not include any predefined pre-processor plugins.

Processing Engine / Converter

The processing engine is responsible for performing the actual data import. STEP Importer is the only available option with the core functionality, and is also the standard functionality used for manual imports. If a special format not supported in the standard STEP Importer is required, the processing engine can be replaced via an extension. Most often, however, creating a converter plugin for the STEP Importer is sufficient for such cases. This approach also has the benefit that the same converter plugin is available for manual imports.

Post-Processor

A post-processor plugin has access to import events (information about what has changed) and, based on the events, can trigger any required system change. By default, STEP does not include any preconfigured post-

processor plugins, since post-processor logic is also typically implemented via business rules referenced from the processing engine configuration.

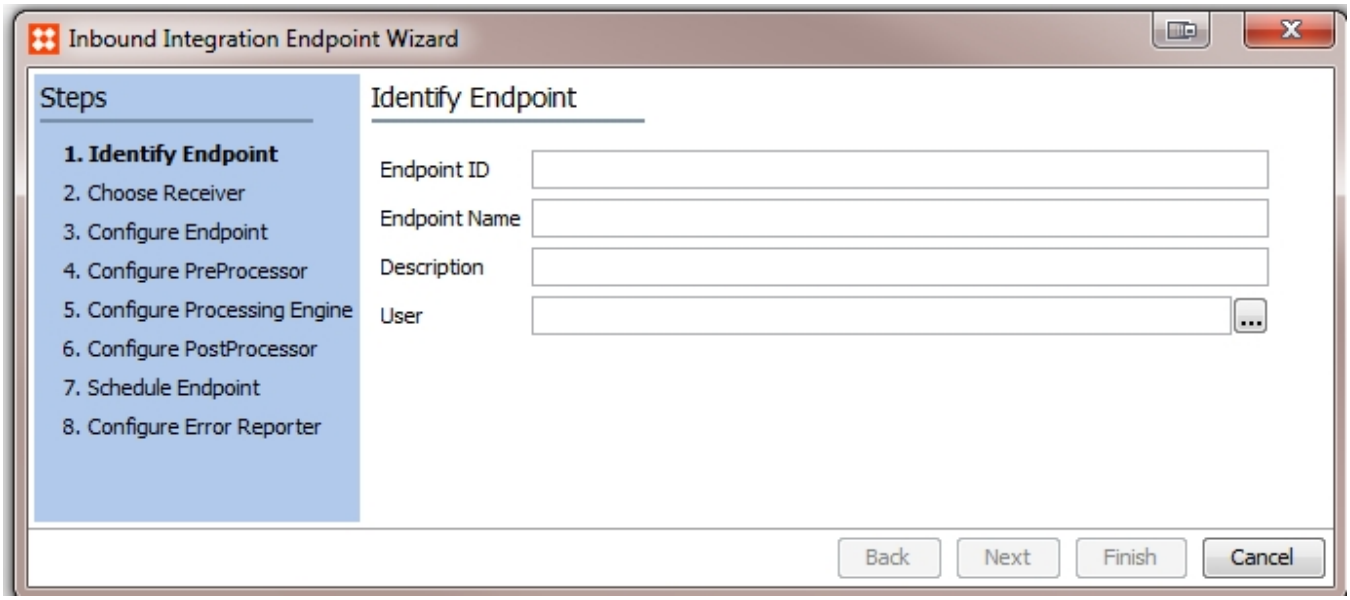
Error Reporter

STEP includes a single Send Error Report plugin to report errors. An email can be sent to a specified address if errors occur when files / messages are handled by the processing engine.

Inbound Integration Endpoint Wizard

Before creating an Inbound Integration Endpoint, you must first have an instance of a Setup Group Type below which IIEPs are legal. For more information on creating a setup group type, see the **Integration Endpoint Setup** section.

In System Setup, right-click the Integration Endpoint setup group and select **Create Inbound Integration Endpoint**. The Inbound Integration Endpoint wizard appears and guides you through the following steps.



Step 1 - Identify Endpoint: Specify the name and ID of the endpoint and identify the user whose privileges are applied.

Step 2 - Choose Receiver: Specify the receiver of the data.

Step 3 - Configure Endpoint: Specify processing, context and queue settings for the integration endpoint.

Step 4 - Configure PreProcessor: Configure defined pre-processors for customer-specific solutions. *This step is only available if you have a customer-specific solution.*

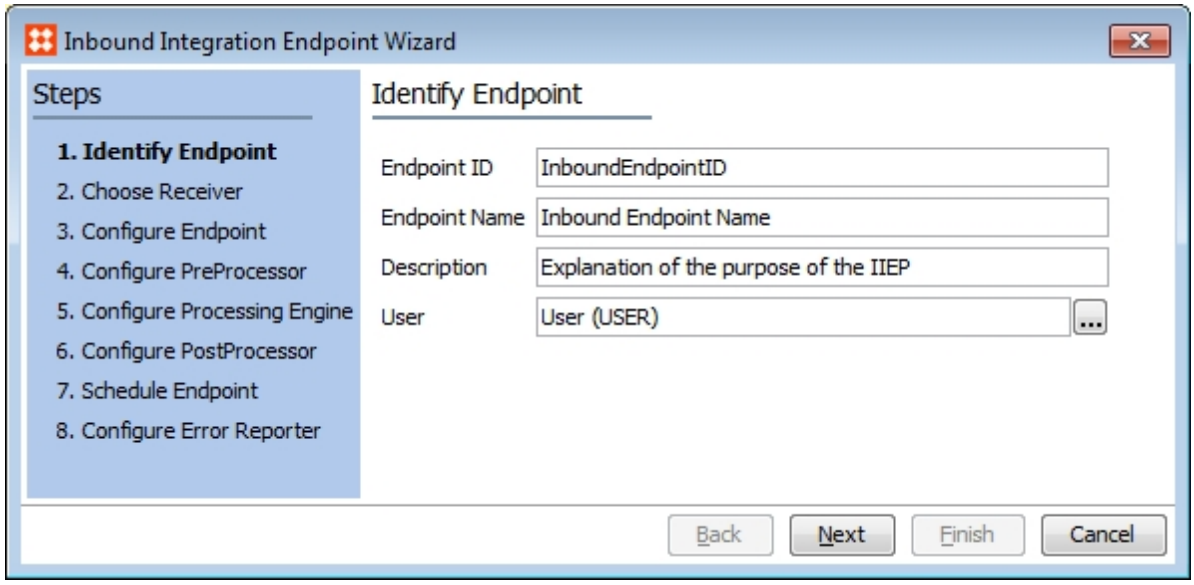
Step 5 - Configure Processing Engine: Specify the data format and map the data, among other options.

Step 6 - Configure PostProcessor: Configure defined post-processes for customer-specific solutions. *This step is only available if you have a customer-specific solution.*

Step 7 - Schedule Endpoint: Specify how often the endpoint should search for data to be processed.

Step 8 - Configure Error Reporter: Specify, select, and configure an error reporter plug-in that is activated if an endpoint-related background process fails.

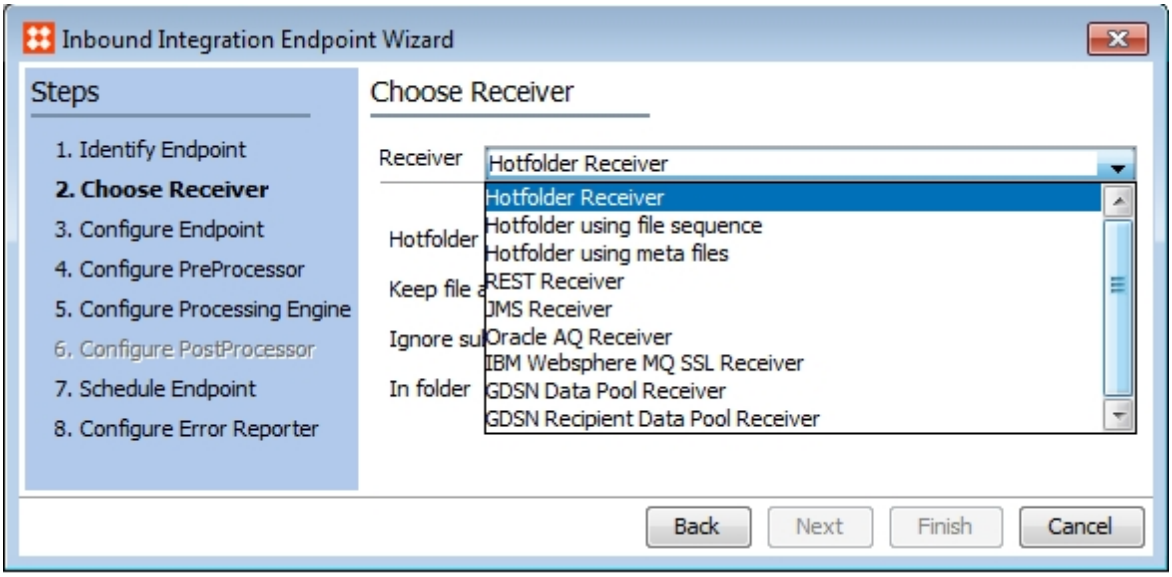
Step 1 - Identify Endpoint



1. Enter an **ID**, **Name**, and **Description** of the integration endpoint.
2. Search or browse for a **User**. The privileges of the selected user determine which actions the integration endpoint can perform when it processes data.

Note: It is common setup to create a special system user account for use by each IIEP. This allows you to trace changes made by the integration and also allows you to restrict (via privileges) what data the integration can change. This is especially relevant for STEP XML based IIEPs since STEP XML allows you to make significant changes to both data and configuration settings.

Step 2 - Choose Receiver



In Step 2, select the receiver to be used for the inbound endpoint. The available standard receivers are:

- GDSN Data Pool Receiver (available with GDSN implementation)
- GDSN Recipient Data Pool Receiver (available with GDSN implementation)
- Hotfolder Receiver
- Hotfolder Receiver Using File Sequence
- Hotfolder Receiver Using Meta Files
- IBM Websphere MQ SSL Receiver
- JMS Receiver
- MQ SSL Receiver
- Oracle AQ Receiver
- REST Receiver

Hotfolder Receiver

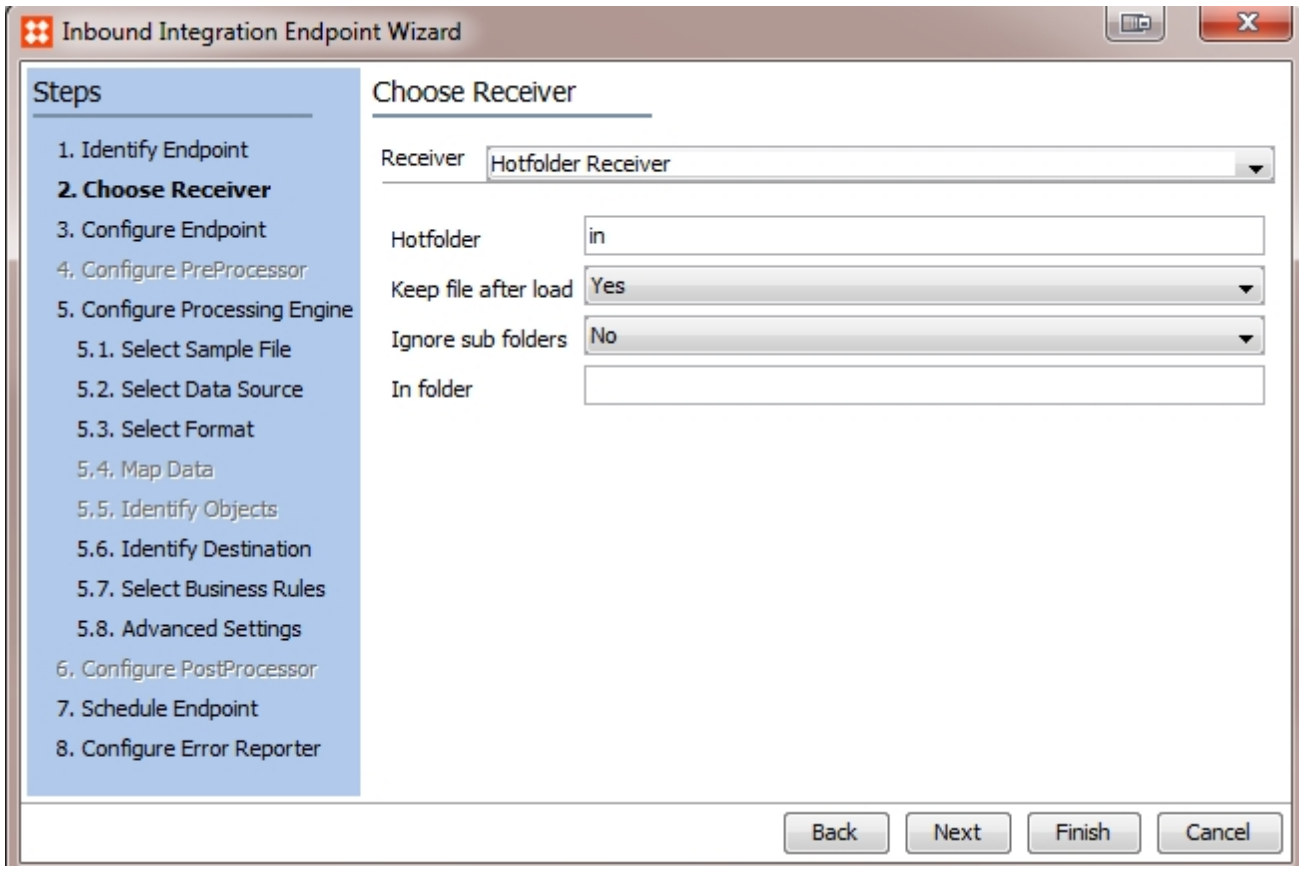
This default hotfolder option enables the setup of a standard data hotfolder as receiver. When the endpoint polls the hotfolder and finds more than one file, the files are handled in sequence starting with the oldest file first, based on the last modified data stamp.

To control the order of the files to be imported, set up either a Hotfolder Receiver Using Meta Files or a Hotfolder Receiver Using File Sequence instead.

Note: The location of the folders is determined by the sharedconfig.properties or config.properties file **Install.HotfolderRoot** property.

---

1. In the Receiver dropdown, select **Hotfolder Receiver**.



2. For **Hotfolder**, enter a name. A folder with this name is created in the upload directory on the application server.
3. For **Keep file after load**, specify whether files dropped in the hotfolder should be removed after processing the files. Selecting 'Yes' will require periodic manual cleanup on the server. Common setup is to select 'No' since the files are cleaned off automatically.
4. For **Ignore sub folders**, specify if subfolders within the hotfolder (which contain import files) should be ignored. Setting to 'Yes' means files in folders below the hotfolder will not be imported. Common setup is to select 'No' in order to process all files, regardless of their location within the hotfolder or a folder below it.

5. For **In folder**, you can optionally enter a name for an In folder. If no In folder is specified, files are dropped at the hotfolder top level.

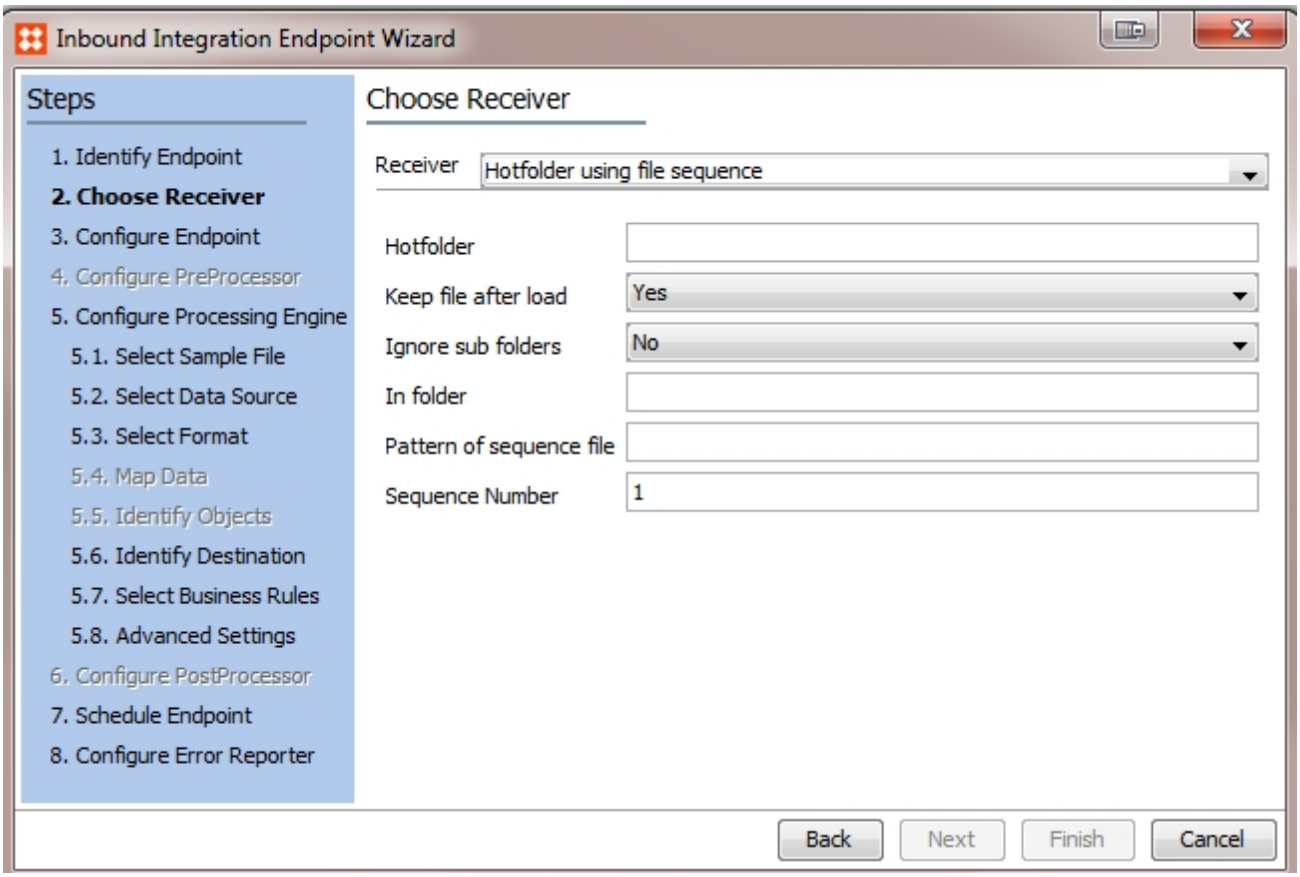
Hotfolder Receiver Using File Sequence

Selecting 'Hotfolder using File Sequence' enables setup of a data hotfolder where a sequence ID (included in the file name) determines the file processing order.

Note: The location of the folders is determined by the sharedconfig.properties or config.properties file **Install.HotfolderRoot** property.

---

1. In the **Receiver** list, select **Hotfolder using file sequence**.



2. For **Hotfolder** dropdown, enter a name. A folder with this name is created in the upload directory on the application server.
3. For **Keep file after load**, specify whether files dropped in the hotfolder should be removed after processing the files. Selecting 'Yes' will require periodic manual cleanup on the server. Selecting 'No' cleans off the files automatically.
4. For **Ignore sub folders**, specify whether subfolders within the hotfolder (which contain files) should be ignored. If set to Yes, files in these folders will not be imported.
5. For **In folder**, you can optionally enter a name for an In folder. If no In folder is specified, files are dropped at the hotfolder top level.

6. For **Pattern of Sequence File**, you must enter the relevant file pattern (required). For example, if the hotfolder is used to parse XML files, the pattern could be: file#.xml. In this case, a file name must begin with *file* and have the extension *xml*. The # indicates a sequence number.
7. For **Sequence Number**, you must enter the sequence number of the first file to be processed (required). For example, if the first number is 1, file1.xml is processed first, then file2.xml, and so on. The sequence number will automatically increment so that it is always possible to see the number for the next file. With this option, sequences can only be skipped if you manually reconfigure the 'Sequence Number'.

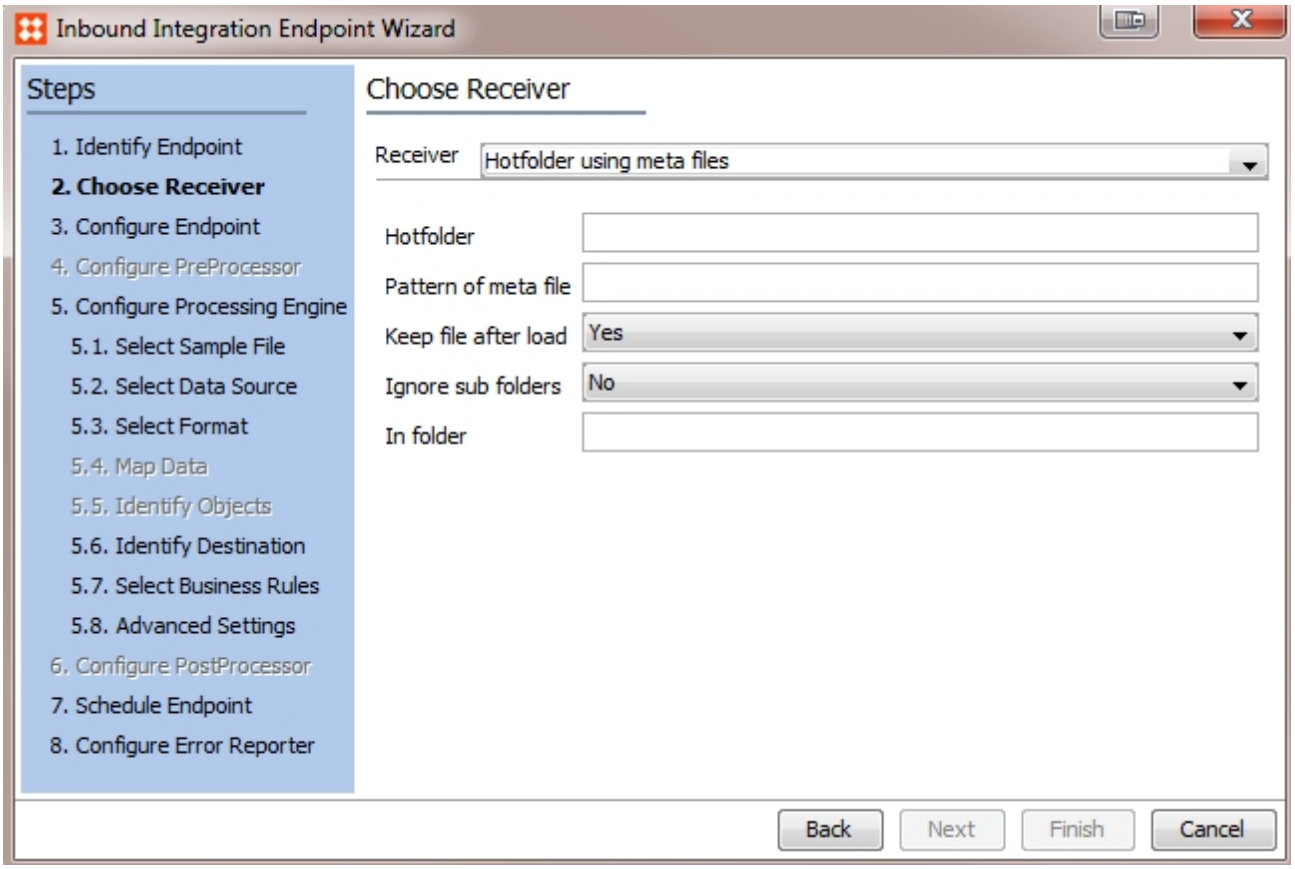
Hotfolder Receiver Using Meta Files

This option enables the setup of a data hotfolder where one or more meta files determine the file processing order. A meta file is a simple .txt file dropped into the hotfolder together with files to be imported. The list of files must be separated by newline.

Note: The location of the folders is determined by the sharedconfig.properties or config.properties file **Install.HotfolderRoot** property.

---

1. In the Receiver list, select **Hotfolder using meta files**.



2. For **Hotfolder** dropdown, enter a name. A folder with this name is created in the upload directory on the application server.
3. For **Pattern of meta file**, you must enter the name of the meta file that will be used to define the order of imported files (required). You can enter:
   - An **exact name** of the meta file, for example, meta.txt. In this case, additional meta files cannot be dropped in the hotfolder at the same time. The hotfolder must complete the import specified in the meta file before additional meta file(s) and files can be imported.
   - A **wildcard name** of the meta file, for example, sequence_*.txt. In this case, multiple meta files with names that match the wildcard name (for example, meta1.txt, meta2.txt, and so on) can exist in a hotfolder and will be processed. The hotfolder handles the meta files based on last modification date. This means that if you

drop a meta1.txt and meta2.txt and the modification date in meta2.txt is older than meta1.txt, the hotfolder will process the files specified in meta2.txt before the files specified in meta1.txt.

4. For **Keep file after load**, specify if files dropped in the hotfolder should be removed after processing the files. Selecting 'Yes' will require periodic manual cleanup on the server. Selecting 'No' cleans off the files automatically.
5. For **Ignore sub folders**, specify if subfolders within the hotfolder (which contain files) should be ignored. If set to Yes, files in these folders will not be imported.
6. For **In folder**, you can optionally enter a name for an In folder. If no In folder is specified, files are dropped at the hotfolder top level.

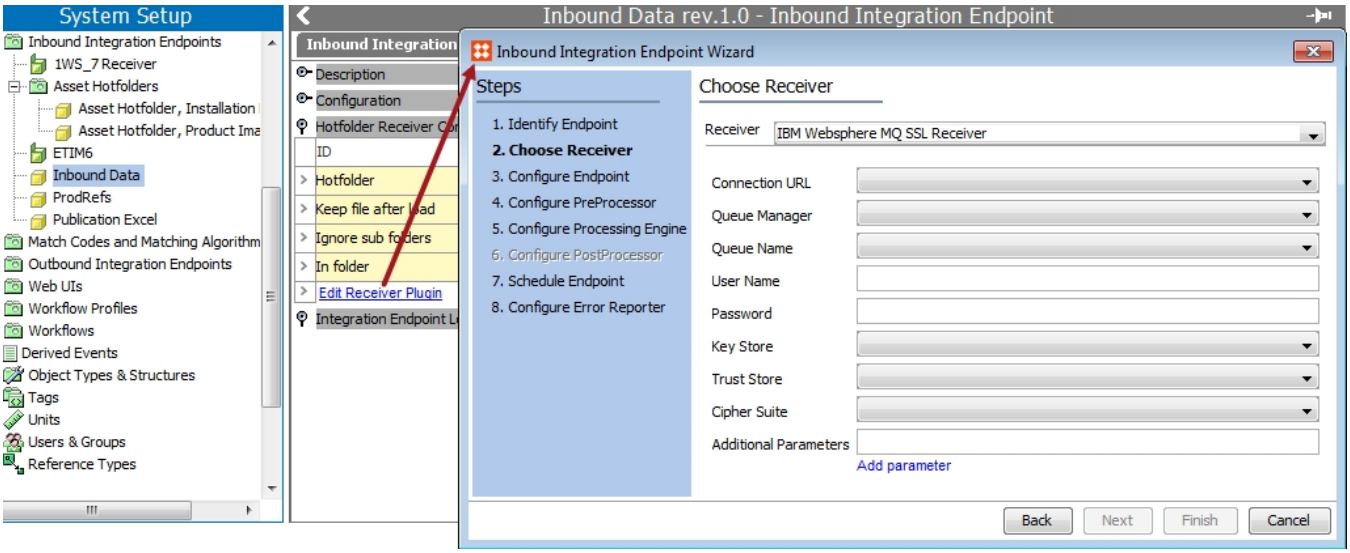
IBM Websphere MQ SSL Receiver

For information on connecting to IBM Websphere MQ in a non-SSL way, see JMS Receiver.

Note: Options available on dropdown fields must be configured in sharedconfig.properties or config.properties. Select a dropdown to display the required key name for each field.

---

1. In the **Receiver** field, choose **IBM Websphere MQ SSL Receiver**.



2. In **Connection URL**, select the URL for connection in the format [host]:[port]/[channel]:
   - [host] = hostname or IP of the Websphere MQ server
   - [port] = port number for the channel
   - [channel] = name of the channel
3. In **Queue Manager**, select the name of the Queue Manager.
4. In **Queue Name**, select the name of the Queue for the connection.
5. In **User Name**, if required, enter the user name that will be used to log onto the JMS provider.
6. In **Password**, if required, enter the password that will be used to log onto the JMS provider.
7. In **Key Store**, select the key store in jks format, with the personal certificate for the Queue Manager. To generate this, consult the manual for IBM Websphere MQ. The password for the key store must be configured in sharedconfig.properties or config.properties using 'WSMQSSLKeyStorePassword'.
8. In **Trust Store**, select the trust store with the CA for the Queue Manager. This can be the same file as key store. To generate this, consult the manual for IBM Websphere MQ. The password for the trust store must be configured in sharedconfig.properties or config.properties use 'WSMQSSLTrustStorePassword'.
9. In **Cipher Suite**, set to the same value as SSL CipherSuite in Websphere MQ. STEP is running on non-IBM jre, so this must be the same value as configured in the Queue Manager.

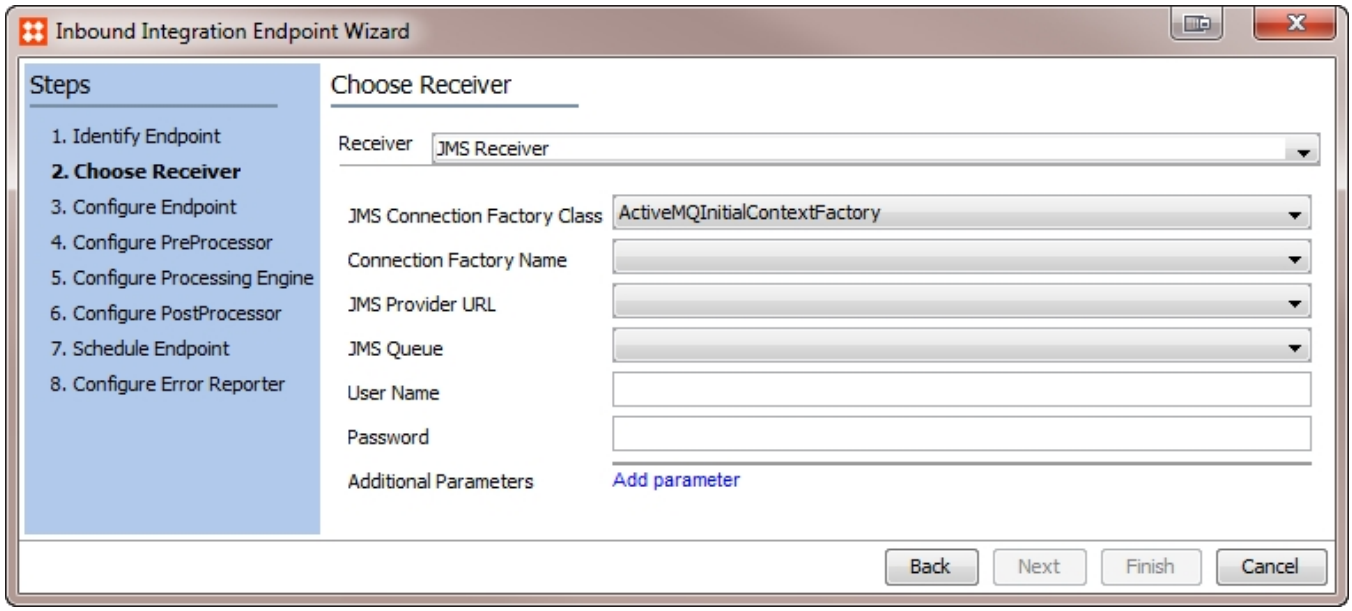
10. If **Additional Parameters** are required, click Add parameter, then enter the Key and Value. For possible keys and values, consult the manual for IBM Websphere MQ.

JMS Receiver

The available options for the Java Message Service (JMS) receiver are system dependent. By default, the JMS Receiver option lets you consume and dequeue messages on queues defined below. If additional JME support is required, contact your Stibo Systems account manager for customizations.

Note: Options available on dropdown fields must be configured in sharedconfig.properties or config.properties. Select a dropdown to display the required key name for each field.

1. In the **Receiver** list, select **JMS Receiver**.



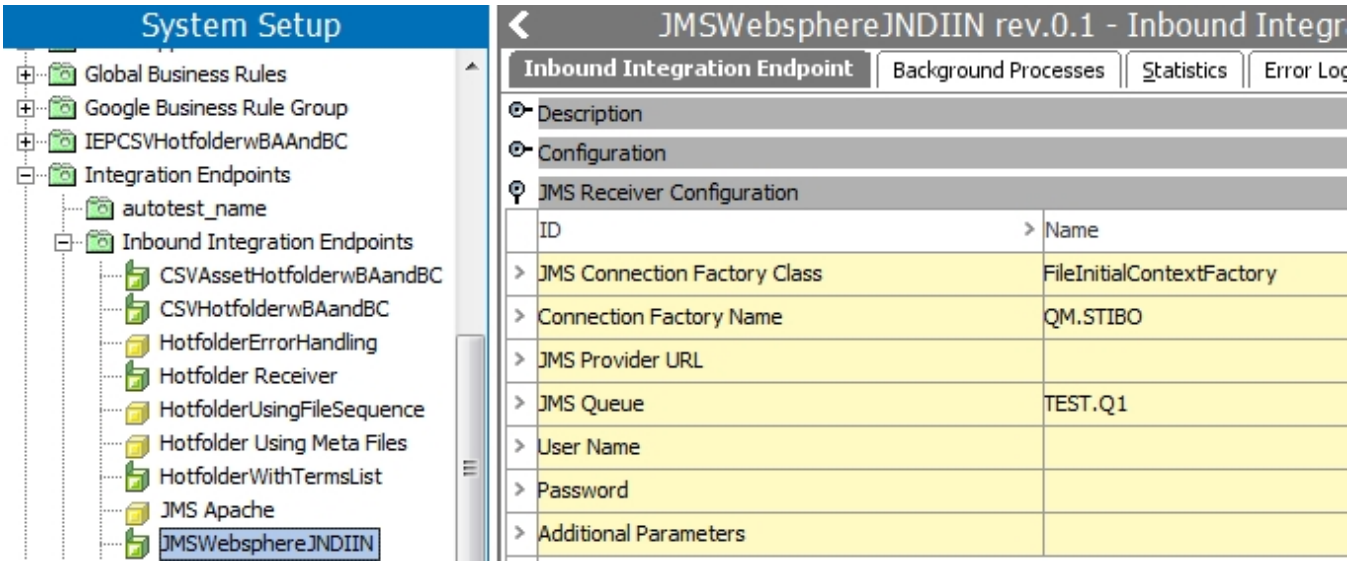
2. In **JMS Connection Factory Class**, select one of the available JMS connection factory classes.
   - **ActiveMQInitialContextFactory**: Allows set up of a JMS receiver connecting to Apache Active MQ. For information, visit apache.org.

   Note: WMQInitialContextFactory is no longer supported, use FileInitialContextFactory instead.

   - **FileInitialContextFactory**: Enables setting up JMS Websphere Inbound and Outbound Integration Endpoints which reference a binding file (created from JMS Websphere Client software). The binding file is a configuration file which includes all details of how STEP should interact with JMS Websphere. For information about JMS Websphere Client Software, visit ibm.com.
2. In **Connection Factory Name**, select a connection factory name in the list.
3. In **JMS Provider URL**, select a JMS URL in the list.
4. In **JMS Queue**, select the JMS Queue to be used on Apache Active MQ or Websphere MQ.
5. In the **User Name** and **Password** fields, enter a user name and a password, if required, to log onto the JMS provider.

6. If **Additional Parameters** are required, click Add parameter, then enter the Key and Value. For possible keys and values, consult the manual for IBM Websphere MQ.

FileInitialContextFactory JMS Receiver Configuration Example



JMS Websphere Delivery Using SSL Configuration Example

IBM Websphere MQ SSL Receiver Configuration

ID	Name
Connection URL	webspheremq-qa.stibo.com:1417/STEP.SVRCONN
Queue Manager	QM.STIBO_SSL
Queue Name	TEST.Q1
User Name	
Password	
Key Store	file:/workarea/JMSWebsphereSSLKeystore/keyStore.jks
Trust Store	file:/workarea/JMSWebsphereSSLKeystore/keyStore.jks
Cipher Suite	TLS_RSA_WITH_AES_256_CBC_SHA256
Additional Parameters	

Apache Active MQ JMS Receiver Configuration Example

Delivery Method

JMS Delivery

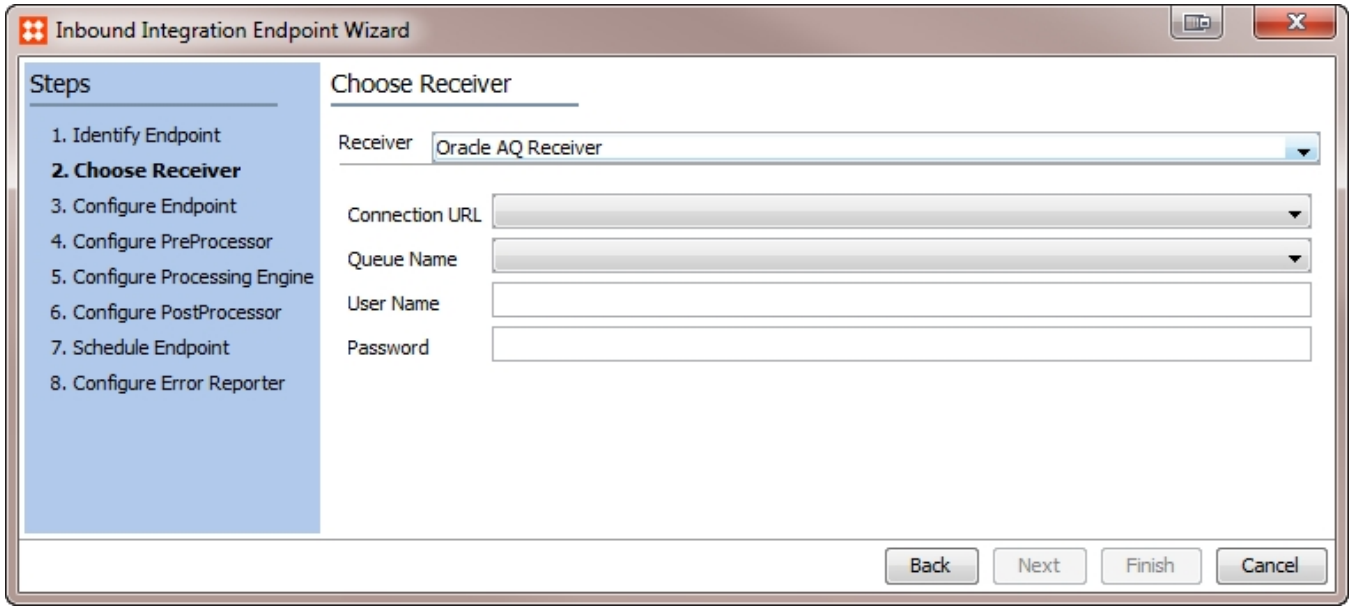
JMS Connection Factory Class	ActiveMQInitialContextFactory
Connection Factory Name	ConnectionFactory
JMS Provider URL	tcp://ATTCM3S9:61616
JMS Queue	testqueue
Binary Payload	No
User Name	
Password	
Additional Parameters	

Edit Delivery

Oracle AQ Receiver

Note: Options available on dropdown fields must be configured in sharedconfig.properties or config.properties. Select a dropdown to display the required key name for each field. Contact Stibo Systems for assistance in configuring additional URLs.

1. In Receiver, click **Oracle AQ Receiver**.



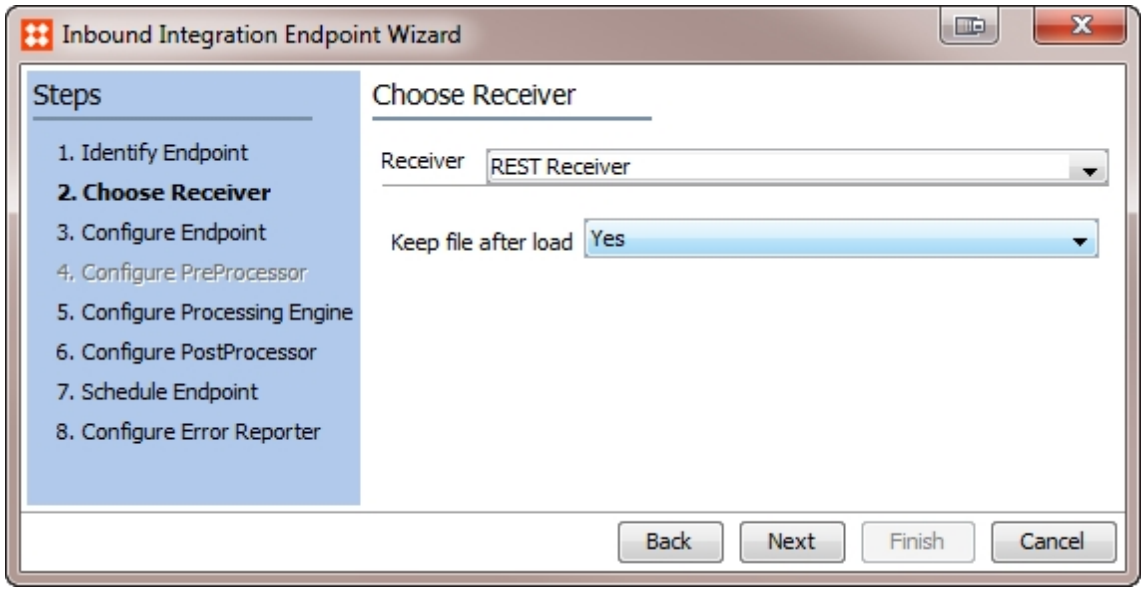
2. For **Connection URL**, select a URL pointing to Oracle AQ.
3. For **Queue name**, select an Oracle AQ queue name.
4. For **User Name**, enter a user name to be used to log on Oracle
5. For **Password**, enter a password to be used to log on Oracle.

REST Receiver

REST stands for Representational State Transfer. It is a software architecture style that you can use to design web services. If you want to learn more about REST, you can find multiple resources on the internet.

For information about how to use the STEP REST API, see the STEP API documentation.

1. From the Receiver list, select **REST Receiver**.



2. For **Keep file after load**, specify if the files uploaded by the REST API to be removed after the files have been processed. Selecting 'Yes' will require periodic manual cleanup on the server. Selecting 'No' cleans off the files automatically. This option is available because files posted via REST are temporarily stored in a directory on the application server (under Hotfolder root).

Note: The path used for the file upload is system specific, and is automatically used when new REST receivers are created. In the configured path, a folder named REST is created and files that are uploaded using REST API appear in this folder.

When an IIEP has been configured to use the REST Receiver, XML files can be posted to the following URL:

```
[Host]/restapi/integrationendpoints/[Endpoint ID]/upload?context=[Existing Context]&workspa
ce=Main
```

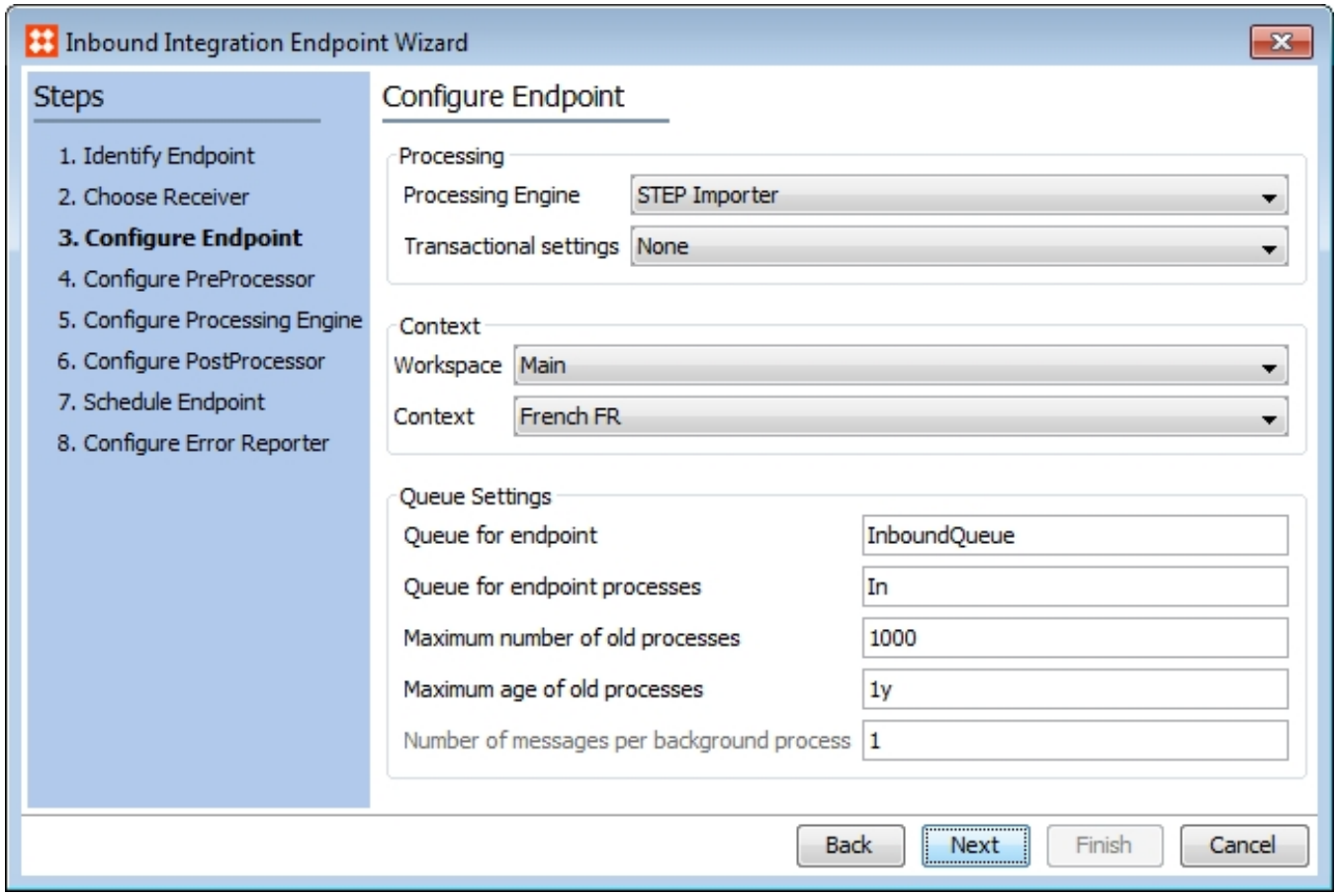
The request header 'Content-Type' must have the value "text/xml" and basic authorization must be used. For example, the 'Authorization' header must have the value "Basic " combined with a 64 bit encoding of [Username]:[Password]. For example, 'Basic c3RlcHN5czpzdGVwc3lz'.

Notice that in the URL shown above, the context and workspace qualifiers have no significance, but they are required for the REST API.

The POST will in itself invoke the endpoint, so there is no reason to schedule a REST-based endpoint. The resource for the REST POST invoke request is:

```
[Host]/restapi/integrationendpoints/[Endpoint ID]/invoke
```

Step 3 - Configure Endpoint



1. For **Processing Engine** on a standard STEP system, you can only select **STEP Importer**. The STEP Importer enables you to use the same import functionality as in the STEP Import Manager.
2. For **Transactional Settings** specify if the processes will be chained, strict, or without any settings. For details on the effects of each option, see **Inbound Integration Endpoint Transactional Settings** in the **Integration Endpoints** documentation.

   Note: When creating or modifying an endpoint for Transactional Setting = Chain, after selecting the Chain option, close the wizard and reopen it to display the 'Maximum number of waiting processes' parameter.

   ---

3. The queue size of a background process with strict transactional settings must be 1. A strict transactional setting does not allow multiple background processes to work in parallel / concurrently.
4. For **Context**, select the workspace and context that you want the endpoint to import data into. Common setup is to use Main.

   The selected context and workspace affects import formats where an import context and workspace is not automatically included in the import file.

If the endpoint is used to import STEPXML files, the context and workspace specified in the STEPXML file is used and the contexts and workspaces you select here are ignored.

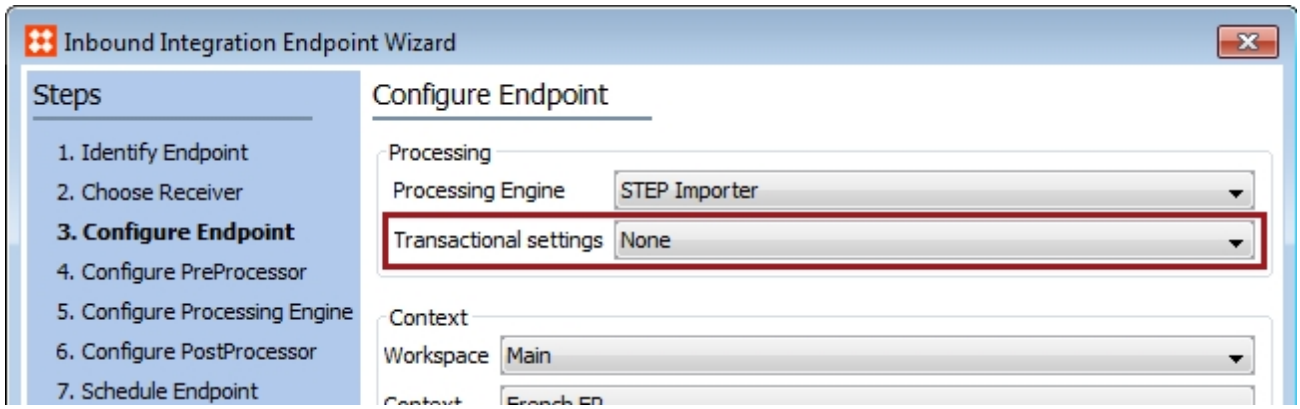
If a CSV or Excel format is used a single context is exported. To output multiple contexts you must configure the Split Context post-processor.

5. For **Queue Settings**, set up the background processes to be used for the endpoint.
   - For **Queue for endpoint**, enter a name for the queue that is used by the IIEP Background Process to poll the endpoint. The first time you activate the endpoint, a queue with the specified name is created if it does not already exist. If in doubt about how to populate this parameter, create a new queue for the IIEP.
   - For **Queue for endpoint processes**, enter a name for the queue that is used by the Background Processes started by the endpoint to handle the actual import. The queue is automatically created on the system if it does not already exist. High priority integrations or integrations with long-running processes should typically have their own queue.
   - For **Maximum number of old processes**, specify the maximum number of ended processes the system is allowed to have. This auto-cleanup option deleted succeeded and ended processes started by the IIEP when the limitation is exceeded. The oldest processes are deleted first.
   - For **Maximum age of old processes**, specify the maximum age of succeeded and ended processes. Use the following case-sensitive notation: y = years, M = months, w = weeks, d = days, h=hours, m = minutes and s = seconds.
   - For **Number of messages per background process**, specify the number of messages that will be handled in one background process before a new background process is generated. This is useful if the endpoint handles many small messages that would otherwise generate a large number of background processes. Common setup is to let several messages / files be handled per background process. This field is only enabled if the transactional setting is strict.

Inbound Integration Endpoint Transactional Settings

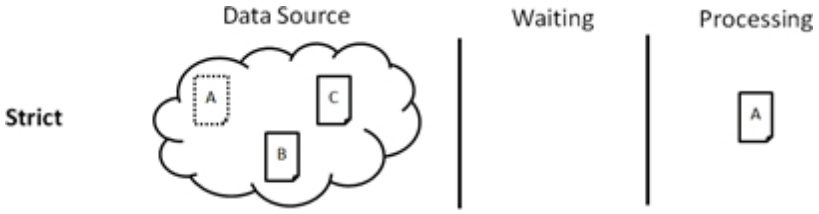
Transactional Settings specify how the messages will be processed. Available options are based on the selected Receiver and the Transactional Setting determines what other parameters are available in the additional wizard steps. There are three possible transactional settings: Strict, Chain, and None.

For more information, see **Step 3 - Configure Endpoint** of the **Integration Endpoints** documentation.



Strict

Using a strict transactional setting processes data in a strict order, one background process at a time. When one process completes successfully, the endpoint starts the next background process. If a background process fails with an error or if it cannot deliver a file, the next process is not started and the 'Failed' status is set.



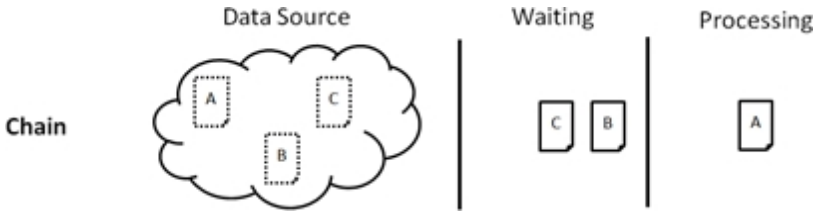
Chain

Using a chained transactional setting processes data in batches of chained background processes. A batch is generated each time the endpoint polls for data and finds more than one message / file that has not yet been processed.

The **Maximum number of waiting processes** field allows you to specify the maximum number of background processes with the status 'Waiting' that is allowed. If a background process in a batch fails, the remaining background processes in the batch will also fail. However, the endpoint remains active with the status 'Running', and continues to process data in the next batch of chained background processes. A chained endpoint also stops if a file cannot be delivered.

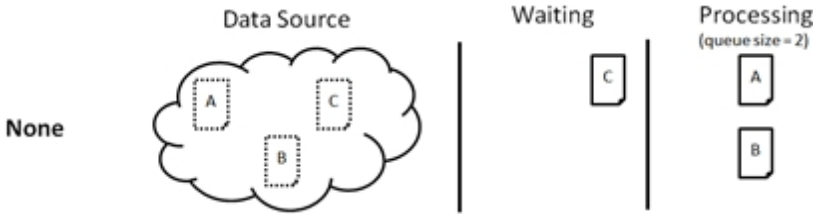
Note: When creating or modifying an endpoint for Transactional Setting = Chain, after selecting the Chain option, close the wizard and reopen it to display the 'Maximum number of waiting processes' parameter.

The queue size of a background process with chained transactional settings must be one (1). A chained transactional setting does not allow multiple background process to work in parallel / concurrently.



None

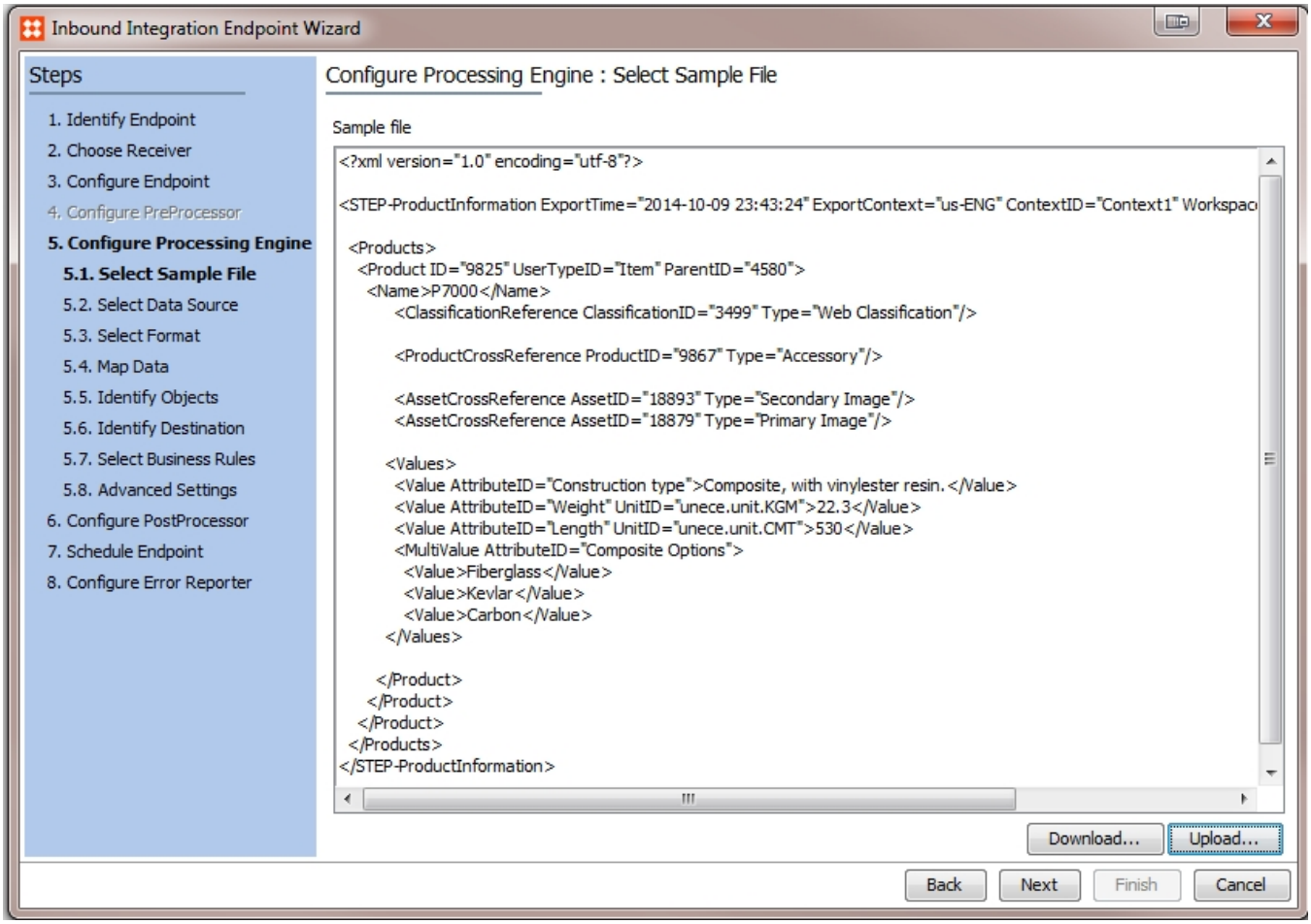
Using the 'none' transactional setting processes data in parallel processes, without any transactional restrictions or data dependencies (useful, for example, when processing assets). This allows the queue size to be larger than one (1). Data is not processed in a strict order and if one background process fails, the endpoint continues to process data in the next background process in queue.



The queue size of a background process with strict transactional settings must be one (1). A strict transactional setting does not allow multiple background processes to work in parallel / concurrently.

Step 5 - Configure Processing Engine

When using the STEP Importer, this step and its potential sub-steps, will typically be identical to the steps in the STEP Import Manager. Additionally, the same options are available for mapping and transforming data, and for referencing business rules to be run during import.



Before you configure the processing engine, you must create a sample file that contains the basic structure of the data.

1. In **Select Sample File**, click **Upload**, to upload the sample file. The sample file is used in the following steps in the wizard to configure how to import data. The sample file is stored in the database, but the file is not imported into your system.

   If you want to modify a sample file that has already been uploaded, click **Download**, and save the sample file to your computer. Make your changes, and then click **Upload** to upload the modified sample file to the endpoint.

   Note: When importing ETIM6, modify the import file to include only the <IXF >, <Header>, and <Groups> sections and save it as your Sample.

   ---

2. In the remaining sub-steps, standard Import Manager functionality is available:

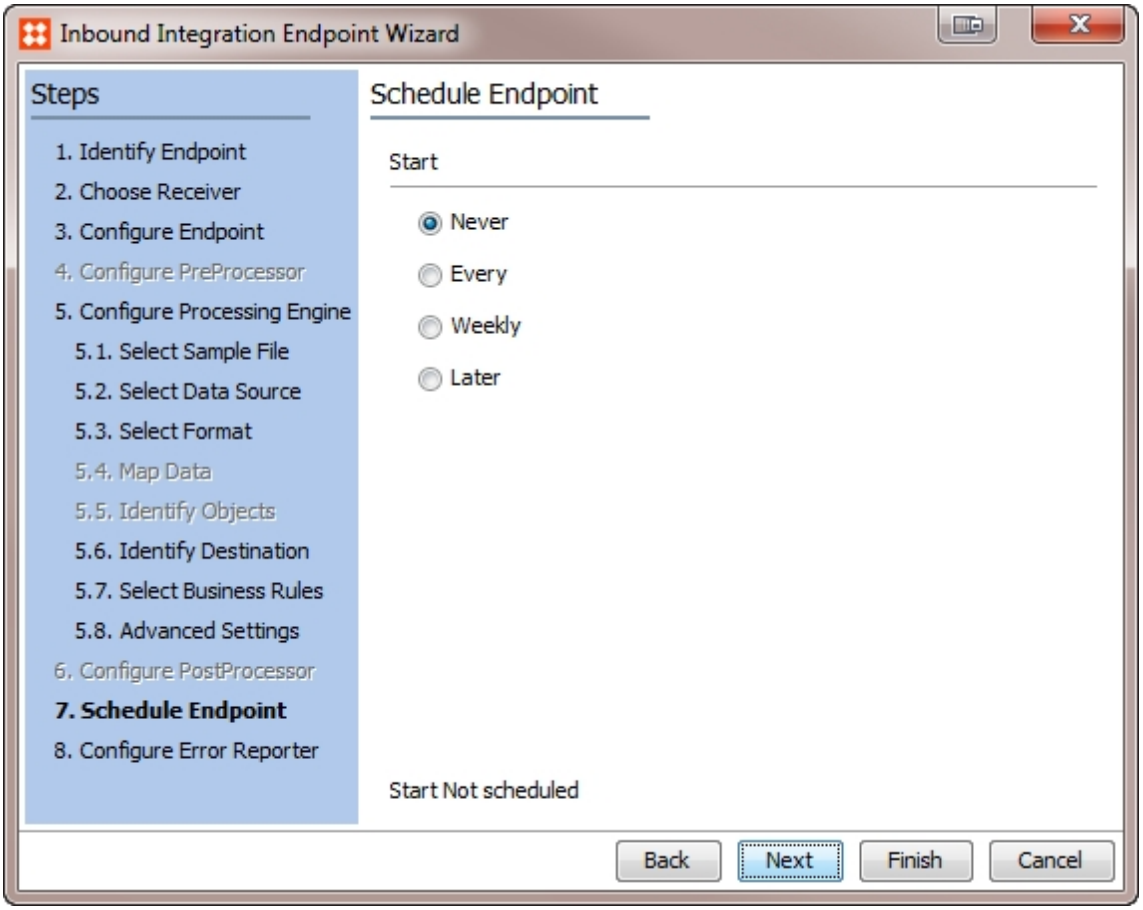
- **5.2 – Select Data Source** of the data. If the source is a File, open a browser and select the load file that you want to use. Your additional source type choices are FTP or an ODBC connection.
- **5.3 – Select Format** to indicate the type and basic parameters of the file you are importing.
- **5.4 – Map Data** to match the columns of data in the import file to the appropriate STEP equivalents, including transformations.
- **5.5 – Identify Objects** to provide a basic confirmation if the products in the input file are new or existing. You can also specify whether to locate products by attribute value rather than by the product ID or name.
- **5.6 – Identify Destination** to specify a number of different settings such as the user who is responsible for approving the objects after import, where to place new objects in STEP, and the object type of new objects. Be cautious if setting default parent and object types.
- **5.7 – Select Business Rules** to specify which business rule(s) to apply when data is imported.
- **5.8 – Advanced Settings** allow you to set a number of options that are not required on a day-to-day basis for data imports but are mainly used to clean up data within STEP. Use these settings with caution to avoid causing data integrity issues.

For detailed information about each step, see **Data Import Manager Wizard** in the **Import Manager User Guide**.

Step 7 - Schedule Endpoint

Use Schedule Endpoint to specify how often the endpoint should search for data to be processed.

IIEPs that are not REST Receiver should typically be scheduled to poll the data source for new files / messages with regular intervals. Non REST Receiver based IIEPs also can be invoked via a REST POST request. With this functionality, for example, a system delivering data to STEP could potentially invoke the IIEP after having uploaded a file to a Hotfolder.

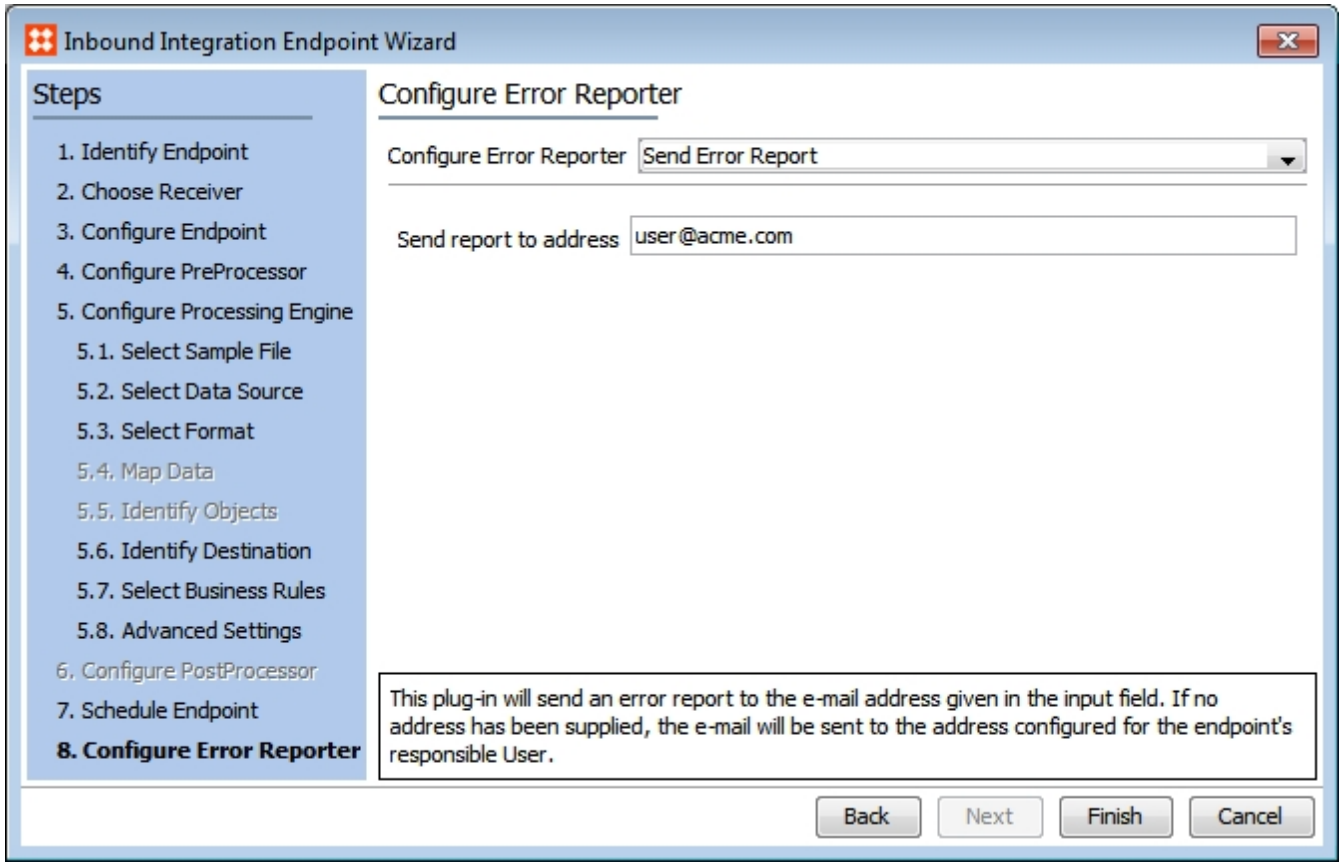


- Click **Never**, if the endpoint process has to be started manually.
- Click **Every**, **Weekly**, or **Later**, to specify how often you want the endpoint process to run. Different options are available depending on the frequency you select. The minimum interval is once every minute.

Step 8 - Configure Error Reporter

In this step, you select and configure an error reporter plug-in that is activated if an endpoint-related background process fails.

The error reporter emails information about the failed endpoint, the background process, the failed file, the failing process step, the cause of the error, and a copy of the file that triggered the error.



1. Click **Configure Error Reporter** and select **Send Error Report**. If you select **Not Defined**, the error reporter is disabled.
2. In **Send report to address**, enter the e-mail address of recipient of the error report.

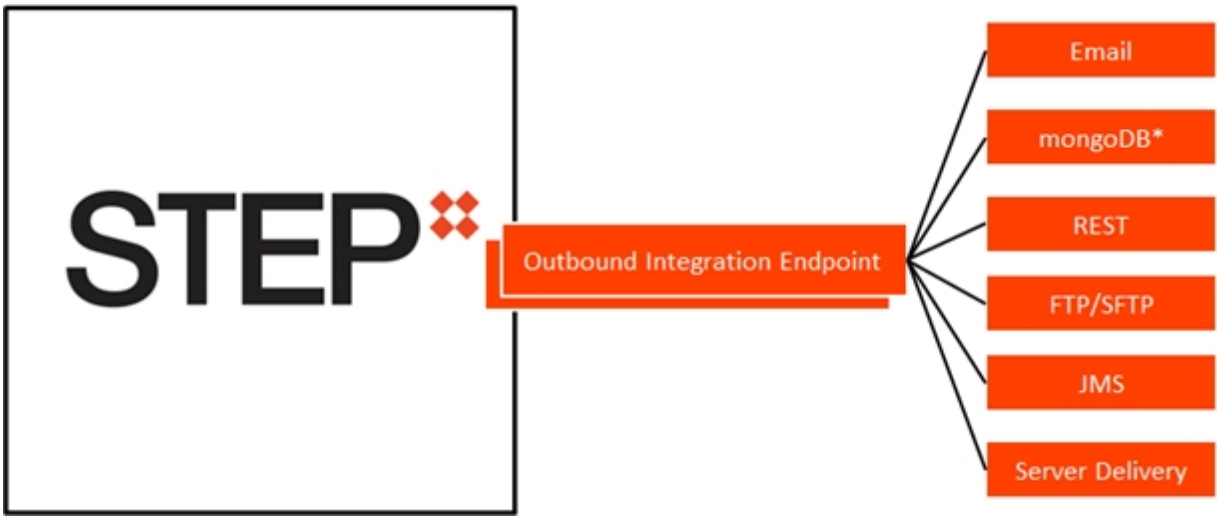
   If you do not enter an e-mail address, the error report is sent to the e-mail address of the user who created the integration endpoint. If no e-mail is defined for this user, or if no mail server is defined in the configuration, the error report is written to the failed Background Process Execution Log.

3. Click **Finish**. The endpoint is created and appears in the setup group. Before the endpoint can start processing data, you need to enable the endpoint. For more information, see the **Integration Endpoint Status** section of the **Integration Endpoint** documentation.

Outbound Integration Endpoints

The outbound integration endpoint (OIEP) functionality can, in many ways, be thought of as wrapper functionality for the standard STEP export functionality. In the simplest form, a static selection OIEP works exactly like a scheduled export that non-incrementally publishes data from selected hierarchies to external systems with scheduled intervals. The only difference is that an OIEP has more standard delivery options than the Export Manager and offers extended monitoring capabilities.

With the core functionality, the following OIEP delivery options are available, although mongoDB requires special license.



OIEPs can also be used for incremental publishing. With and Event-Based OIEP, the data is published based on events occurring in STEP. An event, for example, could be generated when an object in STEP is approved. Multiple OIEPs can be configured to listen for such an event and each will have an associated Event Queue where registered events are queued. The OIEP can then, with scheduled intervals (minimum interval being once every minute), check the Event Queue for new events and publish data accordingly.

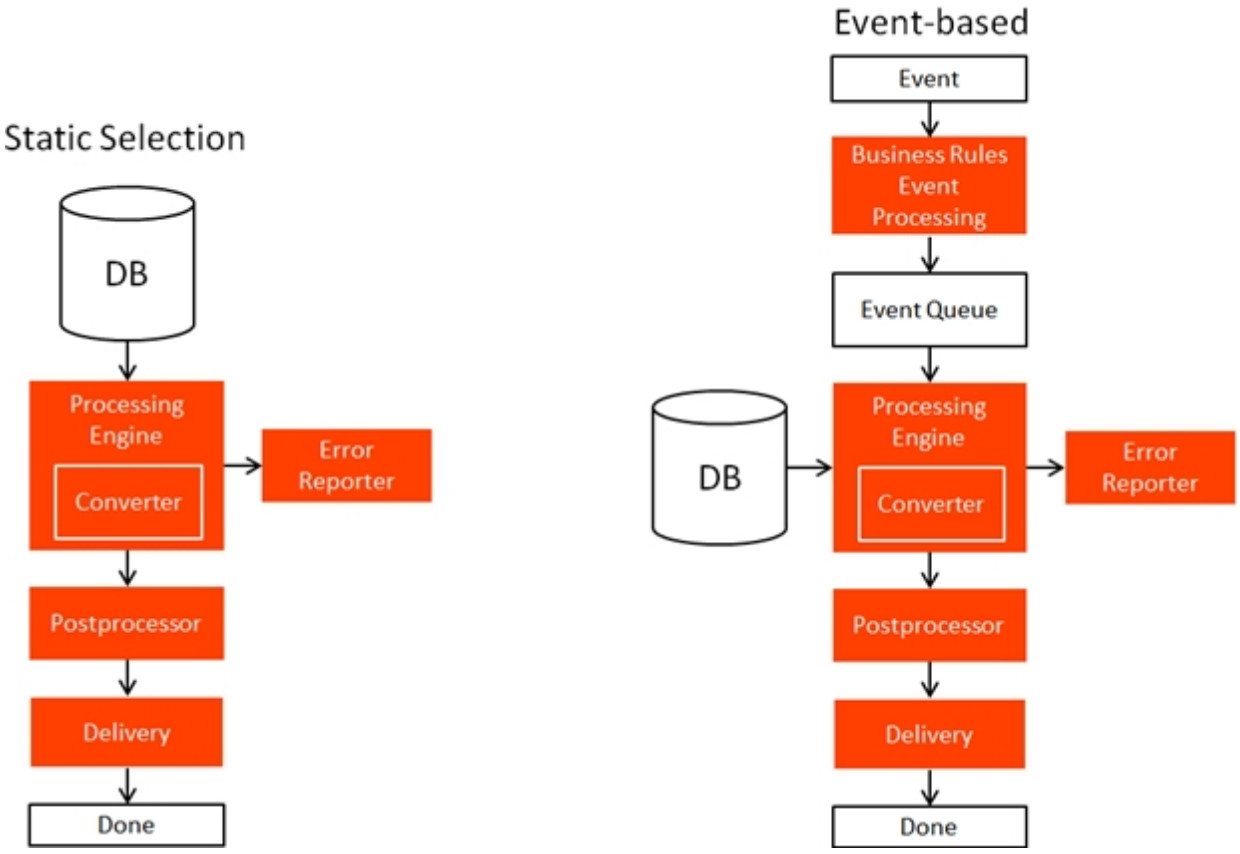
To create an OIEP, you will perform two sets of tasks in **System Setup**:

1. Use the Outbound Integration Endpoint wizard to set up the type of OIEP required:
   - To export a static set of objects independently of events (always export the same data set), see **Select Object Outbound Integration Endpoint Wizard**.
   - To listen for system events tied to specific objects in STEP, see **Event-Based Outbound Integration Endpoint Wizard**.
2. Configure the settings in the Outbound Integration Endpoint editor, which includes:
   - Configure an Outbound Integration Endpoint Output Template including output format and the post-processor, if any. For an Event Queue Data Source, specify the event types that are triggered for the selected objects. The available output formats for both OIEPs and Export Manager are the same regardless of the data source selection.

- Configure Outbound Integration Endpoint Event Messaging for Event Queue Data Source endpoints, including configuring the event triggers and setting the endpoint to read events.
- Configure an Outbound Integration Endpoint Delivery Method to determine how data is delivered. The available delivery options for both OIEPs and Export Manager are the same regardless of the data source selection.
- Apply Outbound Integration Endpoint Derived Events when the system events—create, modify, and delete—are not sufficient for the desired output.
- Schedule Outbound Integration Endpoint to determine when it runs.
- Configure Outbound Integration Endpoint Error Reporter to send an email if an endpoint-related background process fails.

Outbound Integration Endpoint Structure

The OIEP functionality has been created to allow for easy extensions / customizations and can be seen as a plugin framework with interchangeable parts. The graphic below illustrates the structure for both a non-incremental endpoint based on a static selection and an event-based endpoint. The colored boxes represent the interchangeable parts, which are described below.



Business Rules Event Processing

For event-based endpoints, business rules can be used to filter events and generate new events based on others.

Processing Engine / Converter

The processing engine is responsible for performing the actual data export based either on events or a static selection. STEP Exporter is the only available option with the core functionality, and is also the standard functionality used for manual exports. If a special format not supported in the standard STEP Exporter is required, the processing engine can be replaced via an extension. Most often, however, creating a converter plugin for the STEP Exporter is sufficient for such cases. This approach also has the benefit that the same converter plugin is available for manual exports.

Post-Processor

A post-processor plugin has access to the file delivered by the processing engine and can manipulate it and/or split is into multiple files. STEP includes two post-processor plugins, both used for manipulating files output by the processing engine with cross context information.

Error Reporter

STEP includes a single Send Error Report plugin to report errors. An email can be sent to a specified address if errors occur when files / messages are handled by the processing engine.

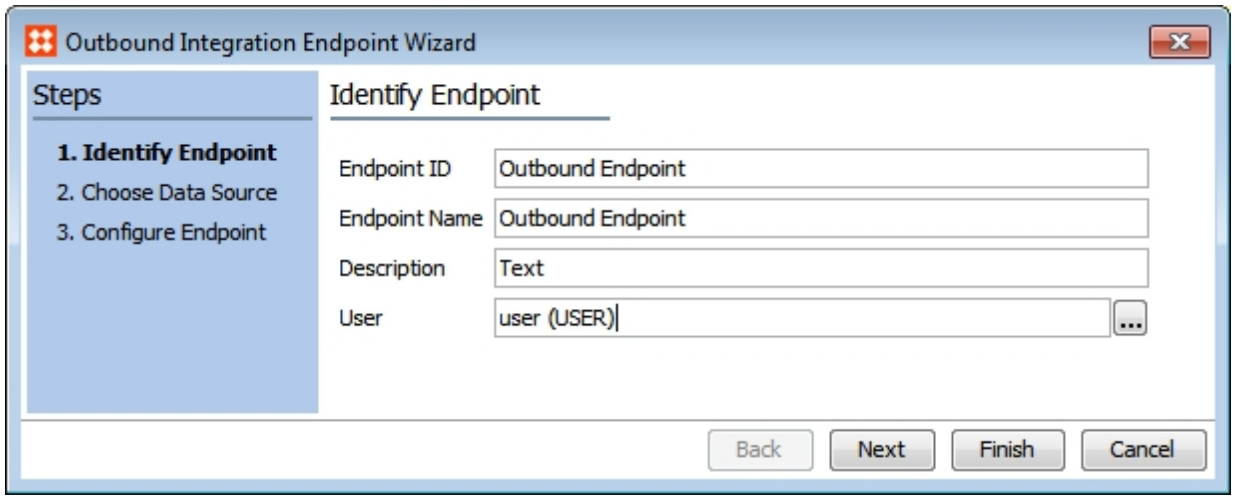
Delivery

A delivery plugin has access to the file(s) output from the processing engine / post-processor. Additional delivery options can be added with the delivery plugin extension.

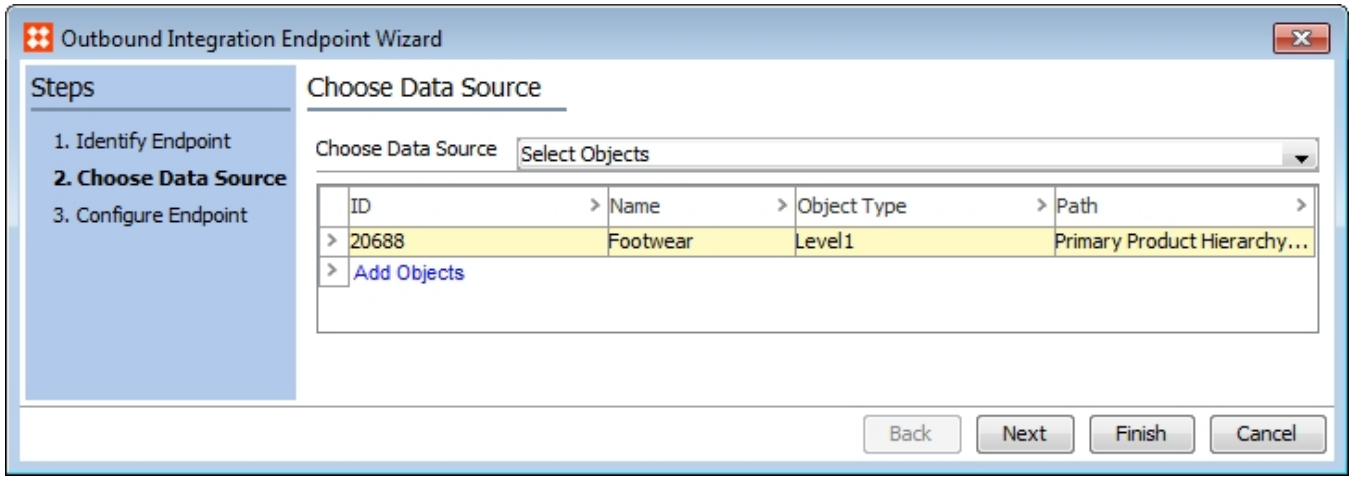
Select Object Outbound Integration Endpoint Wizard

A static selection OIEP works in a way that is similar to Export Manager. Use this option for non-incremental exports where you want to republish all data for specific hierarchies every time the endpoint is invoked.

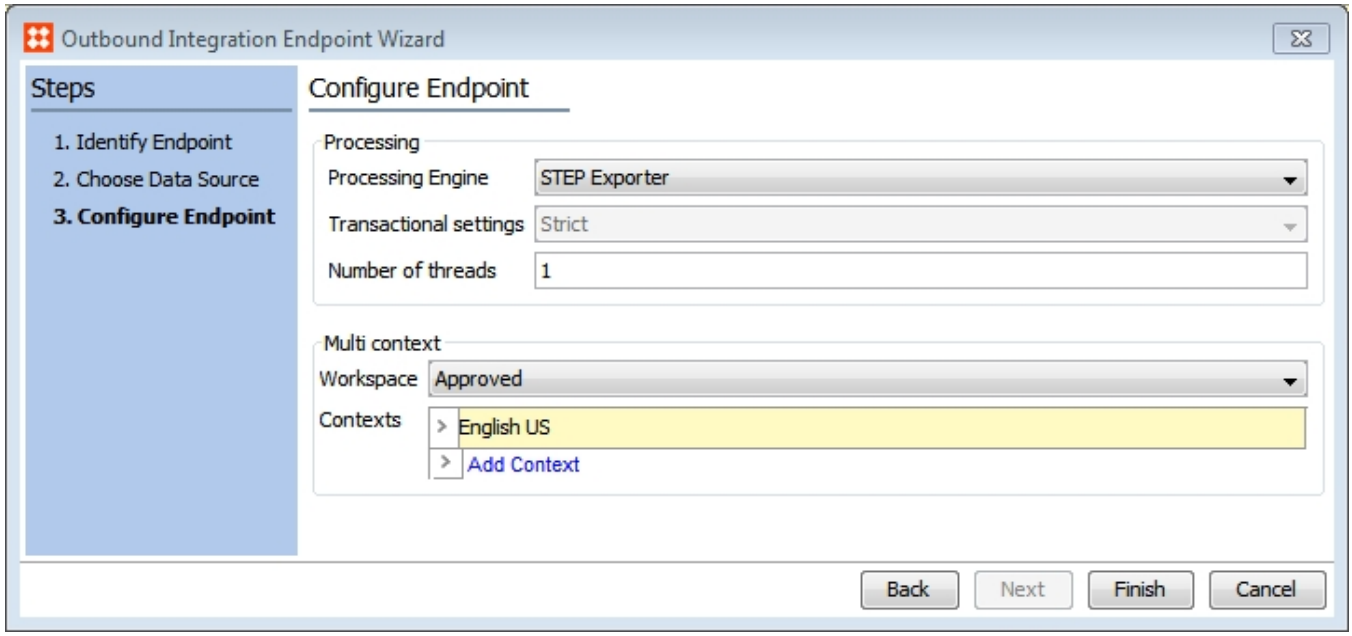
1. In System Setup > right-click Integration Endpoint setup group > **Create Outbound Integration Endpoint**. The Outbound Integration Endpoint wizard appears.
2. On Step 1, identify the outbound endpoint by adding an **ID**, a **Name**, and a **Description**.
3. Use the search or browse for a **User**. The privileges of the selected user determine which actions the integration endpoint can perform when it processes data. Common setup is to create a special system user for this purpose.



4. On Step 2, choose the data source from the following options:
   - **Select Objects** Select one or more root node object hierarchies to be exported. Selecting an object hierarchy is not required if the endpoint will export system-specific data, such as attributes, LOVs, and units that are not used in any specific product, classification, or entity hierarchy.
   - **Event Queue Data Source** For detailed steps, see **Event-Based Outbound Integration Endpoint Wizard**.



5. On Step 3, set the **Processing** options:
   - For the **Processing Engine** select STEP Exporter. This is the only processing engine option on a standard STEP system. If you need a customized processing engine, please contact Stibo Systems.
   - The **Number of threads** option is only valid for an Event Queue Data Source endpoint.



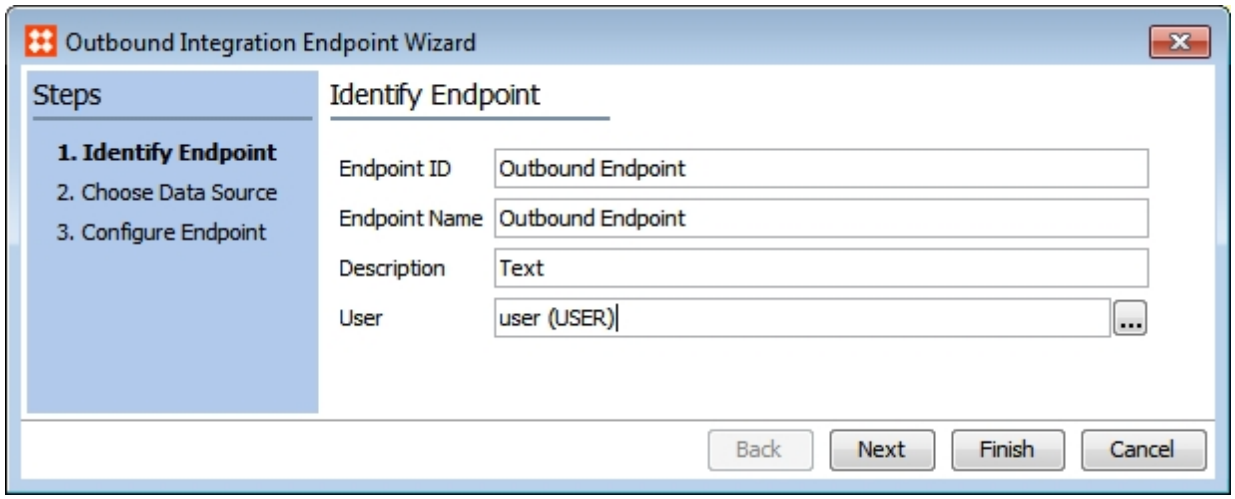
6. On Step 3, the **Transactional Settings** option does not have the same significance for OIEPs as they do for Inbound Integration Endpoints. The 'Strict' mode is set automatically, and while you can specify the Transactional Setting for OIEPs based on a static selection, the setting has no impact on the processing. For more information about Transactions Settings, see the **Inbound Integration Endpoint Transactional Settings** section of the **Creating Inbound Integration Endpoints** documentation.

7. On Step 3, set the **Multi context** options by selecting the workspace and context that the endpoint should use to export data.
8. Click **Finish** to display and complete the configuration in **System Setup** on the **Outbound Integration Endpoint** editor.
9. Follow the steps to configure the remaining settings in the Outbound Integration Endpoint, as discussed in the **Outbound Integration Endpoints** section of the **Integration Endpoint** documentation.

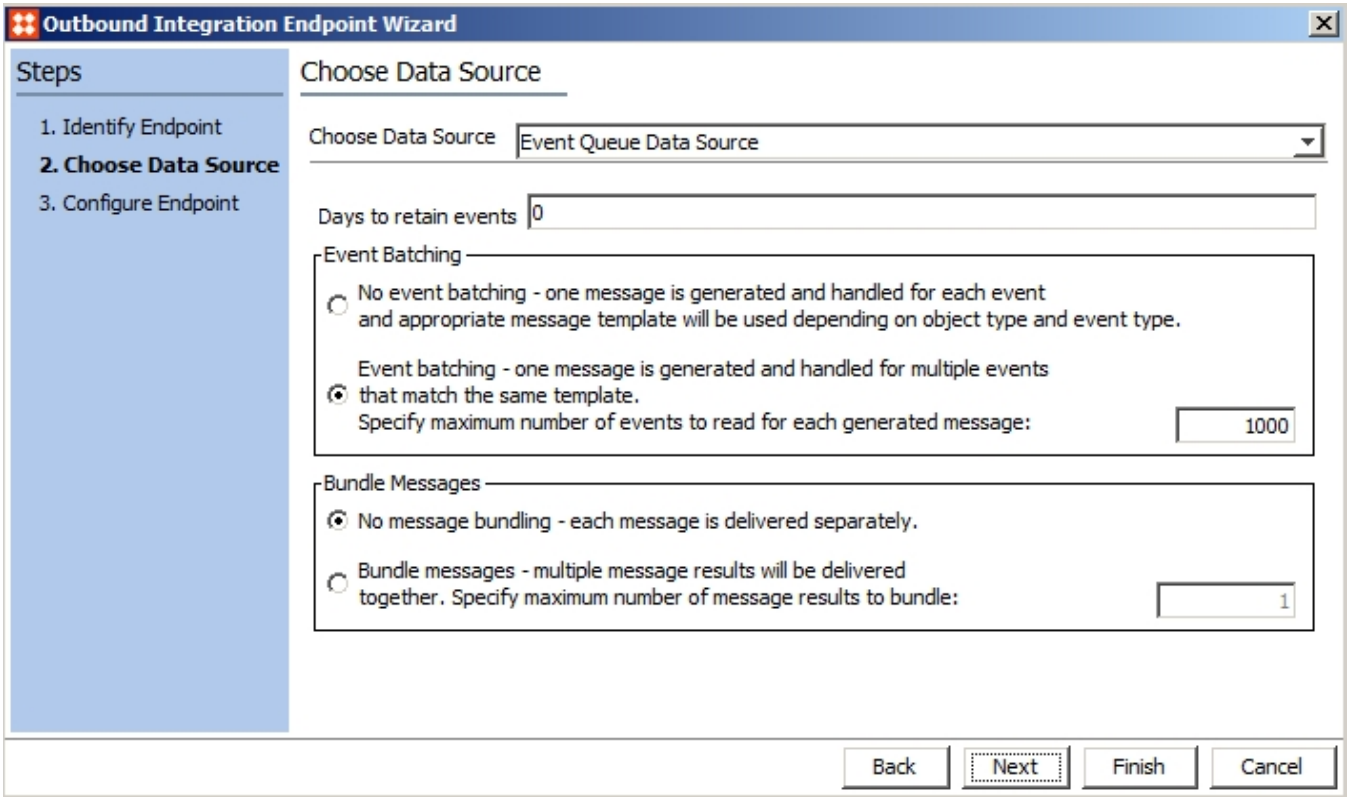
Event-Based Outbound Integration Endpoint Wizard

Events are generated when objects change in STEP. Use this option for incremental exports where data should be published based on events. An Event Queue is automatically created and associated with an event-based OIEP. The OIEP is configured to listen for specific events, and when those events are generated, the event queue holds the events to be exported by the OIEP. When the OIEP is invoked, it handles all unread events in the queue in a single background process, and marks them as read.

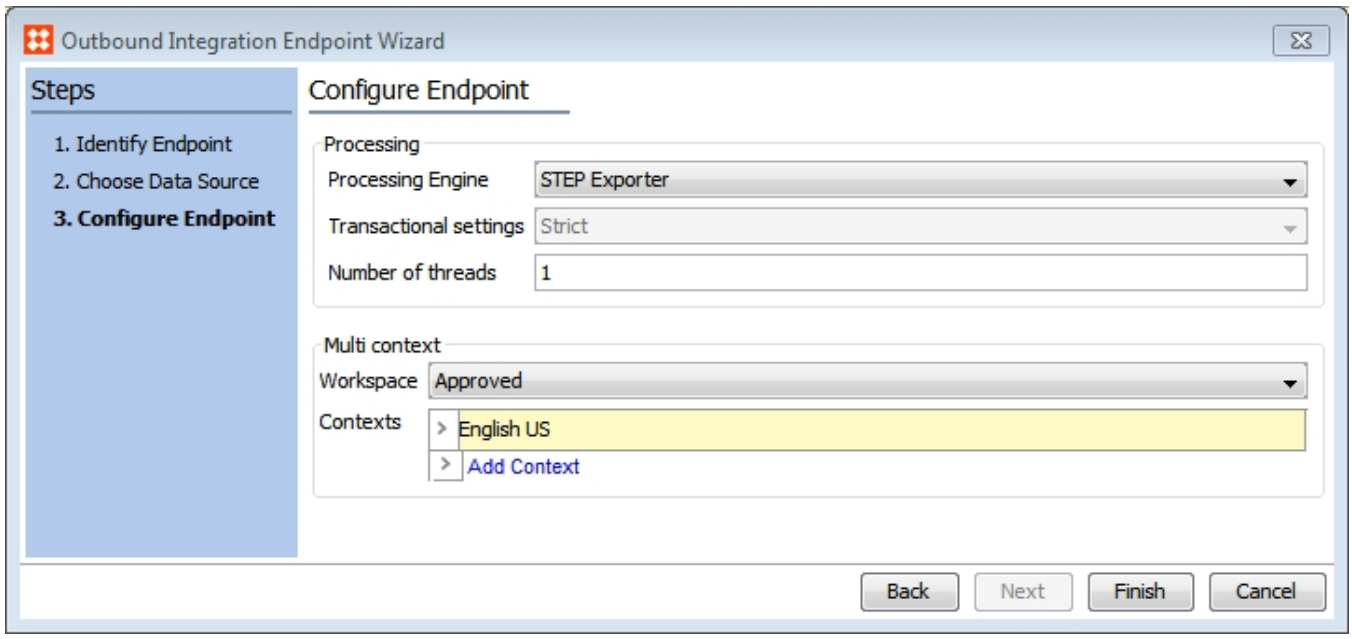
1. In **System Setup**, right-click the **Integration Endpoint** setup group, then click **Create Outbound Integration Endpoint**. The **Outbound Integration Endpoint** wizard appears.
2. On Step 1, identify the outbound endpoint by adding an **ID**, a **Name**, and a **Description**.
3. Use the search or browse for a **User**. The privileges of the selected user determine which actions the integration endpoint can perform when it processes data. Common setup is to create a special system user for this purpose.



4. On Step 2, choose the data source from the following options:
   - **Event Queue Data Source** Select to display the options required for an event-based OIEP.
   - **Select Objects** For detailed steps, see **Select Object Outbound Integration Endpoint Wizard**.



5. For the **Days to Retain Events** field, enter the number of days for the endpoint to retain events once they are read. Setting this number > 0 means that read events will stay on the queue for the specified number of days. This can be used as a security measure. If something goes wrong with the receiver, retaining read events will allow you to republish data for the retained events again later. Common setup is to use between 5 and 10 days.
6. For the **Event Batching** field, select an option to determine how many events should be included in each exported file. For more information, see **Outbound Integration Endpoint Event Batching**.
7. For the **Bundle Messages** field, select an option to determine if multiple files generated by the same background process should be delivered together. Common setup is to use No Message Bundling since this option is only supported by some delivery plugins.
8. On Step 3, set the **Processing** options:
   - For the **Processing Engine** select STEP Exporter. This is the only processing engine option on a standard STEP system. If you need a customized processing engine, please contact Stibo Systems.
   - The **Transactional Settings** option is not enabled for Event Queue Data Source.
   - For **Number of threads**, if multithreading is desired, insert the number here. Increasing the number of threads increases export performance. When using multiple threads, the batch size should be increased proportionally. The batch size is configured in Step 2, **Choose Data Source**. For more information, see **Outbound Integration Endpoint Multithreading Support**.



9. On Step 3, set the **Multi context** options by selecting the workspace and context that the endpoint should use to export data.
   - For **Workspace**, common setup is to use the Approved workspace, except when you need to generate events for objects before they are approved, for example, during import or from a workflow.
   - For **Contexts**, when exporting data in the STEPXML format or when using a custom cross-context enabled format, one file can contain data from multiple contexts. For other formats, the standard Context splitter Post-processor plugin should be used. Using this plugin, separate files are generated for the selected contexts.
10. Click **Finish** to display and complete the configuration in **System Setup** on the **Outbound Integration Endpoint** editor.

For information about setting up an event processor, see **About Event Processors** in the **System Setup / STEP Super User Guide**.

A number of triggers can be specified for each event queue. For detailed information on the configuration of event triggers, see **Event Triggering Definitions**.

For information on how to configure event messaging, see **Outbound Integration Endpoint Event Messaging**.

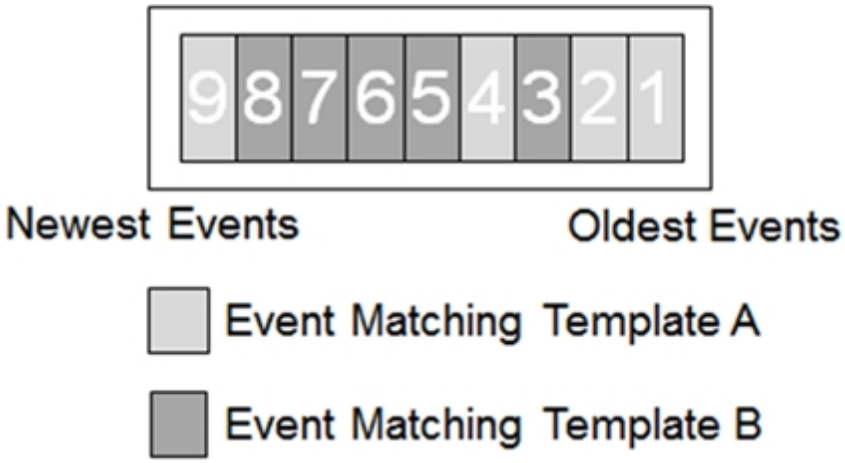
Outbound Integration Endpoint Event Batching

The Event Batching option on an event-based OIEP does not determine the number of objects in a message, but rather, the number of events that will be processed and included in a single message. STEP does not wait for the number of events to occur before generating a message, instead, it caps the number of events in a single message so that 2+ messages are generated if more events are present than the batch number allows when the system is polled.

Unless it is a strict requirement that one file should be exported per event, common setup is to use event batching and leave the default value of 1000, or set even higher if many events are registered and the endpoint is invoked infrequently. Appropriate batch size is typically based on size of messages and downstream system processing capabilities, for example, use smaller batch size for larger messages and vice versa.

STEP batches based on output template, which includes a combination of event type and object type.

Consider the unread events in this illustration, which matches either 'Template A' or 'Template B':



If you work with a batch size of 3, data related to the events will be published as follows:

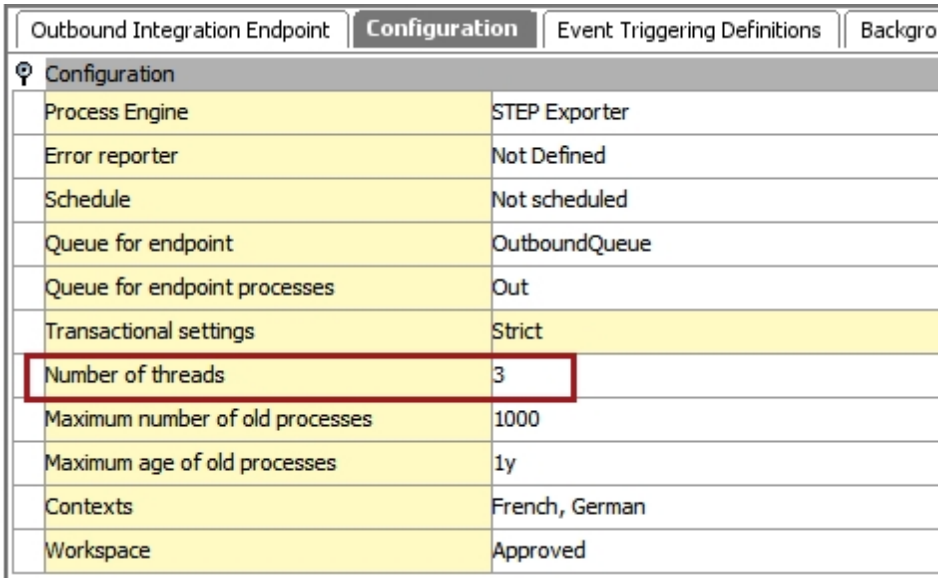
- One file with data for Event 1 & 2 (Template A)
- One file with data for Event 3 (Template B)
- One file with data for Event 4 (Template A)
- One file with data for Event 5, 6 & 7 (Template B)
- One file with data for Event 8 (Template B)
- One file with data for Event 9 (Template A)

Outbound Integration Endpoint Multithreading Support

To increase performance when exporting from an Outbound Integration Endpoint (OIEP) you can increase the number of threads for a given OIEP. This applies only to OIEPs using an event queue as the data source.

Multithreading Setup

The **Number of threads** setting is available on the Configuration tab for both Select Objects and Event-based OIEPs.

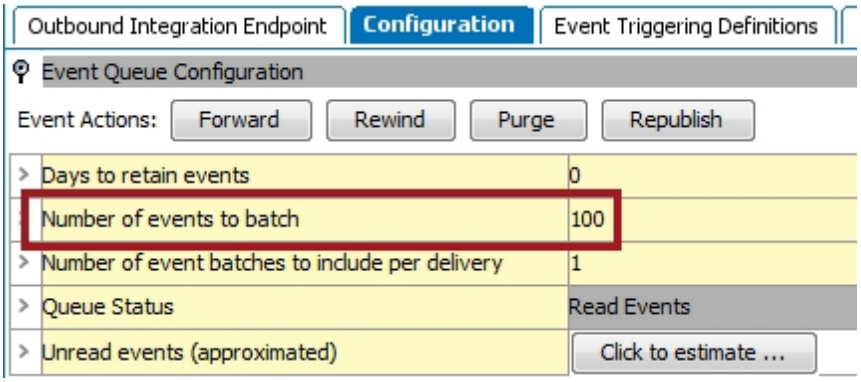


The default thread setting is one (1), in which case the endpoint produces a single message at a time, with all events in the batch processed serially. Increasing the thread number results in each batch size being divided by the thread number so that the contents of a batch can be processed in parallel. Note that when all threads are complete, the batch will still yield a single message, as is consistent with the 'Strict' transactional setting required by event-based OIEPs. Thread size is typically increased for particularly critical information where the speed with which the message is produced is key. However, increasing the setting beyond the capabilities of the hardware will impede the overall performance of the application server, so care must be taken in adjusting the thread size.

Additional Considerations

Increasing the Batch Size

When increasing the number of threads running, the size of the batch will decrease proportionally (e.g., a batch size of 100 using two threads will be split into two batches of 50). When adjusting the number of threads, it is therefore worthwhile to experiment with increasing the batch size, in order for each working thread to have a reasonable amount of data to process.



Serialized and Parallelized

The batch-fetching of the events is run serially and the data distributed as evenly as possible to each thread. From fetching the batch on through the delivery stage, the pre-processing, main processing, and post-processing take place in parallel. Following post-processing (if applicable), the data are handed over to the delivery stage which is again run serially. Events are therefore delivered as if everything was executed serially. With this in mind, it is important to consider the schedule of the endpoint and ensure that it is set to check for events as frequently as if the endpoint were single-threaded.

Note that when a batch is processed in parallel (via multiple threads), it is recorded in the execution report of the endpoint.

Outbound Integration Endpoint Output Template

Use the output template to define the data to be exported and the desired format. Multiple output templates can be created. This enables you to have one type of output for a specific object type, combined with a specific event type, and another type of output for a different object type, combined with a different specific event type.

Common setup is to use a single output template for a static selection OIEP since the selected data determines the template. For event-based OIEPs, using multiple output templates can allow different messages to be generated based on the object type / event type combinations. Keep in mind that batching is restricted to a single template.

Note: A least one output is required for both Select Objects and Event-based OIEPs.

Configuring an output template involves the following steps:

- Configure the object / event type
- Specify Outbound Integration Endpoint Format
- Configure Post-Processor
- Add pre-processor and post-processor, based on availability on your system.

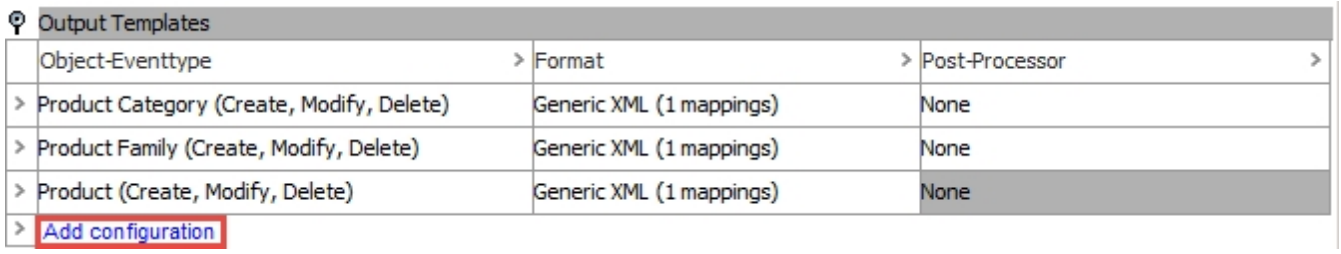
Important: For an event-based outbound integration endpoint, event triggers must be specified before the output template can be configured. However, an output template must also be created for each trigger, otherwise the trigger will not output any data. Events must also be specified for each object type selected. The three basic system events are: Create, Modify, and Delete. For more information, see **Outbound Integration Endpoint Event Triggering Definitions**.

Reasons to Use Multiple Outbound Integration Endpoints

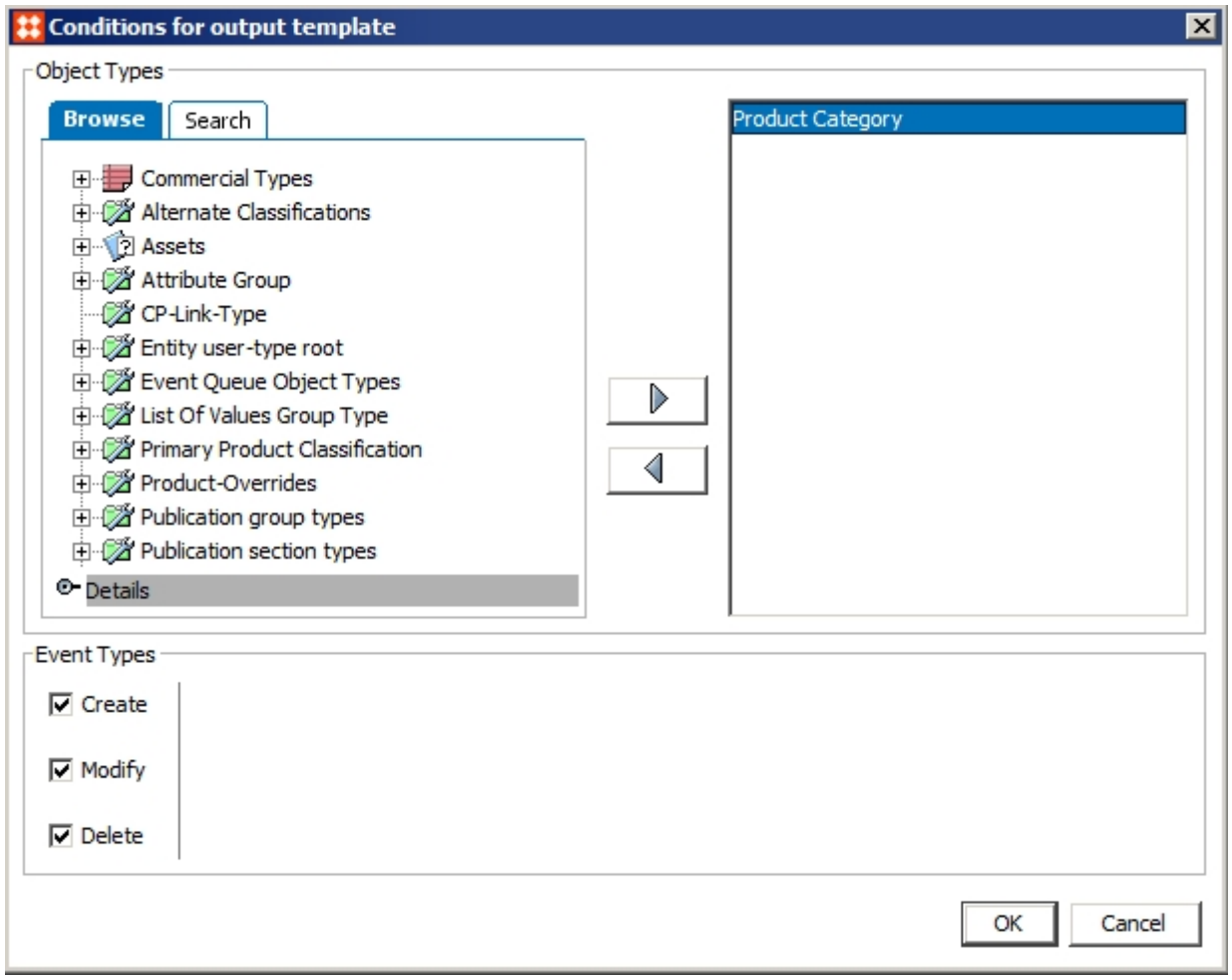
- Triggering events for event-based endpoints are global. For example, you may want a change in STEP name to trigger an event on some objects, but not others. This scenario requires a different configuration of triggers on multiple endpoints.
- Delivery method is per endpoint, not per output format. If you need different outputs go to different external systems, you will need different endpoints.
- Messages generated via events and those generated via object selection cannot be combined in a single endpoint. Even when the format is the same, these scenarios require separate message generation and therefore require separate endpoints.

Configure Output Template

1. In **System Setup**, select the relevant outbound integration endpoint, then click the **Configuration** tab. This image shows an event-based endpoint.
   - Select Objects endpoints can have only one template per object type.
   - Event-based endpoints can have one template for each object type / event type combination.



2. In the **Output Template** area, click **Add configuration** to add one or more output templates.
3. Browse or search for the relevant objects.



Note that the selection of objects in this step defines the template to be used in terms of the object type identified for export (e.g., if exporting an [Article] or [Supplier Item], use this template), but does not necessarily define the objects that are included in the actual message, as the Format of the template may dictate that parent, child, or referenced objects are included. Additionally, the Advanced settings for Format determine if the selected objects, their children, or both should be considered for the export.

Important: If 'Select Objects' was selected as the data source in the Outbound Integration Endpoint wizard, the objects selected during the configuration of the output templates must correspond at root level to the objects selected in the wizard. Furthermore, if Collections were selected as a data source in the wizard, you must use Search to locate the relevant collection.

4. For event -based data sources, the basic system events (**Create**, **Modify**, and **Delete**) are displayed in the **Event Types** area, along with any configured derived events. For more information, see Outbound Integration Endpoint Derived Events. Select the event type that is relevant for the output template.
5. After adding the output configuration, specify the format of the output. For more information, see Outbound Integration Endpoint Format, and if necessary, Post-Processor Configuration.

Configuring Multiple Output Templates

Multiple output templates can be created to allow data output based on specific object types and, for event-based exports, a combination of specific object types and event types.

Important: Be aware that events are processed sequentially and are added to batches, as allowed, by the required output templates. Each batch also involves overhead processing operations. Depending on the data being exported, multiple output templates can result in many batches with relatively few events in each. To optimize processing, it is better to have a fewer large batches instead of many small ones.

Using multiple output templates prevents the same data from being exported across all chosen object types and event types. Multiple output templates also help to reduce data output, which increases performance both for STEP and for downstream systems. In addition, the use of multiple output templates should reduce the need for customizations.

For example, when a change is made to a 'Product Category' object, the export *should not* include child objects, but when a change is made to a 'Product Family' object, the export *should* include child products. In other words, multiple XML output templates can be configured and used, based on the object and event type.

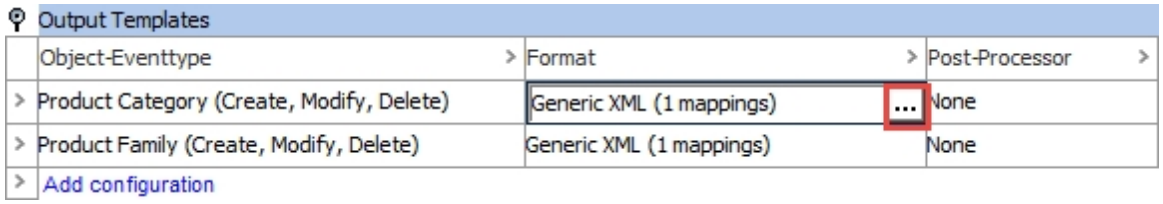
Limitations of Multiple Output Templates

- Output files are only delivered to one destination, regardless of the number of output templates.
- Output templates encompassing all object types are only available if all object types are chosen when making the configuration. This prevents excess output from the endpoints.
- Batching occurs per template type.
- Different output formats cannot be defined for the same object type. Common setup is to avoid combining different output formats (e.g. STEPXML and CSV) in the same endpoint.

Outbound Integration Endpoint Format

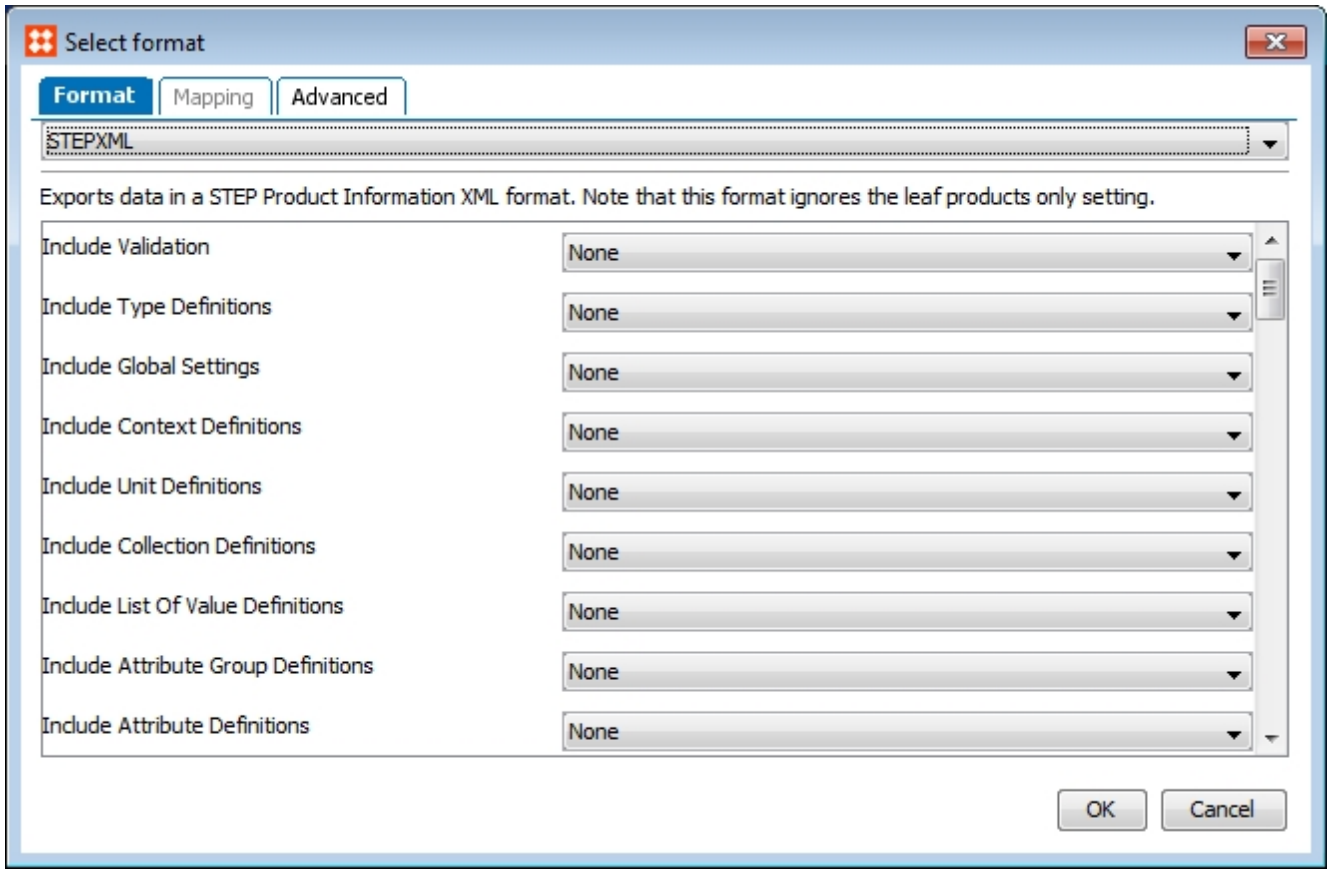
After completing the configuration steps in the Outbound Integration Endpoint wizard, the export format must be specified in the Outbound Integration Endpoint editor.

1. In **System Setup**, locate the relevant outbound integration endpoint.
2. Next to the relevant output template, in the format column, click the ellipsis button **(…)**.



3. In the **Select format** window, on the **Format** tab, select the format of the file to be exported.

   The available export options are the same as in the Export Manager. For more information about format options, see Select Format in the Export Manager documentation.

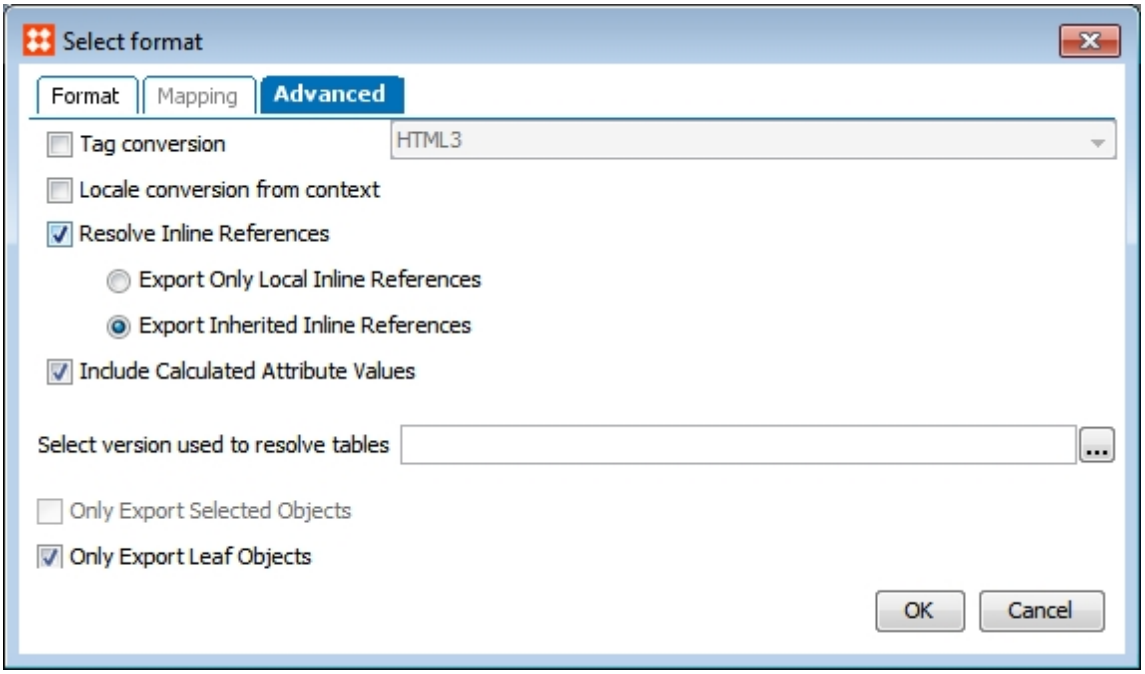


If Advanced STEPXML is selected, a number of different XML tags can be used to control the data that is exported. For more information, see Advanced STEPXML .

4. On the **Map Data** tab, specify how to map data to the export format.

   If the selected format is STEPXML or Advanced STEPXML, the **Mapping** tab is disabled because these formats are automatically mapped. Mappings for Generic XML are described in Generic XML Mappings.

5. On the **Advanced** tab, a number of advanced export options can be specified.



- **Tag conversion**: Converts tags to match the selected output format. This setting is optional.

  If this box is left unchecked, any formatting tags that are included in attribute values will not be converted in the outbound message. For example, if an attribute value contains bold text (which must be made bold with a STEP Style Tag, e.g. <bold>/</bold>), this tag will not be converted to its corresponding HTML output format (e.g. <B>/</B>) in the export. For more information on STEP Tags, refer to System Setup > Tags in the online help.

- **Locale conversion from context**: Converts numbers into the numeric format that corresponds to the selected locale. **Note**: If Smartsheets are used for format, this must remain unchecked.
- **Resolve Inline References**: Resolves inline references when they are exported. This box is checked by default, along with the option 'Export Inherited Inline References.'

  If unchecked, inline references will be exported with the inline reference tagging instead of the actual content pulled in by the inline reference. For example, if an attribute called 'Product Number' has an inline reference to pull in the STEP ID of the object (e.g. 12345), the attribute value will not contain 12345. Instead, it will contain tagging similar to the following:

```
<ref attrid="" equalsign="=" includeattrname="false" resolveto="objid" separator
=";"/>
```

This setting would be valid if the user planned to re-import the file at a later time and not overwrite the inline reference with a static value. For more information on inline references, see **Adding Inline References to Attribute Values** in the **System Setup / STEP Super User Guide**.

- **Include Calculated Attribute Values**: Resolves calculated attribute values when they are exported. If not selected, calculated attribute values are exported with empty values. If many complex calculated attribute values (traversing hierarchies and/or references)are used, consider if they should be exported, since they can impact performance. If required, consider scheduling for non-peak times. Simple calculations are not detrimental to an export, regardless of the quantity.
- **Select version used to resolve tables**: This setting is only available if the format is STEPXML and 'Include Tables' is set to Yes. Specifies which version of a given publication should be used to resolve tables. If a publication is not selected, the version last used in STEP workbench is used.
- **Only Export Selected Objects**: Specifies that only objects from the output template are exported. No children are exported. This setting is unchecked by default and is only available if 'Only Export Leaf Objects' is unchecked. This setting does not apply to STEPXML.
- **Only Export Leaf Objects**: Specifies that only the leaf objects (lowest level of the export object for both events and selected objects) of the selected top hierarchy are included in the export. Selected objects and triggering objects are only included if they have no children. This setting applies to the CSV or Excel format.

Note: Uncheck both ‘Only Export Selected Objects’ and ‘Only Export Leaf Objects’ if all objects (both for triggering events and object selection) and their children should be included in the export.

For more information, see Step 4 - Map Data and Step 5 - Advanced in the Export Manager Wizard documentation.

Advanced STEPXML

Selecting Advanced STEPXML displays a default template in the Select Format window. Advanced STEPXML tags determine how data is exported from STEP. Update or replace the default template with Advanced STEPXML tags to generate the desired output.

The most commonly required output can be generated using the following tags, XML attributes, and setup:

- Attribute Link Tag in STEPXML
- Attribute Value Filtering in STEPXML
- Classification Tag in STEPXML
- Deleting Product References in STEPXML
- Deleting Product, Classifications, and Assets in STEPXML
- Inheriting References in STEPXML
- Inheriting Values in STEPXML
- Product Filtering in STEPXML
- Products Tag in STEPXML
- Reference Filtering in STEPXML
- Referenced and Embedded XML Attributes in STEPXML
- Sequence Product Tag in STEPXML
- STEP Product Information Tag in STEPXML
- Unique Key Tag in STEPXML

Additionally, several STEPXML tags can be used within an Advanced STEPXML template. For more information about the specific tags available, see the **Tags Available for Advanced STEPXML Messages** section.

Attribute Link Tag in STEPXML

The **AttributeLink** tag outputs local attribute links for the products, product-overrides, or classifications in the export.

```
<AttributeLink\>
```

IncludeInherited

When modified with the **IncludeInherited** option, the **AttributeLink** tag also exports inherited attribute links.

```
<AttributeLink IncludeInherited="true">
```

The **AttributeLink** tag is allowed within the following tags:

- Product - Attribute links are inherited from parent products to top-level products.
- Product-Overrides - Attribute links are inherited from overridden product to product-override. Attribute links inherited from Classifications are not included.
- Classification - Attribute links are inherited from parent classifications to top-level classifications.

Template

No Inheritance	IncludeInherited
<?xml version="1.0" encoding="UTF-8"?> <STEP-ProductInformation> <Products> <Product> <Name/> <AttributeLink> <MetaData/> </AttributeLink> </Product> </Products> </STEP-ProductInformation>	<?xml version="1.0" encoding="UTF-8"?> <STEP-ProductInformation> <Products> <Product> <Name/> <AttributeLink **IncludeInherited="true"**> <MetaData/> </AttributeLink> </Product> </Products> </STEP-ProductInformation>

Results

Only local attribute links for the exported product are output.

No Inheritance

```
<STEP-ProductInformation ExportTime="2015-12-23 14:58:34" ExportC

  <Products>
    <Product ID="20878" UserTypeID="Level4" ParentID="20876">
      <Name Changed="true">Batteries  Level4</Name>
      <AttributeLink AttributeID="Voltage"/>

    </Product>
  </Products>
</STEP-ProductInformation>
```

Local and inherited attribute links for the exported product are output.

IncludeInherited

```
<STEP-ProductInformation ExportTime="2015-12-23 15:02:53" ExportCon

  <Products>
    <Product ID="20878" UserTypeID="Level4" ParentID="20876">
      <Name Changed="true">Batteries  Level4</Name>
      <AttributeLink AttributeID="Voltage"/>
      <AttributeLink AttributeID="MaximumVoltage" Inherited="3"/>
      <AttributeLink AttributeID="ETLListed" Inherited="3"/>
      <AttributeLink AttributeID="CSAListed" Inherited="3"/>
      <AttributeLink AttributeID="ULListed" Inherited="3"/>

        <MetaData>
          <Value AttributeID="DisplaySequence">2</Value>
        </MetaData>

    </Product>
  </Products>
</STEP-ProductInformation>
```

Attribute Value Filtering in STEPXML

Use the **Values** tag to output all the attribute values for a specific object.

To filter the attribute values, use the **AttributeID** XML attribute.

- When the output should include a specific attribute's value, include the attribute ID.
- When the output should include attributes within a particular attribute group, include the attribute ID of the attribute group.

For example, the following product message template includes the Brand attribute value and the ProductStatus Attributes group.

```
<?xml version="1.0" encoding="UTF-8"?>
<STEP-ProductInformation ExportDeletedData="true" ResolveInlineRefs="true" ExportDerivedAtt
rs="true">
<Products>
    <Product IncludeParent="true">
       <Name/>
       <Values>
 <Value AttributeID="Brand"/>  or  <Value AttributeID="ProductStatusAttributes"/>
       </Value>
       <AssetCrossReference Type="Main"/>
       <ProductCrossReference Type="Acccessory"/>
       <ClassificationReference Type="Website"/>
    </Product>
</Products>
</STEP-ProductInformation>
```

Classifications Tag in STEPXML

The Classifications tag can include an XML attribute. When an XML attribute is absent, the default setting is used. For example:

```
<Classifications IncludeParent="true">
```

The following XML attribute is available:

IncludeParent

- False - default setting
- True - when a classification is exported from STEP, the file includes the approved classification as well as its parents.

Additionally, several STEPXML tags can be used within an Advanced STEPXML template. For more information about the specific tags available, see the **STEPXML Tags Available Within Advanced STEPXML Tags** section.

Product to Classification links owned by the Classification

By default the product to classification links owned by the classification are exported in two places:

1. Under the Product as a reference to the classification.
2. Under the Classification as a reference to the product.

```
<STEP-ProductInformation>
<Classifications>
<Classification ID="Game Consoles" UserTypeID="Classification Folder" ParentID="Electronics" Referenced="true">
<ProductReference ProductID="XBox 360" Type="Classification Owned Product to Classification Link"/>
</Classification>
</Classifications>
<Products>
<Product ID="Xbox 360" UserTypeID="Sellable unit" ParentID="Microsoft Products">
<ClassificationReference ClassificationID="Game Consoles" Type="Classification Owned Product to Classification Link"/>
</Product>
</Products>
</STEP-ProductInformation>
```

The reason the links are exported under the products is historical, and it has not been changed due to backwards compatibility. However, logically they should be exported under the classification, as they are owned by the classification. And when the links are modified, they affect the revision of the classification - not the product. It is possible to stop the classification owned links from being exported under the products by using the recorder option 'IncludeClassificationOwned' on the 'Products' element. By setting this to "false" the classification links will not be exported under the products:

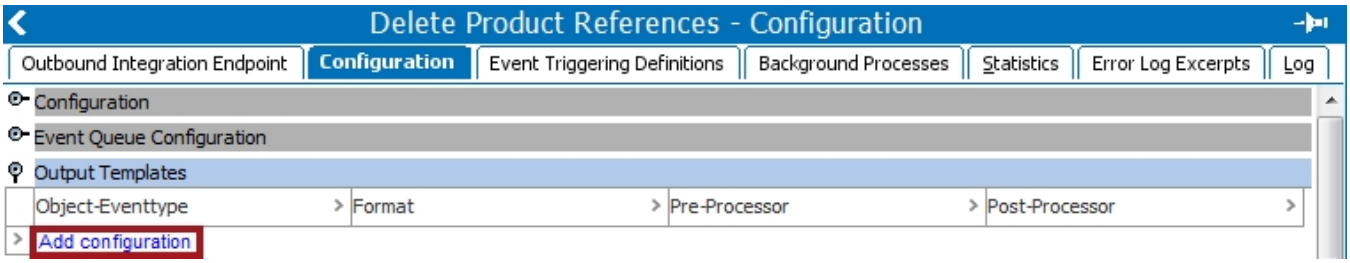
```
<STEP-ProductInformation>
<Products IncludeClassificationOwned="false">
<Product>
</Product>
</Products>
</STEP-ProductInformation>
```

Deleting Product References in STEPXML

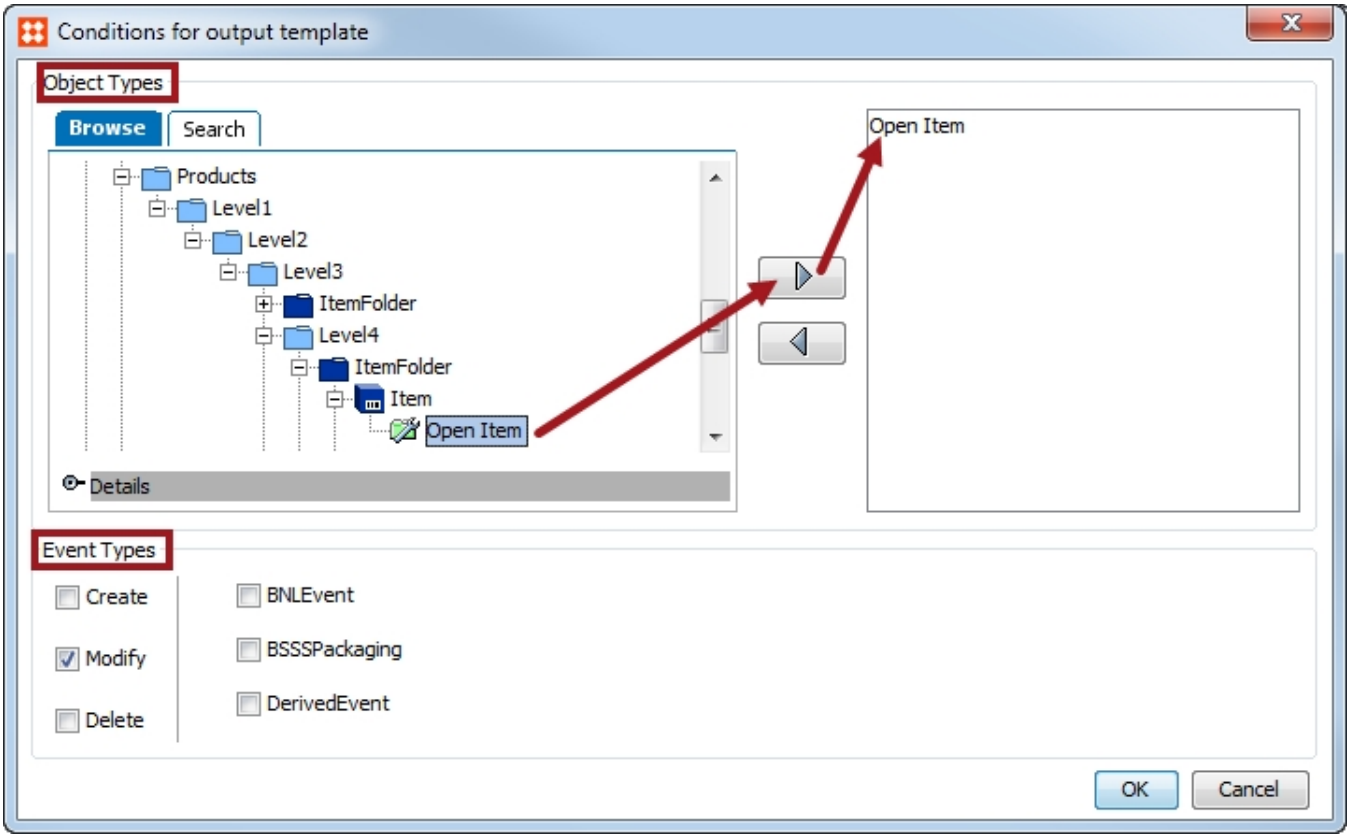
When an Outbound Integration Endpoint (OIEP) is configured to listen for Modify events, use the **STEP-ProductInformation** tag with the **ExportDeletedData** XML attribute to output information about deleted product references. After deleting the reference, approving the product triggers the OIEP.

Object Type and Event Type Selection

View the OIEP and open the Configuration tab. Under the Output Templates flipper, click the **Add configuration** link.



Set the Object Types and Event Types.

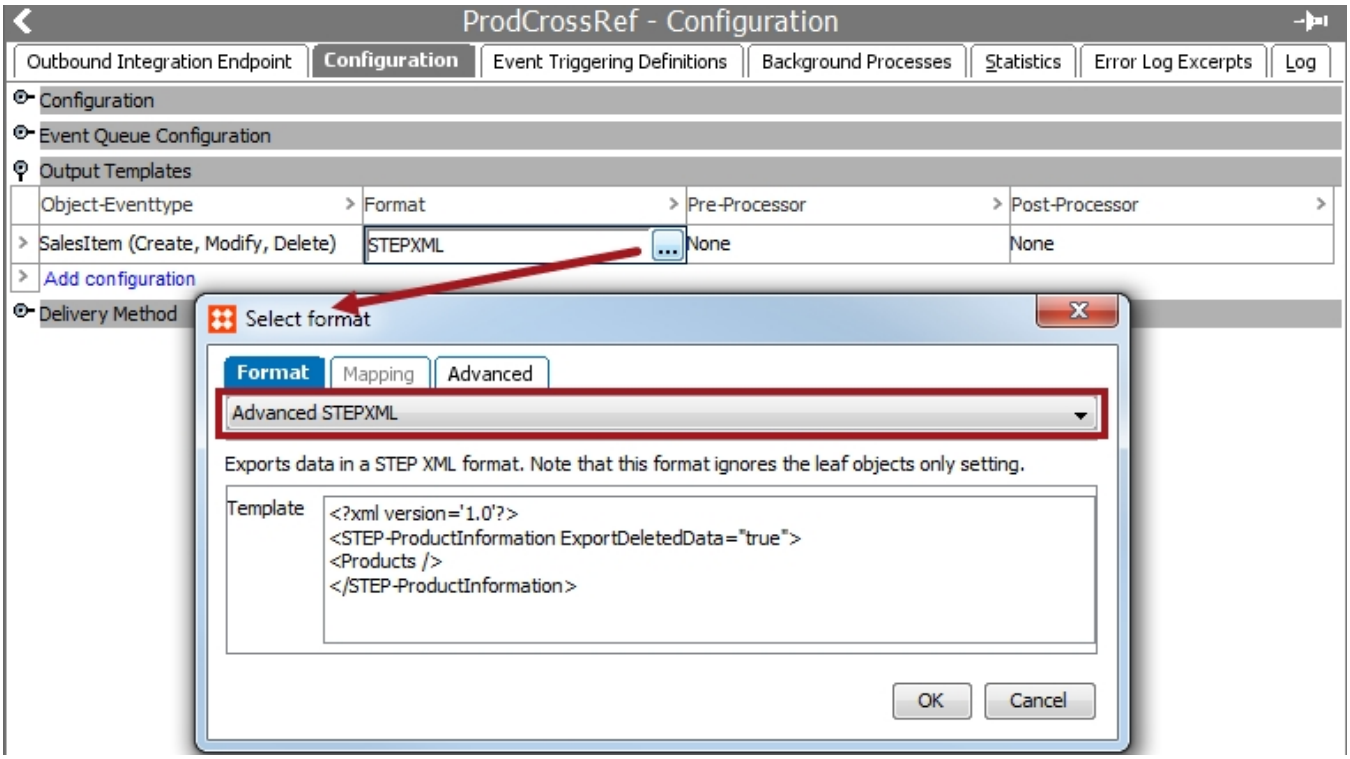


Template

In the Output Template, click the ellipsis button **(…)**, and set the format to **Advanced STEPXML**.

Provide the following text in the Template field.

```
<?xml version='1.0'?>
<STEP-ProductInformation ExportDeletedData="true">
<Products/>
</STEP-ProductInformation>
```



Result

When a Product Reference is deleted and the product is approved, the **DeleteProductCrossReference** tag indicates that the Product Reference is intended to be removed in the downstream system.

The previous settings generated the following Advanced STEPXML output document.

```
<STEP-ProductInformation ExportTime="2015-09-30 11:12:22" ExportContext="Context1" ContextID="Cont

  <Products>
    <Product ID="22163" UserTypeID="SalesItem" ParentID="22157">
      <Name>22163-01</Name>
      <ProductCrossReference ProductID="22155" Type="PrimaryDataSource"/>

      <AssetCrossReference AssetID="22215" Type="PrimaryProductImage"/>

      <Values>
        <Value AttributeID="AnnualSalesForecastMaximum">300</Value>
        <Value AttributeID="AnnualSalesForecast,Minimum">100</Value>
        <Value AttributeID="FeatureBullet3">26 slats of layer-glued birch adjust to your body weig
        <Value AttributeID="SellingPrice" UnitID="iso4217.unit.USD">1299</Value>
        <Value AttributeID="FeatureBullet1">Made of solid wood, which is a durable and warm natura
        <Value AttributeID="DescriptionTable">Comfy bed, expresso, Queen</Value>
        <Value AttributeID="FeatureBullet2">Adjustable bed sides allow you to use mattresses of di
        <Value AttributeID="DescriptionLong">Made of solid wood, which is a durable and warm natur
        <Value AttributeID="DescriptionWeb">Made of solid wood, which is a durable and warm natura
        <Value AttributeID="SalesItemShortDescription">Bed, Expresso, Comfy, Queen</Value>
        <Value AttributeID="SellingPriceUOM">EA</Value>
        <Value AttributeID="Parent" Derived="true">Beds Sales Items</Value>
        <Value AttributeID="Path" Derived="true">Products | Furniture | Bedroom | Beds | Beds Sale
        <Value AttributeID="Category" Derived="true">Primary Product Hierarchy | Products | Furnit
      </Values>
      <DeleteProductCrossReference ProductID="22168" Type="PrimarySupplierItem"/>
    </Product>
  </Products>
</STEP-ProductInformation>
```

Deleting Products, Classifications, and Assets in STEPXML

When an Outbound Integration Endpoint (OIEP) is configured to listen for Delete events, use the **DeleteProducts**, **DeleteClassifications**, or **DeleteAssets** tags to output information about deleted objects. After deleting the object, the OIEP is triggered by the Approve Deletion action within the Recycle Bin.

```
<DeleteProducts/>
<DeleteClassifications/>
<DeleteAssets/>
```

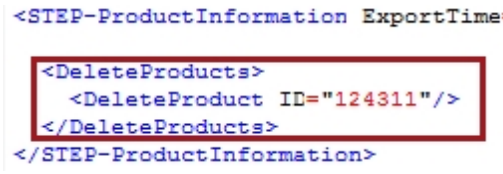
Note: The Force Delete option is not available for products, classifications, or assets when an OIEP is configured to deliver deletion information.

Template

```
<STEP-ProductInformation ExportDeletedData="true">
   <Products>
      <Product>
         <Name/>
      </Product>
   </Products>
<DeleteProducts/>
</STEP-ProductInformation>
```

Results

Once the deleted Product (Classification or Asset) displays in the Recycle Bin, use Approve Deletion to trigger the OIEP. The ID of the deleted object is included in the OIEP output.



Inheriting References in STEPXML

References within STEP are output against the object type where the references are defined. It is possible to inherit references to a specific product type when data is exported from STEP.

Inheriting references to all product types

To inherit references to all object types, use IncludeInherited.

For example:

```
<ClassificationReference IncludeInherited="true"/>
```

Inheriting references to a particular product type

To inherit references to a particular object type, the following tag must highlight the product object type that references should be inherited to.

For example:

```
<AssetCrossReference IncludeInheritedReferences="Level 3"/>
```

Template

```
<?xml version="1.0" encoding="UTF-8"?>
<STEP-ProductInformation ExportDeletedData="true" ResolveInlineRefs="true" ExportDerivedAttrs="true">
<Products>
    <Product IncludeParent="true">
       <Name/>
       <Values/>
       <Values IncludeInheritedAttributes="Level 3"/>
 <AssetCrossReference IncludeInheritedReferences="Level 3"/>  or <AssetCrossReference IncludeInherited="true"/>
       <ProductCrossReference Type="Acccessory"/>
       <ClassificationReference Type="Website"/>
    </Product>
</Products>
</STEP-ProductInformation>
```

Inheriting Values in STEPXML

Within the **Product** message, the **Values** tag outputs only local values. When no local value is available, even when an inherited value exists, no value is output.

```
<Values/>
```

The IncludeInherited and IncludeInheritedAttributes XML attributes allow inherited values to output to top-level objects or to specified object types.

IncludeInherited

Local values are included for all objects in the output. To avoid duplicate values throughout the file, inherited values are only included once, and display for the top-level object type being exported.

```
<Values IncludeInherited="true"/>
```

IncludeInheritedAttributes

Inherited values are output for each specified object type. Use multiple instances of this tag when duplicate values are required for inherited attributes.

```
<Values IncludeInheritedAttributes="ItemFolder"/>
<Values IncludeInheritedAttributes="Item"/>
```

Template

IncludeInherited	IncludeInheritedAttributes
<?xml version="1.0" encoding="UTF-8"?> <STEP-ProductInformation> <Products> <Product> <Name/> <Values **IncludeInherited="true"**/> </Product> </Products> </STEP-ProductInformation>	<?xml version="1.0" encoding="UTF-8"?> <STEP-ProductInformation> <Products> <Product> <Name/> <Values **IncludeInheritedAttributes="ItemFolder"**/> <Values **IncludeInheritedAttributes="Item"**/> </Product> </Products> </STEP-ProductInformation>

Results

Two (2) levels are exported from the product hierarchy: ItemFolder and Item.

- Local and inherited values are output for the top level (ItemFolder)
- Only local values are displayed for the lower level (Item)

IncludeInherited

```
<STEP-ProductInformation ExportTime="2015-12-23 13:52:56" ExportContext="Context1" ContextID="Context1" W

  <Products>
    <Product ID="123942" UserTypeID="ItemFolder" ParentID="20878">
      <Name Changed="true">Batteries Rechargeable ItemFolder</Name>
      <Values>
        <Value AttributeID="PrimaryColor">Green (entered on Object Type ItemFolder)</Value>
        <Value AttributeID="DescriptionLong">Long Description (entered on Object Type Level4)</Value>
      </Values>
      <Product ID="123963" UserTypeID="Item">
        <Name Changed="true">Recharge  C</Name>
        <Values>
          <Value AttributeID="ProductGroupDescription">Description (entered on Object Type Item)</Value>
        </Values>
      </Product>
    </Product>
  </Products>
</STEP-ProductInformation>
```

Three (3) levels are exported from the product hierarchy: Level4, ItemFolder, and Item.

- Only local values are output for the top level (Level4)
- Inherited values are displayed for both of the lower levels (ItemFolder and Item)

IncludeInheritedAttributes

```
<STEP-ProductInformation ExportTime="2015-12-23 13:24:52" ExportContext="Context1" ContextID="Context1" Wor

  <Products>
    <Product ID="20878" UserTypeID="Level4" ParentID="20876">
      <Name Changed="true">Batteries Level4</Name>
      <Values>
        <Value AttributeID="DescriptionLong">Long Description (entered on Object Type Level4)</Value>
      </Values>
      <Product ID="123942" UserTypeID="ItemFolder">
        <Name Changed="true">Batteries Rechargeable ItemFolder</Name>
        <Values>
          <Value AttributeID="PrimaryColor">Green (entered on Object Type ItemFolder)</Value>
          <Value AttributeID="DescriptionLong">Long Description (entered on Object Type Level4)</Value>
        </Values>
        <Product ID="123963" UserTypeID="Item">
          <Name Changed="true">Recharge  C</Name>
          <Values>
            <Value AttributeID="PrimaryColor">Green (entered on Object Type ItemFolder)</Value>
            <Value AttributeID="DescriptionLong">Long Description (entered on Object Type Level4)</Value>
            <Value AttributeID="ProductGroupDescription">Description (entered on Object Type Item)</Value>
          </Values>
        </Product>
      </Product>
    </Product>
  </Products>
</STEP-ProductInformation>
```

Product Filtering in STEPXML

When the output from STEP should include only specific objects, use the **FilterUserType** tag with the **ID** XML attribute. Only objects identified by the ID are exported.

```
<FilterUserType ID="ItemFolder"/>
```

Consider the scenario where a template delivers the parent structure for the approved product. Although six (6) product levels exist, only three (3) of the levels and their parents are required.

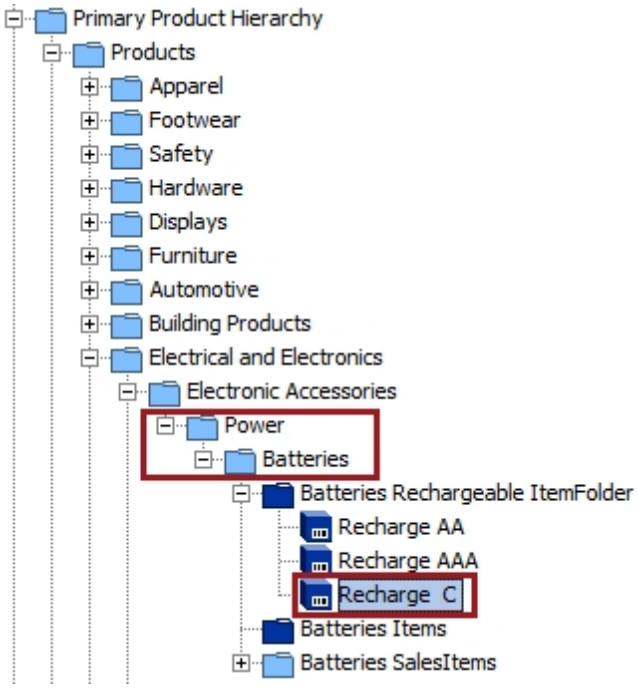
Use multiple instances of the **FilterUserType** tag to include all of the required objects and to filter out the objects that are not needed.

Template

Without Filtering	With Filtering
<?xml version="1.0" encoding="UTF-8"?> <STEP-ProductInformation> <Products> <Product IncludeParent="true"> <Name/> <ProductCrossReference/> </Product> </Products> </STEP-ProductInformation>	<?xml version="1.0" encoding="UTF-8"?> <STEP-ProductInformation> <Products> **<FilterUserType ID="Level3"/>** **<FilterUserType ID="Level4"/>** **<FilterUserType ID="Level5"/>** <Product IncludeParent="true"> <Name/> <ProductCrossReference/> </Product> </Products> </STEP-ProductInformation>

Modified Object

For each template, the same single item (Recharge C) is modified and approved, which could trigger an OIEP.


Primary Product Hierarchy
Products
Apparel
Footwear
Safety
Hardware
Displays
Furniture
Automotive
Building Products
Electrical and Electronics
Electronic Accessories
Power
Batteries
Batteries Rechargeable ItemFolder
Recharge AA
Recharge AAA
Recharge C
Batteries Items
Batteries SalesItems

Results

Without Filtering

```
<STEP-ProductInformation ExportTime="2015-12-17 15:13:24" ExportContext="Context1" ContextID="C

  <Products>
    <Product ID="Product hierarchy root" UserTypeID="Product user-type root" Selected="false">
      <Name>Primary Product Hierarchy</Name>
      <Product ID="ProductsRoot" UserTypeID="Products" Selected="false">
        <Name>Products</Name>
        <Product ID="8302" UserTypeID="Level1" Selected="false">
          <Name>Electrical and Electronics</Name>
          <Product ID="20875" UserTypeID="Level2" Selected="false">
            <Name>Electronic Accessories</Name>
            <Product ID="20876" UserTypeID="Level3" Selected="false">
              <Name>Power</Name>
              <Product ID="20878" UserTypeID="Level4" Selected="false">
                <Name>Batteries</Name>
                  <Product ID="123942" UserTypeID="ItemFolder" Selected="false">
                    <Name>Batteries Rechargable Items</Name>
                    <Product ID="123963" UserTypeID="Level5">
                      <Name Changed="true">Recharge  C</Name>
                        <ProductCrossReference ProductID="20882" Type="PrimaryDataSource"/>
                        <ProductCrossReference ProductID="123943" Type="CrossReference"/>
                        <ProductCrossReference ProductID="123944" Type="CrossReference"/>

                    </Product>
                  </Product>
                </Product>
              </Product>
            </Product>
          </Product>
        </Product>
      </Product>
    </Product>
  </Products>
</STEP-ProductInformation>
```

With Filtering

```
<STEP-ProductInformation ExportTime="2015-12-17 15:23:43" ExportContext="Context

  <Products>
    <Product ID="20876" UserTypeID="Level3" ParentID="20875" Selected="false">
      <Name>Power</Name>
      <Product ID="20878" UserTypeID="Level4" Selected="false">
        <Name>Batteries</Name>
        <Product ID="123963" UserTypeID="Level5" ParentID="123942">
          <Name Changed="true">Recharge C</Name>
            <ProductCrossReference ProductID="20882" Type="PrimaryDataSource"/>
            <ProductCrossReference ProductID="123943" Type="CrossReference"/>
            <ProductCrossReference ProductID="123944" Type="CrossReference"/>

        </Product>
      </Product>
    </Product>
  </Products>
</STEP-ProductInformation>
```

Products Tag in STEPXML

The Products tag can include multiple options. When an option is absent, the default setting is used.

```
<Products ExportSize="Minimum" IncludeParent="true">
```

The following XML attributes are available:

ExportSize

- Minimum - default setting. Other options are Selected and All.

For details about export size, see the **STEPXML Minimum, Referenced, and Selected** section.

IncludeParent

- False - default setting
- True - when a product is exported from STEP, the file includes the approved product as well as its parents.

Product to Classification links owned by the Classification

By default the product to classification links owned by the classification are exported in two places:

1. Under the Product as a reference to the classification.
2. Under the Classification as a reference to the product.

```
<STEP-ProductInformation>
<Classifications>
<Classification ID="Game Consoles" UserTypeID="Classification Folder" ParentID="Electronics" Referenced="true">
<ProductReference ProductID="XBox 360" Type="Classification Owned Product to Classification Link"/>
</Classification>
</Classifications>
<Products>
<Product ID="Xbox 360" UserTypeID="Sellable unit" ParentID="Microsoft Products">
<ClassificationReference ClassificationID="Game Consoles" Type="Classification Owned Product to Classification Link"/>
</Product>
</Products>
</STEP-ProductInformation>
```

The reason the links are exported under the products is historical, and it has not been changed due to backwards compatibility. However, logically they should be exported under the classification, as they are owned by the classification. And when the links are modified, they affect the revision of the classification - not the product. It is possible to stop the classification owned links from being exported under the products by using the recorder option 'IncludeClassificationOwned' on the 'Products' element. By setting this to "false" the classification links will not be exported under the products:

```
<STEP-ProductInformation>
<Products IncludeClassificationOwned="false">
<Product>
</Product>
</Products>
</STEP-ProductInformation>
```

Reference Filtering in STEPXML

When a downstream system requires a limited set of STEP references, use a reference tag with the **Type** XML attribute to indicate the required asset, classification, or product reference.

Use multiple instances of each reference tag to include all of the required objects.

Asset References

For example, to output asset references with an ID of 'PrimaryProductImage' and 'ProductImage', multiple **AssetCrossReference** tags are used with the **Type** XML attribute as follows:

```
<AssetCrossReference Type="PrimaryProductImage"/>
<AssetCrossReference Type="ProductImage"/>
```

Classification References

For example, to output only the classification reference with an ID of 'Supplier', the **ClassificationReference** tag is used with the **Type** XML attribute as follows:

```
<ClassificationReference Type="Supplier"/>
```

Product References

For example, to output only the product reference with an ID of 'CrossReference', the **ProductCrossReference** tag is used with the **Type** XML attribute as follows:

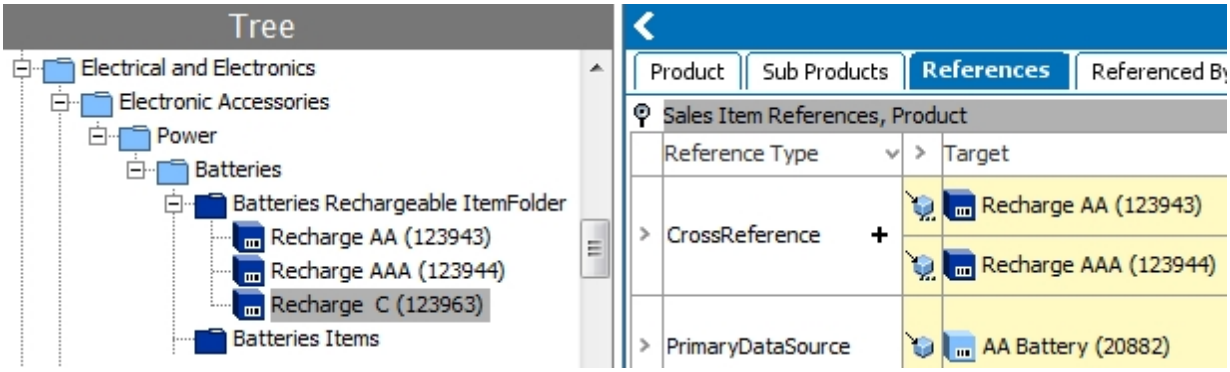
```
<ProductCrossReference Type="CrossReference"/>
```

Template

Without Filtering	With Filtering
<?xml version="1.0" encoding="UTF-8"?> <STEP-ProductInformation> <Products> <Product> <Name/> <ClassificationReference/> <ProductCrossReference/> </Product> </Products> </STEP-ProductInformation>	<?xml version="1.0" encoding="UTF-8"?> <STEP-ProductInformation> <Products> <Product> <Name/> <ClassificationReference> <ProductCrossReference **Type="CrossReference"**/> </Product> </Products> </STEP-ProductInformation>

Modified Object

For each template, the same single item (Recharge C) is modified and approved, which can trigger the OIEP.



Results

Without filtering, all references for the exported product are output.

Without Filtering

```
<STEP-ProductInformation ExportTime="2015-12-16 16:23:02" ExportContext="Cxt

  <Products>
    <Product ID="123963" UserTypeID="Item" ParentID="123942">
      <Name Changed="true">Recharge  C</Name>
      <ClassificationReference ClassificationID="SupplyAll" Type="Supplier">
        <MetaData>
  <Value AttributeID="QtyOfItems" Derived="true">1</Value>
</MetaData>
      </ClassificationReference>

      <ProductCrossReference ProductID="20882" Type="PrimaryDataSource"/>
      <ProductCrossReference ProductID="123943" Type="CrossReference"/>
      <ProductCrossReference ProductID="123944" Type="CrossReference"/>

    </Product>
  </Products>
</STEP-ProductInformation>
```

With filtering, only references of the specified type for the exported product are output.

With Filtering

```
<STEP-ProductInformation ExportTime="2015-12-16 16:23:02" ExportContext="Cxt

  <Products>
    <Product ID="123963" UserTypeID="Item" ParentID="123942">
      <Name Changed="true">Recharge  C</Name>
      <ClassificationReference ClassificationID="SupplyAll" Type="Supplier">
        <MetaData>
  <Value AttributeID="QtyOfItems" Derived="true">1</Value>
</MetaData>
      </ClassificationReference>

      <ProductCrossReference ProductID="123943" Type="CrossReference"/>
      <ProductCrossReference ProductID="123944" Type="CrossReference"/>

    </Product>
  </Products>
</STEP-ProductInformation>
```

Referenced and Embedded XML Attributes in STEPXML

The ID and Type of an object's references can be exported using the **Referenced** XML attribute.

```
<Product Referenced="true">
```

In order to export details about the reference itself, use the **Embedded** XML attribute.

```
<Entity Referenced="true" Embedded="true">
```

When required by a downstream system, the **Referenced** and **Embedded** XML attributes are used together to provide additional details about the reference in line with the product, classification, or asset that owns the reference.

For example, if STEP allows a product-to-product cross reference between products A and B, then the data for both products A and B can be delivered via the product reference. The cross referenced product details are exported within the product that references it.

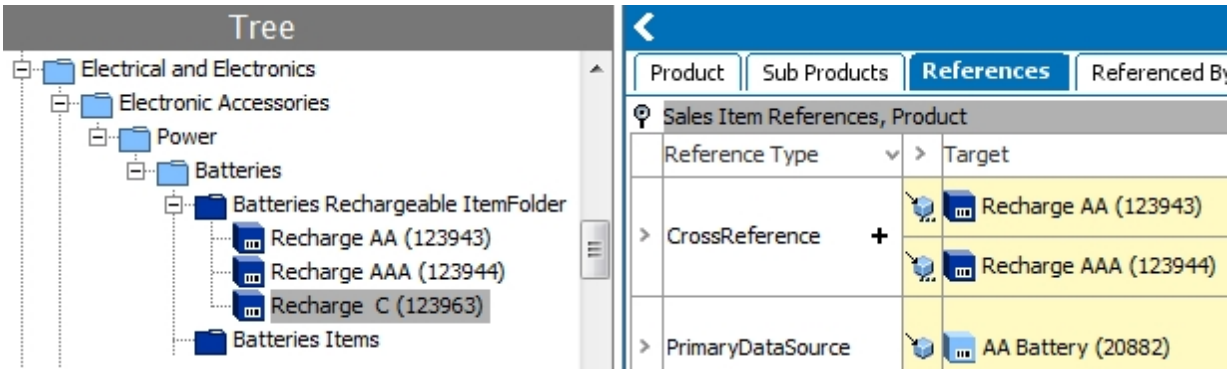
Template

The MetaData tag is used to output attribute values maintained on the product reference itself.

Referenced Without Embedded	Referenced With Embedded
<?xml version="1.0" encoding="UTF-8"?> > <STEP-ProductInformation ResolveInlineRefs="true"> <Products> <FilterUserType ID="Item"/> <Product IncludeParent="true"> <Name/> <ProductCrossReference Type="CrossReference"> <MetaData/> <Product Referenced="true"> <Name/> <Values/> </Product> </ProductCrossReference> </Product> </Products> </STEP-ProductInformation>	<?xml version="1.0" encoding="UTF-8"?> > <STEP-ProductInformation ResolveInlineRefs="true"> <Products> <FilterUserType ID="Item"/> <Product IncludeParent="true"> <Name/> <ProductCrossReference Type="CrossReference"> <MetaData/> <Product Referenced="true" **Embedded="true"**> <Name/> <Values/> </Product> </ProductCrossReference> </Product> </Products> </STEP-ProductInformation>

Modified Object

For each template, the same single item (Recharge C) is modified and approved, which triggers the OIEP.



Results

Referenced Without Embedded

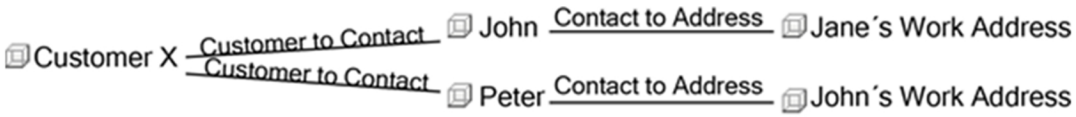
```
<STEP-ProductInformation ExportTime="2015-12-21 15:14:13" ExportContext="Context1" ContextID="Co

  <Products>
    <Product ID="123944" UserTypeID="Item" ParentID="123942" Selected="false" Referenced="true">
      <Name>Recharge AAA</Name>
      <Values>
        <Value AttributeID="SupplierPartNumber">65165165</Value>
        <Value AttributeID="Path" Derived="true">Electrical and Electronics | Electronic Accesso
        <Value AttributeID="WWW1" Derived="true">http://www.stibosystems.com</Value>
      </Values>
    </Product>
    <Product ID="123943" UserTypeID="Item" ParentID="123942" Selected="false" Referenced="true">
      <Name>Recharge AA</Name>
      <Values>
        <Value AttributeID="SupplierPartNumber">32153563</Value>
        <Value AttributeID="Path" Derived="true">Electrical and Electronics | Electronic Accesso
        <Value AttributeID="WWW1" Derived="true">http://www.stibosystems.com</Value>
      </Values>
    </Product>
    <Product ID="123963" UserTypeID="Item" ParentID="123942">
      <Name Changed="true">Recharge C</Name>
        <ProductCrossReference ProductID="123943" Type="CrossReference"/>
        <ProductCrossReference ProductID="123944" Type="CrossReference"/>

    </Product>
  </Products>
</STEP-ProductInformation>
```

Referenced With Embedded

```
<STEP-ProductInformation ExportTime="2015-12-21 15:11:51" ExportContext="Context1" ContextID="Co

  <Products>
    <Product ID="123944" UserTypeID="Item" ParentID="123942" Selected="false" Referenced="true">
      <Name>Recharge AAA</Name>
      <Values>
        <Value AttributeID="SupplierPartNumber">65165165</Value>
        <Value AttributeID="Path" Derived="true">Electrical and Electronics | Electronic Accesso
        <Value AttributeID="WWW1" Derived="true">http://www.stibosystems.com</Value>
      </Values>
    </Product>
    <Product ID="123943" UserTypeID="Item" ParentID="123942" Selected="false" Referenced="true">
      <Name>Recharge AA</Name>
      <Values>
        <Value AttributeID="SupplierPartNumber">32153563</Value>
        <Value AttributeID="Path" Derived="true">Electrical and Electronics | Electronic Accesso
        <Value AttributeID="WWW1" Derived="true">http://www.stibosystems.com</Value>
      </Values>
    </Product>
    <Product ID="123963" UserTypeID="Item" ParentID="123942">
      <Name Changed="true">Recharge  C</Name>
        <ProductCrossReference ProductID="123943" Type="CrossReference">
<Product ID="123943" UserTypeID="Item" ParentID="123942">
      <Name>Recharge AA</Name>
      <Values>
        <Value AttributeID="SupplierPartNumber">32153563</Value>
        <Value AttributeID="Path" Derived="true">Electrical and Electronics | Electronic Accesso
        <Value AttributeID="WWW1" Derived="true">http://www.stibosystems.com</Value>
      </Values>
    </Product></ProductCrossReference>
        <ProductCrossReference ProductID="123944" Type="CrossReference">
<Product ID="123944" UserTypeID="Item" ParentID="123942">
      <Name>Recharge AAA</Name>
      <Values>
        <Value AttributeID="SupplierPartNumber">65165165</Value>
        <Value AttributeID="Path" Derived="true">Electrical and Electronics | Electronic Accesso
        <Value AttributeID="WWW1" Derived="true">http://www.stibosystems.com</Value>
      </Values>
    </Product></ProductCrossReference>

    </Product>
  </Products>
</STEP-ProductInformation>
```

Additionally, several STEPXML tags can be used within an Advanced STEPXML template. For more information about the specific tags available, see the **STEPXML Tags Available Within Advanced STEPXML Tags** section.

As another example, the following Advanced STEPXML template is defined to export a customer entity hierarchy. Each customer has references to contacts, and each contact has one or more references to addresses. All objects are in this example are modeled with entity objects.



The following is a general Advanced STEPXML template that exports all used entity objects below the customer hierarchy.

```
<?xml version='1.0'?>
<STEP-ProductInformation>
<Entities>
   <Entity>
      <EntityCrossReference Type="Customer to Contact">
         <Entity Referenced="true">
            <Name />
            <EntityCrossReference Type="Contact to Address">
               <Entity Referenced="true">
                  <Name />
               </Entity>
            </EntityCrossReference>
         </Entity>
      </EntityCrossReference>
   </Entity>
</Entities>
</STEP-ProductInformation>
```

The template exports customer entity objects and their references of the type **Customer to Contact** and **Contact to Address**. However, the referenced objects are not exported as embedded in the customer hierarchy. To export the referenced contacts and their address objects embedded in the customer hierarchy, you have to insert <Entity Referenced="true" Embedded="true"> in the advanced template as illustrated in the following.

```
<?xml version='1.0'?>
<STEP-ProductInformation>
<Entities>
   <Entity>
      <EntityCrossReference Type="Customer to Contact">
         <Entity Referenced="true" Embedded="true">
            <Name />
            <EntityCrossReference Type="Contact to Address">
               <Entity Referenced="true" Embedded="true">
                  <Name />
               </Entity>
            </EntityCrossReference>
         </Entity>
```

```
      </EntityCrossReference>
   </Entity>
</Entities>
</STEP-ProductInformation>
```

Sequence Product Tag in STEPXML

The sequence number is the order of the products in STEP. When used with the **IncludeParent** tag, the **SequenceProduct** tag exports the sequence numbers of all exported products.

```
<SequenceProduct/>
```

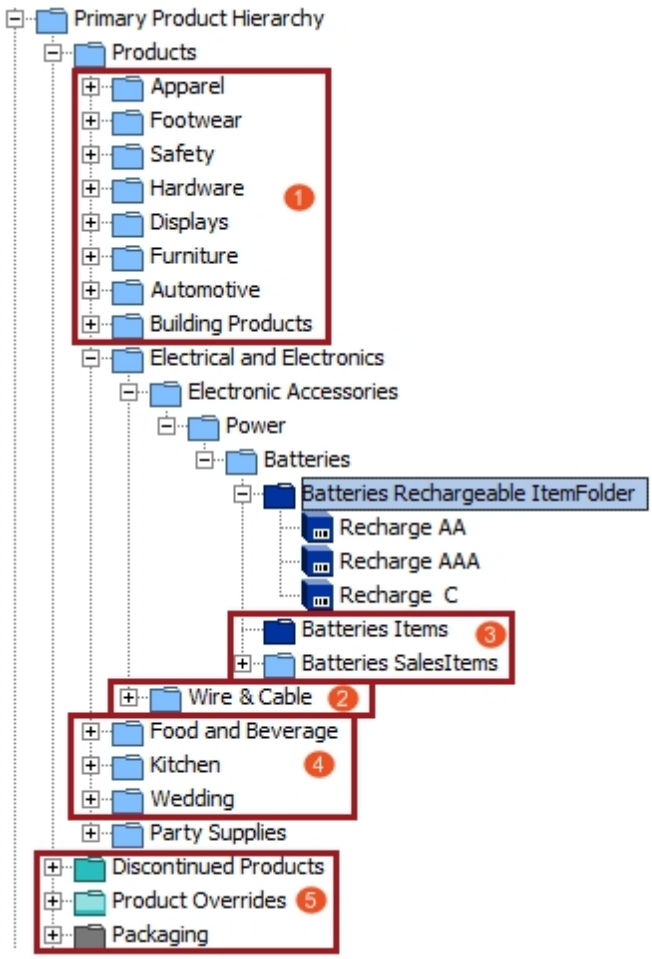
Classifications are always exported in the sequence they are displayed in the workbench.

Template

With SequenceProduct	Without SequenceProduct
<?xml version="1.0" encoding="UTF-8"?> <STEP-ProductInformation> <Products> <Product IncludeParent="true"> <SequenceProduct/> <Name/> </Product> </Products> </STEP-ProductInformation>	<?xml version="1.0" encoding="UTF-8"?> <STEP-ProductInformation> <Products> <Product IncludeParent="true"> <Name/> </Product> </Products> </STEP-ProductInformation>

Modified Object

For each template, the same single item (Batteries Rechargeable ItemFolder) is modified and approved, which triggers the OIEP.



Results

All parents for the exported product are output.

Sequence numbers for all nodes under each of the parents are output.

Sequence numbers for nodes equal to the changed object are output.

With SequenceProduct

```
<STEP-ProductInformation ExportTime="2016-01-04 10:04:58" ExportContext="Context1" ContextID="

  <Products>
    <Product ID="Product hierarchy root" UserTypeID="Product user-type root" Selected="false">
      <Name>Primary Product Hierarchy</Name>
      <Product ID="ProductsRoot" UserTypeID="Products" Selected="false">
        <Name>Products</Name>
        <SequenceProduct ID="18200"/>
        <SequenceProduct ID="20688"/>
        <SequenceProduct ID="20718"/>
        <SequenceProduct ID="20864"/>
        <SequenceProduct ID="22005"/>  1
        <SequenceProduct ID="22150"/>
        <SequenceProduct ID="8298"/>
        <SequenceProduct ID="8299"/>
        <Product ID="8302" UserTypeID="Level1" Selected="false">
          <Name>Electrical and Electronics</Name>
          <SequenceProduct ID="8313"/> 2
          <Product ID="20875" UserTypeID="Level2" Selected="false">
            <Name>Electronic Accessories</Name>
            <Product ID="20876" UserTypeID="Level3" Selected="false">
              <Name>Power</Name>
              <Product ID="20878" UserTypeID="Level4" Selected="false">
                <Name>Batteries</Name>
                <SequenceProduct ID="20879"/>
                <SequenceProduct ID="20880"/> 3
                <Product ID="123942" UserTypeID="ItemFolder">
                  <Name Changed="true">Batteries Rechargeable  ItemFolder</Name>
                </Product>
              </Product>
            </Product>
          </Product>
        </Product>
        <SequenceProduct ID="8303"/>
        <SequenceProduct ID="8304"/> 4
        <SequenceProduct ID="109946"/>
      </Product>
      <SequenceProduct ID="DiscontinuedProductsRoot"/>
      <SequenceProduct ID="ProductOverridesRoot"/>  5
      <SequenceProduct ID="PackagingRoot"/>
    </Product>
  </Products>
</STEP-ProductInformation>
```

All parents for the exported product are output.

Without SequenceProduct

```
<STEP-ProductInformation ExportTime="2016-01-04 10:13:31" ExportContext="Context1" ContextID="(

  <Products>
    <Product ID="Product hierarchy root" UserTypeID="Product user-type root" Selected="false">
      <Name>Primary Product Hierarchy</Name>
      <Product ID="ProductsRoot" UserTypeID="Products" Selected="false">
        <Name>Products</Name>
        <Product ID="8302" UserTypeID="Level1" Selected="false">
          <Name>Electrical and Electronics</Name>
          <Product ID="20875" UserTypeID="Level2" Selected="false">
            <Name>Electronic Accessories</Name>
            <Product ID="20876" UserTypeID="Level3" Selected="false">
              <Name>Power</Name>
              <Product ID="20878" UserTypeID="Level4" Selected="false">
                <Name>Batteries</Name>
                <Product ID="123942" UserTypeID="ItemFolder">
                  <Name Changed="true">Batteries Rechargeable ItemFolder</Name>
                </Product>
              </Product>
            </Product>
          </Product>
        </Product>
      </Product>
    </Product>
  </Products>
</STEP-ProductInformation>
```

STEP Product Information Tag in STEPXML

The **STEP-ProductInformation** tag determines the data exported for Products. It allows multiple XML Attributes (described below) to be listed on a single line, each separated by a single space. XML Attributes that are absent use the default setting.

```
<STEP-ProductInformation ExportDeletedData="true" ResolveInlineRefs="true"
FollowOverrideSubProducts="true" ExportDerivedAttrs="true" Validation="XSD"
DefinitionsAsComments="true">
```

DefinitionsAsComments

- False - default setting
- True - configuration definitions for Business Rules, Web UIs, IEPs, or Workflows are exported as comments.

For more details, see the **Export Configuration Definitions as Comments** section of the **Configuration Management** documentation

ExportDeletedData

- False - default setting
- True - change markers indicate if data has been added to or removed from a product. This includes attribute values, product references and classification references.

ExportDerivedAttrs

- False - default setting
- True - calculated attribute values are derived when an object is output from STEP.

FollowOverrideSubProducts

- False - default setting
- True - all Sub-Products of Product Overrides are included in the output XML.

ResolveInlineRefs

- False - default setting
- True - all inline references are resolved within the attributes values that are exported.

Validation

- None - default. Other options are XSD and DTD.

Tags Available for Advanced STEPXML Messages

You can use the following tags within a product, classification, or asset Advanced STEPXML message to output specific object data.

Tag	Output	Valid Parent tag
`<Name/>`	object name	• Product • Classification • Asset
`<Values/>`	attribute values	• Product • Asset
`<AssetCrossReference/>`	all asset cross references	• Product
`<ProductCrossReference/>`	all product cross references	• Product
`<ClassificationReference/>`	all classification cross references	• Product • Classification
`<MetaData/>`	metadata	• Classification • Asset

For more information about cross reference output, see the **Filtering References** section.

For more information about values output, see the **Inheriting Values** section.

Unique Key Tag in STEPXML

A unique key is a database item that represents a specific product in the STEP system. The item is composed of (transformed) product data. A key is a definition of how unique keys are composed. It specifies which object data is used and how it should be combined and transformed. The combination and transformation is done using the function framework (key formula). There can be exactly one unique key for each object for each defined key. No two objects can have the same unique key for an active key.

Unique keys are especially used when STEPXML is delivered to external systems like ERP systems , websites, and so on. These systems do not know the STEP ID of the exported products and therefore need a specific unique ID. For example, a SAP ID must be a 7-digit number.

In the following example, an advanced template is defined to export unique key IDs based on the unique key 'SAP_ID'.

To export a key (with ID SAP_ID) through the Export Manager or the Event Messenger, you must include the XML tag <IncludeKey KeyID="SAP_ ID"/> after the XML tag <STEP-ProductInformation ….>. For example:

```
<STEP-ProductInformation ExportDerivedAttrs="false" FollowOverrideSubProducts="true">
<IncludeKey KeyID="SAP_ID"/>
<Products ExportSize="Minimum">
<Product>
   <Name/>
   <AttributeLink/>
   <ClassificationReference/>
   <ProductCrossReference/>
   <AssetCrossReference/>
   <Values/>
</Product>
</Products>
</STEP-ProductInformation>
```

This instructs the exporter to also export keys of the corresponding IDs:

```
<STEP-ProductInformation
   ...
   <DeleteProduct ID="77">
      <KeyValue KeyID="SAP_ID">SAP_77</KeyValue>
      ...
   <Product ID="77" ...>
      <KeyValue KeyID="SAP_ID">SAP_77</KeyValue>
      ...
      <ProductCrossReference ProductID="77" ...>
         <KeyValue KeyID="SAP_ID">SAP_77</KeyValue>
         ...
```

```
<Entity ID="77" ...>
   <KeyValue KeyID="SAP_ID">SAP_77</KeyValue>
   ...
...
```

Generic XML

Generic XML is the XML-based language used to specify how data is extracted from or written to an XML document during import or export. The Generic XML mapping step requires selection of the kind of data to be mapped. Various aspects are available depending on the selected data.

See section Generic XML of the Exporting Data and Images / STEP Export Manager User Guide documentation for more information.

Exports Unique to OIEP

OIEP can trigger an export file upon deletion and approval of the following data:

- Assets
- Classifications
- Entities
- Product References

OIEP can also generate an export file for Units.

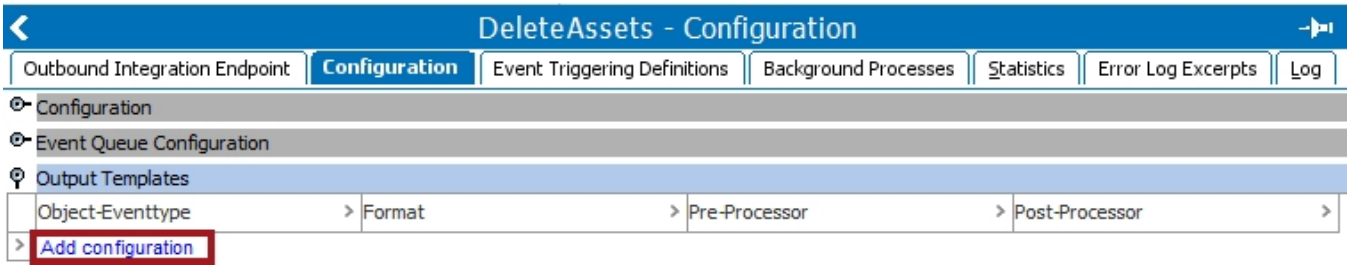
- Units

Deleting Assets with Generic XML

When assets are deleted and approved in STEP, Outbound Integration Endpoints (OIEPs) can pass deletion events when the OIEP is configured to listen for Delete events on the relevant objects. As a result, the downstream system is alerted to the deleted asset.

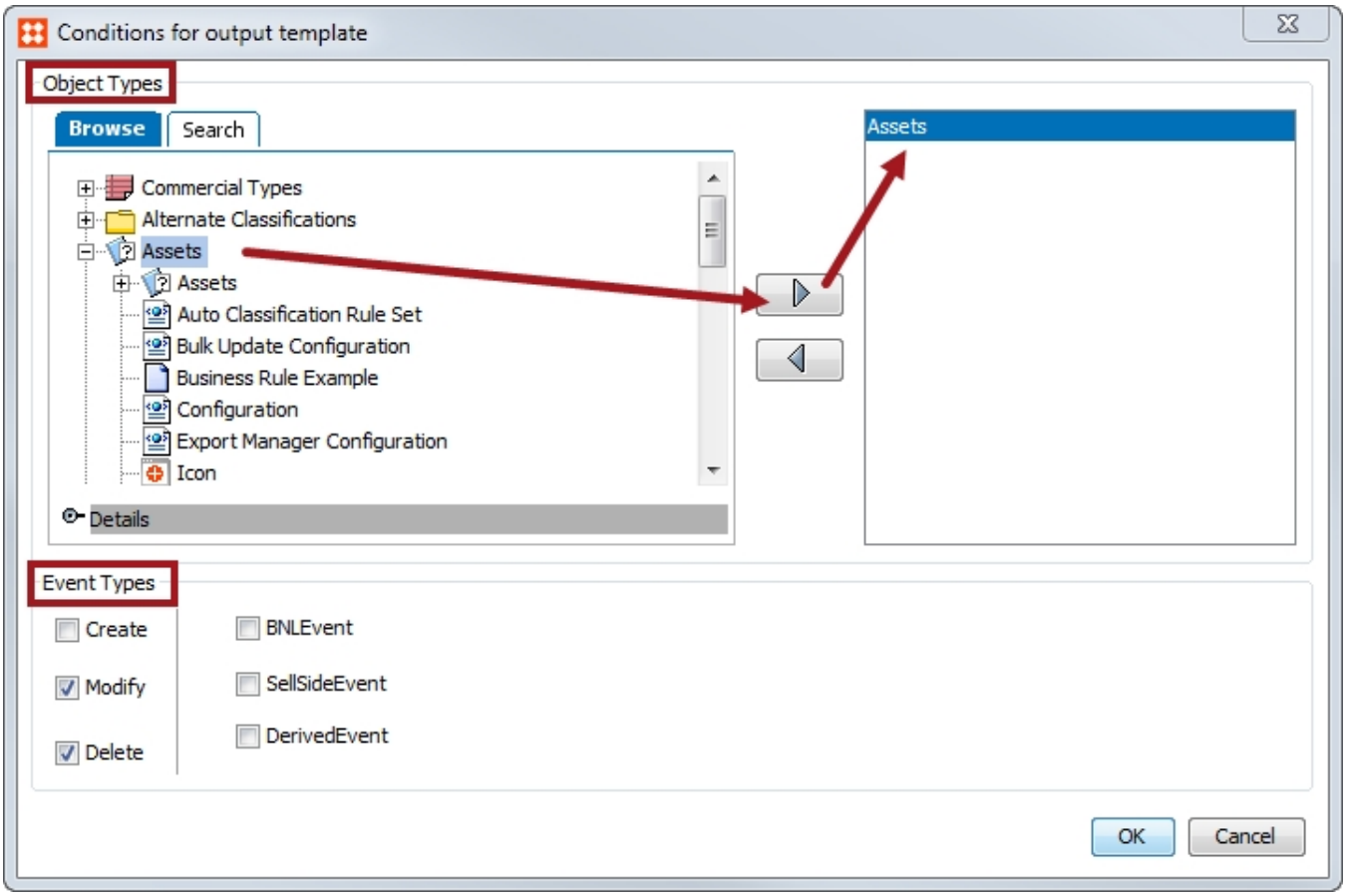
Object Type and Event Type Selection

On the OIEP, open the Configuration tab. Under the Output Templates flipper, click the **Add configuration** link.



Set the Object Types and Event Types.

For this example, Modify events are also included to demonstrate the available functionality.



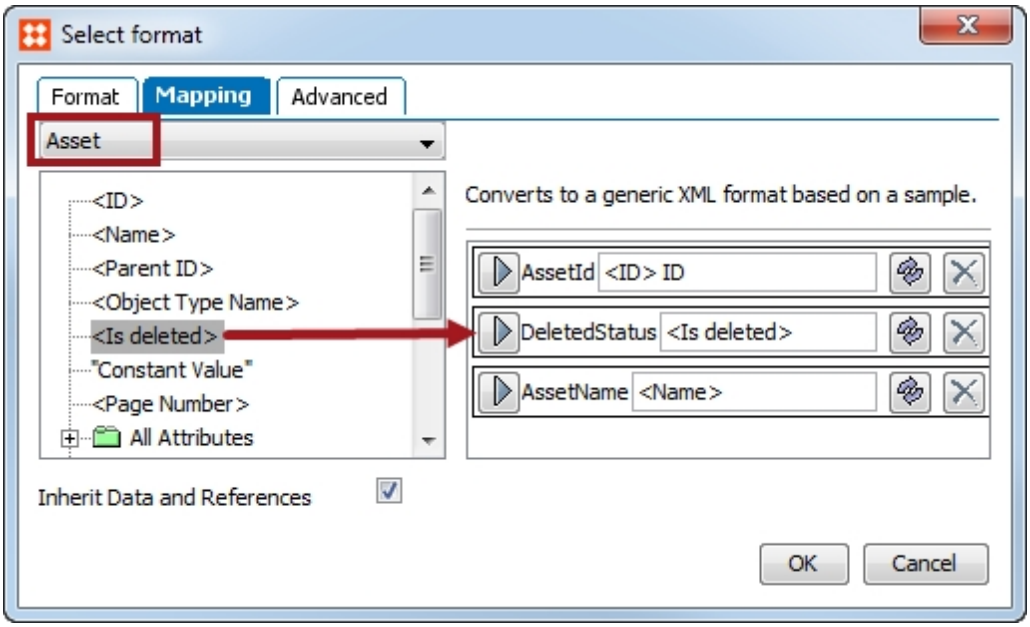
Template

In the Output Template, click the ellipsis button **(…)**, and set the format to **Generic XML**.

Provide the following text in the Sample field.

```
<root>
   <Assets>
      <?Record?>
      <Asset AssetID="[?Target AssetId?]" DeletedStatus="[?Target DeletedStatus?]">
      <AssetName> <?Target?> </AssetName>
      </Asset>
   </Assets>
</root>
```

Mapping



Results

Selecting **Approve Deletion** for an Asset in the Recycle Bin and/or **Approve Object** for modification of an Asset on the Tree triggers the OIEP.

Invoking the OIEP generates the following output:

```
<root>
  <Assets>
    <Asset AssetID="MSDS_00020" DeletedStatus="IsDeleted">
      <AssetName/>
    </Asset>
  </Assets>
  <Assets>
    <Asset AssetID="MSDS_00019" DeletedStatus="">
      <AssetName>MSDS_00019 modified</AssetName>
    </Asset>
  </Assets>
</root>
```

For deleted Assets, the Deleted Status displays **"IsDeleted"**. Only ID is retained for export on Delete events.

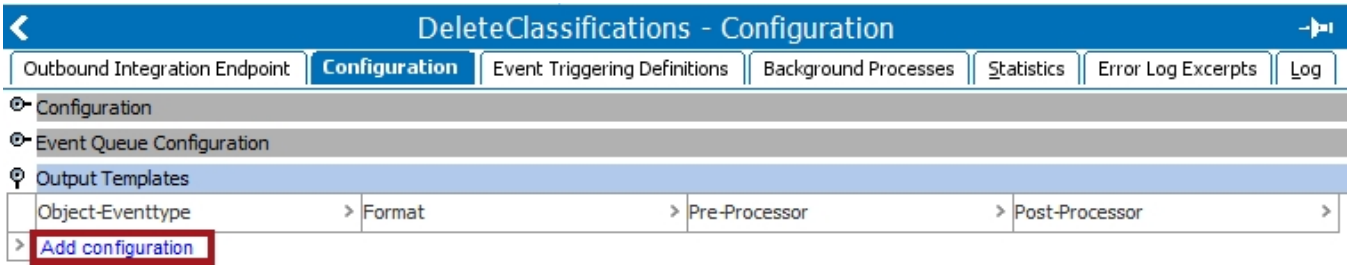
For modified Assets, the Deleted Status displays **" "**, which indicates the asset is not deleted. All mapped fields are exported for Create and Modify events.

Deleting Classifications with Generic XML

When Classifications are deleted and approved in STEP, Outbound Integration Endpoints (OIEPs) can pass deletion events when the OIEP is configured to listen for Delete events on the relevant objects. As a result, the downstream system is alerted to the deleted classification.

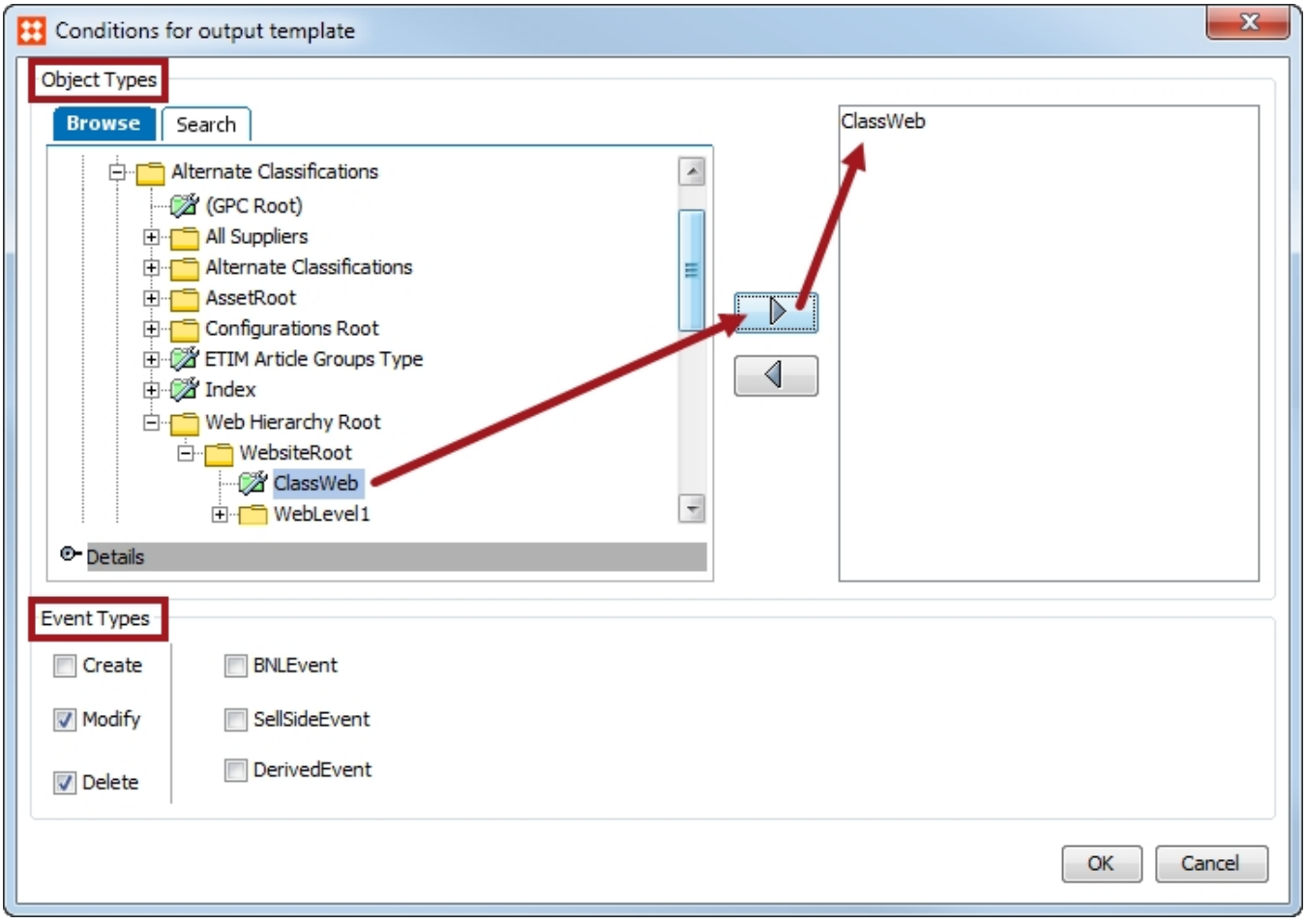
Object Type and Event Type Selection

On the OIEP, open the Configuration tab. Under the Output Templates flipper, click the **Add configuration** link.



Set the Object Types and Event Types.

For this example, Modify events are also included to demonstrate the available functionality.



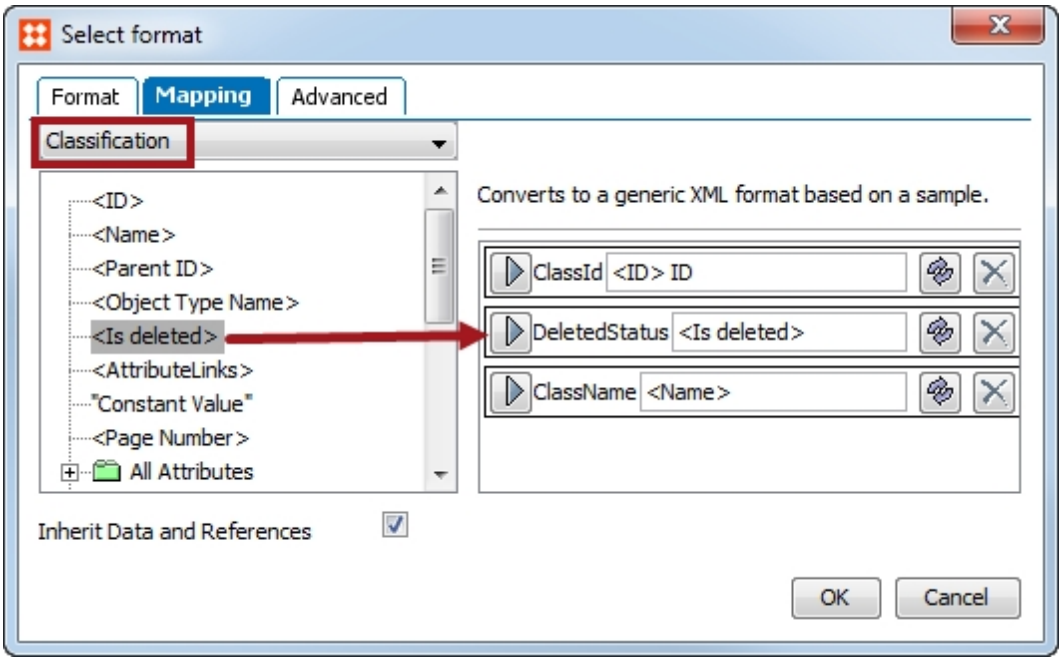
Template

In the Format field, click the ellipsis button **(…)**, and set the format to **Generic XML**.

Provide the following text in the Sample field.

```
<root>
   <Classifications>
      <?Record?>
      <Class ClassID="[?Target ClassId?]" DeletedStatus="[?Target DeletedStatus?]">
      <ClassName> <?Target?> </ClassName>
      </Class>
   </Classifications>
</root>
```

Mapping



Result

Approve Deletion for a Classification in the Recycle Bin and/or **Approve Object** for modification of a Classification on the Tree triggers the OIEP.

Invoking the OIEP generates the following output:

```
<root>
  <Classifications>
    <Class ClassID="SofasAndChairs" DeletedStatus="IsDeleted">
      <ClassName/>
    </Class>
  </Classifications>
  <Classifications>
    <Class ClassID="TablesAndChairs" DeletedStatus="">
      <ClassName>Tables And Chairs modified</ClassName>
    </Class>
  </Classifications>
</root>
```

For deleted Classifications, the Deleted Status displays **"IsDeleted"**. Only ID is retained for export on Delete events.

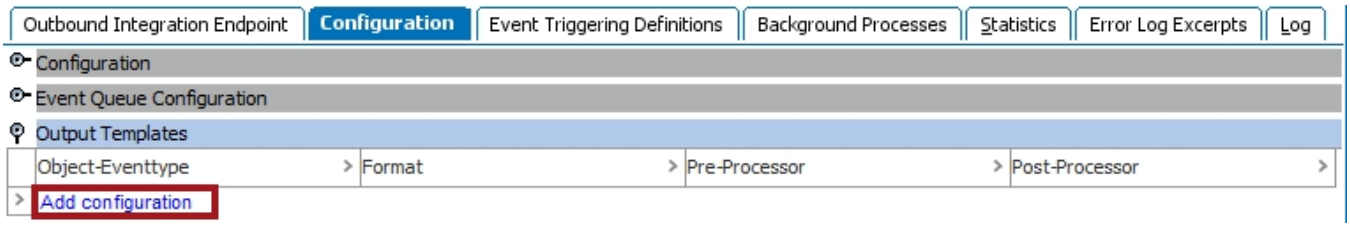
For modified Classifications, the Deleted Status displays **" "**, which indicates the classification is not deleted. All mapped fields are exported for Create and Modify events.

Deleting Entities with Generic XML

When Entities are deleted and approved in STEP, Outbound Integration Endpoints (OIEPs) can pass deletion events when the OIEP is configured to listen for Delete events on the relevant objects. As a result, the downstream system is alerted to the deleted asset. Entities must have the Revisability field set to Workspace Revisable, not Global Revisable.

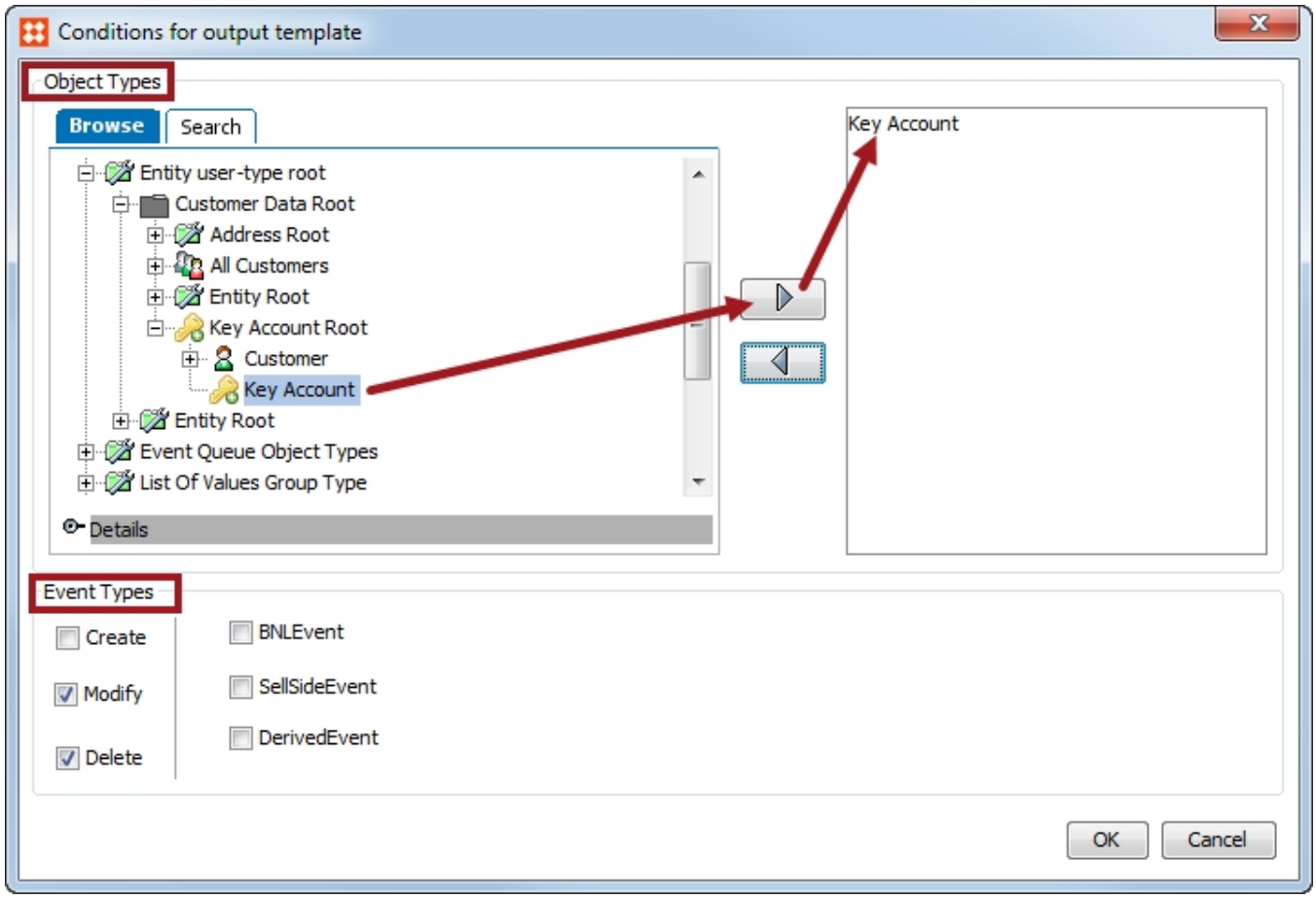
Object Type and Event Type Selection

On the OIEP, open the Configuration tab. Under the Output Templates flipper, click the **Add configuration** link.



Set the Object Types and Event Types.

For this example, Modify events are also included to demonstrate the available functionality.



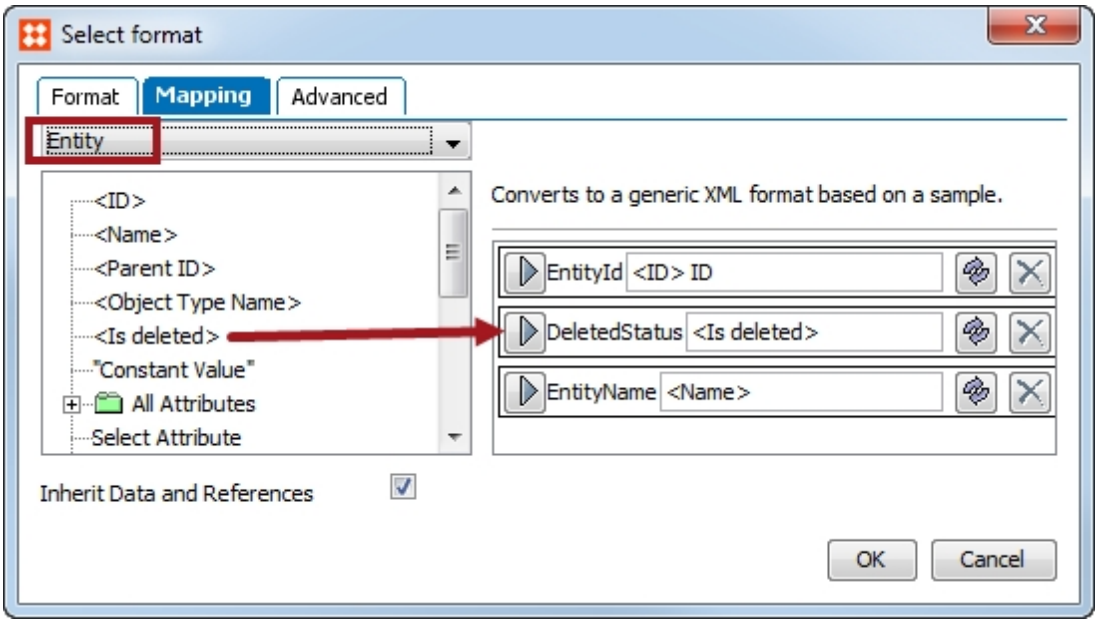
Template

In the Output Template, click the ellipsis button **(…)**, and set the format to **Generic XML**.

Provide the following text in the Sample field.

```
<root>
   <Entities>
      <?Record?>
      <Entity EntityID="[?Target EntityId?]" DeletedStatus="[?Target DeletedStatus?]">
      <EntityName> <?Target?> </EntityName>
      </Entity>
   </Entities>
</root>
```

Mapping



Results

Approve Deletion for an Entity in the Recycle Bin and/or **Approve Object** for modification of a Entity that is Workspace Revisable on the Tree triggers the OIEP.

Invoking the OIEP generates the following output:

```
<root>
  <Entities>
    <Entity EntityID="KeyB" DeletedStatus="IsDeleted">
      <EntityName/>
    </Entity>
  </Entities>
  <Entities>
    <Entity EntityID="KeyA" DeletedStatus="">
      <EntityName>Key A modified</EntityName>
    </Entity>
  </Entities>
</root>
```

For deleted Entities, the Deleted Status displays **"IsDeleted"**. Only ID is retained for export on Delete events.

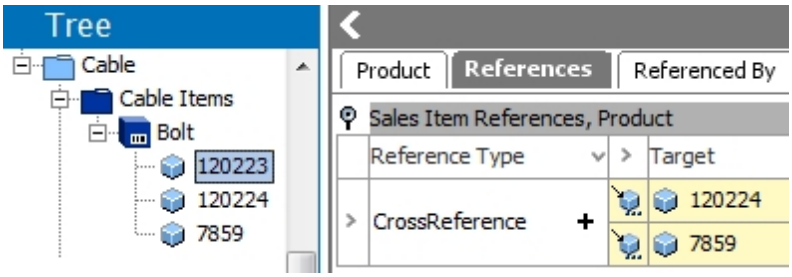
For modified Entities, the Deleted Status displays **" "**, which indicates the entity is not deleted. All mapped fields are exported for Create and Modify events.

Deleting Product References with Generic XML

Outbound Integration Endpoints (OIEPs) can pass deletion events for Product References when the OIEP is configured to listen for Modify events on the relevant objects. The Modify Event Type is necessary since the object itself is not being deleted, but rather a reference on the object which continues to exist.

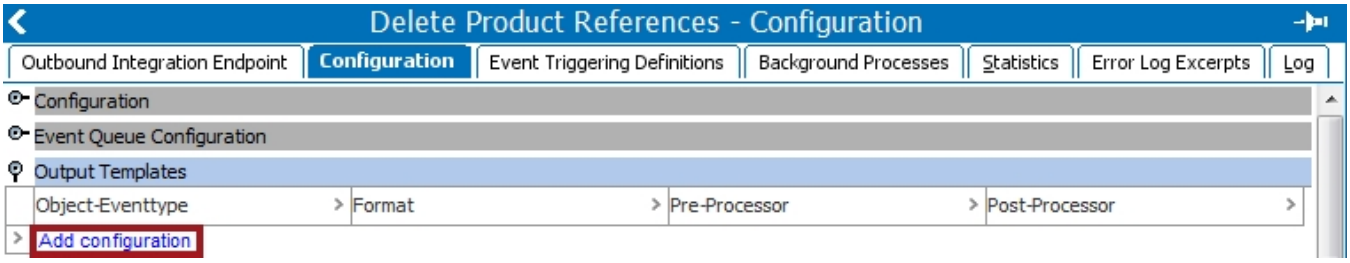
Product

On product 120223, the Reference Type of CrossReference to Target 7859 will be deleted in order to trigger the OIEP.

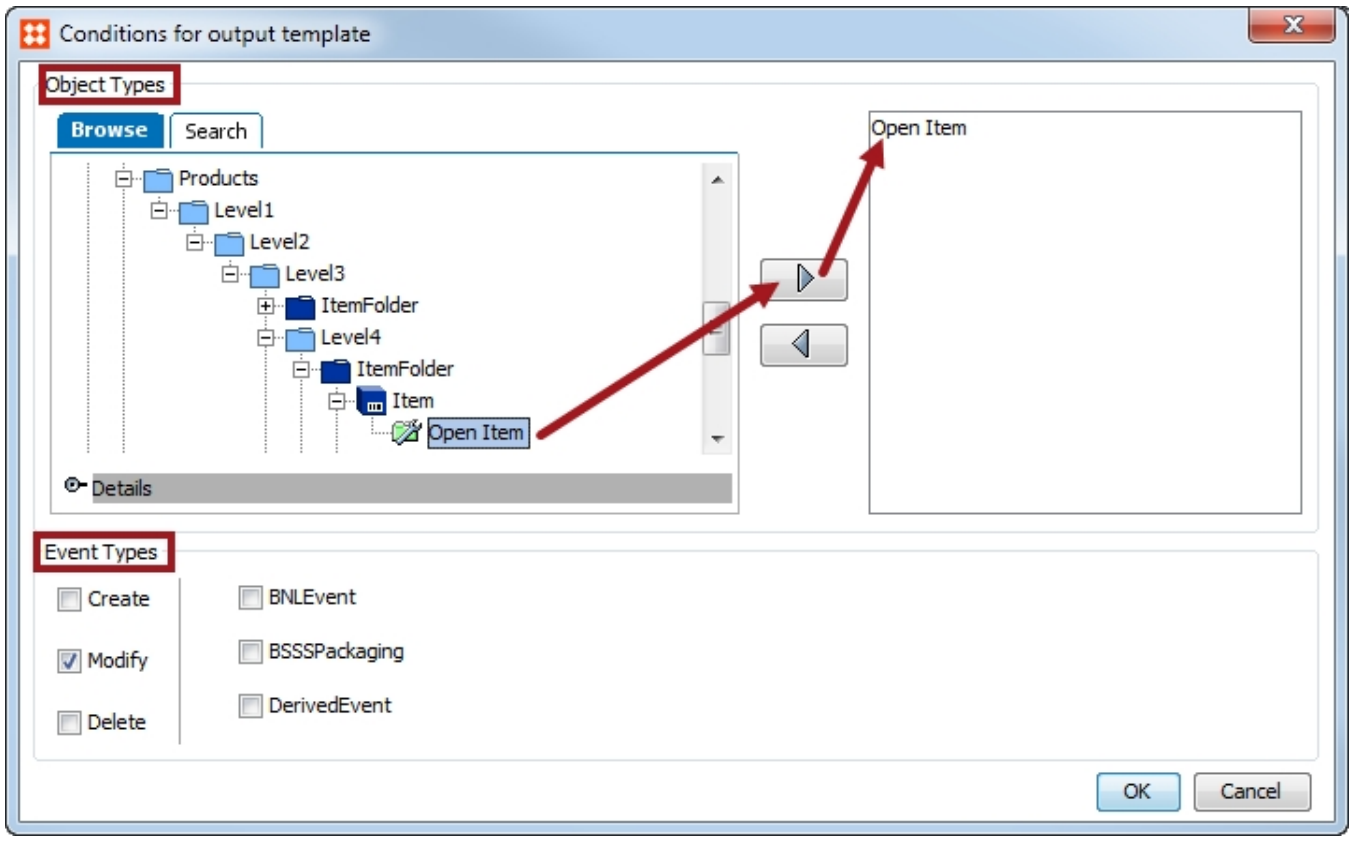


Object Type and Event Type Selection

On the OIEP, open the Configuration tab. Under the Output Templates flipper, click the **Add configuration** link.



Set the Object Types and Event Types.



Template

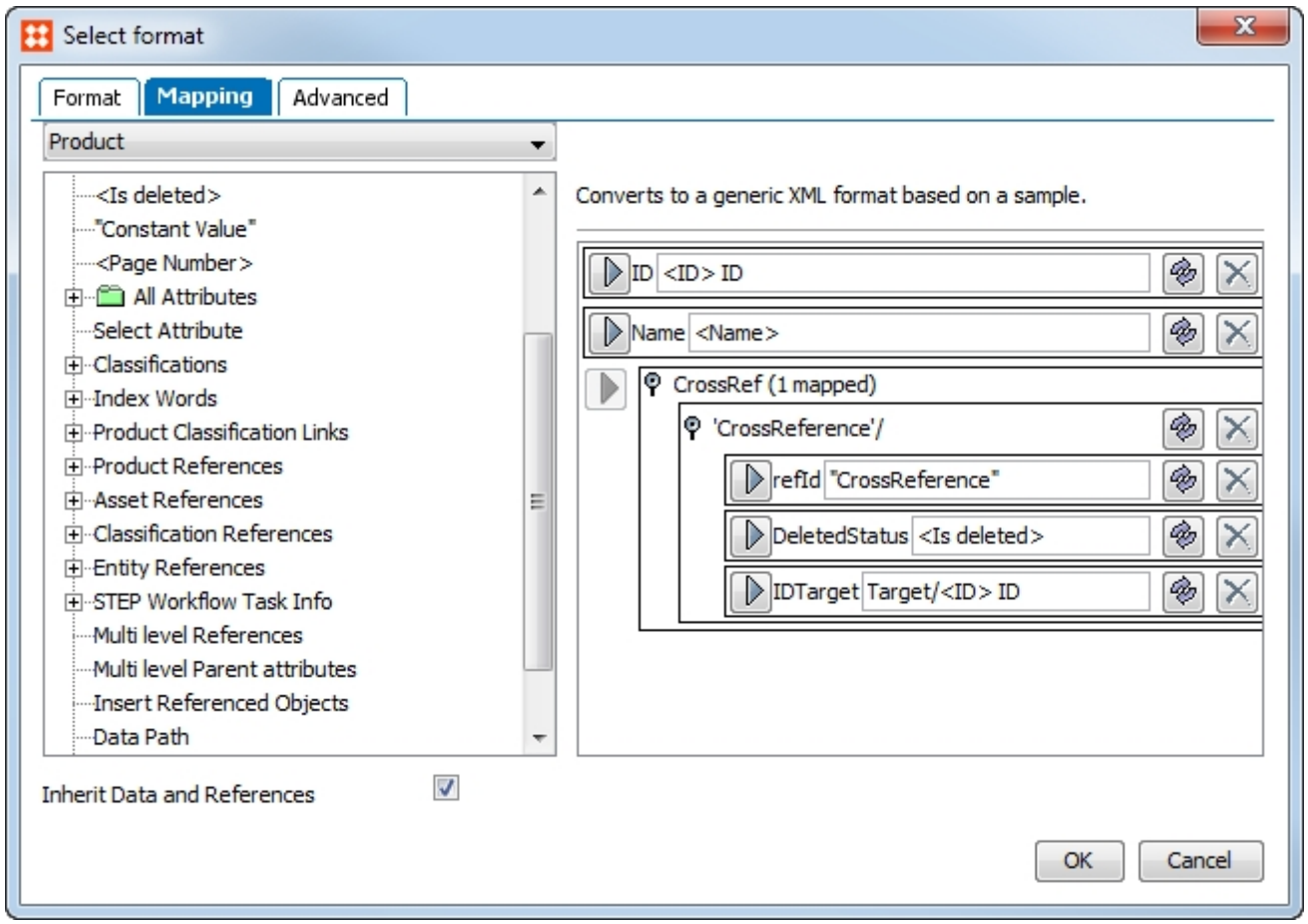
In the Output Template, click the ellipsis button **(…)**, and set the format to **Generic XML**.

Provide the following text in the Sample field.

```
<Products>
   <Product ID="[?Target ID?]">
   <?Record?>
   <Name><?Target?></Name>
   <CrossRefs>
      <CrossRef refTypeID="[?Target refId?]" DeletedStatus="[?Target DeletedStatus?]">
      <?MultiTarget?>
      <IDTarget><?Target?></IDTarget>
      </CrossRef>
   </CrossRefs>
   </Product>
</Products>
```

Mapping

The Constant Value 'CrossReference' is used to identify the type of reference being monitored. The <Is deleted> aspect monitors for deletions. Data Path mapping to the Target reference provides the deleted ID.



See the 'Mapping to Delete Product References with Generic XML' section of the STEP Integration Endpoint User Guide / Integration Endpoints for Data Exchange documentation for detailed mapping steps.

Results

When the Product Reference for item 7859 is deleted and the product 120223 is approved, the OEIP is triggered. Invoking the OIEP generates the following file:

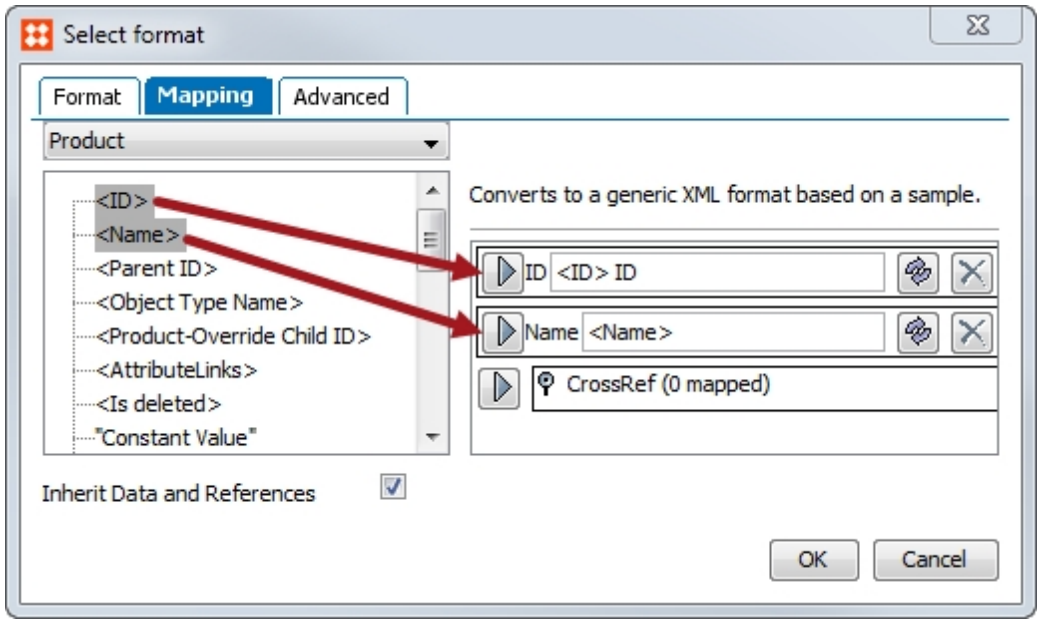
```
<Products>
  <Product ID="120223">
    <Name>120223</Name>
    <CrossRefs>
      <CrossRef refTypeID="CrossReference" DeletedStatus="">
        <IDTarget>120224</IDTarget>
      </CrossRef>
      <CrossRef refTypeID="CrossReference" DeletedStatus="IsDeleted">
        <IDTarget>7859</IDTarget>
      </CrossRef>
    </CrossRefs>
  </Product>
</Products>
```

Note: Both of the product references originally on the product are exported. The DeletedStatus target shows blank for the reference that still exists, and shows "IsDeleted" for the removed reference.

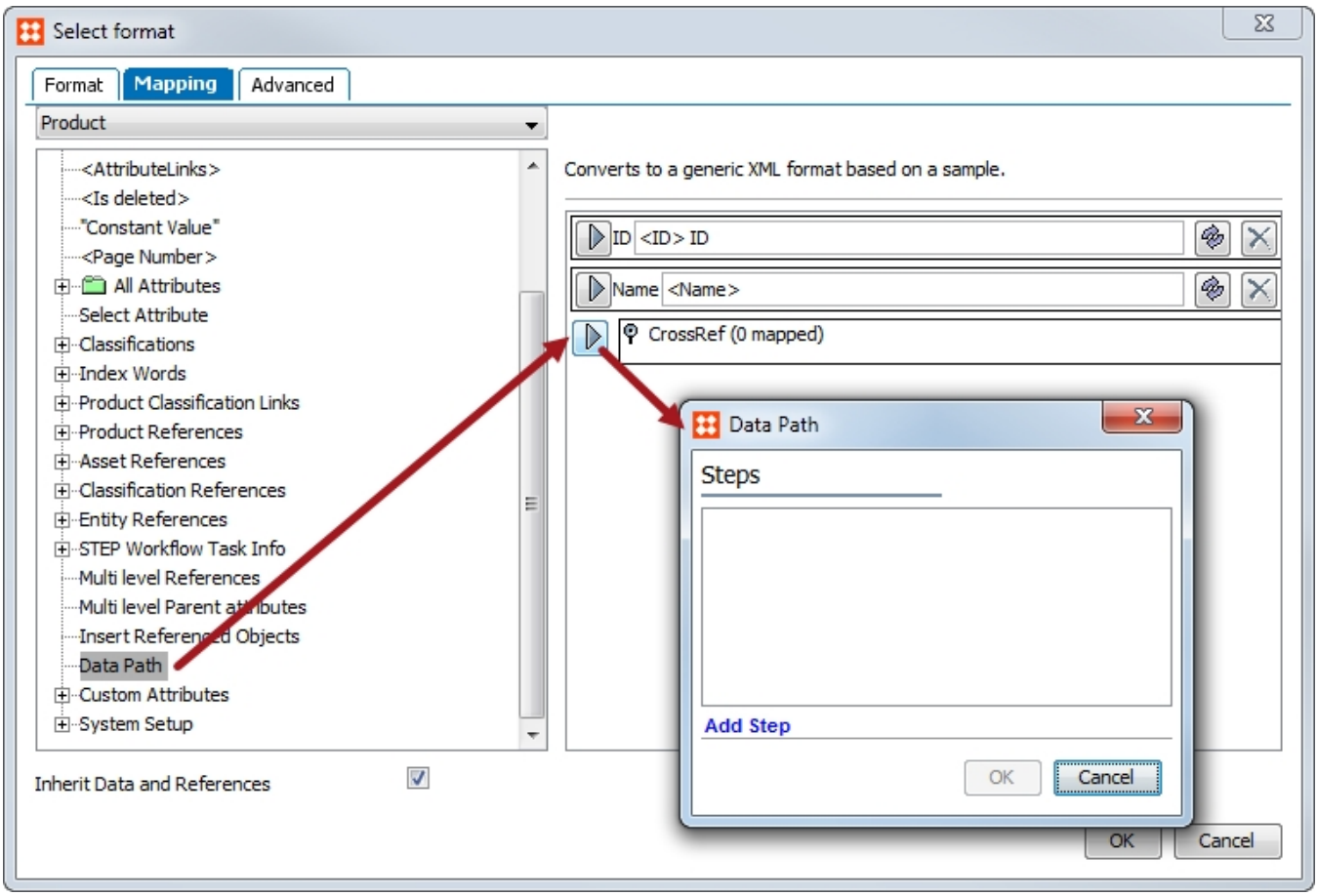
Mapping to Delete Product References with Generic XML

The Deleting Product References with Generic XML section discusses additional setup for a Generic XML export. Use the following steps to map the Product and monitor for deletion of product references.

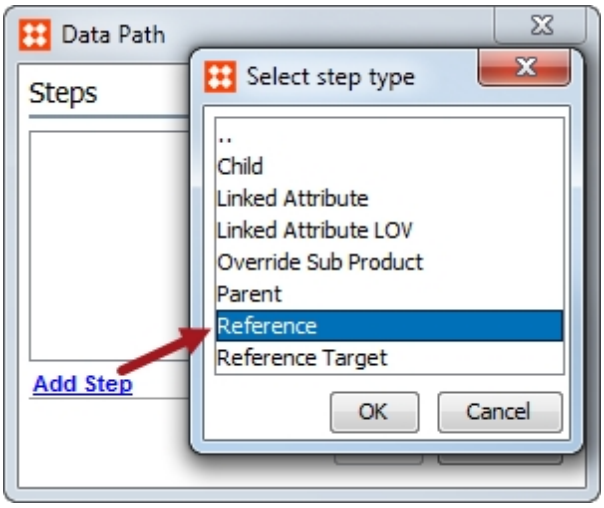
1. Map **ID** and **Name** for the Product.



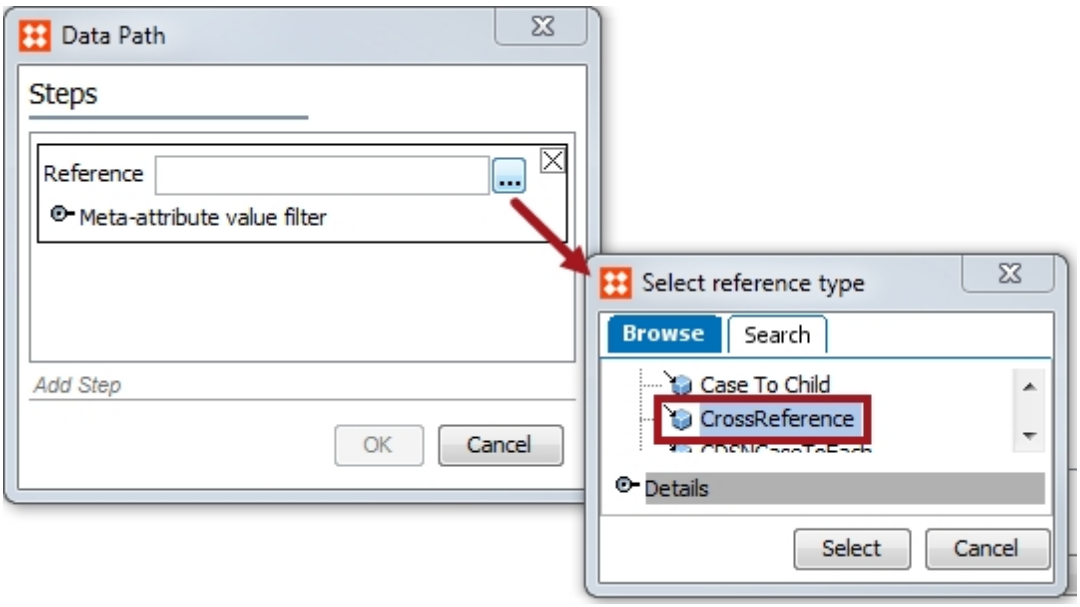
2. Map **Data Path** to CrossRef.



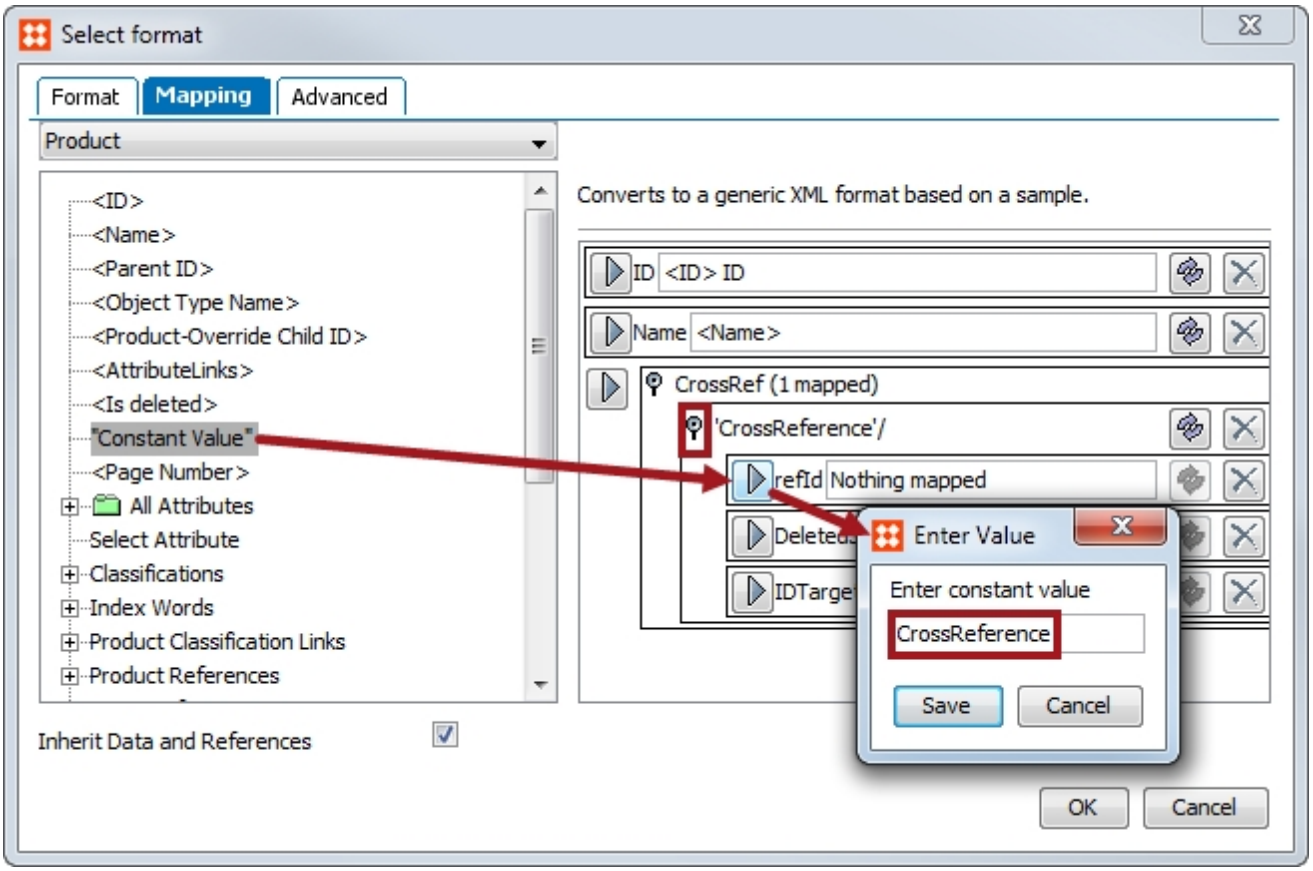
3. Click the **Add Step** link and select **Reference**.



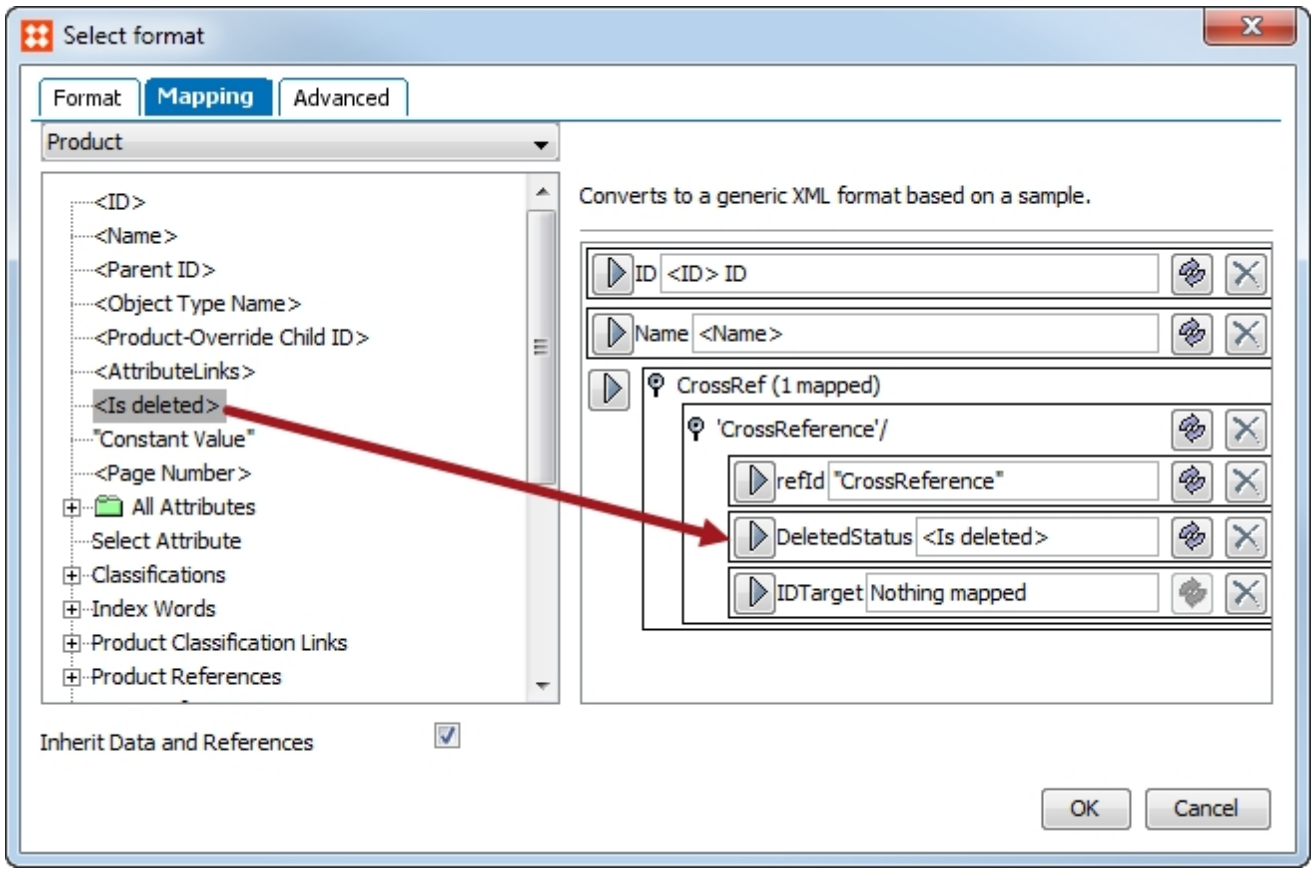
4. Click **OK** to close the **Select step type** dialog.
5. On the Data Path dialog, click the ellipsis button **(…)** and select the **Reference Type** to monitor, then click the **Select** button.



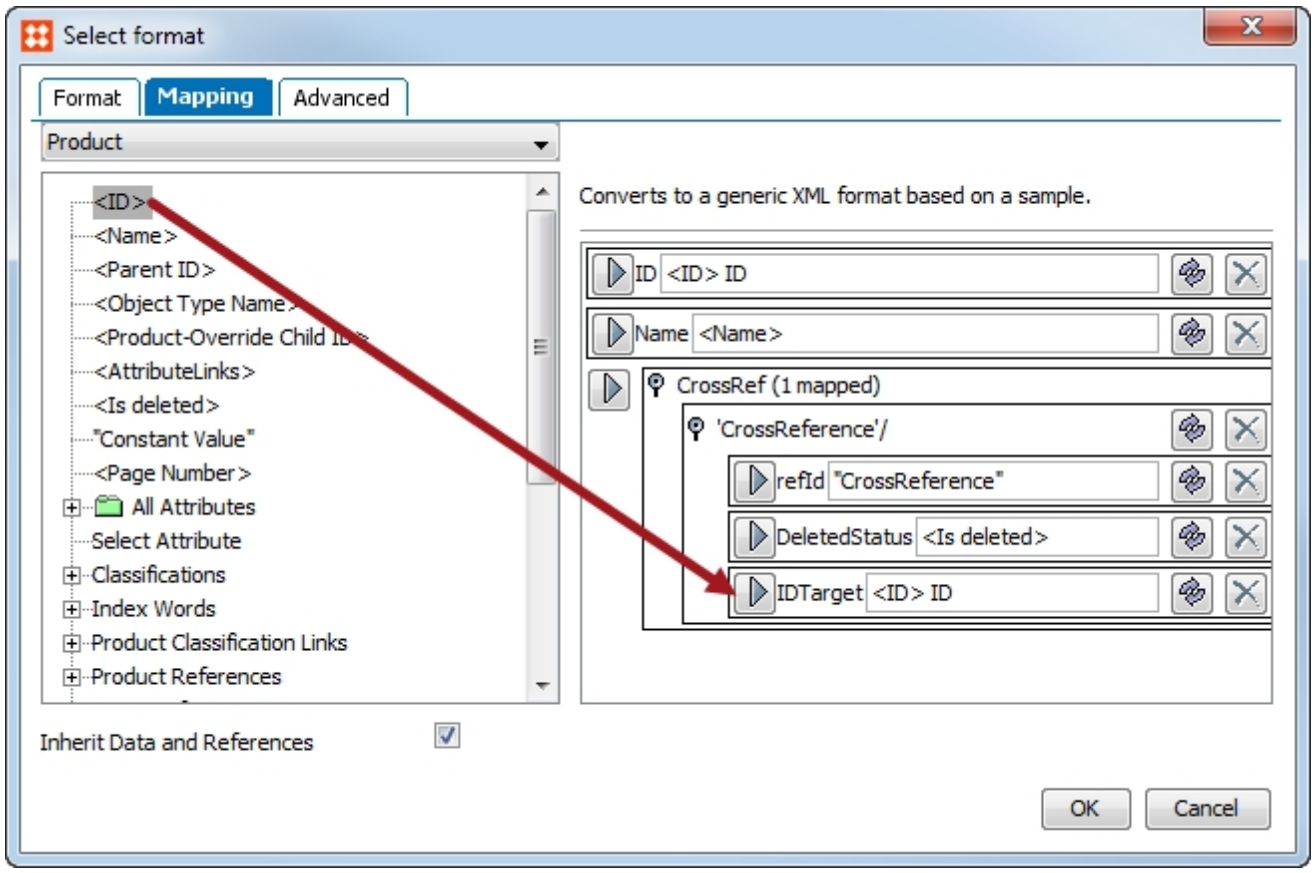
6. Click **OK** to close the **Data Path** dialog.
7. Open the 'CrossReference'/ flipper, map **Constant Value** to refID, add the text, and click the **Save** button.



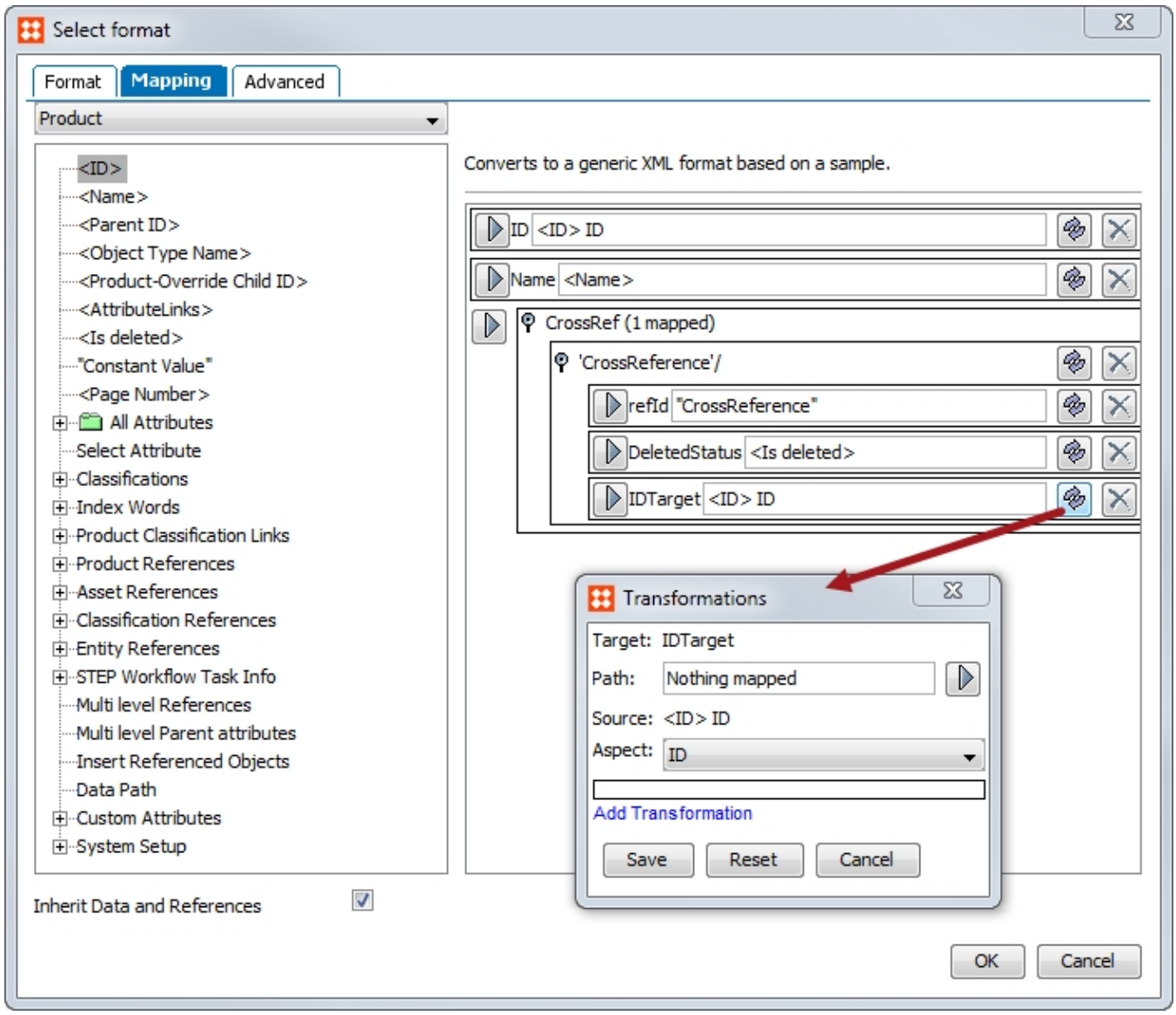
8. Map **<Is deleted>** to DeletedStatus.



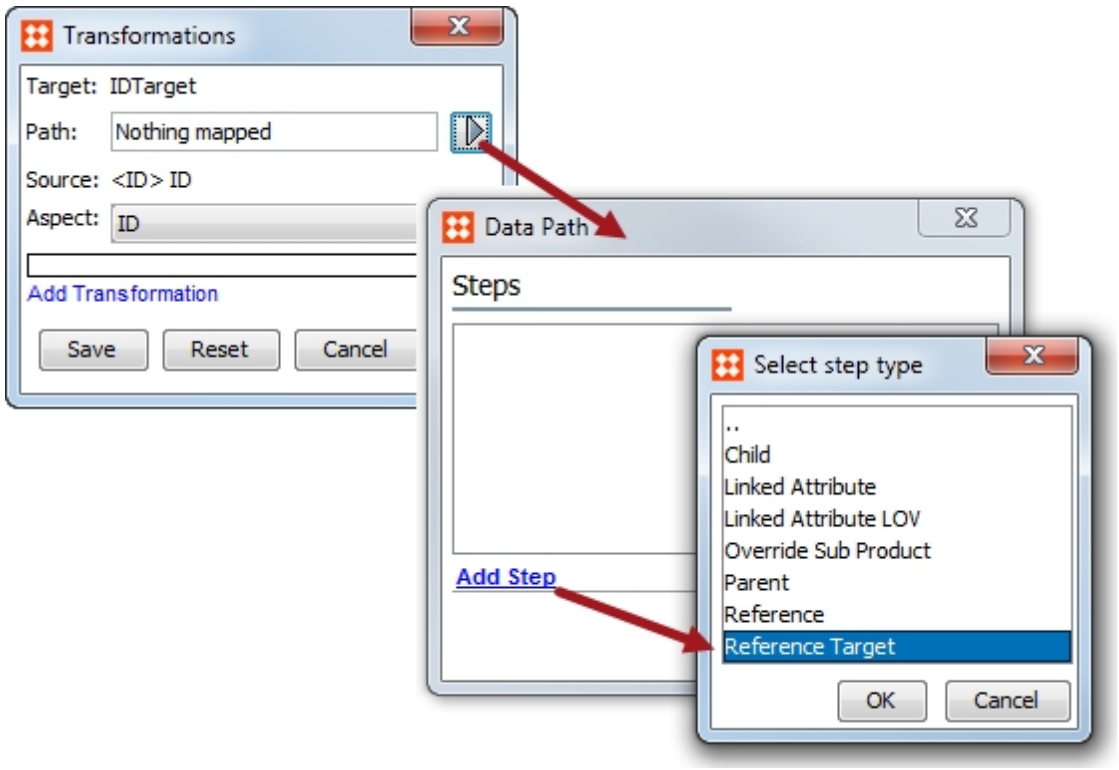
9. Map **ID** to IDTarget.



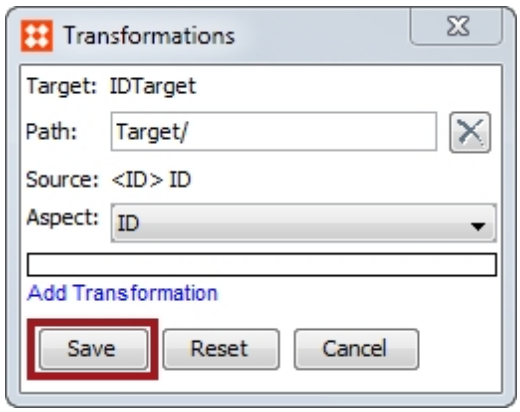
10. Click the **Transformations** button.



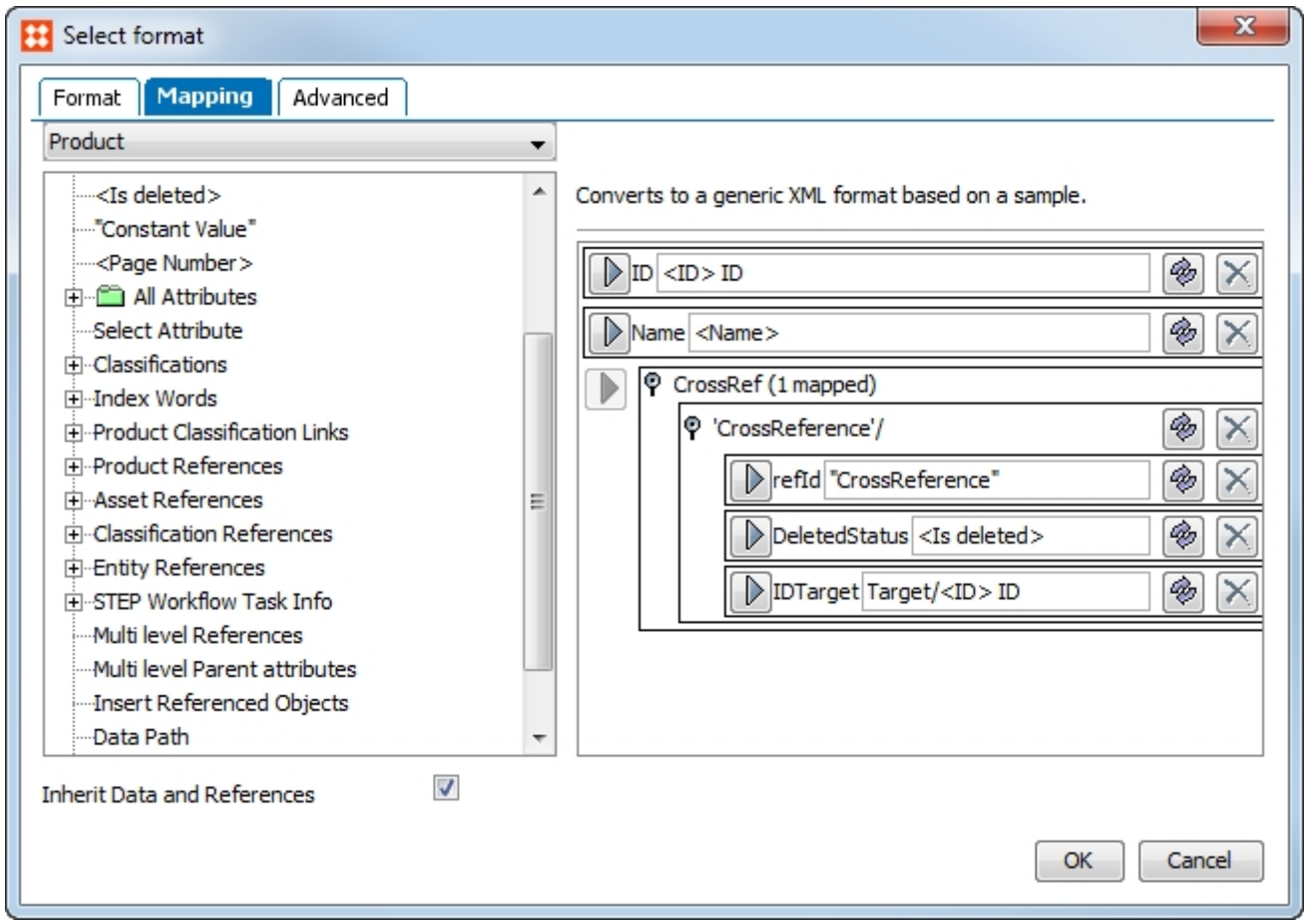
11. Click the **Path** button, click the **Add Step** link, and select **Reference Target**.



12. Click **OK** to close the **Select step type** and **Data Path** dialogs.
13. Click **Save** to close the **Transformations** dialog.



All mapping is complete.

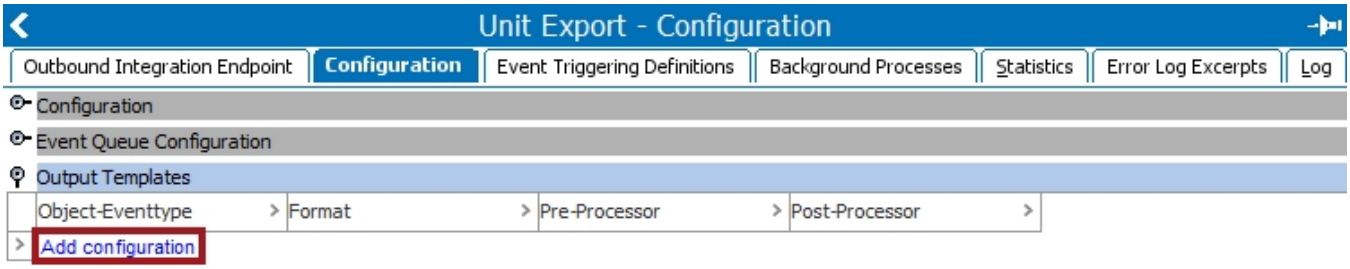

Select format
Format
Mapping
Advanced
Product
<Is deleted>
"Constant Value"
<Page Number>
All Attributes
Select Attribute
Classifications
Index Words
Product Classification Links
Product References
Asset References
Classification References
Entity References
STEP Workflow Task Info
Multi level References
Multi level Parent attributes
Insert Referenced Objects
Data Path
Converts to a generic XML format based on a sample.
ID <ID> ID
Name <Name>
CrossRef (1 mapped)
'CrossReference'/
refId "CrossReference"
DeletedStatus <Is deleted>
IDTarget Target/<ID> ID
Inherit Data and References
OK
Cancel

Exporting Units with Generic XML

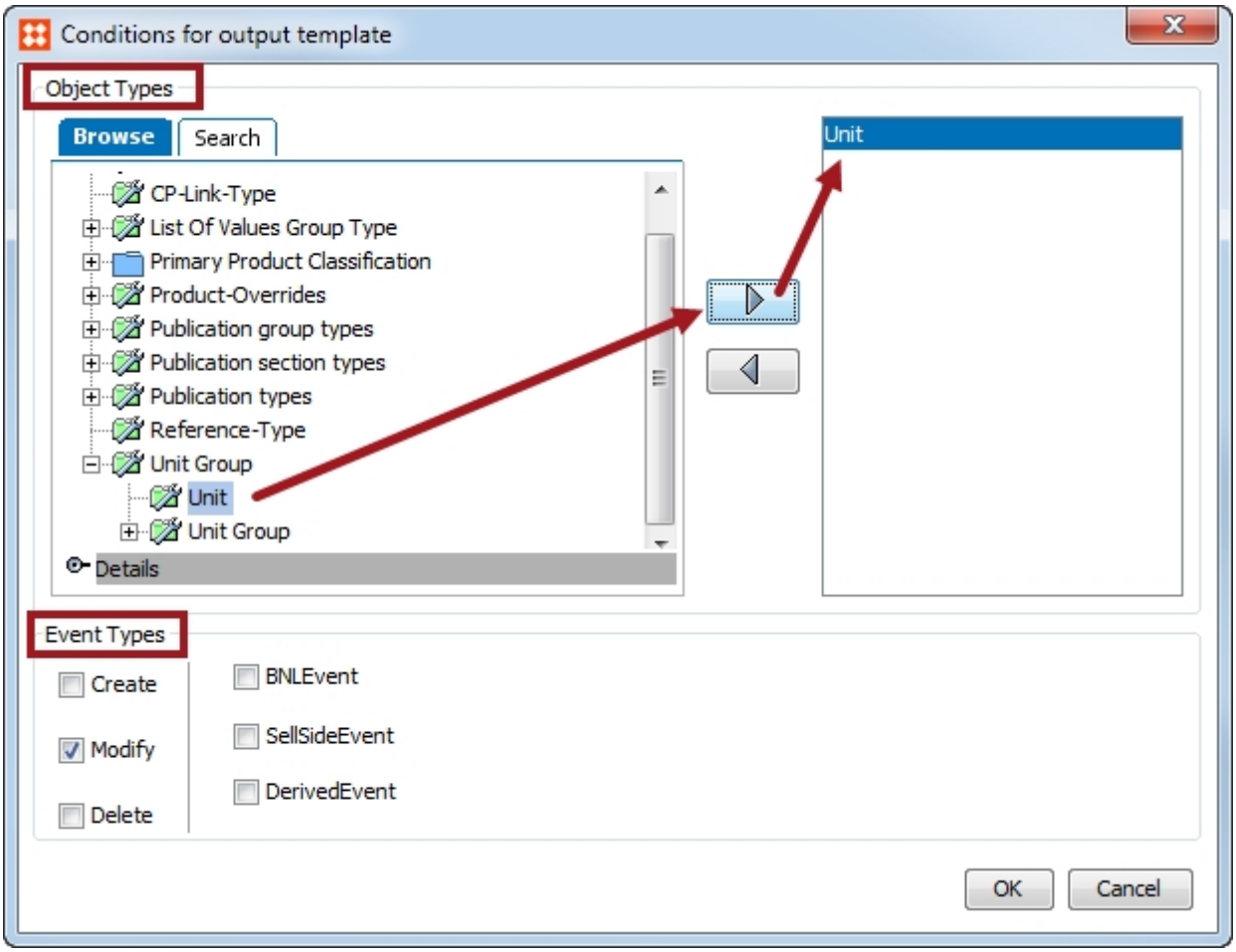
Outbound Integration Endpoints (OIEPs) allow the export of unit information.

Object Type and Event Type Selection

On the OIEP, open the Configuration tab. Under the Output Templates flipper, click the **Add configuration** link.



Set the Object Types and Event Types.



Template

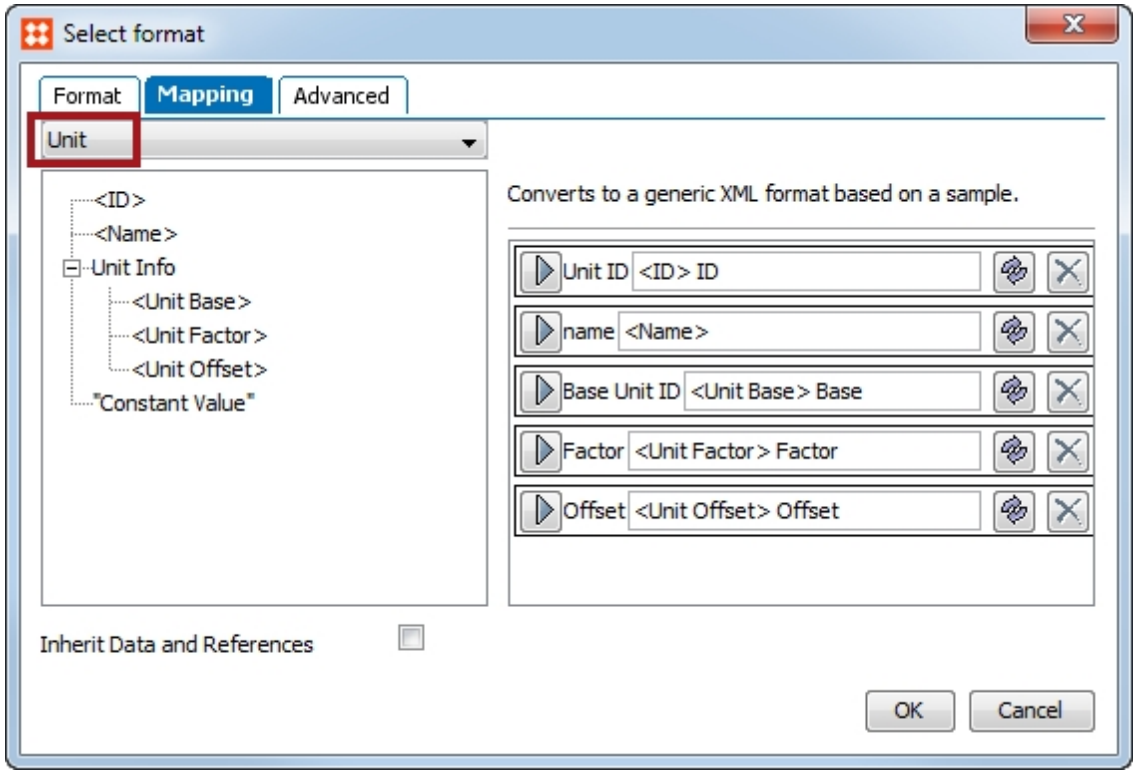
In the Output Template, click the ellipsis button **(…)**, and set the format to **Generic XML**.

Provide the following text in the Sample field.

```
<ExportUnits>
   <Units>
      <Unit ID="[?Target Unit ID?]">
         <?Record?>
         <Name>
             <?Target name?>
         </Name>
         <UnitConversion BaseUnitID="[?Target Base Unit ID?]" Factor="[?Target Factor?]" Of
fset="[?Target Offset?]"/>
      </Unit>
   </Units>
</ExportUnits>
```

Mapping

Select **Unit** from the dropdown and then map the data sources to the targets provided by the template.



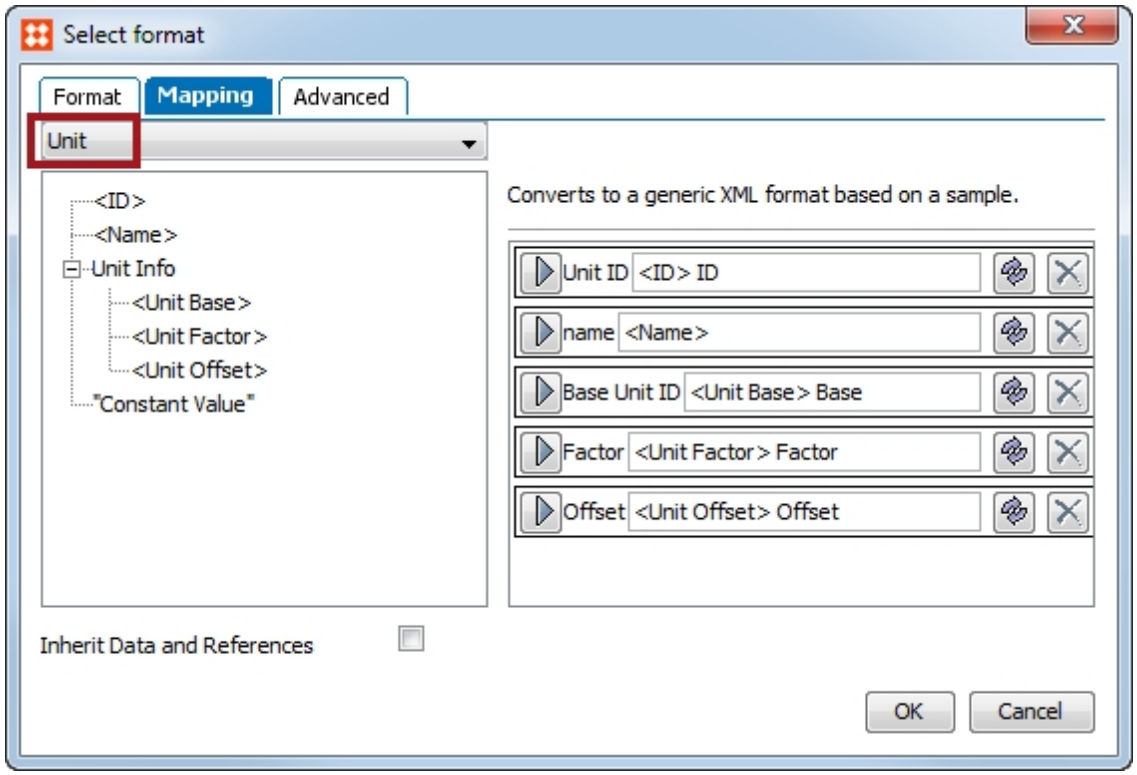
See the Unit Aspects section for more information.

Results

```
<ExportUnits>
  <Units>
    <Unit ID="unece.unit.INK">
      <Name>in²</Name>
      <UnitConversion BaseUnitID="unece.unit.MTK" Factor="6.4515E-4" Offset="0.0"/>
    </Unit>
    <Unit ID="unece.unit.ACR">
      <Name>acre</Name>
      <UnitConversion BaseUnitID="unece.unit.MIK" Factor="3.0" Offset="1.0"/>
    </Unit>
    <Unit ID="unece.unit.FTK">
      <Name>ft²</Name>
      <UnitConversion BaseUnitID="unece.unit.MTK" Factor="0.09290305" Offset="0.0"/>
    </Unit>
    <Unit ID="unece.unit.CMK">
      <Name>cm²</Name>
      <UnitConversion BaseUnitID="unece.unit.MTK" Factor="2.0E-4" Offset="0.0"/>
    </Unit>
  </Units>
</ExportUnits>
```

Mapping Unit Aspects with Generic XML

Aspects, also called data sources, are displayed when mapping Units for export. The description of each Aspect is included below.



Aspect	Description
ID	Extracts the ID of the unit. For example, if a unit has the ID 1234 and the name 'Pound,' then '1234' is extracted.
Name	Extracts the name of the unit. For example, if a unit has the ID 1234 and the name 'Pound,' then the word 'Pound' is extracted.
Unit Base	Extracts the ID of the base unit for the unit. For example, if a unit has base unit 'm,' then the 'm' is extracted.
Unit Factor	Extracts the conversion factor that should be multiplied to a value with the unit to get the value in the base unit. For example, if a unit has a base unit conversion of 'value(g) = .001 * value(kg),' then '.001' is

Aspect	Description
	extracted.
Unit Offset	Extracts the conversion factor that should be added to a value with the unit to get the value in the base unit. For example, if a unit has a base unit conversion of 'Value(K) = .556 * Value(°F) + 255.3722,' then '255.3722' is extracted.

For a description of Constant Value, see the Map Data section of Exporting Data User Guide / Exporting Data and Images.

Pre-Processor Configuration

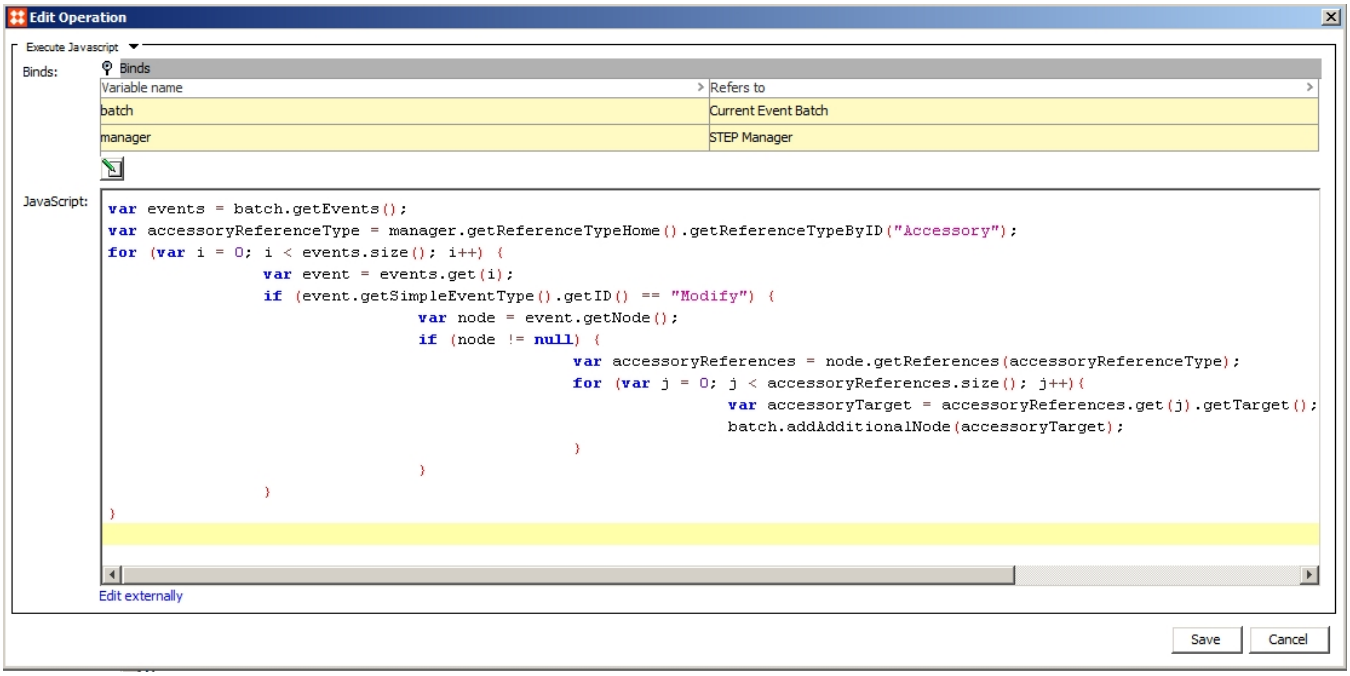
A business action pre-processor can be applied before exporting events from a batch in an event queue.

By applying the pre-processor, a node can be added to a current batch, or an event can be removed from the current batch before a message is created for an export. A Triggering Object Type event filter or generate event option can be used in place of a pre-processor. For more information, see the **Outbound Integration Endpoint Event Triggering Definitions** section of the **Integrated Endpoints** documentation.

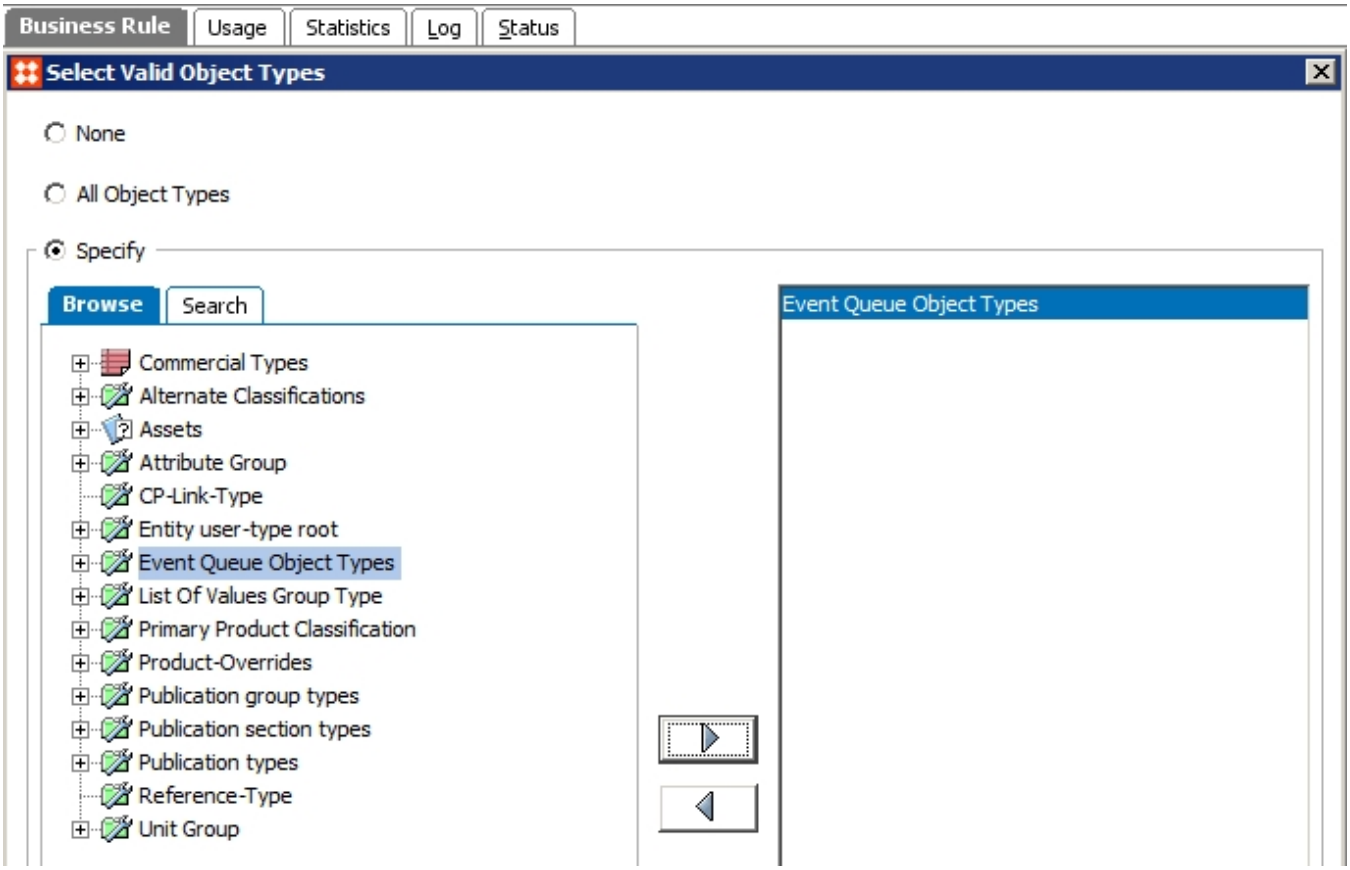
Example Using a Pre-Processor Business Action to Add a Referenced Product

In the following example, an event-based OIEP has been configured to deliver messages that inform a downstream system about changes to a product. A pre-processor business rule (business action) will be used to add referenced products to the products that are contained in the message.

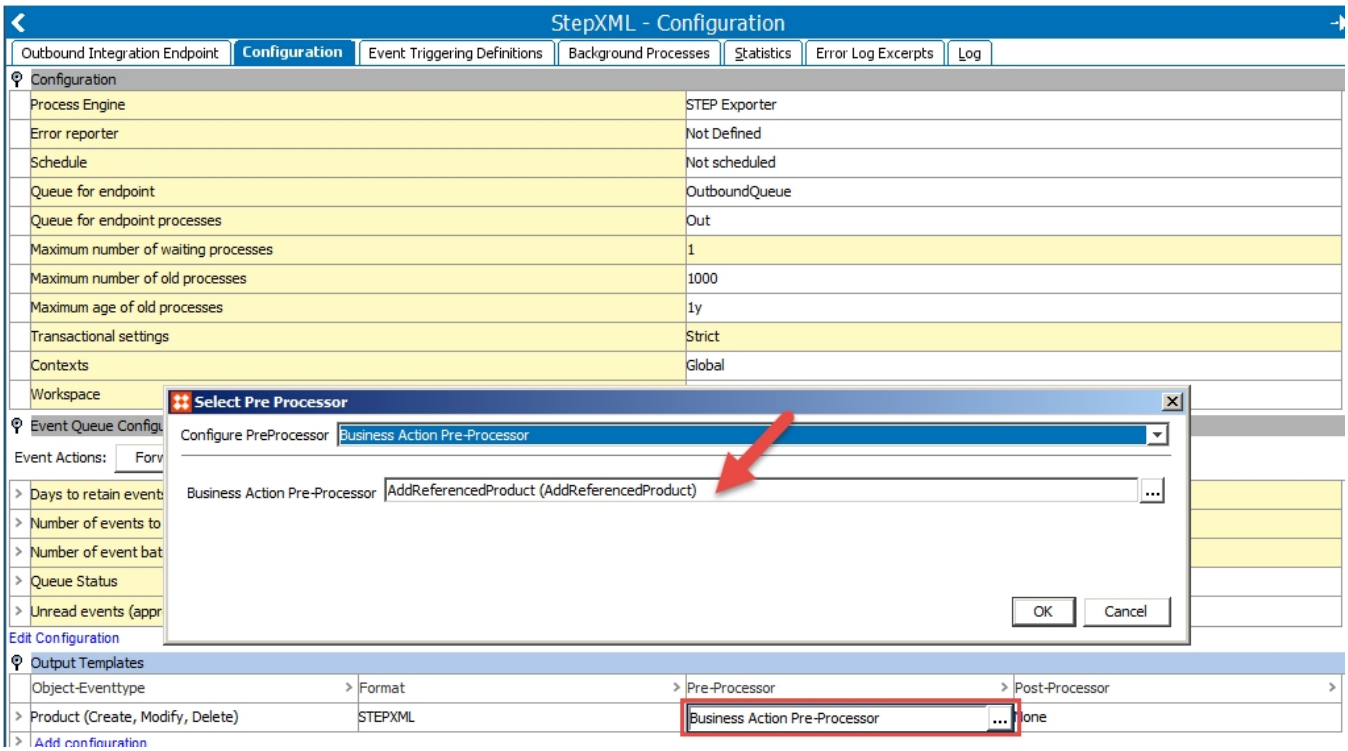
First, a business action is created where the current event batch is addressed to either add a node or to remove an event from the event batch. In this example, a node is added to each source product to include the target products that are referenced via the 'Accessory' Product Reference Type.



To make the business action valid, it must be made valid for 'All Object Types' or specifically for 'Event Queue Object Types':



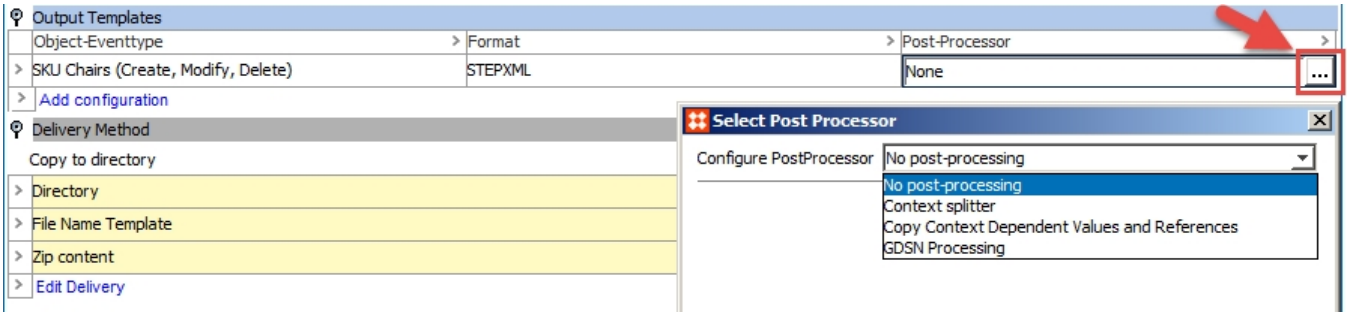
A pre-processor can then be configured for any of the output templates:



1. In **System Setup**, locate the relevant outbound integration endpoint.
2. On the **Configuration** tab, navigate to the **Event Queue Configuration** flipper and locate the relevant output template. In the **Pre-Processor** column, click the ellipsis button **(…)**, and then select the relevant pre-processor.
3. In **Configure PreProcessor**, the following options are available:
   - Select **No pre-processing** to export files without pre-processing.
   - Select **Business Action Pre-Processor** to execute a given business action on an event batch before export.
   - Select a pre-made business action to add a node or to remove an event from the event batch.

Post-Processor Configuration

Standard post-processing options should be evaluated if multiple contexts are included in the output. You can configure a post-processor for each of the output templates. The Triggering Object Type generate event option can be used in place of a post-processor. For more information, see the **Outbound Integration Endpoint Event Triggering Definitions** section of the **Integrated Endpoints** documentation.



1. In **System Setup**, locate the relevant outbound integration endpoint.
2. On the **Configuration** tab, navigate to the **Event Queue Configuration** flipper and locate the relevant output template. In the **Post-Processor** column, click the ellipsis button **(…)**, and then select the relevant post-processor.
3. In **Configure Post-processor**, the following options are available:
   - Select **No post-processing** to export files using the standard export manager.
   - Select **Context splitter** to have the endpoint generate an export file for each configured context. Each exported file will only contain context specific data.

   Note: When using Excel or CSV format, and multiple contexts are configured for export, you must select Context Splitter, or a single context will be included in the export.

     - Set **Copy inherited product values** to **Yes** to copy and save inherited product values to the child product.
     - Set **Copy inherited product values** to **No** to use inherited product values in the export, but leave the child product unmodified.
   - Select **Copy Context Dependent Values and References** to export context-dependent values and references and add the corresponding ContextID attribute to the value or reference in question. The endpoint generates one file containing values and references for each context specified in the wizard **Step 3: Configure Endpoint**.
     - Set **Copy inherited product values** to **Yes** to copy and save inherited product values to the child product.
     - Set **Copy inherited product values** to **No** to use inherited product values in the export, but leave the child product unmodified.

   Note: If you use the **Copy Context Dependent Values and References** post-processor to add ContextIDs to a cross-context export, it is easier for the downstream system to interpret the exported file.

This is because it can use the <ContextID> tag to identify specific contexts. It also makes it easier to re-import the exported data into the correct contexts in STEP after processing by a third-party service or application.

For more information, see Post-Processor Copy Context Dependent Values and References section.

Post-Processor Copy Context Dependent Values and References

The **Copy Context Dependent Values and References** post processor adds the <ContextID> and <QualifierID> XML tags to the export. A standard cross-context export only adds the <QualifierID>. The <QualifierID> tag specifies the ID of the Dimension Point in which the attribute value exists but does *not* provide the specific context. For example, the <QualifierID> of 'std.lang.all' could be the ID of the 'Language Root' Dimension Point (which would typically only be used in a Global context). But, without the addition of the <ContextID> to the export, the additional contexts that an attribute value has inherited down to cannot be determined.

When the <ContextID> XML tag is used in combination with the <QualifierID> XML tag, it is possible for the downstream system to determine the inheritance relationship between a value or reference and a Context.

This is useful when a service tries to re-import processed data into STEP. Because the downstream system knows the context the data must be imported into, it is easier to target specific contexts within STEP for import.

This example illustrates the differences between using the Export Manager for a standard cross-context export and using an outbound integration endpoint with the **Copy Context Dependent Values and References** post-processor enabled.

In the following, attribute values are used, but the post-processor also works with inherited references such as classification or asset references.

The product has a language-dependent attribute, 'Body Copy', with values defined in two different contexts: 'GL (Global)' and 'FR All All (French)'.

Global context attribute value:

Name		Value
Body Copy	abc	Stylish, modern and lightweight, ideal for use in cafes, bars, bistros and clubs.

French context attribute value:

Name		Value
Body Copy	abc	Elégantes, modernes et légères, idéales pour les cafés, les bars, les bistros ou les discothèques. Assortis aux tables frêne page 282, 283.

A third context, 'DE All All (German)', inherits its attribute values from the Global context. This is because no values have been entered in the DE All All context yet, and DE All All has an inheritance relationship with the GL context:

German context attribute value:

Name		Value
Body Copy	abc	Stylish, modern and lightweight, ideal for use in cafes, bars, bistros and clubs.

XML Output Example - Standard Cross-Context Export

Using a standard cross-context export, the XML looks like the following:

```
<Product ID="CHAIR" UserTypeID="Product Folder" ParentID="MyChairFolder">
  <Name QualifierID="std.lang.all">SitWell Armchair</Name>
  <Values>
    <ValueGroup AttributeID="Body Copy">
      <Value QualifierID="std.lang.all">Stylish, modern and lightweight, ideal for use in cafes, bars, bistros and clubs.</Value>
      <Value QualifierID="fre">Elégantes, modernes et légères, idéales pour les cafés, les bars, les bistros ou les discothèques.</Value>
    </ValueGroup>
  </Values>
</Product>
```

Only the attribute values specific to particular dimension points—indicated by QualifierIDs—are exported, and only the attribute values that are different from the Global context. Global values are taken from 'std.lang.all' and the French values are taken from 'fre'. Since the 'Name' attribute is inherited from the 'std.lang.all' qualifier, no specific French or German 'Name' is exported. Because standard export methods collapse duplicate values, only the top value of the inheritance tree is exported.

XML Output Example - Copy Context Dependent Values and References

The following shows an XML output of an OIEP cross-context export with the **Copy Context Dependent Values and References** post-processor enabled:

```
<Product ID="CHAIR" UserTypeID="Product Folder" ParentID="MyChairFolder">
    <Name ContextID="GL" QualifierID="std.lang.all">SitWell Armchair</Name>
    <Name ContextID="FR All All" QualifierID="std.lang.all">SitWell Armchair</Name>
    <Name ContextID="DE All All" QualifierID="std.lang.all">SitWell Armchair</Name>
    <Values>
        <Value AttributeID="Body Copy" ContextID="GL" QualifierID="std.lang.all">Stylish, modern and lightweight, ideal for use in cafes, bars, bistros and clubs.</Value>
        <Value AttributeID="Body Copy" ContextID="FR All All" QualifierID="fre">Elégantes, modernes et légères, idéales pour les cafés, les bars, les bistros ou les discothèques.</Value>
        <Value AttributeID="Body Copy" ContextID="DE All All" QualifierID="std.lang.all">Stylish, modern and lightweight, ideal for use in cafes, bars, bistros and clubs.</Value>
    </Values>
</Product>
```

The <ContextID> tag has been added to the export along with the <QualifierID> tag, and each value—including inherited values—is explicitly exported. This means that the German values are included in the export, even though the DE All All context inherits from the GL context. Once the value is overwritten in the German context, the <QualifierID> would change to the ID of the German language dimension point.

Outbound Integration Endpoint Event Messaging

STEP event messaging is based on event queues and can be created as outbound integration endpoints. When event messaging is enabled, messages are sent from STEP to the downstream system each time a specific event occurs; for example, when products are deleted or modified.

Note: Changes on data under revision control need manual approval before an event is released, whereas changes on data that is not under revision control do not require approval.

The two types of events are Core Events and Derived Events. For cases where you want to be able to use different output templates under conditions different from the core conditions, see the **Outbound Integration Endpoint Derived Events** section of the **Integration Endpoint** documentation.

Core Events

You can use different output templates (export configurations) within the OIEP framework to export data depending on the event type (create / modify / delete for core events) and for the type of object for which the event is generated. This means that with the core events, you can use different formats for different types of objects when a create, modify or delete event is generated.

As demonstrated in the table below, events are always tied to an object in STEP. To simplify, an event can be said to include data about its creation time, the object for which the event was generated, and the event type.

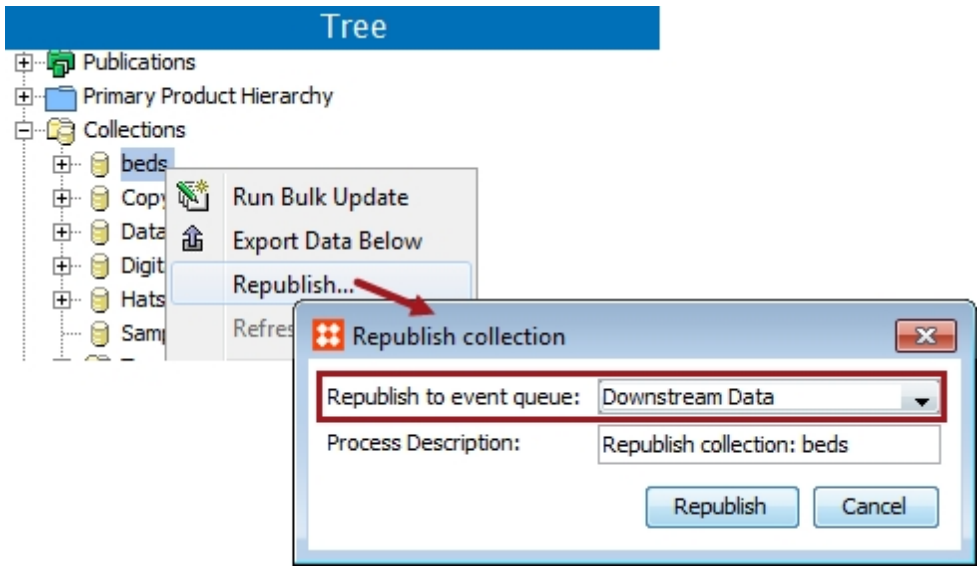
Core Event Type	Internal Event Type ID	External Event Type	Description
APPROVECREATED	0	Create	Generated when a revisable object is approved for the first time.
APPROVEMODIFIED	1	Modify	Generated when a revisable object already in Approved is approved again.
DELETEAPPROVAL	2	Delete	Generated when the deletion of a revisable object in Approved is approved.
GLOBALATTRIBUTECHANGE	11	Modify	Generated when the value for an Externally Maintained Attribute is changed.
GLOBALEVENT	35	Create	Generated when one of these System Setup objects is created: attribute, attribute group, list of values group, list of values, unit group, unit, product to classification link type, and reference type objects

Core Event Type	Internal Event Type ID	External Event Type	Description
GLOBALEVENT	14	Delete	Generated when one of these System Setup objects is deleted: attribute, attribute group, list of values group, list of values, unit group, unit, product to classification link type, and reference type objects
GLOBALEVENT	3	Modify	Generated when one of these System Setup objects is modified: attribute, attribute group, list of values group, list of values, unit group, unit, product to classification link type, and reference type objects
GLOBALREFERENCETYPECHANGE	14	Modify	Generated when an Externally Maintained Reference or Link is modified.
GLOBALTERMCHANGE	13	Modify	Generated when a Terms List is modified.
REPUBLISH	12	Create	Generated for objects on demand from Business Rules, GUI or Bulk Update.
REVIVED	10	Create	Generated when a revisable object is revived.

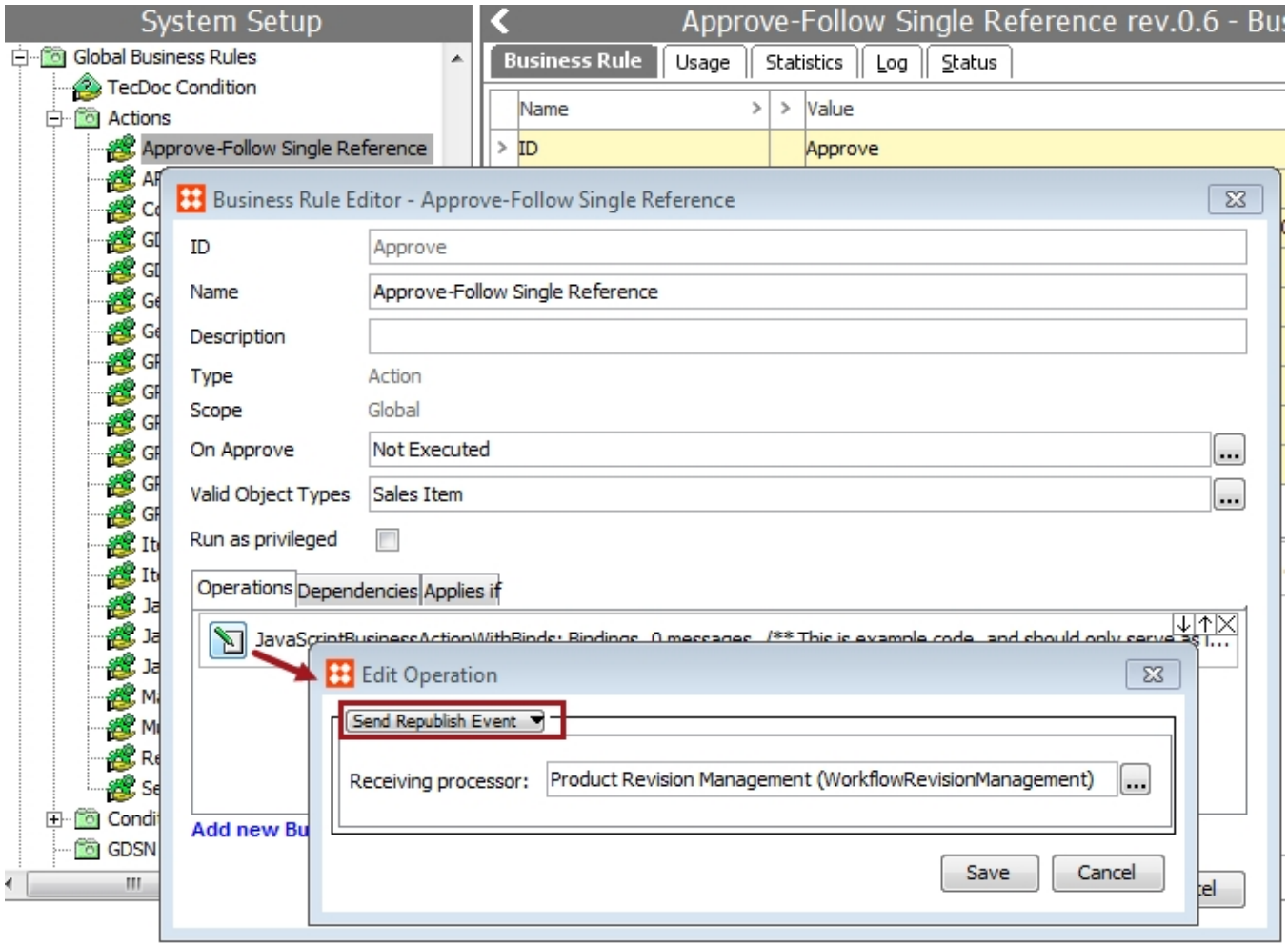
Except for the republish event, the events described above are all generated automatically by STEP when certain actions are performed.

Republish events can be generated manually in STEP Workbench in two ways:

- From a collection



- Via a dedicated bulk update / business action plugin



Outbound Integration Endpoint Derived Events

Core event functionality can be used to publish data to a downstream system when an object is created or updated via an import, or when an object reaches a specific state in a workflow. These situations will not normally trigger any of the other core events. Since an event-based OIEP cannot discriminate the republish event from other types of events, it would also be registering events and publishing data, for example, based on an object being approved for the first time. This problem can be avoided by instead using a derived event, which is an event triggered based on the triggering of another event. Use a derived event when the trigger you need is not available as a core event.

Another case for derived events is when you want to use different output templates under conditions different from the core conditions. For example, when publishing data to a website. If the website categories are modeled using classification in STEP, you may want to send files formatted in different ways or include different data when a product is first linked into a category, when a product linked to a category is modified, and when a product is unlinked from a category.

You may also want to generate derived events on referenced images, referenced products, parent product, or on products that are linked, removed, or modified in a specific classification structure.

When you set up triggering object types on an event-based outbound integration endpoint, you can apply business rules to generate derived events based on the original object type triggering. Derived events are triggered based on business rules, and can be set up depending on the need.

Note: To be able to create successful derived events, you need a thorough understanding of business rules. For more information, see the **Business Rules Overview** section of the **Business Rules** documentation.

Add a Derived Event Type

Before you can add a derived event, you need a derived event type.

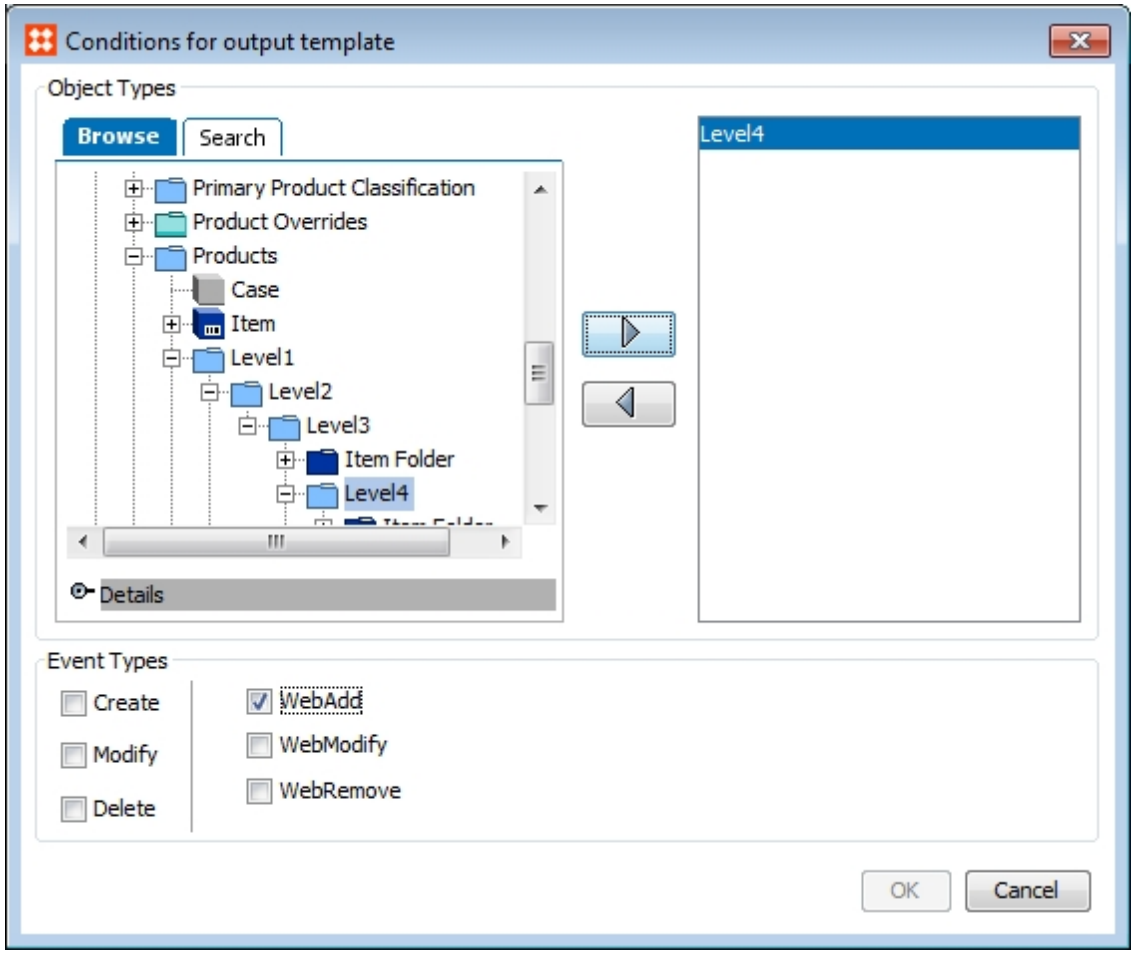
1. In System Setup, click **Derived Events**.
2. On the Derived Events tab, click the **Add Derived Event Type** link.
3. Type an ID and click the **Add** button. The derived event type is now available for use in an output template for an endpoint.

Derived Event Example

The goal of this example is to create a business rule that generates derived events when products are linked, removed, or modified in a specific classification hierarchy, and to have STEP distinguish if the products have been created, modified, or deleted in the classification structure. This enables you to specify different output formats for the different event types.

1. Create the following three derived event types:
   - WebAdd
   - WebModify
   - WebRemove

2. Create an event-based outbound integration endpoint, click the Output Templates **Add configuration** link to display the 'Conditions for output template' dialog.
3. On the **Conditions for Output Template** dialog, select the object type and the derived event type WebAdd.



4. Create additional conditions for output templates using the object type and configuration for WebRemove and WebModify.

Output Templates

Object-Eventtype	Format	Pre-Processor	Post-Processor
Level4 (WebAdd)	Excel (3 mappings)	None	None
Level4 (WebModify)	Excel (3 mappings)	None	None
Level4 (WebRemove)	Excel (3 mappings)	None	None
Add configuration			

In this example, the format for all output templates is Excel but the mapping differ in each template. You can also specify different output formats for each output template such as Generic XML, STEPXML, or CSV, depending on the specific requirements.

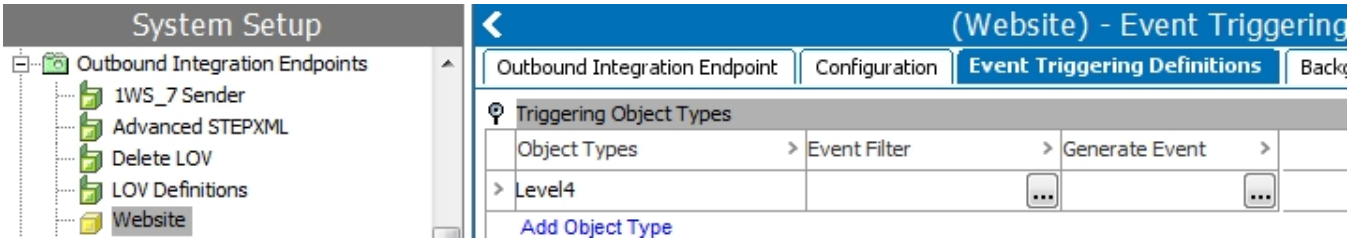
Using Business Rules to Generate Events on Derived Event Types

Once defined, derived events can be generated from JavaScript-based business actions and, when configuring an event-based OIEP, you can use business conditions to filter away events of certain types and actions to create new events based on others.

The filtering option can also be used to handle the problem with the REPUBLISH event described in the previous section. Instead of having a bulk update or business action generate REPUBLISH events that will be seen as core 'create' events, you can instead generate a derived event and then configure the relevant OIEP to discard all events that are not of this type.

Note: You can only use standard derived event types together with JavaScript-based business actions. Additional information on JavaScript syntax for STEP is available in the STEP API documentation.

The following example shows how you can set up a business rule to generate events on different derived event types. Apply the business rules to be used for the derived events on the **Event Triggering Definitions** tab, in the **Triggering Object Types** table.



The following derived event types are used in this example:

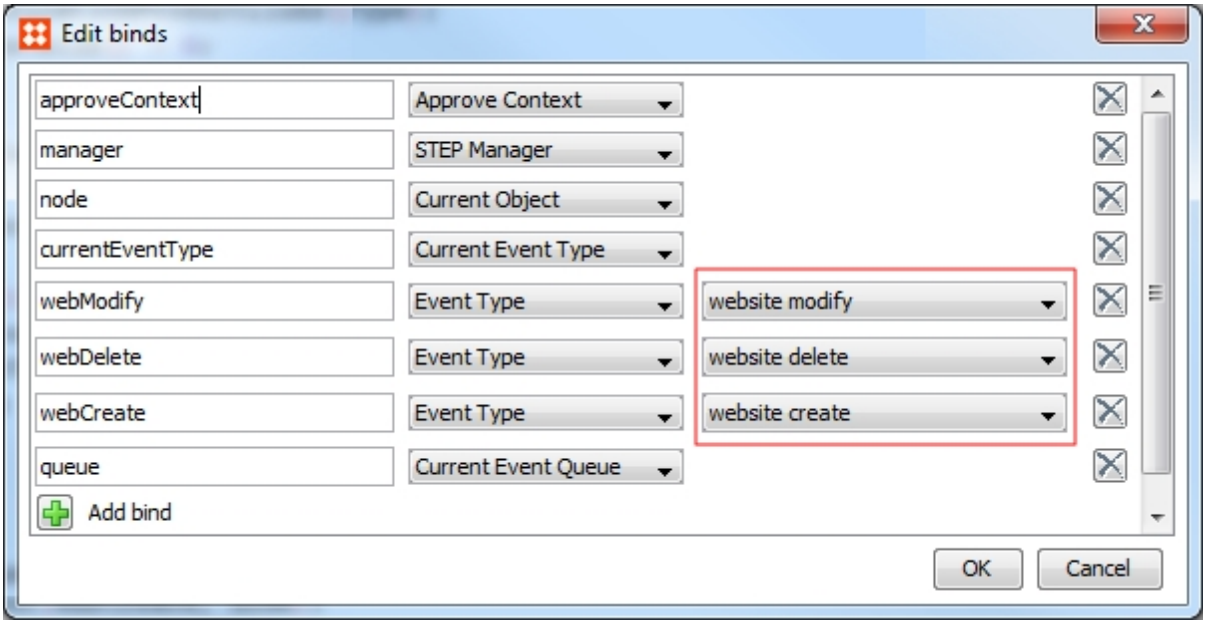
- A product is linked into a classification and approved for the first time. In this case, a derived event with the event type 'website create' is generated.
- A product is approved and linked into a classification, but data on the product is approved with modifications. In this case, a derived event with event type 'website modify' is generated.
- A product is approved and linked into a classification, but approved with a deletion of the product to classification link. In this case, a derived event with event type 'website delete' is generated.

Note: In the example, the product to classification link type is named 'Web Classification'. However, to use this business rule, you may have to correct the script to match a product-to-classification link type that exists in your system.

Set Up a Business Action

1. In **System Setup**, right-click a business rules setup group, and then select **New Business Action**.
2. Enter a **Name** and an **ID** for the business rule.
3. In the business rule editor, click **Valid Object Types**, and select the object types that you want the business action to be valid for.
4. On the **Business Rule** tab, click **Add New Business Action**, and then in the list, select Execute JavaScript.

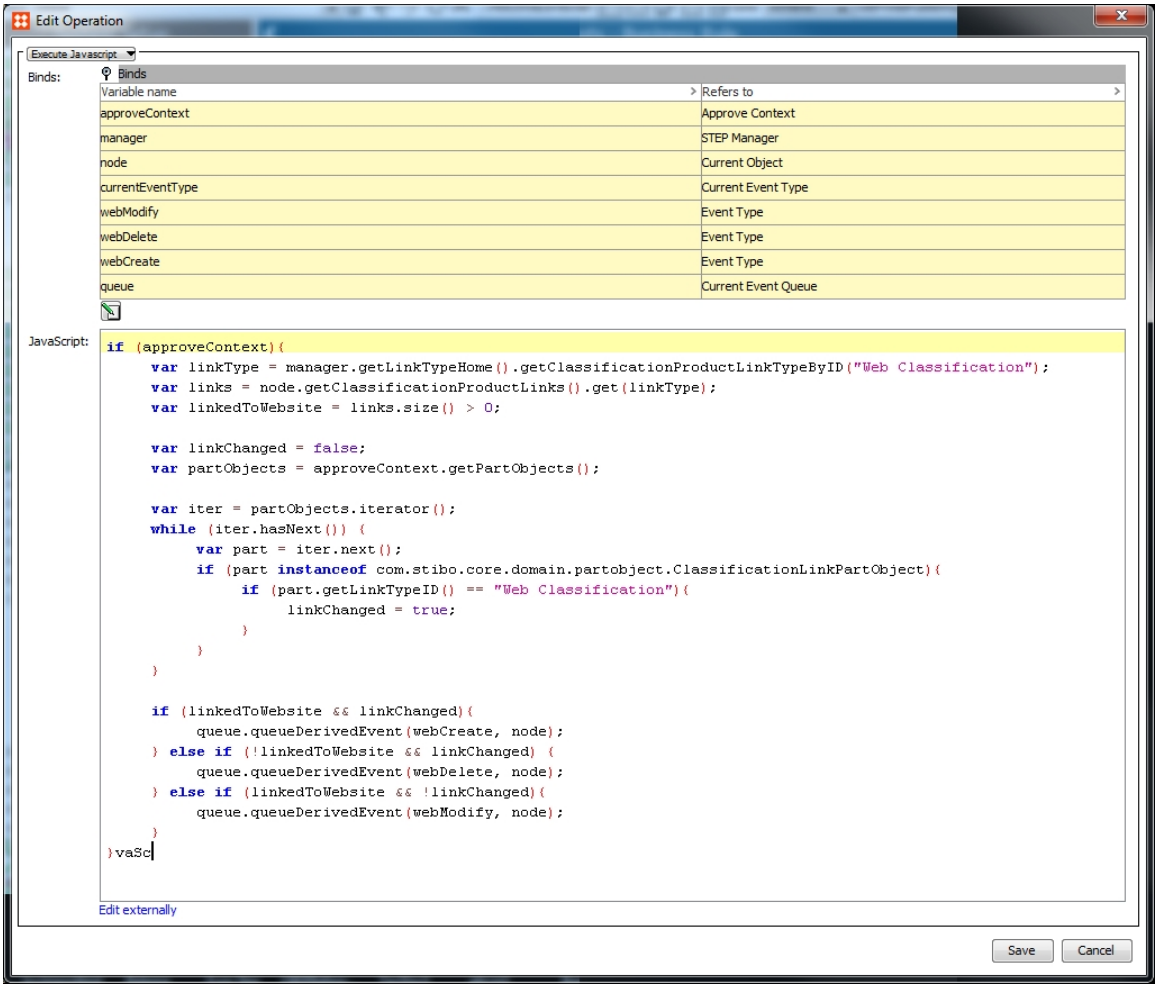
5. Click the **Edit Binds** icon.
6. Add the binds shown in the following screen-shot, and for each event types, select the specific derived event.



7. Copy and paste the following JavaScript code to create a derived event based on the approved change on the product.

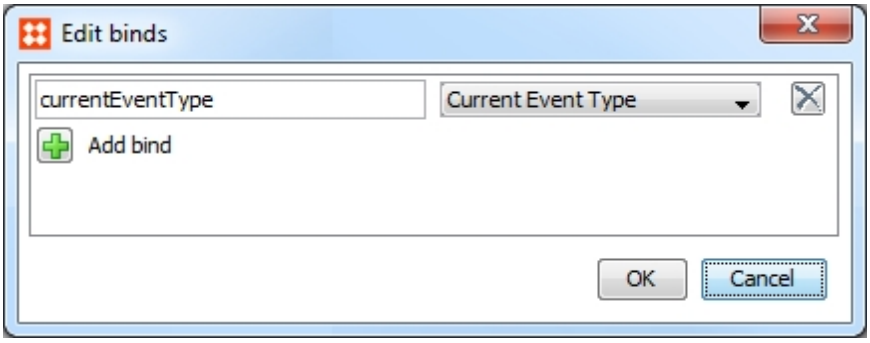
```
if (approveContext){
   var linkType = manager.getLinkTypeHome().getClassificationProductLinkTypeByI
D("Web Classification");

       var links = node.getClassificationProductLinks().get(linkType);
       var linkedToWebsite = links.size() > 0;
       var linkChanged = false;
       var partObjects = approveContext.getPartObjects();

       var iter = partObjects.iterator();
       while (iter.hasNext()) {
               var part = iter.next();
       if (part instanceof com.stibo.core.domain.partobject.ClassificationLinkPa
rtObject){
               if (part.getLinkTypeID() == "Web Classification"){
                       linkChanged = true;
               }
       }
       }

        if (linkedToWebsite && linkChanged){
               queue.queueDerivedEvent(webCreate, node);
       } else if (!linkedToWebsite && linkChanged) {
               queue.queueDerivedEvent(webDelete, node);
       } else if (linkedToWebsite && !linkChanged){
               queue.queueDerived
```

```
Event(webModify, node);
        }
}
```



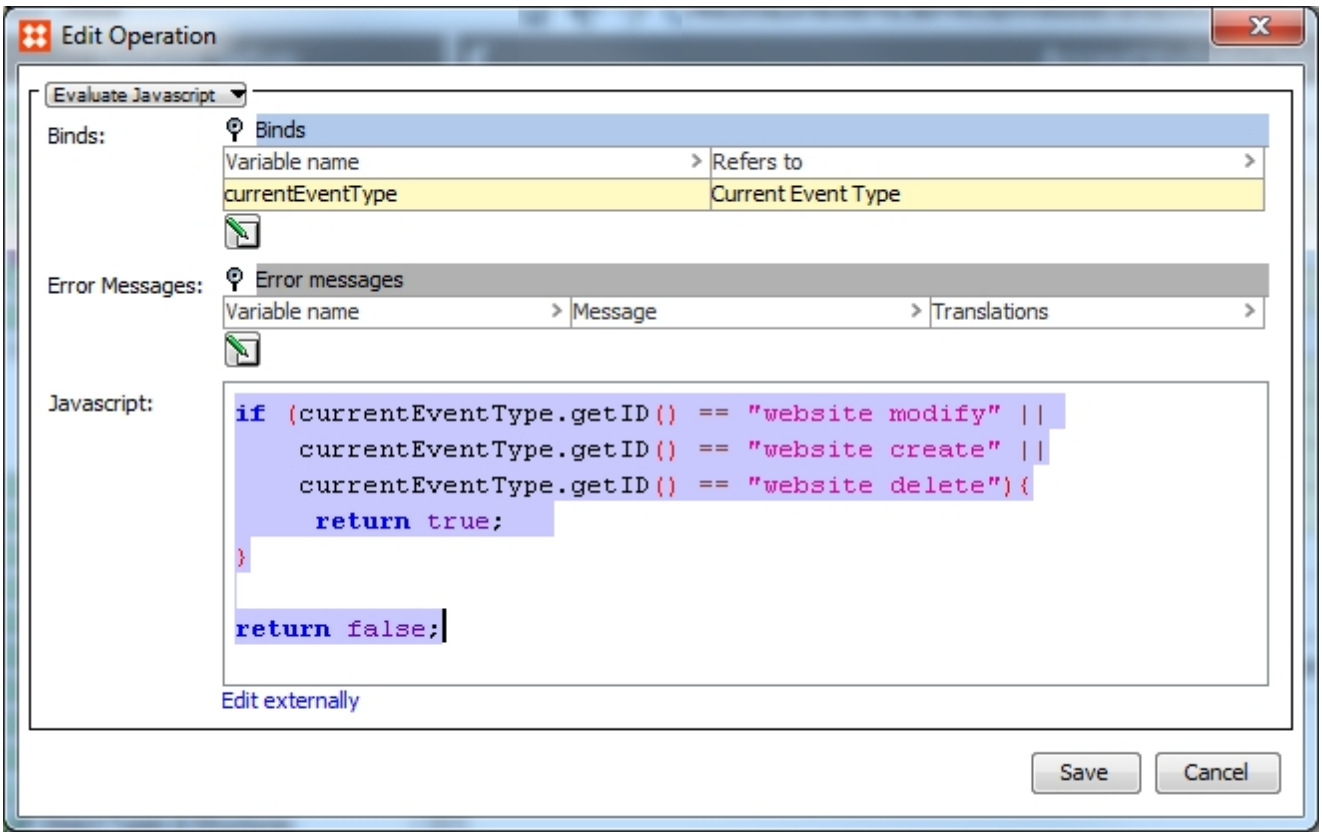
Set Up an Event Filter

1. In **System Setup**, right-click a business rules setup group, and then select **New Business Action**.
2. Enter a **Name** and an **ID** for the business rule.
3. In the business rule editor, click **Valid Object Types**, and select the object types that you want the business action to be valid for.
4. On the **Business Rule** tab, click **Add New Business Action**, and then in the list, select **Execute JavaScript**.
5. Click the **Edit Binds** icon, in the list select **Current Event Type**, and then enter a name.



6. Copy and paste the following JavaScript code.

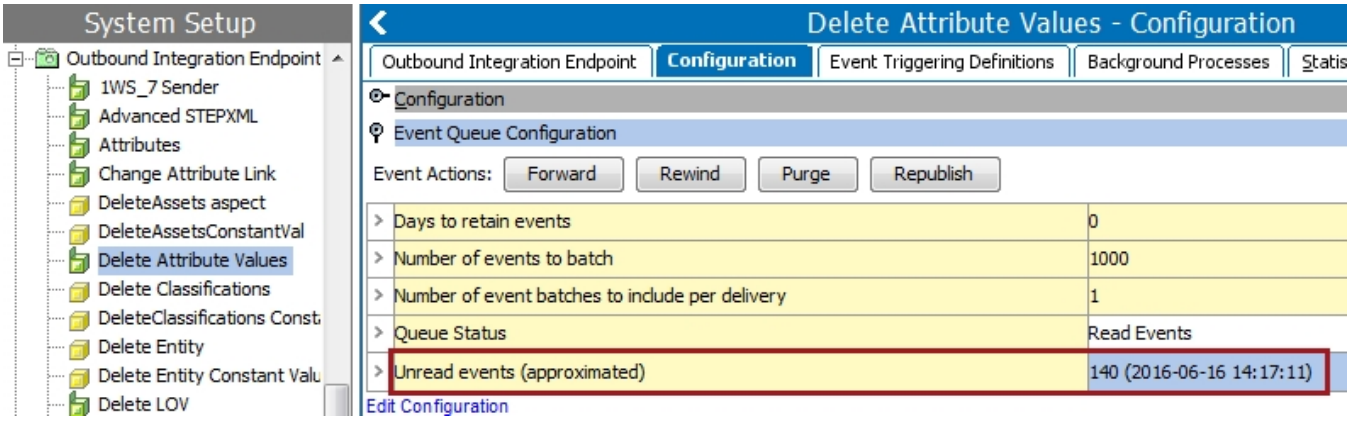
```
if (currentEventType.getID() == "website modify" ||
    currentEventType.getID() == "website create" ||
    currentEventType.getID() == "website delete"){
       return true;
}

return false;
```



Outbound Integration Endpoint Queued Events

You can view an estimate of the current number of unread events on an event-based OIEP in the Event Queue Configuration area 'Unread events (approximated)' field.

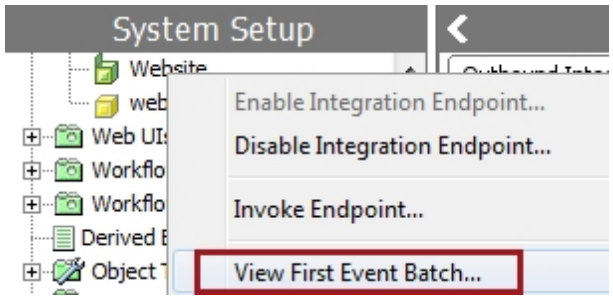
Click the 'Click to estimate' button to display the data.



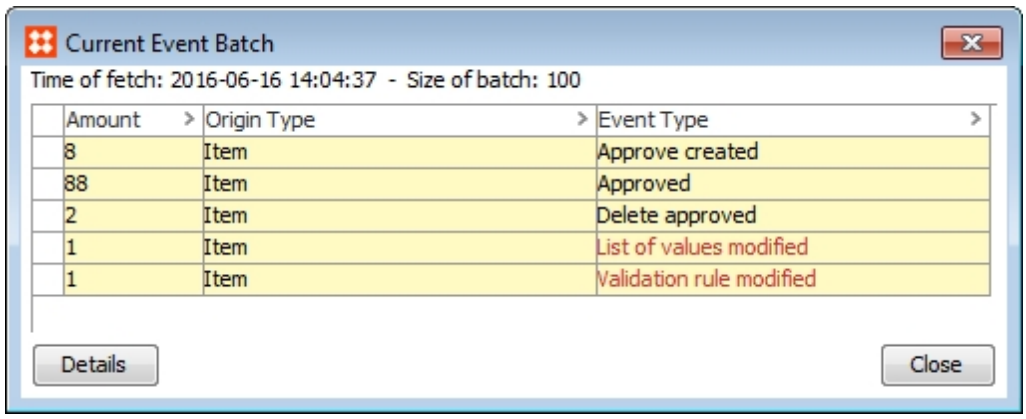
Note: Using the forward, rewind, purge, or republish events options causes the number of unread events on the queue to change.

To view more than just a count of the events, use the View First Event Batch option.

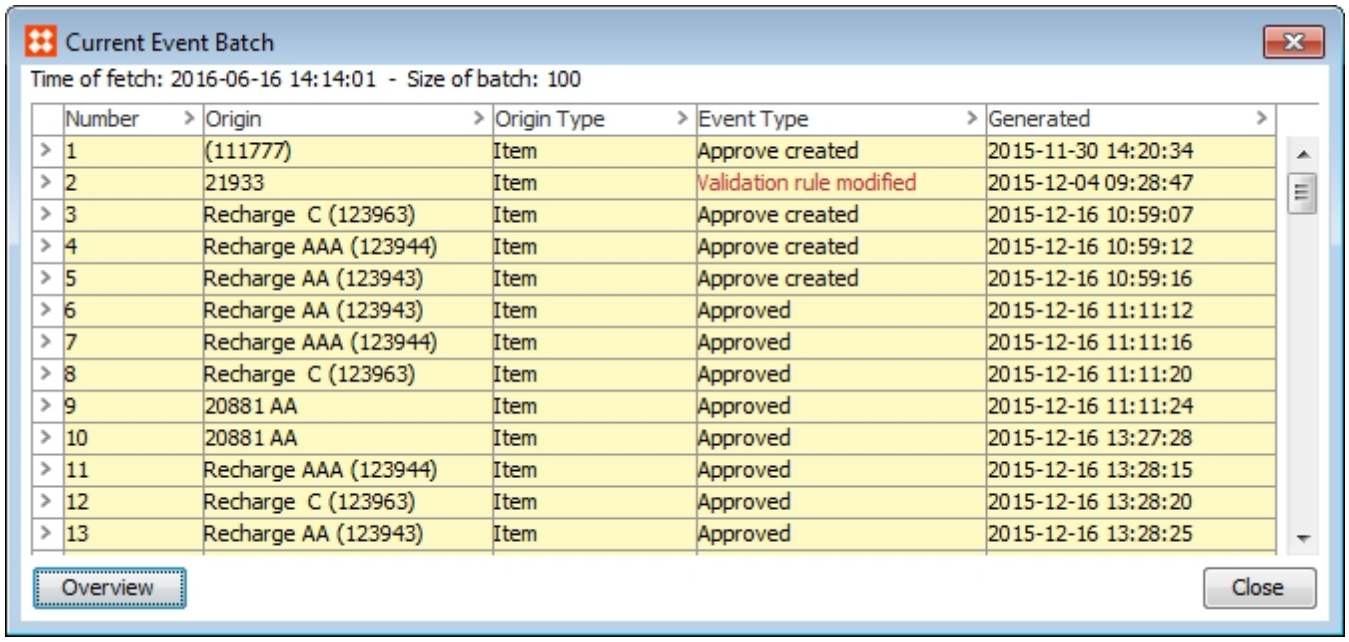
1. Right-click the OIEP and select View First Event Batch.



2. The Current Event Batch dialog displays the first 100 unread events on the queue. Derived events are represented by a number instead of by their ID.

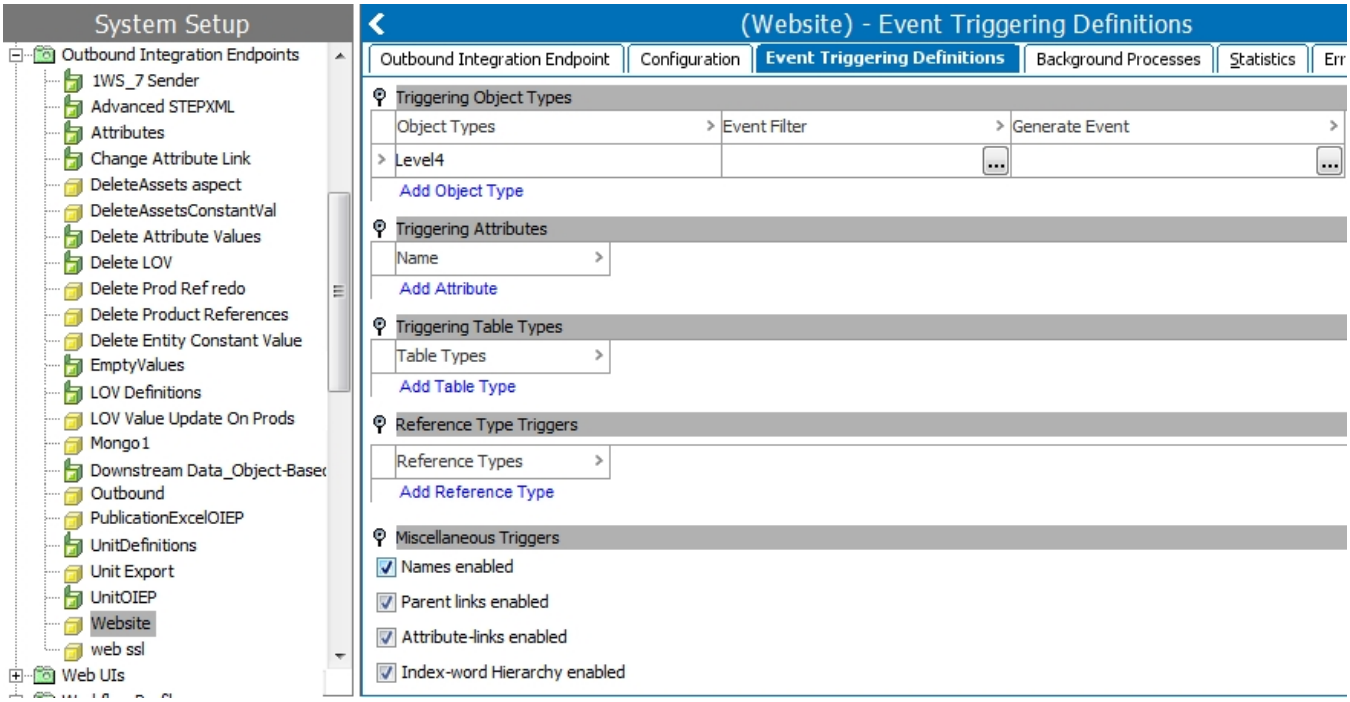


3. For more information about the events, click the Details button. Click the Overview button to return to the previous view.



Outbound Integration Endpoint Event Triggering Definitions

When Event Queue has been selected as the data source in the Outbound Integration Endpoint wizard, you can specify whether modification, deletion, or creation of specific object types triggers an event.



For objects that are under revision control, such as a product, the changes must be approved for the event to be triggered. If changes are made to an object that is not under revision control, such as an attribute, an event is released right away because approval is not required.

You can also specify business actions that apply derived events. For more information, see Outbound Integration Endpoint Derived Events.

Important: The event triggers must be specified before output templates can be configured for event-based outbound integration endpoints. Furthermore, an output template must be created for each trigger, otherwise the trigger will not output any data.

- Click the **Event Triggering Definitions** tab. Here you can add or edit triggers for object types, attributes, table types, reference types, and a selection of miscellaneous triggers. This tab is only available if you selected **Event Queue** as data source in the wizard.

Triggers can be specified for object types, attributes, table types, reference types, and miscellaneous elements. No default triggers exist, so any triggers that are required must be added manually.

Triggering Object Types for Filtering

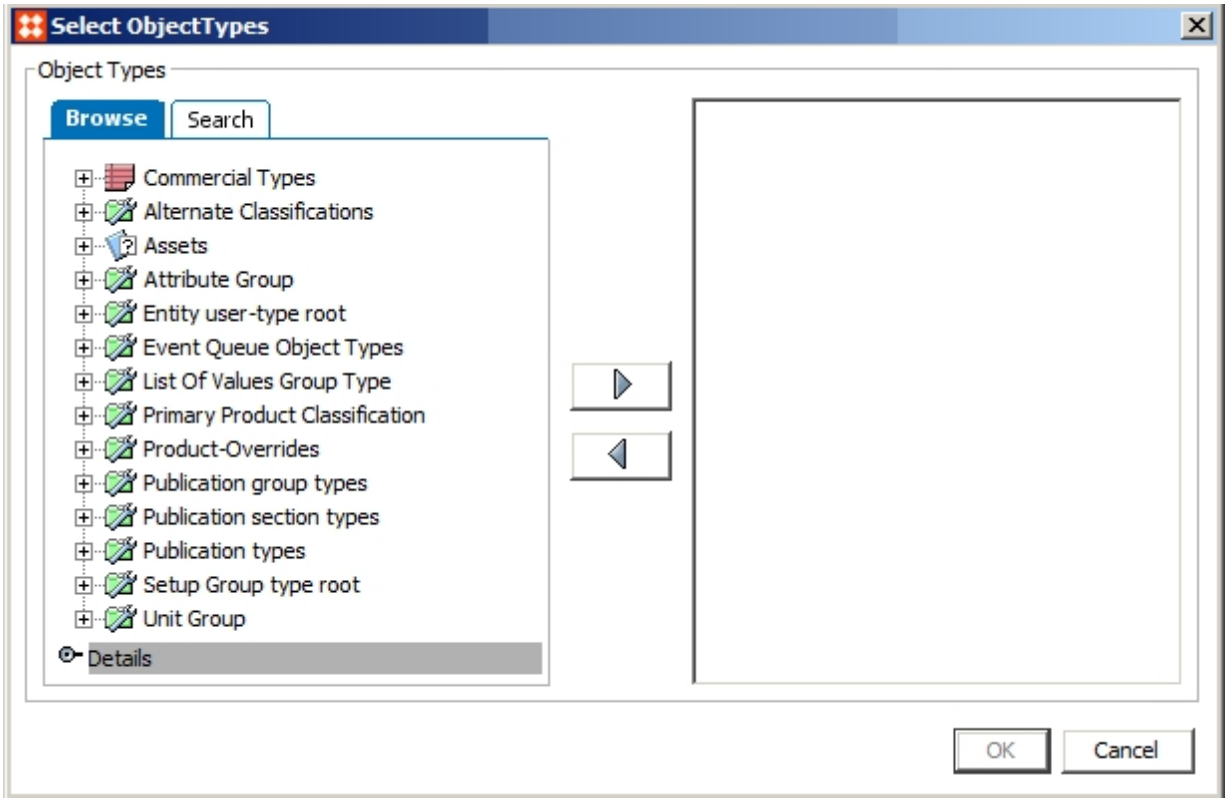
Consider the following points when setting up triggering object types:

- If functionality should vary by object type, a unique endpoint should be created for each object type.
- Common setup is to include at least one attribute or attribute group *unless* the integration endpoint handles System Setup data only, or Create / Delete events only.
- If standard object types (product, entity, classification) are selected as triggers and the output template includes a Modify event, at least one attribute or attribute group must be selected to trigger the Modify event.

Object Types

Events will only be registered for the object types selected in the 'Triggering Object Types' section. Notice that to register events on changes to configuration objects like attributes, the object type for these must be selected as well. If you are not using any business rules to filter and process events (functionality described below), all object types can be added in a single row.

- Click the **Add Object Type** link, then browse or search for the relevant object type. Only the most commonly used object types are displayed in the browsing area. However, you can search for other object types.
- System Setup data (references, attributes, units, etc.) can be selected.
- At least one object type must be selected.



Event Filter

Event filters can be used in place of a pre-processor. Event filters are business conditions that can be added for each row of the Object Type Triggering section and can be used for applying additional event filtering. If the condition evaluates to true, the event passes and is registered on the Event Queue, if the condition evaluates to false, the event is discarded (unless a corresponding derived event is generated).

For example, when a change happens on a parent object, but you only want to output the children, you can use a business condition to filter out the parent. Then use a business action (see the Generate Event section below) to trigger an export of the children.

Conditions used as event filters will typically be written in JavaScript since the built-in condition plugins, in most cases, will not provide the required flexibility.

For JavaScript based event filter conditions, the following available binds are of special relevance:

- **Event Type** lets you bind in a derived event so that you can check if the event currently being handed is of a specific derived event type.
- **Current Event Type** lets you test if the event currently being handed is a BasicEventType or a DerivedEventType.

  Assuming that 'Current Event Type' is bound to the variable 'currentEventType':

  ```
  if (currentEventType instanceof com.stibo.core.domain.eventqueue.BasicEventType) {
  // It is a core event
  }
  if (currentEventType.equals(com.stibo.core.domain.eventqueue.BasicEventType.Modify))
  {
  // It is a core Modify event
  }
  ```

  You can also test derived events and if you have created an 'Event Type' binding, test if current event is of a specific type. Assuming that 'Current Event Type' is bound to the variable 'currentEventType' and the derived event 'webModify' is bound to the variable 'webModify':

  ```
  if (currentEventType instanceof com.stibo.core.domain.eventqueue.DerivedEventType) {
  // It is a derived event
  }
  if (currentEventType.equals(webModify)) {
  // It is a derived WebModify event
  }
  ```

- **Approve Context** lets you compare an object in Main and Approved before the actual synchronization is carried out.It is important to understand that event filter conditions are evaluated immediately when events are generated. If the event is generated based on an approval, in the event filter condition, you will have access to the Approve Context. See the 'ApprovePlugin.ApproveContext' interface in the Public Java API JavaDoc.

Generate Event

Event generators are business actions that can be added for each row of the object type triggering section. The purpose of an event generator is to produce derived events based on other events (most often core events). Event generators will have access to the current event regardless of what the chosen event filter has to do with it (an event filter is not required). This means it can generate derived events from events discarded by the filter. As with event filters, the event generator logic is also executed immediately when an event is generated.

Generator actions must typically be formulated in JavaScript. In addition to the binds mentioned in the event filter section, the following are of special relevance for event generators.

- **Current Event Queue** lets you bind in the current event queue so that you can easily put new derived events on the queue (after filtering). Assuming that 'Current Event Queue' is bound to 'currentEventQueue', the derived event type 'webModify' is bound to the variable 'webModify', and 'Current Object' is bound to 'currentObject', an event can be generated as follows:
  ```
  currentEventQueue.queueDerivedEvent(webModify, currentObject);
  ```
- **Event Queue** lets you place an event on a queue different from current. use this binding if you produce events from actions not hooked in as generators.

Important: Beware of creating endless loops since any derived event generated by an event generator for the current event queue will pass through the object type filter and any event filters and event generators again.

Triggering Attributes for Filtering

You can set up an event-based OIEP so that it only registers events for objects of the object types selected in the Triggering Object Types section when values for specific attributes change. The 'Triggering Attributes' section is a positive list, so if no attributes are selected, events will not be registered when, for example, a value changes and the object holding the value is approved.

- Click **Add Attribute**, then browse or search for the relevant attribute(s) and/or attribute group(s). Only approved value changes in the selected attributes (or attributes within the listed groups) will trigger events, and then only if an object type on which the attributes are valid is selected as a triggering object type.
  - At least one attribute or attribute group must be selected, unless the integration handles ONLY System Setup data OR Create / Delete events.
  - When an attribute or attribute group is selected as a trigger, an output template must be configured with a Modify event or the endpoint will skip the event since it cannot find a corresponding template. For more information, see the **Outbound Integration Endpoint Output Template** section of the **Integration Endpoints for Data Exchange** documentation.

Be aware that you select both revisable and externally maintained attributes here. Changes to values for:

- Externally maintained attributes cause an event to be generated immediately.
- Revisable attributes cause an events to be generated when the object holding the values is approved.

Triggering Table Types for Filtering

Allows you to specify table types for which you want events registered when tables owned by objects of the type listed in the Triggering Object Types section change. Common setup is to use tables and this setting only for print output.

- Click **Add Table Type**, then browse or search for the relevant table type. Modifications to the System Setup configuration of the selected table types will trigger an event for the endpoint.

Reference Type Triggers for Filtering

Use reference / link types to define which reference / link types to list to for changes. This is a positive list and both revisable and externally maintained reference / link types can be added. If functionality should vary by object type, a unique endpoint should be created for each object type.

- Click **Add Reference Type**, then browse or search for the relevant reference type. Only approved changes in the selected references will trigger events (if an object type on which the reference is valid is selected as a triggering object type).

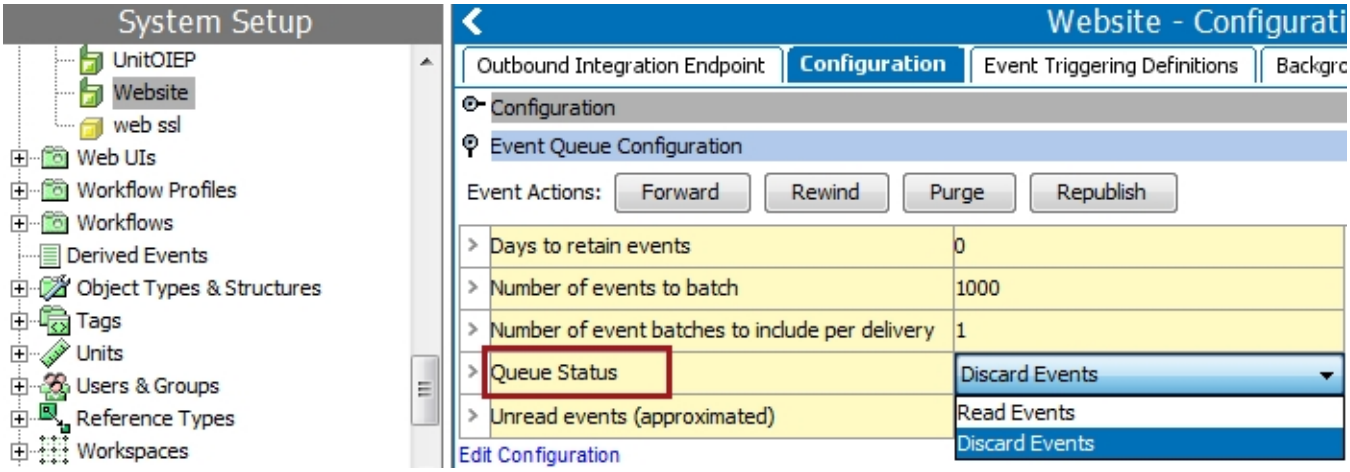
Miscellaneous Triggers for Filtering

If functionality should vary by object type, a unique endpoint should be created for each object type. On the **Event Triggering Definitions** tab, the following miscellaneous triggers are available:

- **Names enabled**: Triggers an event when approved changes are made to STEP Names of triggering object types. An output template must be configured with a Modify event or the endpoint will skip the event since it cannot find a corresponding template.
- **Parent links enabled**: Triggers and event when approved changes are made to the parentage of triggering object types. For example, when a product or asset is moved into another parent folder. An output template must be configured with a Modify event or the endpoint will skip the event since it cannot find a corresponding template.
- **Attribute-links enabled**: Triggers an event when attribute hierarchy links are changed on an object of the types selected in the Object Type triggering section. For example, when an attribute is linked to a product or classification.
- **Index-word hierarchy enabled**: Triggers an event when an index word changes or is added on a triggering object type. Common setup is to use indexes and this option for print output.

Outbound Integration Endpoint Event-Based Queue Status

1. In **System Setup**, locate an event-based outbound integration endpoint and view the **Configuration** tab.
2. Open the **Event Queue Configuration** flipper.
3. In **Queue Status**, select the desired option.



For information about how the Endpoint Status and Queue Status settings work together, see the **Event-Based Outbound Integration Endpoint Status and Queue Status** section of the **Integration Endpoints** documentation.

Event-Based Outbound Integration Endpoint Status and Queue Status

For event-based integration endpoint, the endpoint status (enabled / disabled) setting and the queue status (read events / discard events) setting are independent of each other, but work together to determine how data is processed.

This means that, for example, if a downstream system that receives deliveries from an OIEP is offline, you can disable the endpoint so that you do not keep publishing data, but by leaving it set to read events, the endpoint continues to register events in STEP for data to be published once the external system is back online.

Common setup includes the following combination of queue status and endpoint status settings:

- **Enabled + Read Events = Active**. Use this setting for an active endpoint that should deliver data to downstream systems.
- **Disabled + Read Events = Paused**. Use this setting to temporarily disable the feed, while retaining access to events being generated, even while the endpoint is disabled.
- **Disabled + Discard Events = Inactive**. Use this setting when no new events should be processed (now or later) and data should not be delivered downstream.
- **Enabled + Discard Events = Transition**. Not commonly used, but can be employed when one endpoint will take over for another, or prior to running a bulk update process that should not be sent downstream. Allows the old endpoint to process queued events, but not generate any new ones as new events should be set to queue on the new endpoint (or discarded if bulk update should not be sent out).

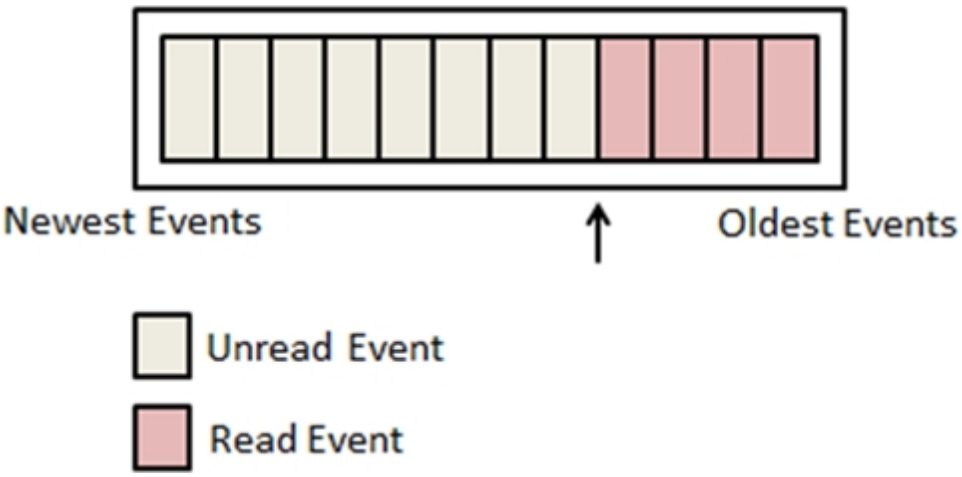
For information about setting the enabled / disabled parameter, see the **Integration Endpoint Status** section of the **Integration Endpoints** documentation.

For information about setting the Read Events / Discard Events parameter, see the **Outbound Integration Endpoint Event-Based Queue Status** section of the **Integration Endpoints** documentation.

Event-Based Outbound Integration Endpoint Forward, Rewind, Purge, and Republish

If the receiving system loses one or more messages, you can rewind the event queue and reprocess already processed events from a specific point in time.

For example, if the queue is configured to retain events after they have been read, read events can be distinguished from unread events, as illustrated by the arrow in the following image.



The Forward and Rewind actions essentially lets you move this arrow:

- Forward action marks unread Events as read (moving the pointer left)
- Rewind action marks read Events as unread (moving the pointer right)

For information on the number and types of unread events, see the **Outbound Integration Endpoint Queued Events** section of the **Outbound Integration Endpoints** documentation.

To forward, rewind, purge, or republish events:

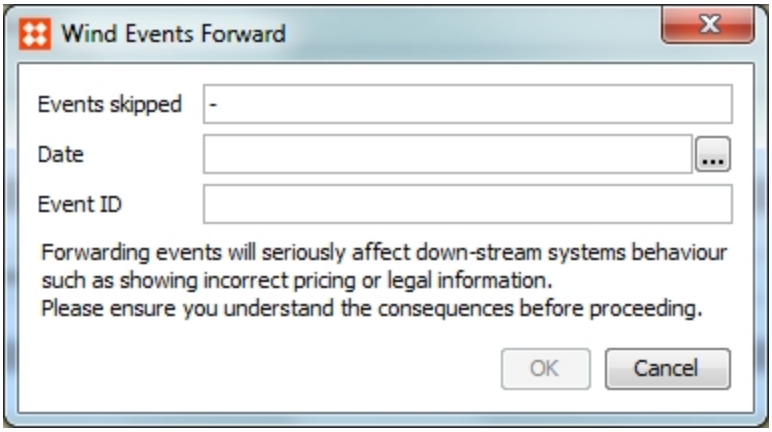
1. In **System Setup**, locate the relevant outbound integration endpoint.
2. On the **Configuration** tab, in the **Event Queue Configuration** area, click the **Forward**, **Rewind**, **Purge**, or **Republish** button. Depending on which event action you choose, complete one of the following tasks:

Forward Events

1. In the **Wind Events Forward** dialog, click the ellipsis button **(…)** in the **Date** field.
2. In the **Date Picker**, select the date from which you want to skip events.

   When you have selected a date, the **Events skipped** field shows the number of events that will be skipped.

Note: Event IDs are not shown in the UI but are read from the database.

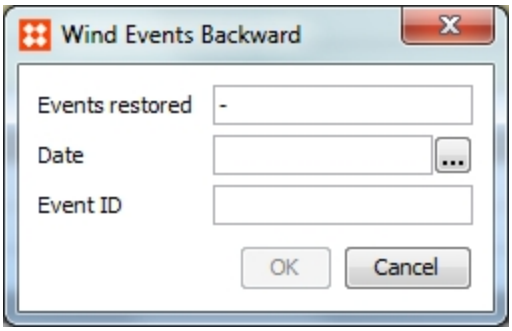


Rewind Events

1. In **Wind Events Backward** dialog, click the ellipsis button **(…)** in the **Date** field.
2. In the **Date Picker**, select the date from which you want to restore events.

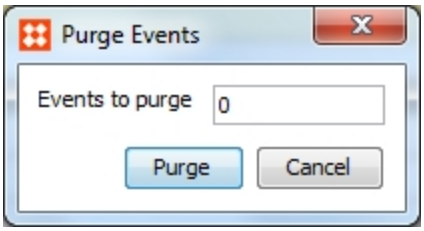
   When you have selected a date, the **Events restored** field shows the number of events that will be restored.

Note: Event IDs are not shown in the UI but are read from the database.



Purge Events

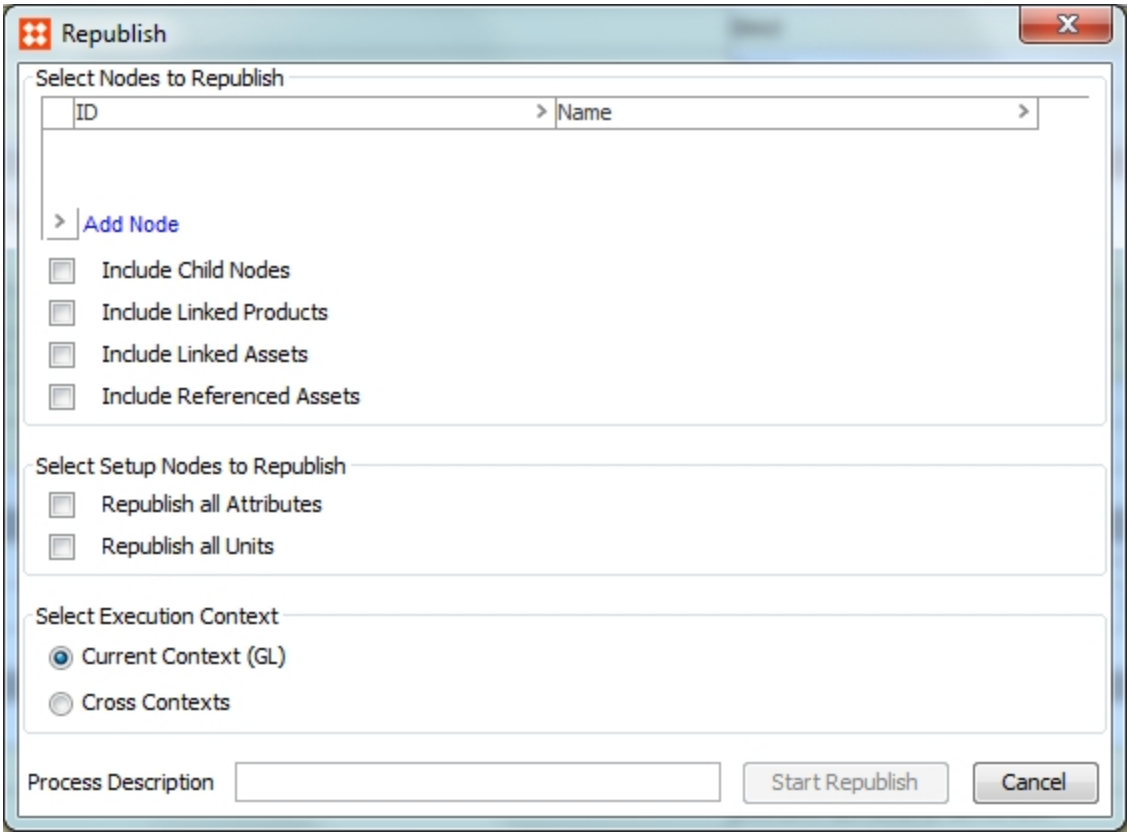
1. In **Event Actions**, click **Purge**.
2. The Purge Events dialog, the number of **read** events that you can delete from the system is displayed.
3. Click **Purge** to delete the events.



Republish Events

The Republish action generates events on products, classifications, or assets in one or more selected hierarchies. Common setup is to use this option for on-demand creation of republish events. For example, to generate events for all objects in a hierarchy that has never been published by an event queue.

1. In **Event Actions**, click **Republish** to display the Republish dialog.



2. Click **Add node** and then browse or search for the hierarchy you want to republish.
3. Select the objects you want to republish. You have the following options:
   - **Include Child Nodes** republishes all children of the selected object.
   - **Include Linked Products** republishes all products linked to the selected object.
   - **Include Linked Assets** republishes all assets linked to the selected object.
   - **Include Referenced Assets** republishes all assets referenced by the selected object.
4. Select the setup nodes you want to republish. You have the following options:
   - **Republish All Attributes** republishes all attributes linked to the selected object.
   - **Republish all Units** republishes all units linked to the selected object.
5. Select the Execution Context. You have the following options:

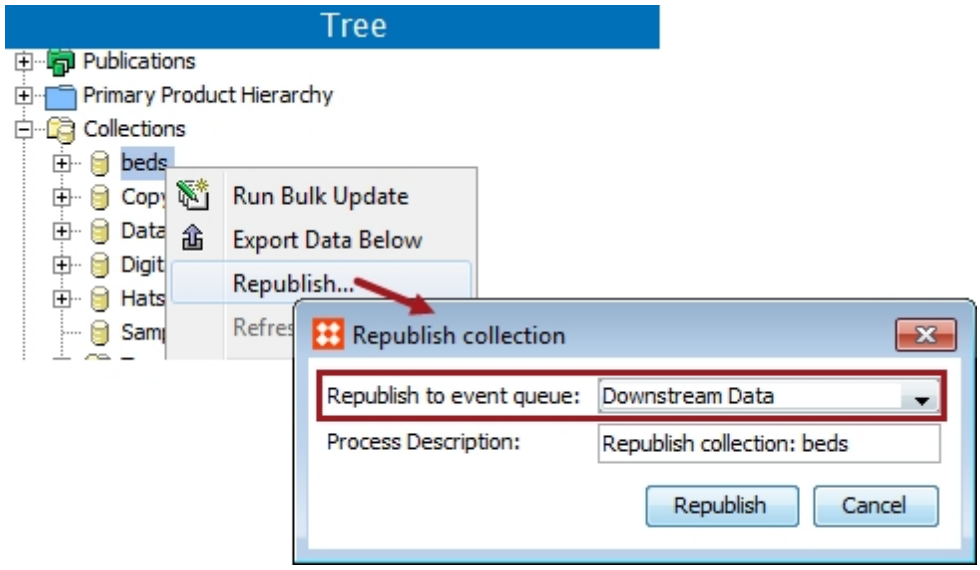
- Select **Current Context** to republish in the current context. (shown in parenthesis)
- Select **Cross Contexts** to republish child nodes in all contexts.

6. In **Process Description** enter a name for the process. This is a mandatory field.
7. Click **Start Republish**. Events are triggered on objects that match the selection.

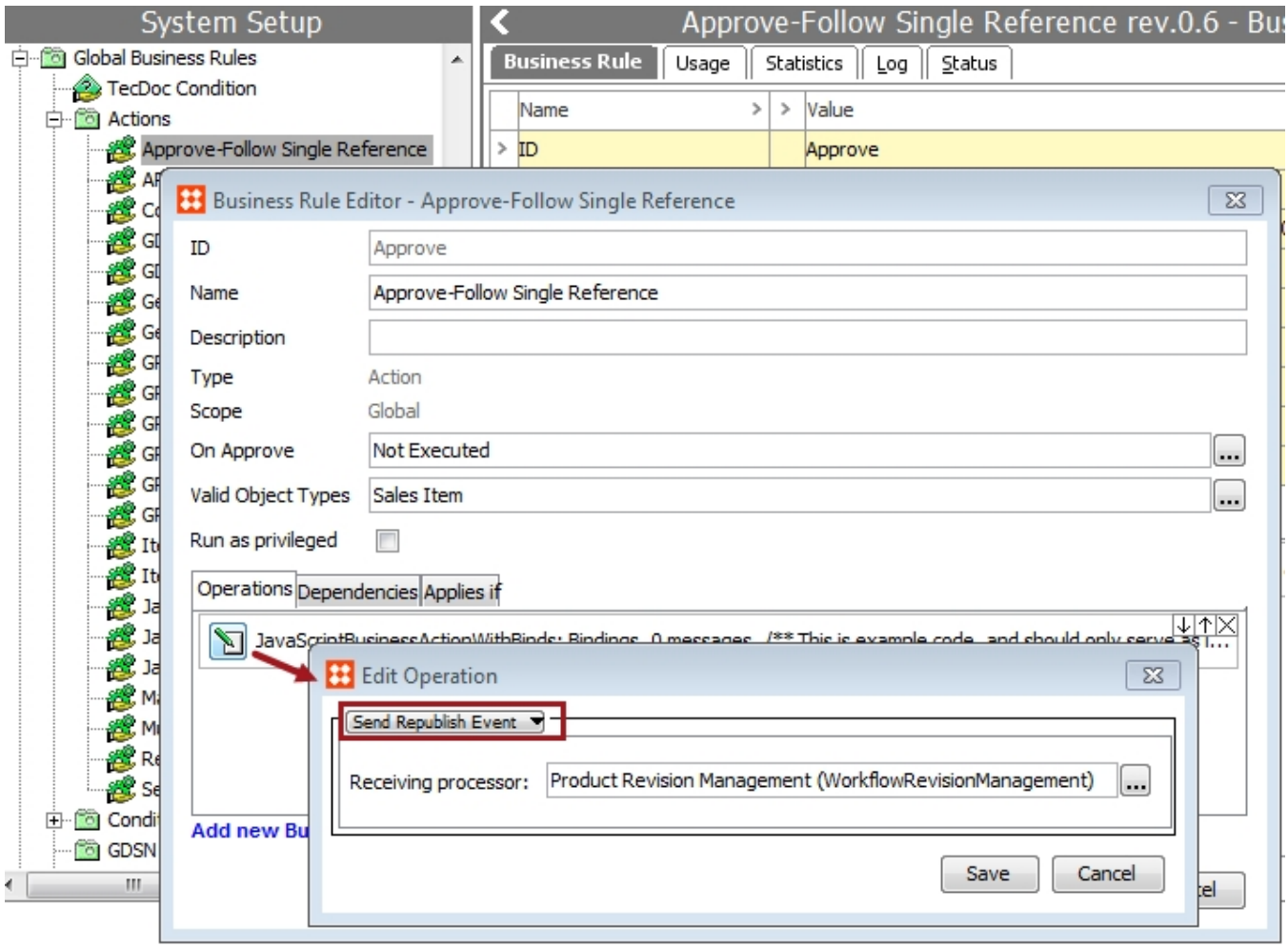
Republish Manually

Additionally, republish events can be generated manually in STEP Workbench in two ways:

- From a collection



- Via a dedicated bulk update / business action plugin


System Setup
Global Business Rules
TecDoc Condition
Actions
Approve-Follow Single Reference
Approve-Follow Single Reference rev.0.6 - Bus
Business Rule
Usage
Statistics
Log
Status
Name
Value
ID
Approve
Business Rule Editor - Approve-Follow Single Reference
ID
Approve
Name
Approve-Follow Single Reference
Description
Type
Action
Scope
Global
On Approve
Not Executed
Valid Object Types
Sales Item
Run as privileged
Operations
Dependencies
Applies if
Edit Operation
Send Republish Event
Receiving processor:
Product Revision Management (WorkflowRevisionManagement)
Save
Cancel
Add new Bu
GDSN

Event-Based Outbound Integration Endpoint Examples

This section includes event-based OIEP configuration examples for three semi-advanced to advanced cases:

- OIEP Example Publishing of Leaf Products with Inherited Data
- OIEP Example Upward Inheritance
- OIEP Example Publishing of Products Linked Into Specific Classification Hierarchy

OIEP Example Publishing of Leaf Products with Inherited Data

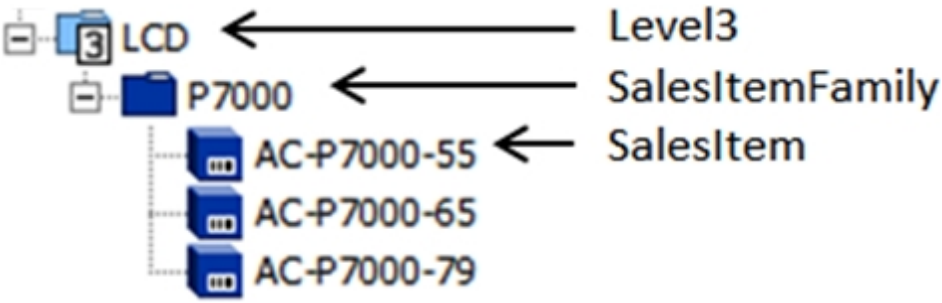
Problem: You only want to publish the leaf product level to a downstream system and the leaves inherit data from ancestors, so events will not be generated for the leaves when inherited data changes.

Solution: When values are inherited from the parent or grandparent level, use Event Filters and Event Generators to ensure that:

- events generated for the parent or grandparent level are discarded
- derived events are generated for all descendant leaf level objects that exist in Approved mode (have been published earlier)

Important: The following scripts are only an example and should not be used as-is without thorough testing.

Using the following object types:



Use the following triggering object types setup:

Triggering Object Types

Object Types	Event Filter	Generate Event
Sales Item		
Level 3, Sales Item Family	LP Always False (LPAlwaysFalse)	LP Republish Sales Item Descendants (

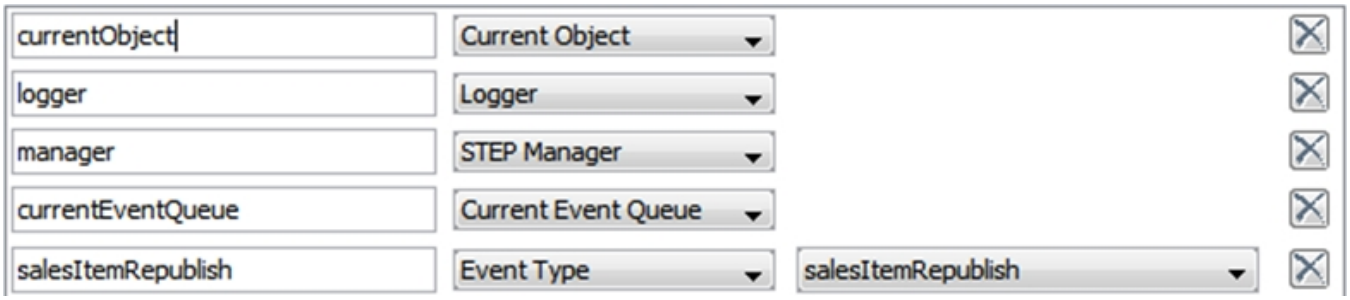
Add Object Type

No filters or generators are used for Sales Items, so events generated for this object type always get registered on the queue.

Use the following logic for the grandparent and parent levels (Level 3 and Sales Item Family):

Event Filter - No Events are registered for objects of these object types, but we still need to catch them to be able to generate derived events for descendant Sales Items. The event filter should be a JavaScript business condition that always evaluates to false, as shown below:

Event Generator - The business action for event generation is a little more complex and uses the following binds (logger is optional):



Notice the 'Event Type' bind to a derived event named 'salesItemRepublish'. This is the event that will be generated for descendant Sales Items and it must be defined in advance via the 'Derived Events' node in System Setup.

Important: The following script is only an example and should not be used as-is without thorough testing.

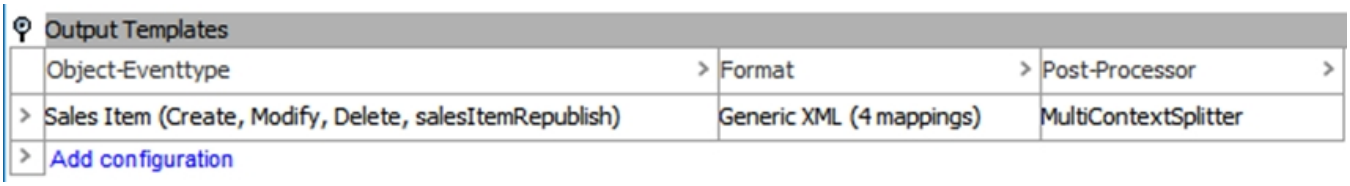
The script to generate derived events for approved Sales Items could look as follows:

```
function prodIsInApp(prodID) {
var appProd;
manager.executeInWorkspace("Approved", function(appManager) {
appProd = appManager.getProductHome().getProductByID(prodID);
});
return (appProd != null);
}
function generateEventForApprovedLeafSalesItems(ancestor) {
```

```
var children = ancestor.getChildren();
for (var i = 0; i < children.size(); i++) {
var current = children.get(i);
if (current.getObjectType().getID() == "SalesItem") {
if (prodIsInApp(current.getID())) {
logger.info("Generating 'salesItemRepublish' event for approved descendant Sales Item '" +
current.getID() + "'.");
currentEventQueue.queueDerivedEvent(salesItemRepublish, current);
}
}
else generateEventForApprovedLeafSalesItems(current);
}
}
generateEventForApprovedLeafSalesItems(currentObject);
```

The script uses as recursive function 'generateEventForApprovedLeafSalesItems' to find descendent Sales items that are in the Approved workspace.

Output Template - Using the configuration described above, you will need an output template for Sales Items and the different types of possible events. This can be one or multiple lines dependent on the need for different messages for the different events.



OIEP Example Upward Inheritance

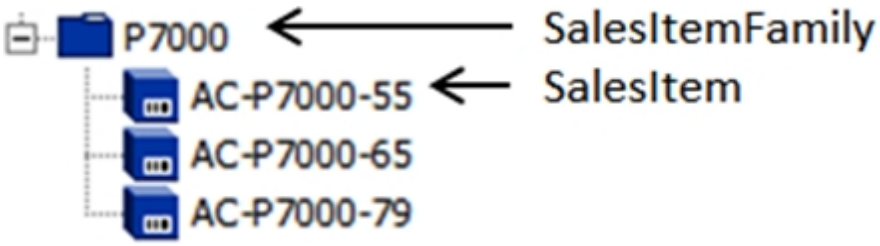
Problem: For outbound integrations where not only the leaf level is published, sometimes data from children is used on parents, for instance, via calculated attributes. When the children change, the parents are not automatically republished.

Solution:

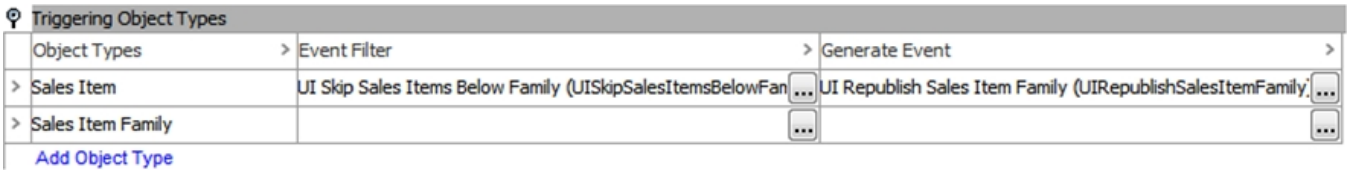
- If children get exported along with the parent, whenever an event is generated for a child with appropriate parent, catch it, discard it, and generate a derived event for the parent instead.
- If children are not published with parents, do not discard the children events.

Important: The following scripts are only an example and should not be used as-is without thorough testing.

The first option is described below, using the following object types:



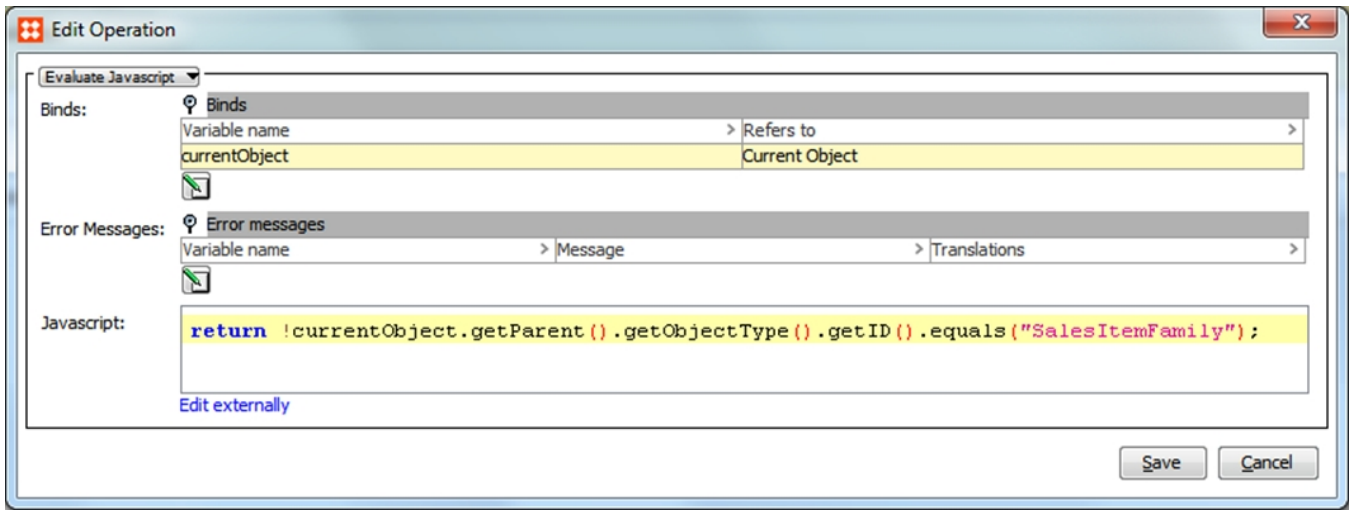
Use the following triggering object types setup:



No Filter or Generator is used for the Sales Item Family Object Type as all Events should pass for this type.

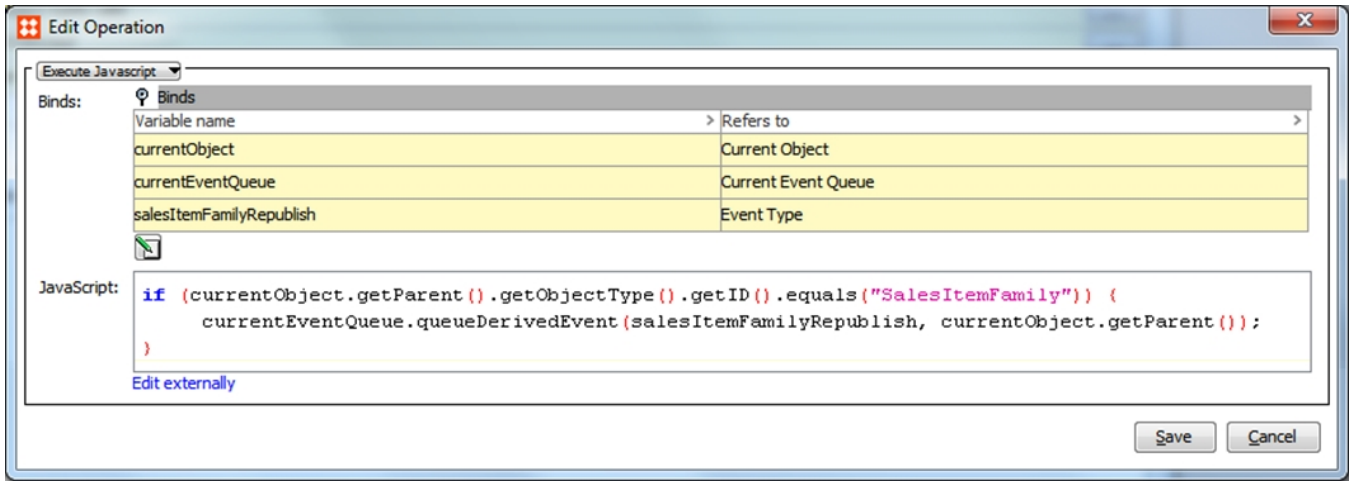
Use the following logic for the Sales Items:

Event Filter - If Sales Items can be children of both Sales Item Family and other types, a filter that checks the parent object type is required. The Script below discards events for Sales Items that are children of Sales Item Family objects:



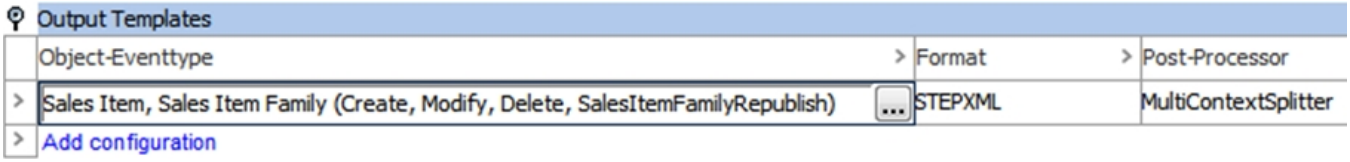
If Sales Items can only exist below Sales Item Family objects, instead an 'Always False' condition like the one used in **OIEP Example Publishing of Leaf Products with Inherited Data** can be used instead.

Event Generator - This business action script is also fairly straight forward, notice that it is assumed that a 'Derived Event' with ID 'salesItemFamilyRepublish' has already been defined in STEP.



If Sales Items can only exist below Sales Item Family objects, the 'if' statement can be left out.

Output Template - Using the configuration described above, you will need one or more output templates that cover both Sales item and Sales Item Family, and the different types of possible events must be defined. Below, a single output template is used.

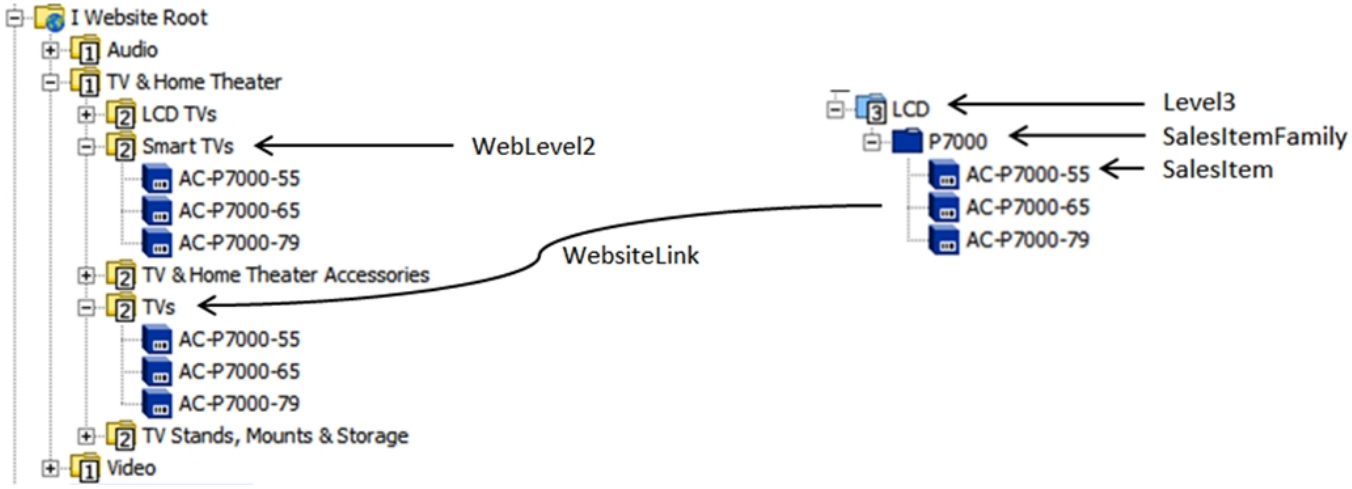


OIEP Example Publishing of Products Linked Into Specific Classification Hierarchy

Problem: Sometimes an external system is only interested in data about products linked into a specific classification hierarchy (e.g., a website hierarchy) and needs different types of messages depending on whether a product has just been linked into the hierarchy ('create' to the external system), is in the hierarchy and has been edited ('modify' to the external system) or has been taken out of the hierarchy (remove / delete' to the external system).

Assumptions: For this example the setup shown below is used and a number of assumptions are made.

- Only leaf level (Sales Item) can be linked into the classification hierarchy and only leaves should be published.
- The Product to Classification Link Type ('WebsiteLink') is only used for this specific website and all Sales Item objects in Approved that has the link should be on website.



Solution: A naive approach to this problem could be to simply have an event filter condition hooked in for Sales Items that only let events for the objects with a WebsiteLink pass. The problem with this approach is that it would not allow you to detect the cases where a link has just been removed, so you would not be able to communicate this to the external system. Also, you would not be able to differentiate between cases where the link has just been added ('create' to external system) and cases where a Sales Item already linked into the classification hierarchy has been modified.

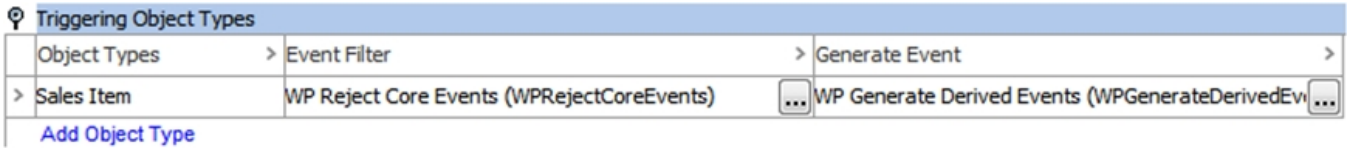
A better approach is to use three derived events with IDs 'webAdd', 'webModify', and 'webRemove'. The conditions for triggering the different events are listed below:

- **webAdd** - Sales Item with WebsiteLink is approved for the first time; earlier approved Sales Item where WebsiteLink has been added is approved
- **webModify** - Sales Item with WebsiteLink has been modified and is approved; externally maintained data for Sales Item with WebsiteLink in Approved is modified

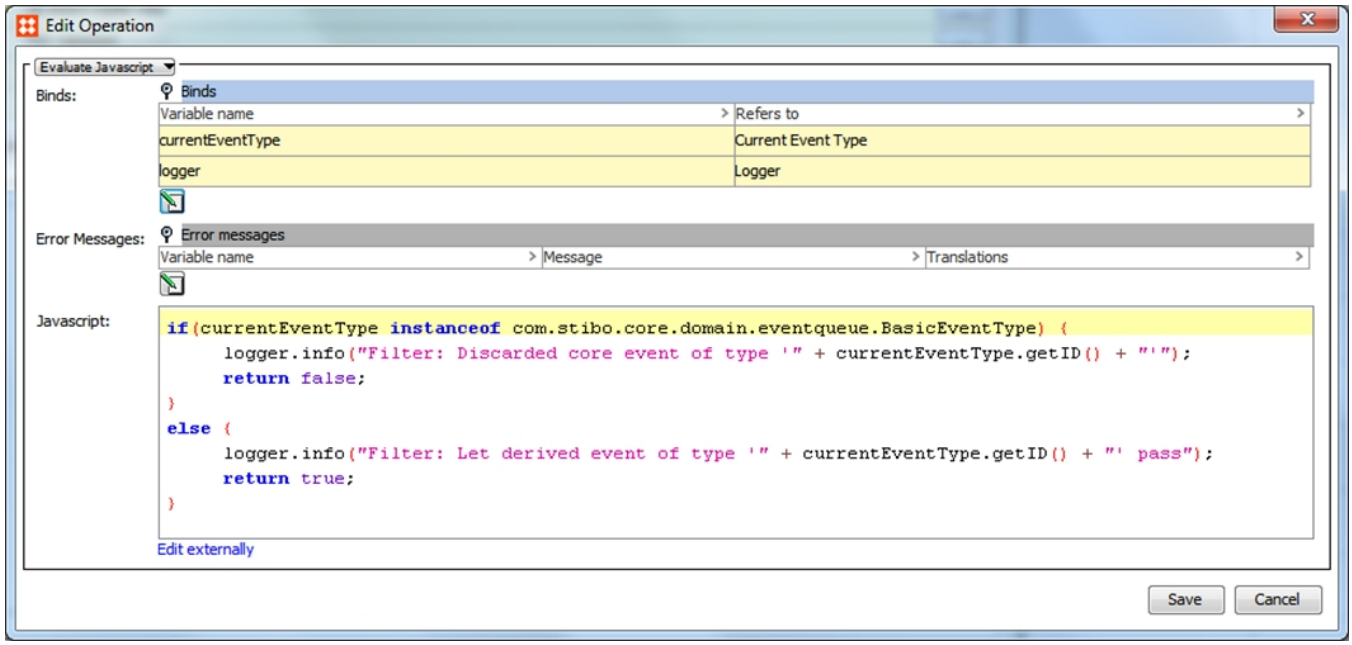
- **webRemove** - Earlier approved Sales Item where WebsiteLink has been removed is approved; Sales Item with WebsiteLink is approve deleted (core Delete Event)

Important: The following scripts are only an example and should not be used as-is without thorough testing.

A setup for this case could use the following triggering object types:



Event Filter - As data will be published based on derived events alone, all core events for Sales Items should be discarded. The script below does this and also, for debugging purposes, logs to the main Java log.

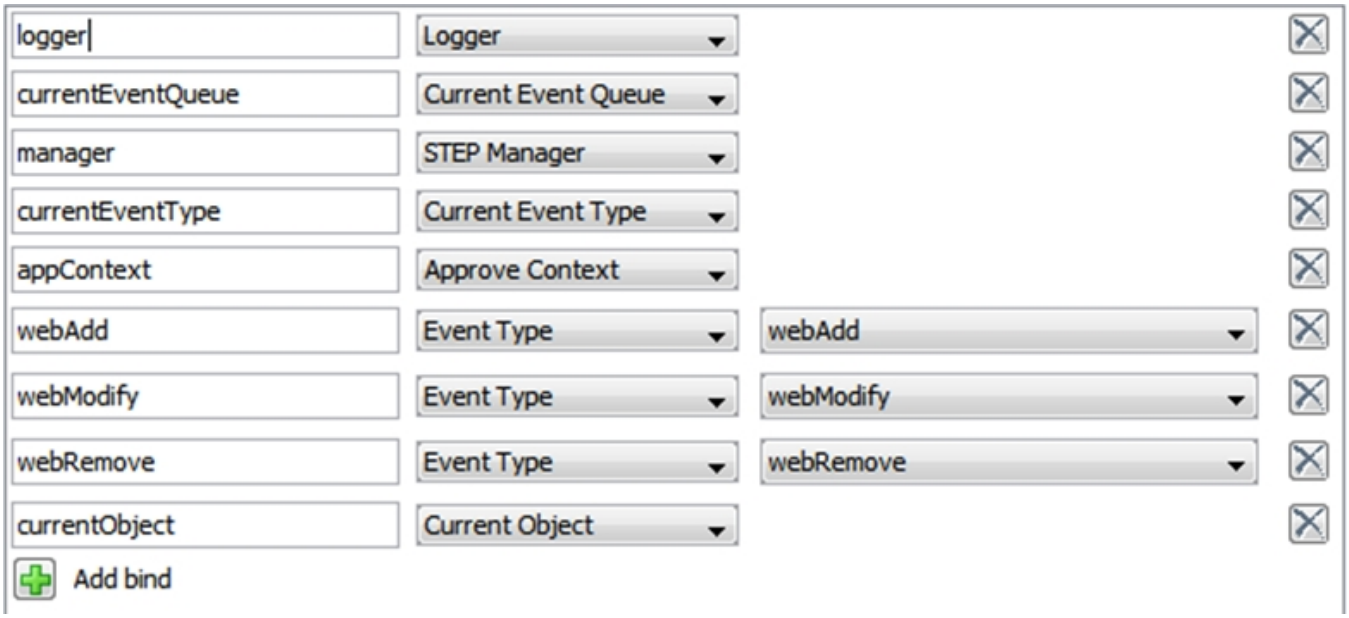


If logging is not required, the script could be:

```
return !(currentEventType instanceof com.stibo.core.domain.eventqueue.BasicEventType);
```

Event Generator - This logic is much more complex as it will have to perform the analysis to figure out which type of derived event should be generated. An example without logging is shown below. Notice that the business action should only generate derived events based on core events. If this check is not in place, an endless loop will be created.

The script uses the following bindings:



Keeping in mind that the following script is strictly to illustrate the solution, and can be difficult to read, this list outlines the intended logic:

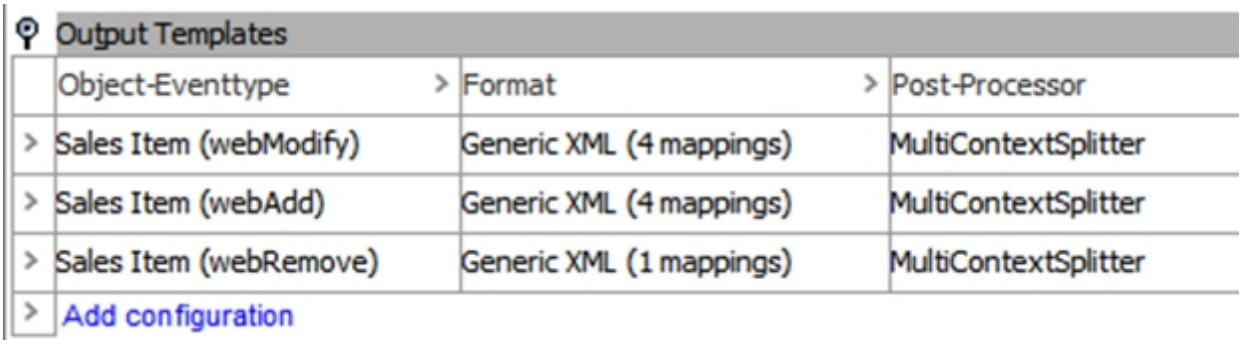
- Check if current event is a core event, do nothing if it is a derived event.
- Check if the Approve Context is available. If so, the event is generated based on an approval. If not, the event is based on some other change, for example, externally maintained data being modified.
- If Approve Context is available, use it to get hold of current object in both Main and Approved.
- Check if current core event is a Delete.
- Check for WebsiteLinks in Main and Approved.
- If current event is a Delete, and current object has a WebsiteLink in Approved, generate a derived 'webRemove' event.
- If current object has WebsiteLink in both Main and Approved, generate a derived 'webModify' event.
- If none of the above, the Approve Context is used to figure out whether the addition or removal of a WebsiteLink is currently being approved (is a part object). Notice here, that currently you cannot get the link type directly from a ClassificationLinkPartObject and thus, the target classification object type is used to figure out if the link type is correct. Also, because you could risk having a situation where the target classification does not exist in Approved and you therefore would not get the link synchronized even though it is in the part object set, it is checked that the classification exists there.
- If WebsiteLink exists in Main and is to be approved, generate a derived 'webAdd' event.
- If WebsiteLink exists in Approved and is to be approved, generate a derived 'webRemove' event.
- If no Approve Context is available, check whether current object exists in Approved with WebsiteLink. If it does, generate derived 'webModify' event.

```
var prodClassLinkTypeID = "WebsiteLink"
var ClassObjTypeID = "WebLevel2";
```

```
var linkType = manager.getLinkTypeHome().getClassificationProductLinkTypeByID(prodClassLink
TypeID);
//Utility function to check whether a Classification exists in Approved
function classInApp(classID) {
var res;
manager.executeInWorkspace("Approved", function(appManager) {
res = appManager.getClassificationHome().getClassificationByID(classID) != null;
});
return res;
}
if(currentEventType instanceof com.stibo.core.domain.eventqueue.BasicEventType) {
var appProd;
//Event generated based on an approval
if(appContext) {
var mainProd = appContext.getMainNode();
appProd = appContext.getApprovedNode();
var stepDelete = currentEventType == com.stibo.core.domain.eventqueue.BasicEventType.Delet
e;

var linkInMain = !mainProd.getClassificationProductLinks().get(linkType).isEmpty();
var linkInApp = false;
if(appProd) {
linkInApp = !appProd.getClassificationProductLinks().get(linkType).isEmpty();
}
//Case A - Prod has been delete approved and has link in Approved > WebRemove
if(stepDelete && linkInApp) {
currentEventQueue.queueDerivedEvent(webRemove, mainProd);
}
//Case B - Prod has link in both Main and Approved > WebModify
else if(linkInMain && linkInApp) {
currentEventQueue.queueDerivedEvent(webModify, mainProd);
}
else {
var partObjects = appContext.getPartObjects().toArray();
var linkToBeApproved = false;
for(var i = 0; i < partObjects.length; i++) {
if(partObjects[i] instanceof com.stibo.core.domain.partobject.ClassificationLinkPartObject)
{
var currentClassID = partObjects[i].getClassificationID();
var currentClassObjTypeID = manager.getClassificationHome().getClassificationByID(currentCl
assID).getObjectType().getID();
```

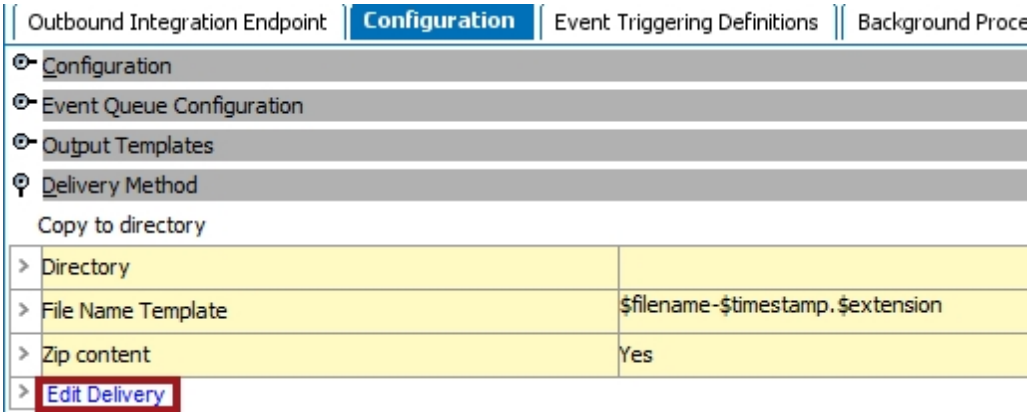
```
if(currentClassObjTypeID == ClassObjTypeID && classInApp(currentClassID)) {
linkToBeApproved = true;
break;
}
}
}
//Case C - Prod has link in Main but not in Approved and link is part object > WebAdd
if(linkInMain && linkToBeApproved) {
currentEventQueue.queueDerivedEvent(webAdd, mainProd);
}
//Case D - Prod has link in Approved but not in Main and link is part object > WebRemove
if(linkInApp && linkToBeApproved) {
currentEventQueue.queueDerivedEvent(webRemove, mainProd);
}
}
}
//Event based on republish or change to externally maintained data
else {
manager.executeInWorkspace("Approved", function(appManager) {
appProd = appManager.getProductHome().getProductByID(currentObject.getID());
});
//Case E - Non approval based event has been generated and prod is in Approved with link >
WebModify
if (appProd && !appProd.getClassificationProductLinks().get(linkType).isEmpty()) {
currentEventQueue.queueDerivedEvent(webModify, currentObject);
}
}
}
```

Output Template - If different messages are to be output dependent on the derived event type, three output templates should be configured as shown below.



Outbound Integration Endpoint Delivery Methods

Both static selection and event-based endpoints require a delivery method. To specify the delivery method for an outbound integration endpoint, on the **Configuration** tab, in the **Delivery Method** area, click the **Edit Delivery** link.



The following options are available by default:

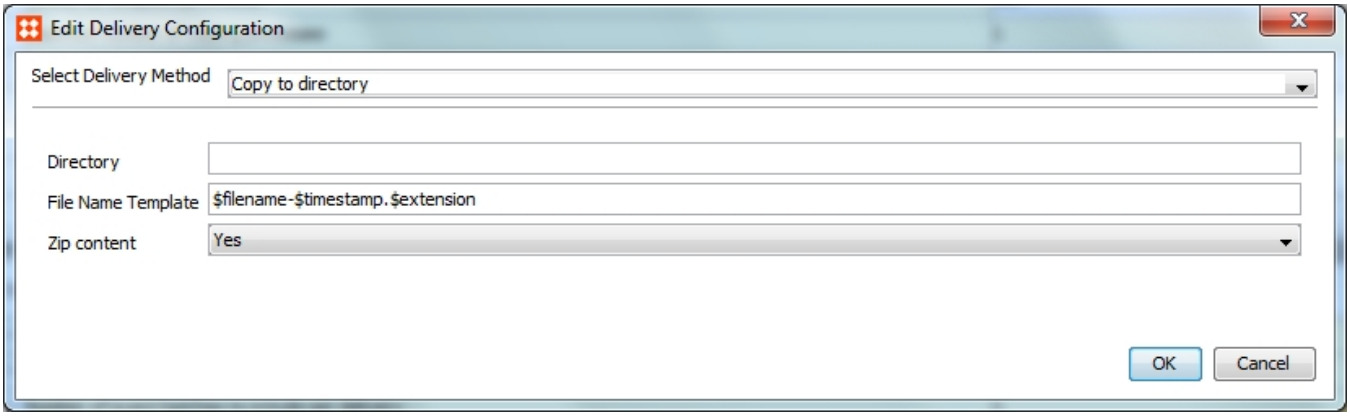
- Copy to Directory Delivery Method
- Deploy Delivery Method
- Email Delivery Method
- FTP Delivery Method
- IBM Websphere MQ SSL Delivery Method
- JMS Delivery Method
- Mongo Delivery Method
- Oracle AQ Delivery Method
- REST Delivery Method
- REST Direct Delivery Method
- SFTP Delivery Method

If other options are required, please contact Stibo Systems.

Copy to Directory Delivery Method

There is no single delivery plugin for server side delivery, but the 'Deploy' and 'Copy to directory' plugins provide the same function in that they both deliver files to a directory on the application server. Since the 'Copy to directory' is a little more flexible because it allows the delivery to be zipped, this is the common setup.

1. On the **Configuration** tab, navigate to the **Delivery Method** flipper, then click **Edit Delivery**.
2. In **Select Delivery Method**, choose **Copy to Directory**.



3. In **Directory**, specify a directory path on the application server where the exported files will be saved.
4. In **File Name Template**, enter the file naming template to be used for the deliveries. The default template is $filename-$timestamp.$extension. The following options are available:
   - **$filename** expands to 'exported' For CSV and Excel exports. Expands to the Event ID(s) for event-based STEPXML. $filename can be replaced if a different file name should be used.
   - **$timestamp** expands to the time in milliseconds between January 1, 1970 and the time when the file is uploaded (e.g., 1438634758430). You can use $timestamp(ddMMyyyy) to format the timestamp as a date instead.
   - **$extension** expands to the extension of the output file

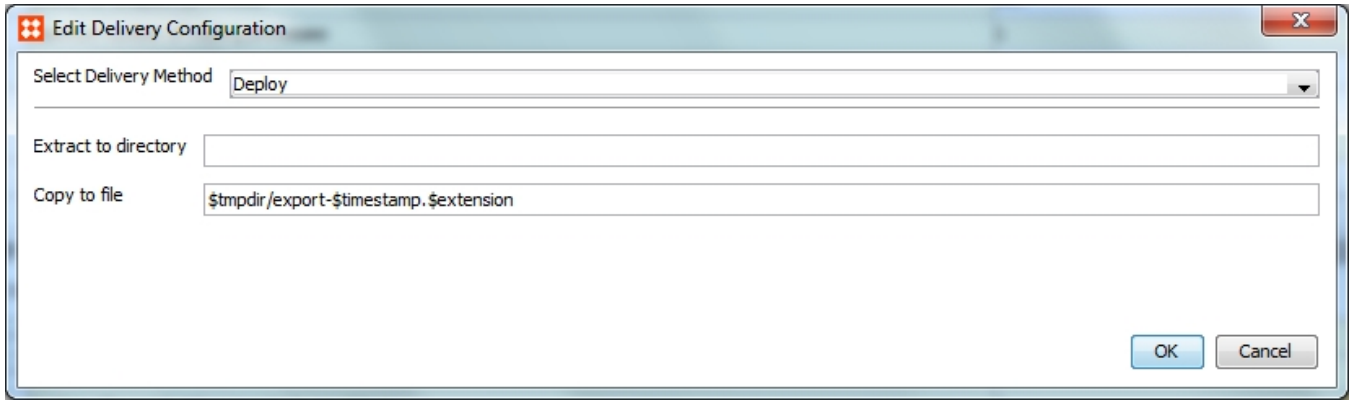
Note: The File Name Template does not support conversions of file formats and can only be used to deliver files in the format specified on the Configuration tab of the outbound integration endpoint.

5. For **Zip content**, specify whether the output file should be delivered in a .ZIP file.

Deploy Delivery Method

There is no single delivery plugin for server side delivery, but the 'Deploy' and 'Copy to directory' plugins provide the same function in that they both deliver files to a directory on the application server. Since the 'Copy to directory' is a little more flexible because it allows the delivery to be zipped, this is the common setup. For more information, see the **Copy to Directory Delivery Method** section of the **Select Delivery Method** documentation.

1. On the **Configuration** tab, navigate to the **Delivery Method** flipper, then click **Edit Delivery**.
2. In **Select Delivery Method**, choose **Deploy**.



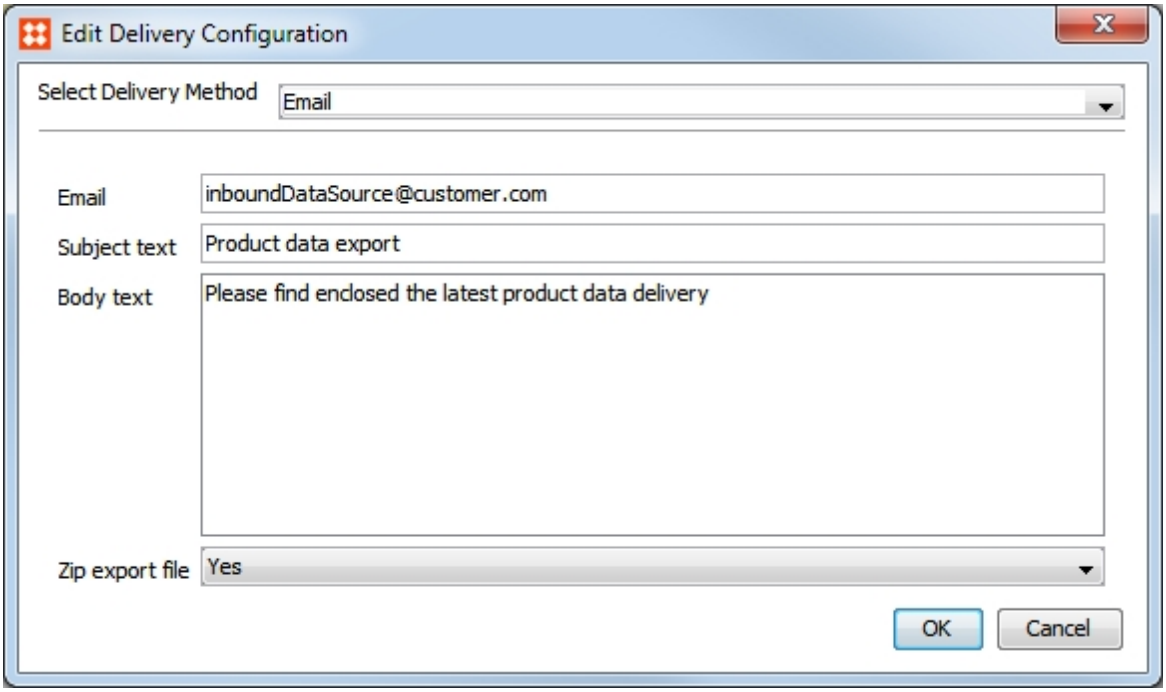
3. In **Extract to Directory**, specify a path on the application server where the exported .ZIP file should be unzipped. The field will only have an effect if the exported file is a .ZIP file, otherwise this field will be ignored.
4. In **Copy to file**, specify a file name template and a path where exported file should be saved.
   - **$tmpdir** points to the $tmpdir path, defined in the configuration file of the server
   - **export-** configurable text that can be replaced with any content that should be in the file name. For example, the Copy to File field could read '$tmpdir/EmergencySKUExport-$timestamp.$extension' instead.
   - **$timestamp** expands to the time in milliseconds between January 1, 1970 and the time when the file is uploaded (e.g., 1438634758430). You can use $timestamp(ddMMyyyy) to format the timestamp as a date instead.
   - **$extension** expands to the extension of the output file

Note: The 'Copy to file' template does not support conversions of file formats and can only be used to deliver files in the format specified on the Configuration tab of the outbound integration endpoint.

Email Delivery Method

The email delivery option is very straight forward and, like 'Copy to directory', it allows the delivery to be zipped.

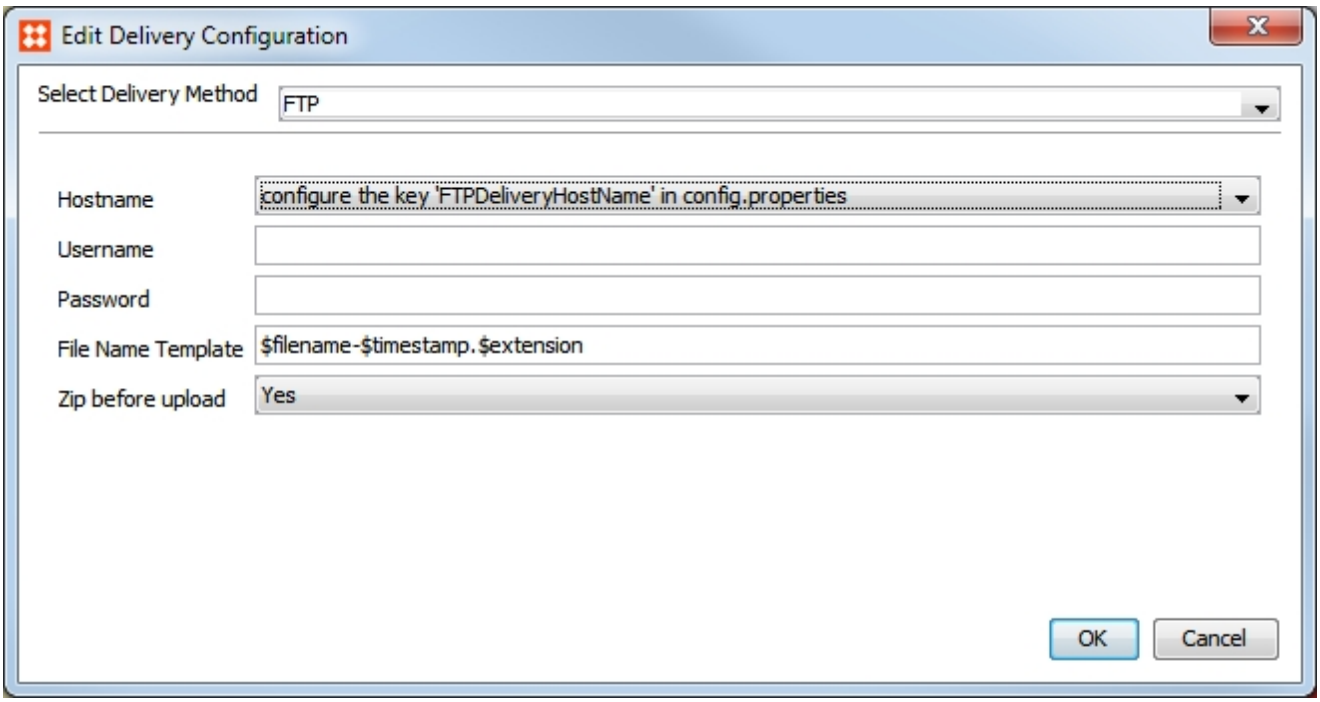
1. In **Select Delivery Method**, choose **Email**.
2. Enter an email address, subject text, and body text.



3. In **Zip export file**, specify if you want to deliver the exported contents (email attachment) as a .ZIP file.

FTP Delivery Method

1. In **Select Delivery Method**, choose **FTP**.



2. In **Hostname**, select the host name to be used for the delivery. Options are available based on the FTPDeliveryHostName property in the sharedconfig.properties file on the application server. The required format of the property is (square brackets not included): FTPDeliveryHostName=1=[host1],2=[host2] Contact Stibo Systems if you need assistance with configuration.
3. In **Username**, enter the user name that will be used to log on to the FTP server.
4. In **Password**, enter the password that will be used to log on to the FTP server.
5. In **File Name Template**, enter the file naming template to be used for the deliveries. The default template is $filename-$timestamp.$extension. The following options are available:
   - **$filename** expands to 'exported' for CSV and Excel exports; expands to the Event ID(s) for event-based STEPXML. $filename can be replaced if a different file name should be used.
   - **$timestamp** expands to the time in milliseconds between January 1, 1970 and the time when the file is uploaded (e.g., 1438634758430). You can use $timestamp(ddMMyyyy) to format the timestamp as a date instead.
   - **$extension** expands to the extension of the output file

Note: The File Name Template does not support conversions of file formats and can only be used to deliver files in the format specified on the Configuration tab of the outbound integration endpoint.

6. In **Zip before upload**, click **yes** to zip contents before uploading the files to the host.

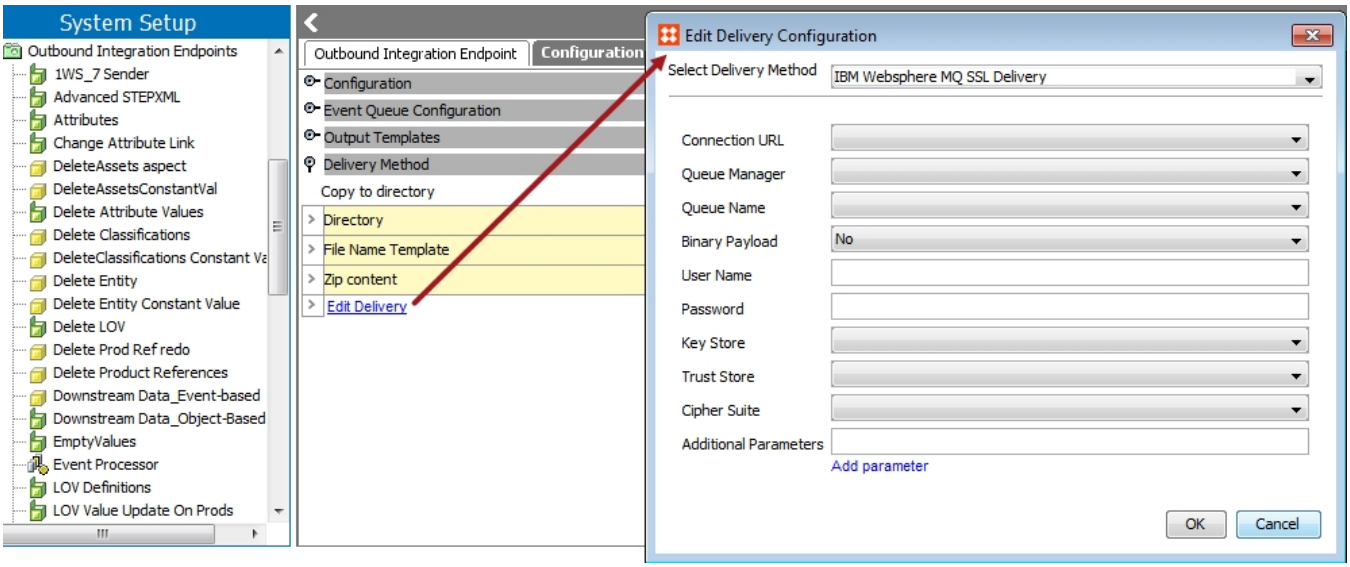
IBM Websphere MQ SSL Delivery Method

For information on connecting to IBM Websphere MQ in a non-SSL way, see JMS Delivery Method.

Note: Options available on dropdown fields must be configured in sharedconfig.properties or config.properties. Select a dropdown to display the required key name for each field.

---

1. In the **Select Delivery Method** field, choose **IBM Websphere MQ SSL Delivery**.



2. In **Connection URL**, select the URL to connect to in the format [host]:[port]/[channel]
   - [host] = hostname or IP of the Websphere MQ server
   - [port] = port number for the channel
   - [channel] = name of the channel
3. In **Queue Manager**, select the name of the Queue Manager.
4. In **Queue Name**, select the name of the Queue for the connection.
5. In **Binary Payload**, select 'Yes' if the message contents will be in a binary format (such as Excel).
6. In **User Name**, if required, enter the user name that will be used to log onto the JMS provider.
7. In **Password**, if required, enter the password that will be used to log onto the JMS provider.
8. In **Key Store**, select the keystore in jks format, with the personal certificate for the Queue Manager. To generate this, consult the manual for IBM Websphere MQ. The password for the key store must be configured in sharedconfig.properties or config.properties using 'WSMQSSLKeyStorePassword'.
9. In **Trust Store**, select the trust store with the CA for the Queue Manager. This can be the same file as key store. To generate this, consult the manual for IBM Websphere MQ. The password for the trust store must be configured in sharedconfig.properties or config.properties using 'WSMQSSLTrustStorePassword'.

10. In **Cipher Suite**, set to the same value as SSL CipherSuite in Websphere MQ. STEP is running on non-IBM jre, so this must be the same value as configured in the Queue Manager.
11. If **Additional Parameters** are required, click Add parameter, then enter the Key and Value. For possible keys and values, consult the manual for IBM Websphere MQ.

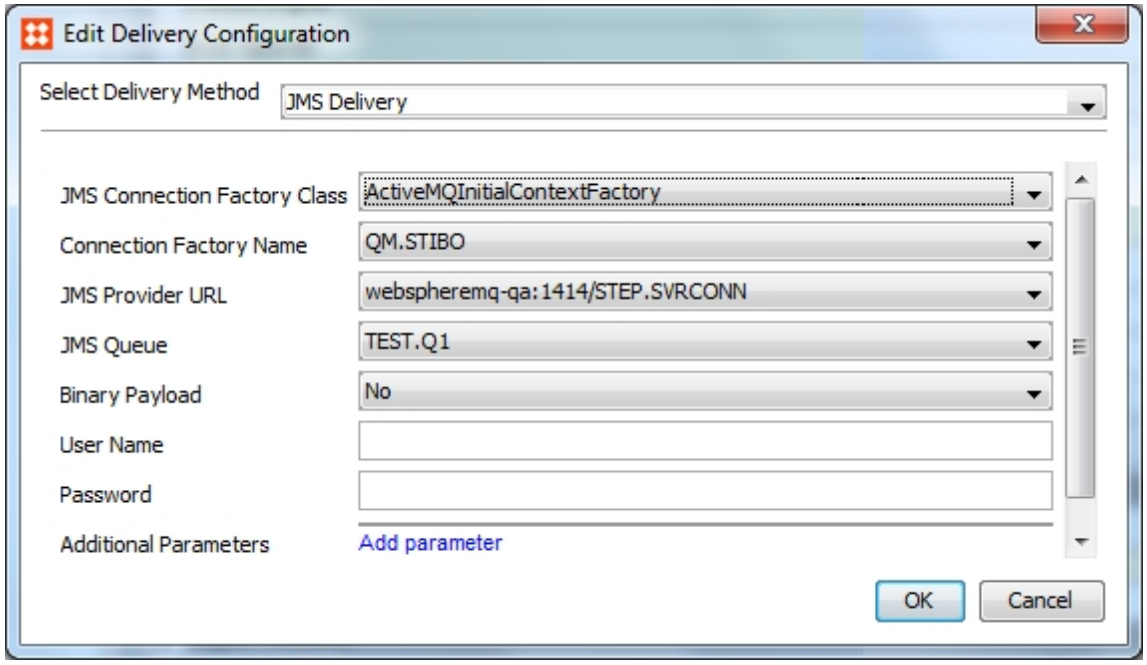
JMS Delivery Method

The available options for the Java Message Service (JMS) delivery method are system dependent. STEP has three standard JMS delivery plugins available: Apache Active MQ, Websphere MQ, and Oracle AQ. Each of these can be used to deliver messages to a JMS receiver without any further customizations. To deliver messages to other JMS providers, contact your Stibo Systems account manager for further customizations.

Note: Options available on dropdown fields must be configured in sharedconfig.properties or config.properties. Select a dropdown to display the required key name for each field. Contact Stibo Systems if you need assistance with configuration.

---

1. On the **Configuration** tab, in the **Delivery Method** area, click **Edit Delivery**.
2. In **Select Delivery Method**, choose **JMS Delivery**.



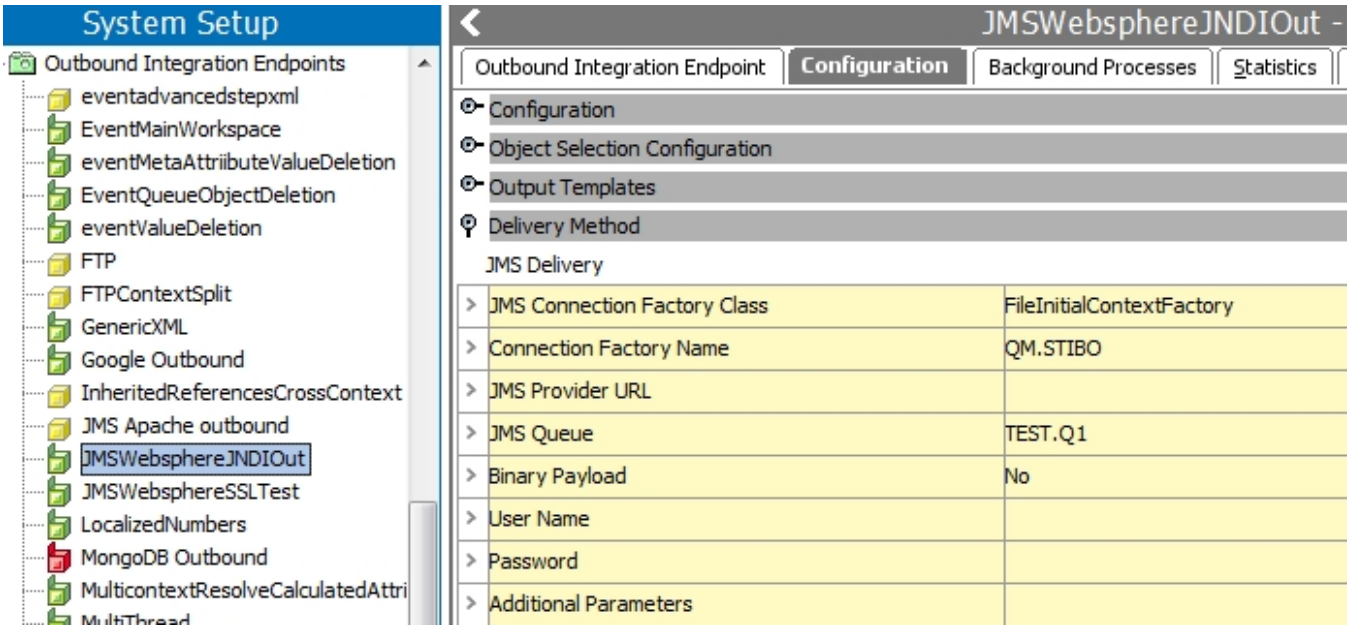
3. In **JMS Connection Factory Class**, choose the JMS connection factory class that corresponds to the driver class from the JMS provider vendor. On a standard system, the following options are available:
   - **ActiveMQInitialConnectionFactory**: Connects the JMS delivery to Apache Active MQ. For information, visit apache.org.

   Note: WMQInitialContextFactory is no longer supported, use FileInitialContextFactory instead.

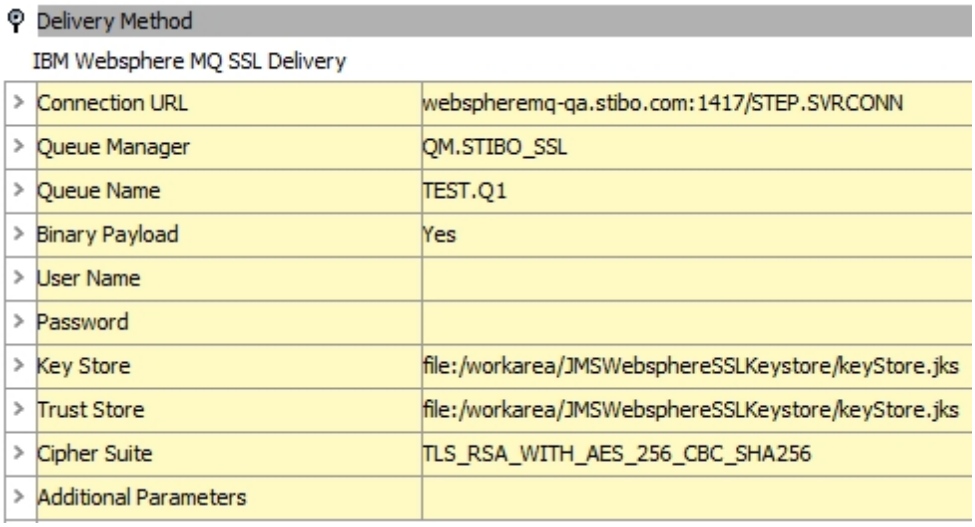
   - **FileInitialContextFactory**: Enables setting up JMS Websphere Inbound and Outbound Integration Endpoints which reference a binding file (created from JMS Websphere Client software). The binding file is a configuration file which includes all details of how STEP should interact with JMS Websphere. For information about JMS Websphere Client Software, visit ibm.com.

4. In **Connection Factory Name**, select a connection factory name from the list.
5. In **JMS Provider URL**, select a JMS Provider URL from the list. For example: [protocol]://[address]:[port]
6. In **JMS Queue**, select the physical name of the JMS Queue to be used on Apache Active MQ or Websphere MQ.
7. In **Binary Payload**, select 'Yes' if the message contents will be in a binary format (such as Excel).
8. In **User Name**, if required, enter the user name that will be used to log onto the JMS provider.
9. In **Password**, if required, enter the password that will be used to log onto the JMS provider.
10. If additional parameters are required, click **Add parameter**, then enter the Key and Value.

FileInitialContextFactory JMS Delivery Configuration Example



JMS Websphere Delivery Using SSL Configuration Example



Apache Active MQ JMS Delivery

Delivery Method

JMS Delivery

JMS Connection Factory Class	ActiveMQInitialContextFactory
Connection Factory Name	ConnectionFactory
JMS Provider URL	tcp://ATTCM3S9:61616
JMS Queue	testqueue
Binary Payload	No
User Name	
Password	
Additional Parameters	

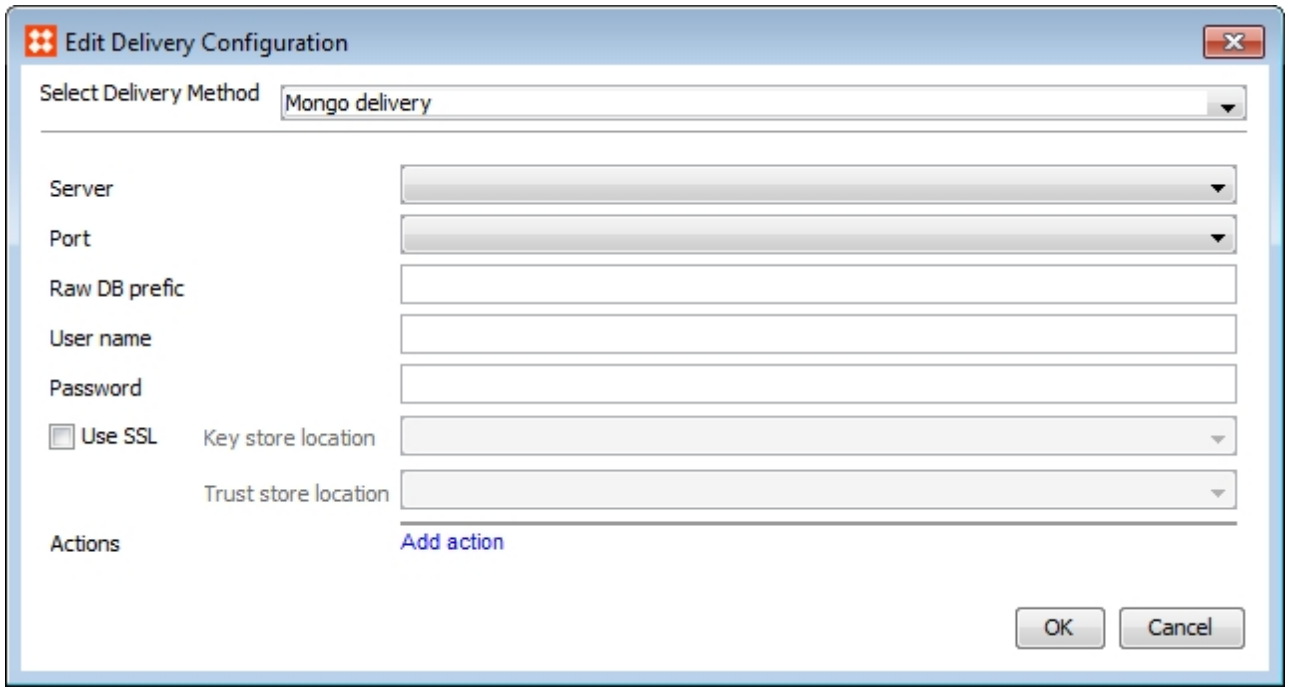
Edit Delivery

Mongo Delivery Method

Prior to setting the Mongo Delivery Method, ensure your setup meets the prerequisites explained in **Prerequisites for Configuring the MongoDB Adapter** section of the **Integration Endpoints** documentation.

Note: Options available on dropdown fields must be configured in sharedconfig.properties or config.properties.

1. In **Select Delivery Method**, choose **Mongo delivery**.



2. In **Server**, enter the server where the MongoDB instance used by the endpoint is running. The available servers are configured using the MongoDB.server property.
3. In **Port**, enter the network port used by the MongoDB instance on the specified server. The possible values are configured using the MongoDB.port property.
4. In **Raw DB prefix**, if required, enter a prefix for all raw databases the adapter creates. Optional.
5. In **User name**, if MongoDB is configured to use authentication, enter the MongoDB user. Optional.
6. In **Password**, if MongoDB is configured to use authentication, enter the MongoDB password. Optional.
7. Check **Use SSL** if MongoDB requires the client to use SSL encryption. Optional. If no further parameters are configured, it is assumed that the MongoDB uses a SSL certificate that is issued by a trusted certificate authority (like Verisign or Thawte). If this is not the case, or if more security of the connection is requested, you can configure an SSL key store and an SSL trust store.
   - In **Key store location**, enter a key store location, for example, file://[path]/key_store.jks. The Key store holds the STEP Mongo Adapter private key and SSL certificate in a keystore file as generated by the Java utility 'keytool' (in jks format). The password for the key store is configured in sharedconfig.properties or

config.properties, using MongoDB.SSLKeyStorePassword. For example,

```
MongoDB.SSLKeyStorePassword = keystore_password
```

- In **Trust store location**, enter a trust store location, for example, file://[path]/trust_store.jks. The trust store holds the Mongo DB public key and certificate in a keystore file as generated by the Java utility 'keytool' (in jks format). The trust store configuration is only needed if the certificate is self-signed and not issued by a trusted certificate authority. The password for the key store is configured in sharedconfig.properties or config.properties, using MongoDB.SSLTrustStorePassword. For example,

```
MongoDB.SSLTrustStorePassword = truststore_password
```

8. In **Actions**, click the **Add action** link to enter one or more STEP business actions used to create and populate aggregated collections. These actions are run after the raw collections have been populated.

For additional information about the Mongo Delivery Method, see the **Mongo Delivery Method Overview** section of the **Integration Endpoints** documentation.

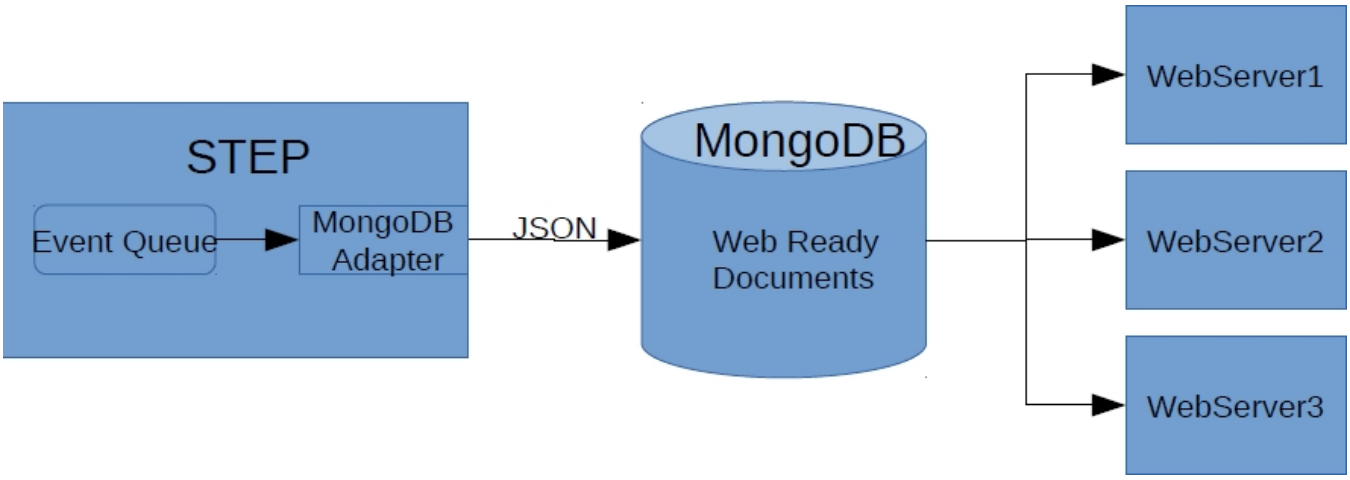
For more information, see **MongoDB Adapter Setup Quick Guides** section of the **Integration Endpoints** documentation.

Mongo Delivery Method Overview

The STEP Mongo Adapter is an outbound integration endpoint delivery plugin that receives data from a STEP event queue and loads it into a MongoDB database. The MongoDB is often used for web site back-end, reporting, and high performance feeds to other back-end systems.

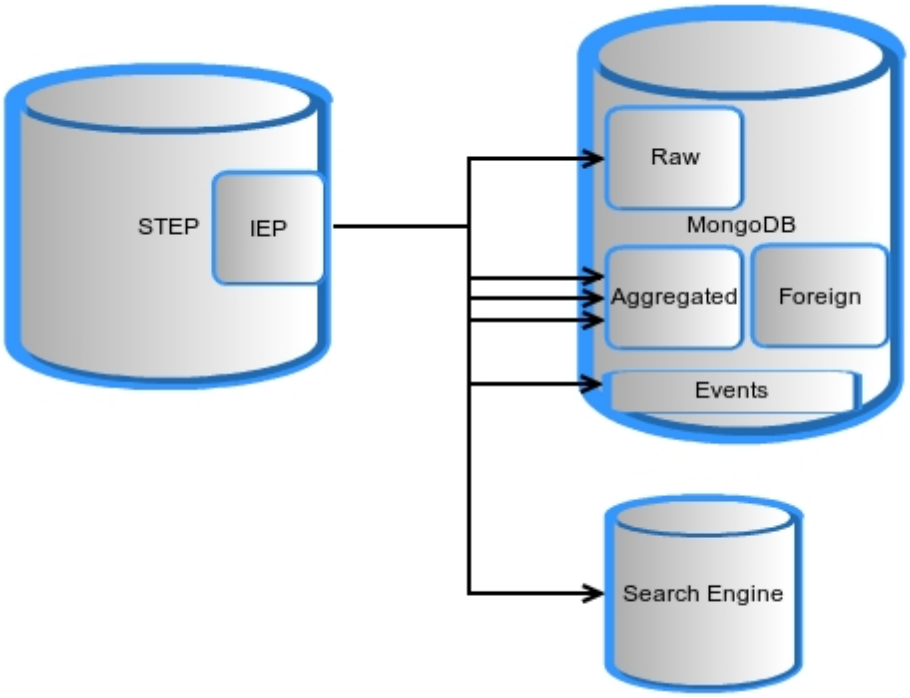
The Mongo adapter converts STEPXML to JSON and keeps a set of 'raw' collections in MongoDB that are in sync with STEP. Additionally, it allows you to add JavaScript triggers to maintain aggregated collections.

The following illustration shows how the Mongo Adapter can be used to feed a Mongo database that is used as the back-end to a website:



By keeping collections of web-ready JSON documents available in MongoDB, the JSON can be passed directly to the browser's JavaScript. This results in the web server having little processing to do, which means that it simply needs to pass the right collection result directly back to the browser.

The following drawing illustrates the collections involved:



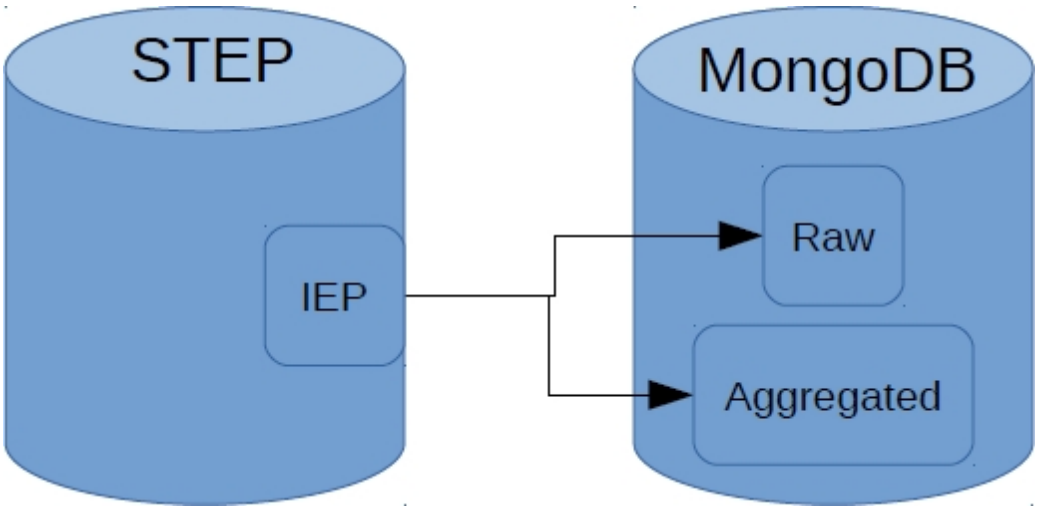
The STEP Mongo Adapter ensures that the raw collections are a one-to-one mirror of data in STEP. The adapter does not embed nodes found through references or child associations. These nodes are stored separately in the raw collections.

It is possible to configure the adapter so that it creates one or more customized, aggregated MongoDBs (databases or collections) where the documents can be tailored to meet the exact usage requirements, like reporting, analysis, and so on.

The JavaScript triggers allow maintaining aggregated collections, for example:

- List of children, including title and other data necessary in a tree-like view.
- Information from the target end of references that is needed in a given view.
- Mapping to a JSON structure that is closer to the web site data model.

The following image shows a MongoDB database containing a raw database that is maintained directly by the STEP Mongo Adapter and an aggregated database maintained by JavaScript triggers.



A JavaScript trigger is executed under certain conditions, for example, only for a specific object type. This reduces the amount of checking necessary in the JavaScript code. Once the raw collection is updated and the JavaScript triggers (if configured) have completed, an event is inserted into the events collection. Also, an update is sent to the search engine.

Note: Aggregated collections and raw collections do not need to be in the same MongoDB instance. However, if required, different collections in the aggregate can be combined into their own MongoDB instance.

Default Databases and Collection

The STEP Mongo Adapter maintains a number of MongoDBs, referred to as 'raw databases', and a number of collections in the raw databases, referred to as 'raw collections'. A raw database is created for each context that is exported and the database is named after the context. For example, if the two contexts EN and FR are exported, two raw databases are created, one named EN and one named FR.

It is possible to configure a string that prepend the raw database name when the STEP Mongo Adapter is configured. When data is exported from multiple contexts, a JSON document is created per context, per exported object. Continuing the previous example, two JSON documents are created for each object: one for the object in the EN context and one for the object in the FR context.

A raw collection is only created when the particular STEP data type is exported. So, if no assets are exported from STEP, no asset collection is created. Each raw database contains all or some of the following raw collections:

Raw Collection	Purpose
product	Exported products
asset	Exported assets
classification	Exported classifications

Raw Collection	Purpose
entity	Exported entities
attribute	Exported attribute definitions
referenceType	Exported reference types
listofvalues	Exported list of values (LOVs)
unit	Exported unit definitions

Inherited values, calculated values, and unit names on values are handled already in the raw collections. Since changes to these affect replicated information, STEP ensures that appropriate updates are made to sync the replicated information.

STEP JSON and Mongo JSON

The STEP Mongo Adapter converts STEPXML input to STEP JSON (JavaScript Object Notation) documents, and then stores the JSON documents in the MongoDB. The STEP JSON schema can be downloaded from your STEP server at: http://*[enter step-server]*/files/StepSchema.json

Formally, MongoDB documents are BSON documents, which is a binary representation of a JSON document. For more information, see the MongoDB documentation on https://www.mongodb.com/

For an expanded example of the basic conversion using STEP Mongo Adapter, see the **Mongo Delivery Method Conversion Example** section of the **Integration Endpoints** documentation.

JavaScript Triggers

STEP business action triggers with JavaScript populate the aggregated MongoDB databases and collections. You can configure one or more business action triggers for the STEP Mongo Adapter. The business action triggers are fired after the STEP Mongo Adapter has populated the raw collections.

During configuration of the STEP Mongo Adapter, the configuration wizard allows you to select multiple JavaScripts to run under different conditions by adding preconditions to the business actions. A precondition is, for example, a JavaScript trigger that is only fired for updates to a given type of objects. For more information, see the **To Add Preconditions to a Business Rule** section of the **Editing Global Business Rules** documentation.

Due to the nature of inheritance and calculated attributes, the trigger may be called even if there are no changes to the underlying raw objects.

The business action triggers are executed once per JSON document sent to the MongoDB database. If two objects are exported from STEP, two JSON documents are sent to the MongoDB database and the business action triggers are executed twice. If the objects are exported in multiple contexts, one JSON document is

generated per context, per object. This means that if two products in two contexts are exported, the business action triggers are executed four times.

Business action triggers are executed on the STEP server. Business actions triggers need access to the MongoDB database to fetch data from the raw collections and write or read data to / from the aggregated collections. For the best performance, network latency between the STEP server and the MongoDB server should be minimal.

Important: Place aggregated collections in their own database so that the raw database contains only collections.

Bound Variables

The JavaScript trigger have accesses to information in the Mongo Adapter environment by binding script variables when the JavaScript trigger is defined.

Type	Gives access to
MongoDBContext	• The current STEP context. • The name of the raw MongoDB database. • The MongoDB database via the com.mongodb.Mongo object. For more information, see: http://api.mongodb.org/java/current/com/mongodb/Mongo.html. The actual Mongo object that is passed can be a wrapper, but it adheres to the interface
JSON context	The JSON object with the data that has been persisted in MongoDB.

For more information about executing JavaScript in business rules, see the **Execute JavaScript** section of the **Business Rules** documentation.

Example JavaScript to Maintain a Collection in a Classification

The following code-snippet illustrates an example JavaScript that maintains a collection containing the list of products (ID and Name) in a classification.

The structure of the collection is:

```
{_id: "<classification-id>", products : { { id: "<product-id1>", name: "<product-name1>"},
{ id: "<product-id2", name: "<product-name2>"}, ... }
function getClassificationIDs(classrefs) {
    var result = new Array();
    if (classrefs != null) {
        for (i = 0; i < classrefs.targets.size(); ++i) {
            result.push(new String(classrefs.targets[i].targetID));
        }
    }
    return result;
```

```
}

var db = mongoContext.getMongo().getDB('extra');
var collection = db.getCollection('classificationproducts');

var classrefs = mongoData.references ? mongoData.references["Web Classifications"] : null;

var classificationIDs = getClassificationIDs(classrefs);

var deletequery = {"products.id" : mongoData._id , "_id" : { $nin :  getClassificationIDs(c
lassrefs) }};
var adelete = { $pull : { products : { id : mongoData._id } } };

collection.update(deletequery, adelete, false, true);

var query = {"products.id" : mongoData._id };
var name = mongoData.name;
var update = { $set : { "products.$.name" : name } };

collection.update(query, update, false, true);

classificationIDs.map(function(item) {
    var query = { _id : item };
    var update = { $addToSet : { products : { id : mongoData._id, name : name } } };
    collection.update(query, update, true, false);
});
```

Prerequisites for Configuring the MongoDB Adapter

The following are prerequisites for setting up the MongoDB adapter:

1. A MongoDB instance with suitable amounts of disk space.
2. The STEP MongoDB Adapter uses a network port to access the MongoDB database on the server where the MongoDB is running. The network port must be open and accessible from the STEP servers.
3. Update sharedconfig.properties or config.properties with the **name of the MongoDB server**, using the MongoDB.Server configuration property and a list of number-to-value pairs, separated by commas, as follows:

   ```
   MongoDB.Server = 1=MongoDBServer1, 2=MongoDBServer2, 3=MongoDBServer3
   ```

4. Update sharedconfig.properties or config.properties with the **MongoDB port** using the MongoDB.Port configuration property, and a list of number-to-value pairs, separated by commas, as follows:

   ```
   MongoDB.Port = 1=10001, 2=2002, 3=3003
   ```

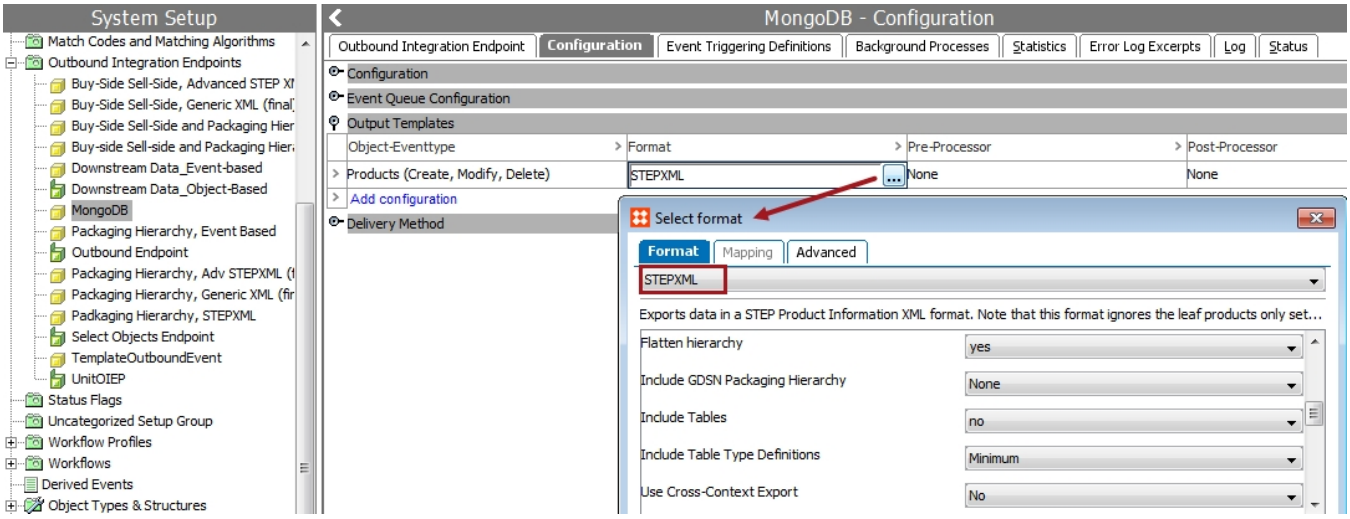
   For example, to configure the adapter to use a MongoDB instance running on either 'Server1' or 'Server2' using port '27712' on both servers, the properties would be:

   ```
   MongoDB.Server = 1=Server1, 2=Server2
   MongoDB.Port = 1=27712
   ```

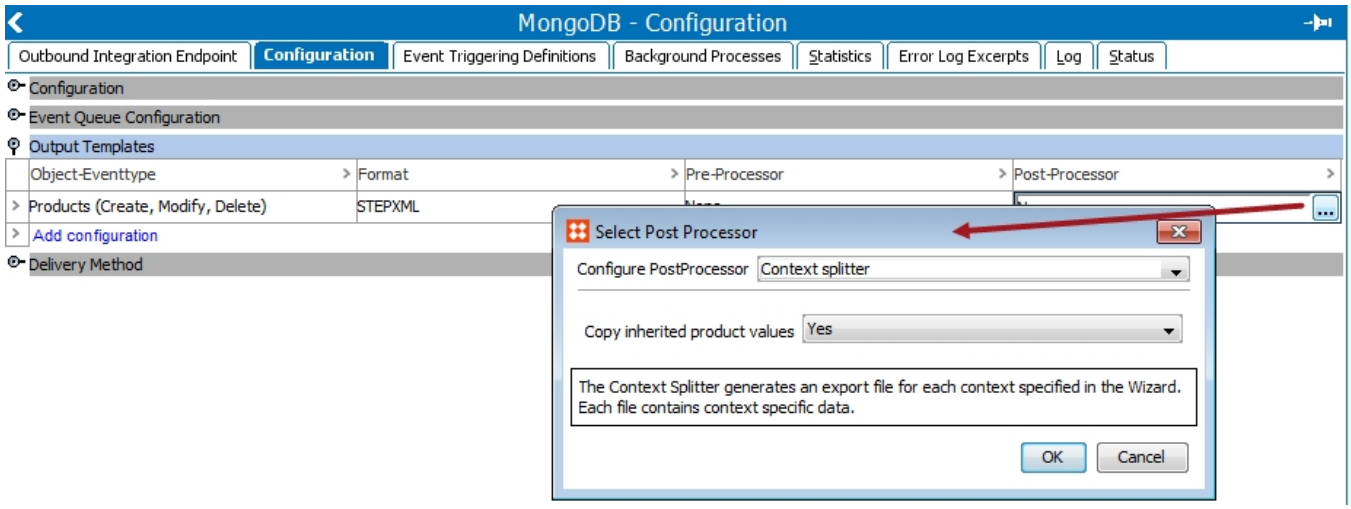
Configure the OIEP

The STEP MongoDB Adapter is run within an Outbound Integration Endpoint (OIEP) and configured as a delivery plugin for the endpoint.

1. Create an OIEP, selecting STEP Exporter as the Process Engine.
2. For the new OIEP, open Configuration > Output Templates > Add configuration > Format > click the ellipsis button **(…)**.



3. On the Select Format dialog, leave objects as default values except for the following parameters:
   - Select STEPXML export format in the dropdown at the top of the dialog
   - Set Flatten hierarchy = Yes
   - Set Include Type Definitions = Minimum
   - Set Include Unit Definitions = Minimum
   - Set Include List of Values Definitions = Minimum
   - Set Include Attribute Group Definitions = Minimum
   - Set Include Assets = Selected
   - Set Include Classifications = Selected
   - Set Include Products = Selected
   - Set Include Entities = Selected
4. In the Post-Processor field, click the ellipsis button **(…)** and select **Context splitter**, then click OK.



5. For Delivery Method, if MongoDB requires user authentication, configure a MongoDB user and password. The MongoDB user must be configured in the MongoDB admin database and have permissions to create new MongoDB databases (the MongoDB role "readWriteAnyDatabase").

Configuring the JavaScript Triggers

JavaScript triggers are fired for all updates to the raw collections. The purpose of the triggers is to maintain the aggregated collections. The JavaScript triggers are configured as business actions in STEP and then applied when configuring the MongoDB Adapter.

Continue with setting up the Mongo Delivery Method options, by following the **Mongo Delivery Method** section of the **Integration Endpoints** documentation.

Mongo Delivery Method Conversion Example

The following examples use this product with type = 'Item', ID 'XYZ', and Name = 'XYZ Name'. The product is located in the family 'FamilyXYZ2'.

The STEP Mongo Adapter transforms this to the STEP JSON document:

```
{
_id : "EXA-5002-1001",
objectTypeID : "Chair",
parentID : "Chairs",
name : "Arm chair",
type : "product"
}
```

The id of the product is used as unique ObjectId in the MongoDB database, and therefore the STEPXML id field is transformed to _id. The UserTypeID XML attribute is converted to the objectTypeID field. The type field describes the kind of STEP object that is exported. For Basic STEP objects, the types are 'product', 'asset', 'classification', and 'entity'. Other STEP objects are described in the following topics.

Values

In this example, a product has two attributes:

- A single value attribute Color with the value 'Brown'
- A multi value attribute Height with the value = '43', the unit = 'inches', and the value = '120', with the unit = 'cm'.

When this is transformed to a STEP JSON document by the STEP MongoDB adapter, it results in the following:

```
{
  _id : "EXA-5002-1001",
  objectTypeID : "Chair",
  parentID : "Chairs",
  name : "Arm chair",
  type : "product"
  values : {
    Color : "Brown",
    Height : [ "43 inches", "120 cm"]
  }
}
```

The data displayed includes:

- Values of the product are stored as a subdocument with the name 'values'.
- The value of the single-valued attribute 'Color' is represented as a field-and-value pair: the ID of the attribute is the field, and the value is the value of the attribute represented as a string.
- The multi-valued attribute 'Height' is represented as a field-to-arrays of strings. Again, the ID of the attribute is the field, and the string array contains a string for each value.
- The name of the unit is appended to the attribute value with a space for all values with a unit. In the example above, '120 cm' is the concatenation of the value '120' and 'cm' from the unit named 'unece.unit.CM'.

Values of specification attributes that are inherited to the product are included in the JSON document of the product. For example, if the parent contains an attribute called Brand with the value 'Office chairs', the JSON document looks like the following:

```
{
  _id : "EXA-5002-1001",
  objectTypeID : "Chair",
  parentID : "Chairs",
  name : "Arm chair",
  type : "product"
  values : {
    color : "Brown",
    Height : [ "43 inches", "120 cm"],
    Brand: "Office chairs"
    }
}
```

References and Links

References and links are converted to STEP JSON in a subdocument called 'references'. This subdocument contains a field-and-value pair, where the field is the ID of the reference type, and the value is either a

subdocument that defines the target of a reference, or an array of subdocuments that each define the target of a reference.

If the reference can only reference one target for the same source, the value is a subdocument. If the reference can reference more targets for the same source, the value is an array.

In this example, the product uses a 'Primary Image' asset reference to reference the image for the product. The asset has the ID 'Image_EXA-5002-1001'. The STEP JSON then exports as:

```
{
  _id : "EXA-5002-1001",
  objectTypeID : "Chair",
  parentID : "Chairs",
  name : "Arm chair",
  type : "product"
  values : {
    color : "Brown",
    Height : [ "43 inches", "120 cm"],
  Brand: "Office chairs"
  }
  references : {
    Primary Image : { targetID : "Image-EXA-5002-1001" }
  }
}
```

To find the asset for the reference, it is necessary to find the reference type 'Primary Image', which is stored in the raw collection 'referenceType'. The target type of the reference comes from the reference type.

The JSON of the reference target is found by searching the raw collection given by the reference type of an object with 'targetID'.

In the example above, the target type of the 'Primary Image' reference type is 'asset'. By looking in the 'asset' raw collection, the asset given by Image-EXE-5002-1001 is found.

Adding the reference 'Secondary Image', which can reference more targets such as ('Image1' and

'Image2') to the product, results in the following STEP JSON:

```
{
  _id : "EXA-5002-1001",
  objectTypeID : "Chair",
  parentID : "Chairs",
  name : "Arm chair",
  type : "product"
  values : {
    color : "Brown",
    Height : [ "43 inches", "120 cm"],
```

```
  Brand: "Office chairs"
}
references : {
  Primary Image : { targetID : "Image-EXA-5002-1001" },
    Secondary Image : [ { targetID : "Image1" }, { targetID : "Image2" } ]
  }
}
```

If the reference contains a meta data attribute, the values of the meta data attribute are added as a subdocument to the reference subdocument that contains the value of the attribute:

```
{
  _id : "EXA-5002-1001",
  objectTypeID : "Chair",
  parentID : "Chairs",
  name : "Arm chair",
  type : "product"
  values : {
    color : "Brown",
    Height : [ "43 inches", "120 cm"],
  Brand: "Office chairs"
}
  references : {
    Primary Image : { targetID : "Image-EXA-5002-1001",
                      values: { ShowOnWeb : "true" },
                     },
      Secondary Image : [ { targetID : "Image1" }, { targetID : "Image2" } ]
  }
}
```

Here, the reference type 'Primary Image' has a meta data attribute called 'ShowOnWeb'. The attribute is set to 'true' for the reference from the product to the asset 'PrimaryAsset'.

Classification to product links are special because they are owned by either the product or the classification. This will always be the same for a specific classification to product link type. The JSON document only contains the classification-to-product links owned by the object. That is, the JSON document for a product only contains classification-to-product links owned by the product.

Attribute Types

Exporting an attribute type with the ID 'AttributeTypeID', and the name 'Name' linked in two parent attribute groups 'Parent1' and 'Parent2' results in the following JSON:

```
{
_id : "AttributeTypeID",
```

```
parentID : [ "Parent1", "Parent2"],
name : "Name",
listOfValuesID : "ListOfValuesID",
validUnitIDs : ["unece.unit.MMT", "unece.unit.CMT"],
type : "attribute"
}
```

Reference types are stored in the raw collection 'attribute'. To export reference types, verify that 'Include Attribute Definitions' is set to at least 'Selected' in the Process Engine configuration.

Reference Types

Exporting a product-to-product reference type with the ID 'ReferenceTypeID' and the name 'ReferenceName' generates the following JSON:

```
{
_id : "ReferenceTypeID",
name : "ReferenceName",
type : "referenceType",
targetType : "product"
}
```

In this example, 'targetType' is the target type of the reference. Possible target types are 'product','asset', 'classification', and 'entity'.

Reference types are stored in the raw collection 'referenceType'. When you export reference types, verify that 'Include Type Definitions' is set to at least 'Minimum' in the Process Engine configuration.

Asset Push Locations

Asset Push Locations are stored in the assetPushLocations subdocument, in the root of the Asset documents in the asset collection. It provides the relative path to different versions of the asset. The assetPushLocations subdocument contains field-and-value pairs, where the field name is the assetpush configuration id, and the value is the relative path that the image file has been pushed to. The relative path is relative to the AssetPushClients root folder, so to get the full path of the image, you have to prepend the path to the assetpush client root dir, as shown in the following example:

```
assetPushLocations : {
    AllAssets-approved : "AllAssets-approved/std.lang.all/73/15/7315.pdf",
    AllWebsiteImages-approved : "AllWebsiteImages-approved/std.lang.all/73/15/7315.pdf",
    small : "small/std.lang.all/73/15/7315.jpg",
    large : "large/std.lang.all/73/15/7315.pdf"
}
```

Tables

Tables are defined on classifications or products. The table subdocument is stored in the root of the product and classification document, in the product and classification collection. The table subdocument contains field-and-value pairs, where the field name is the table Type name of the table on the product or classification.

A table is exported from STEP in a resolved and transformed state. That is, any transformations defined in STEP have already been applied to the table and the row and column layout has therefore been resolved.

Formatting is not applied to the table before it is exported. So, a web application is not bound by the STEP table formatting. However, if any formatting has been defined for the table in STEP, it is included in the export so that the website application can be configured to render the formatting on the website.

The table contains an array of column elements that contain the formatting meta data for each column and an array of rows. The row contains the formatting information for each row, plus an array of cells. Each cell has a number of formatting meta data attributes and a text field that contains the content of the table cell.

Table Formatting

Formatting meta data can be stored on table, column, row, and cell level, and is inherited downwards from table level to column level to row level to cell level.

If a column is specified to use an underlined text style, and a row is specified to be bold text style, the formatting information is accumulated so that the cell at the intersection of the row and column is rendered as both underlined and bold.

Formatting meta data defined at a lower level overrides meta data at a higher level. For example, if a table has a gray background color, and a column has a red background color, then cells in that row are rendered with a red background.

A simple way of formatting a table in HTML is to map the tableType, columnType, and rowType to element classes in HTML, and then use a CSS style sheet to define the table formatting, as shown in the following example.

```
tables : {
  Description Table : {
    columns :
      [ { columnType : "Description" }, { columnType : "Description" }, { columnType : "Des
cription" } ],
    rows : [
      { rowType : "Header",
        cells : [
          { ruleRight : "0.5 pt",
            text : "Header 1",
            verticalAlignment : "top",
            cellStoryDirection : "horizontal",
            backgroundColor : "Light Blue",
            column : "0",
            ruleBelow : "0.5 pt",
```

```
            ruleAbove : "0.5 pt",
          textStyle : "TableHeader—Description",
            ruleLeft : "0.5 pt" },
          { ruleRight : "0.5 pt",
            text : "Header 2",
            verticalAlignment : "top",
            cellStoryDirection : "horizontal",
            backgroundColor : "Light Blue",
            column : "1",
            ruleBelow : "0.5 pt",
            ruleAbove : "0.5 pt",
            textStyle : "TableHeader—Description",
            ruleLeft : "0.5 pt" }] },
      { rowType : "Header",
          cells : [
          { ruleRight : "0.5 pt",
            text : "Value 1",
            verticalAlignment : "top",
            cellStoryDirection : "horizontal",
            backgroundColor : "Light Blue",
            column : "0",
            ruleBelow : "0.5 pt",
            ruleAbove : "0.5 pt",
            textStyle : "TableHeader—Description",
            ruleLeft : "0.5 pt" },
          { ruleRight : "0.5 pt",
            text : "Value 2",
            verticalAlignment : "top",
            cellStoryDirection : "horizontal",
            backgroundColor : "Light Blue",
            column : "1",
            ruleBelow : "0.5 pt",
            ruleAbove : "0.5 pt",
            textStyle : "TableHeader—Description",
            ruleLeft : "0.5 pt" }] }
] } }
```

Attribute Links

AttributeLinks define the validity of attributes for products linked into the product and classification hierarchies. The attributeLinks subdocument is stored at the root of the product and classification document structure. The attribute links can themselves contain a values subdocument that defines the meta data values attached to the attributeLink.

The attributeLinks subdocument contains field value pairs, where the field name is the attributeID, and the value is a subdocument that contains any meta data values related to the attribute link. The following is an example of an attributeLinks subdocument:

```
attributeLinks : { Voltage range : { }
                   Rotary address switches : { },
                   Default address : { },
                   Power consumption : { }
                   }
```

Unit

Exporting an unit with the ID 'unece.unit.CMT', and the name 'cm' results in the following JSON:

```
{
"_id" : "unece.unit.CMT",
"values" : {
"4941" : "centimetre"
},
"conversionToBase" : {
    "factor" : "100",
    "unitID" : "unece.unit.MTR",
    "offset" : "0"
},
"name" : "cm",
"type" : "unit"
}
```

The unit document contains the 'conversionToBase ' object that can be used for converting values from a unit to its base unit. The 'conversionToBase ' object of the above example shows how to convert values of the unit centimeters to meters.

Units are stored in the raw collection 'unit'. To export units, verify that 'Include Unit Definitions' is set to at least 'Minimum' in the Process Engine configuration.

List Of Values

Exporting a ListOfValues object with the ID 'LOVID', and the name 'List of values' results in the following JSON:

```
{
```

```
"_id" : "LOVID",
"values" : {
"6823" : "This is a metadata value of the lov'"
},
"name" : "List of values",
"validUnitIDs" : [
"unece.unit.CMT",
"unece.unit.MTR"
],
"type" : "listofvalues"
}
```


List Of Values are stored in the raw collection 'listofvalues'. To export List Of Values, verify that 'Include List Of Values Definitions' is set to at least 'Minimum' in the Process Engine configuration.

MongoDB Adapter Setup Quick Guides

The following information is available to assist in the setup of Mongo Delivery Method.

Configuring Mongo Authentication Quick Guide

The following quick guide describes how to configure a Mongo database to use authentication. Details may be found in the Mongo documentation: http://docs.mongodb.org/manual/core/authentication/#inter-process-auth

The guide assumes that the setup is used on a Mongo database installation running as a single server installation (i.e., the Mongo database is running as a standalone server).

1. Start the Mongo database without authentication.
2. Log onto the database using the Mongo client.
3. Create a system administrator user.

```
> use admin
switched to db admin
> db.createUser(
   {
     user: "admin",
     pwd: "password",
     roles:
     [
       {
         role: "userAdminAnyDatabase",
         db: "admin"
       }
     ]
   }
 )
Successfully added user: {
"user" : "admin",
"roles" : [
{
"role" : "userAdminAnyDatabase",
"db" : "admin"
}
]
}
```

4. Create the user that will be used when the MongoDB adapter logs onto the Mongo database. This user should be created in the admin database with roles to read and write any databases in Mongo:

```
> use admin
switched to db admin
> db.createUser( {"user" : "stepsys", "pwd" : "stepsys", "roles" : [ "readWriteAnyDatabase"
] })
Successfully added user: { "user" : "stepsys", "roles" : [ "readWriteAnyDatabase" ] }
```

5. Stop the Mongo database and enable authentication. This may be done in the mongo configuration file:

```
auth = true
```

or from the command line:

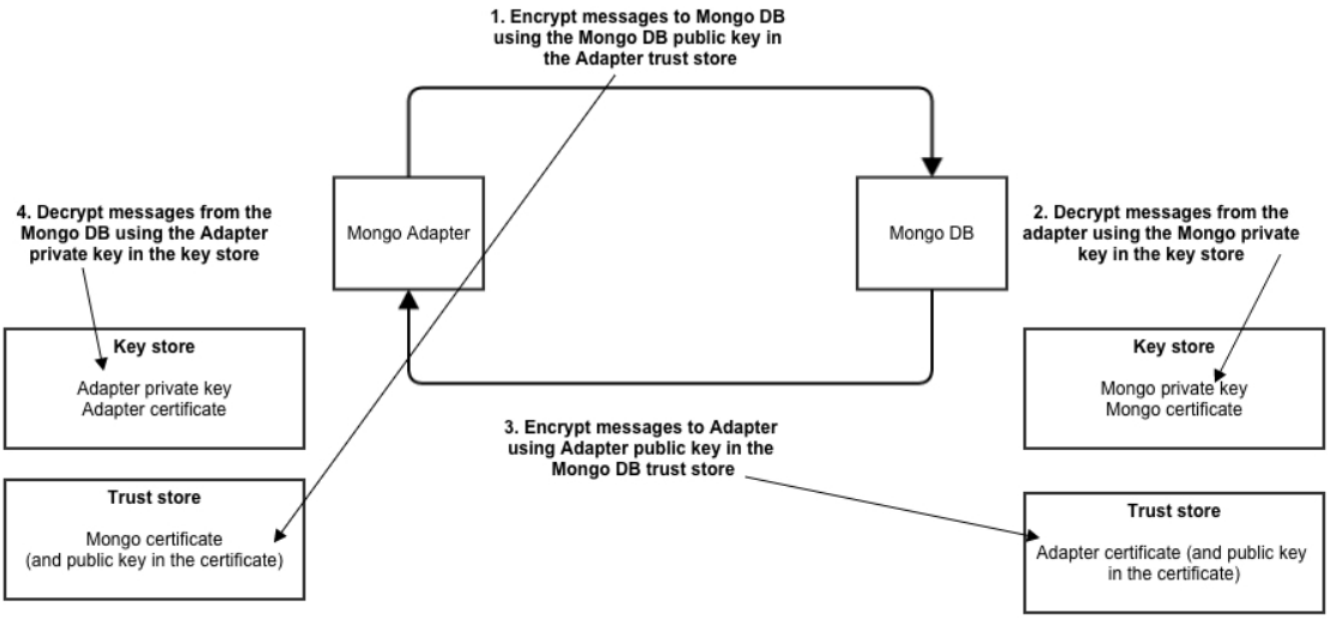
```
mongod --auth --config
```

SSL Configuration Quick Guide

When the Mongo DB adapter is configured for SSL, a key store and a trust store must be configured.

- The key store is a key store file in jks format containing the Mongo Adapter (STEP) private key and certificate.
- The trust store is a trust store file in jks format containing the Mongo DB public key and certificate.

Key store files may be created from the public and private key files and certificates using utilities like the java keytool utility. The diagram below illustrates the uses of the key store and trust store.



A quick guide to configure SSL encryption on the STEP-to-Mongo database connection includes:

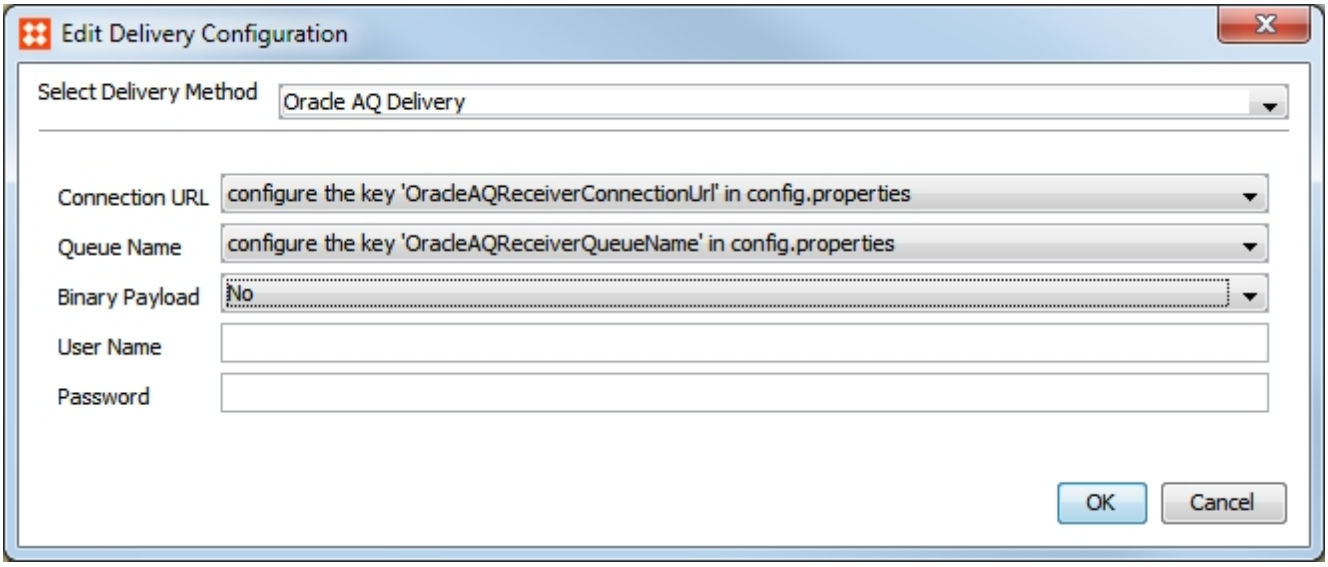
- Configure the Mongo database to use SSL: http://docs.mongodb.org/manual/tutorial/configure-ssl/
- Configure the Mongo client to connect to a Mongo database using SSL: http://docs.mongodb.org/manual/tutorial/configure-ssl-clients/

The Mongo database available as a free download does not support SSL. (SSL requires the Enterprise edition of Mongo.) Instead, use the appropriate link below to download the source and compile your own version of the Mongo database:

- For Mac OS X, see http://docs.mongodb.org/manual/tutorial/install-mongodb-on-os-x/
- For Linux/Windows, see http://www.mongodb.org/about/contributors/tutorial/build-mongodb-from-source/

Oracle AQ Delivery Method

1. In the **Select Delivery Method** list, choose **Oracle AQ Delivery**.

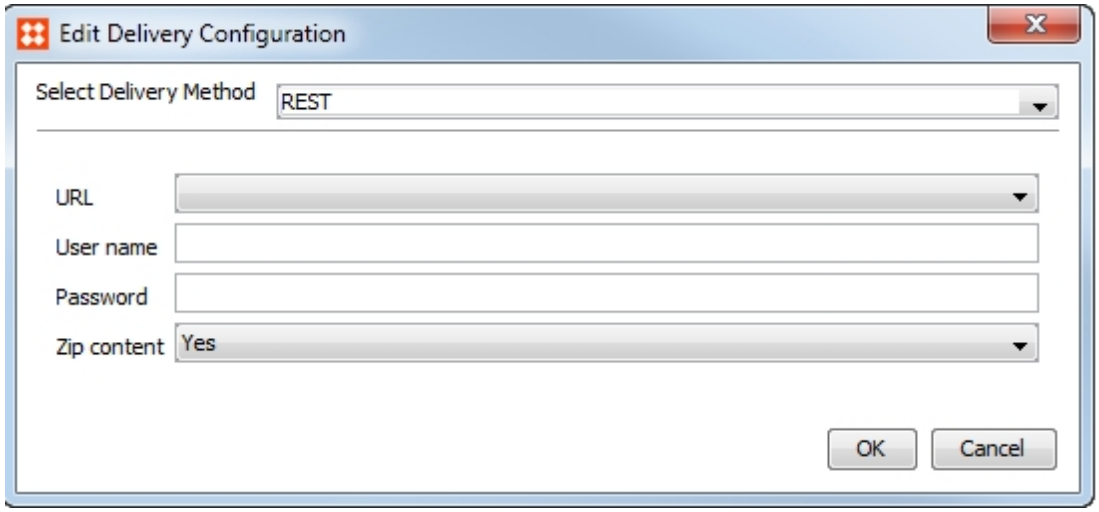


2. In the **Connection URL** list, select a URL that points to Oracle AQ. Options are available based on the OracleAQReceiverConnectionURL property in the sharedconfig.properties file on the application server. Contact Stibo Systems if you need assistance with configuration.
3. In the **Queue name** list, select an Oracle AQ queue name. Options are available based on the OracleAQReceiverQueueName property in the sharedconfig.properties file on the application server. Contact Stibo Systems if you need assistance with configuration.
4. For **Binary Payload**, select Yes or No ('Yes' will be selected for non-text files).
5. In the **User Name** field, enter the user name that will be used to log on to Oracle
6. In the **Password** field, enter the password that will be used to log on to Oracle.

REST Delivery Method

The REST Delivery Method delivers a call-back URL to the REST service and does not include actual STEP data. The data can be fetched from the call-back URL by the receiving REST service.

1. For **Select Delivery Method**, choose **REST**.



2. In **URL**, select the URL that points to the REST endpoint where you would like to receive the delivered data. This should be a URL to a REST PUT method, for example, http://server/rest_URL. A URL where the result of the endpoint can be fetched is returned in the PUT call.

   Note: The available URLs are system dependent and must be configured in the sharedconfig.properties file under the property name RESTDeliveryURL. Contact Stibo Systems if you need assistance with configuration.

3. In **User Name**, enter the user name that will be used to log on to the REST endpoint.
4. In **Password**, enter the password that will be used to log on to the REST endpoint.
5. In **Zip content**, specify whether to zip the contents before upload.

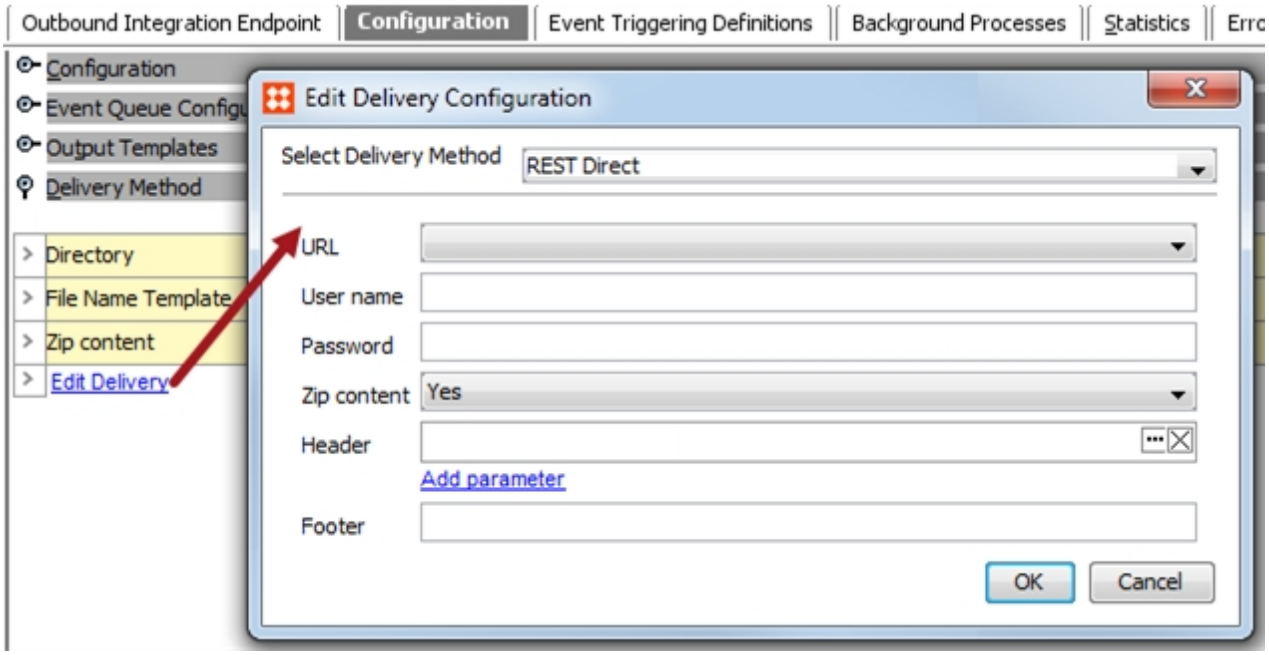
For information about how to use the REST API to upload files to REST, see the STEP API documentation.

REST Direct Delivery Method

The REST Direct delivery method differs from the standard REST delivery method in that the data is delivered directly to the REST service and no call-back URL is required.

The REST Direct delivery method reads the outgoing file into memory before sending. To handle the size of the outgoing file and prevent time-outs and rejections you must scale the heap size. If the delivery fails, the OIEP will be disabled. There is no special resilience handling.

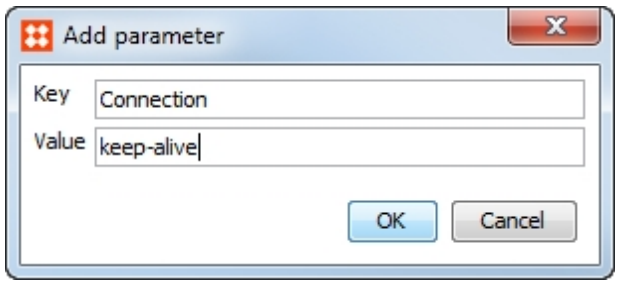
1. For **Select Delivery Method**, choose **REST Direct**.



2. In **URL**, select the URL that points to the REST endpoint where you would like to receive the delivered data.

   Note: The available URLs are system dependent and must be configured in the sharedconfig.properties file under the property name RestDirectDeliveryURL. Contact Stibo Systems if you need assistance with configuration.

3. In **User Name**, enter the user name that will be used to log on to the REST endpoint.
4. In **Password**, enter the password that will be used to log on to the REST endpoint.
5. In **Zip content**, specify whether to zip the contents before upload.
6. For **Header**, click the **Add Parameter** link and add a key and a value. This is part of the HTTP network protocol.



7. On the Add parameter dialog, click the OK button and the Header is displayed in the field. If multiple headers are needed, use the **Add Parameter** link to add each additional key and value pairs

Note: Once a header is displayed, use the ellipsis button (...) to edit it or the X button to remove it.

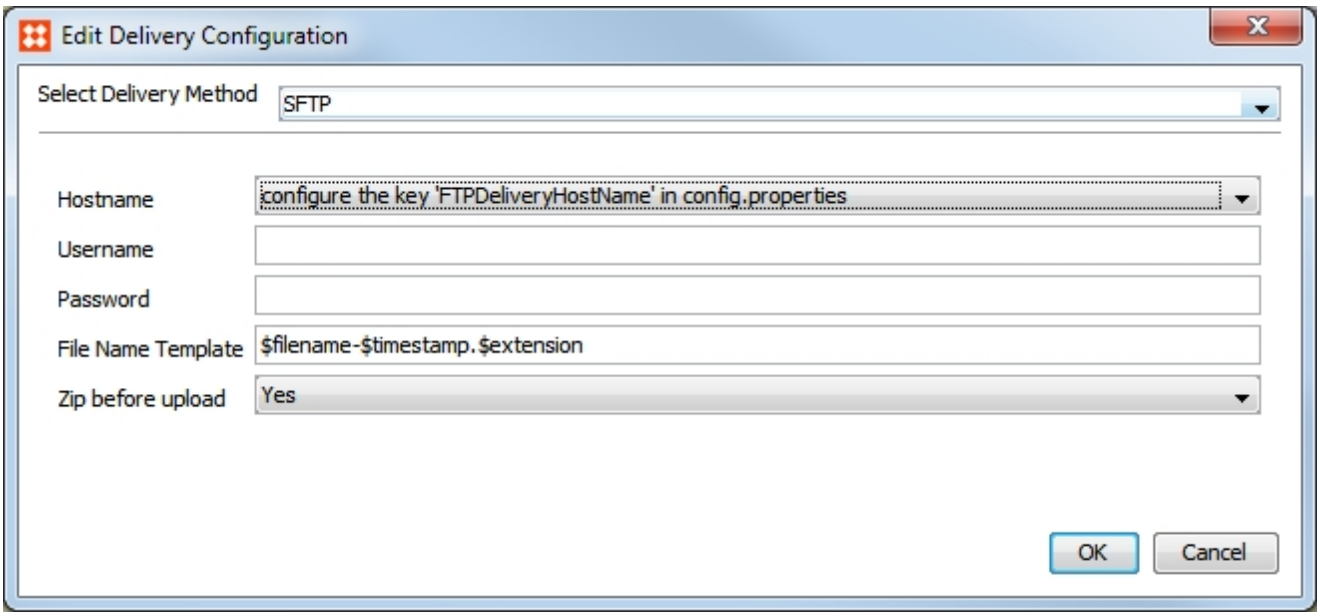
8. In **Footer**, add data required for the recipient to verify that the full message was received. This is part of the HTTP network protocol.

   The footer can be used to mark the end of a multi-part REST call (that is a REST call containing the payload split in more packages). The footer could also contain a checksum that the receiver can use to detect if the payload in a multi-part message has been changed by the middleware.

9. On the Edit Delivery Configuration dialog, click the OK button to save the delivery method.

SFTP Delivery Method

1. In the Select Delivery Method list, select **SFTP**.



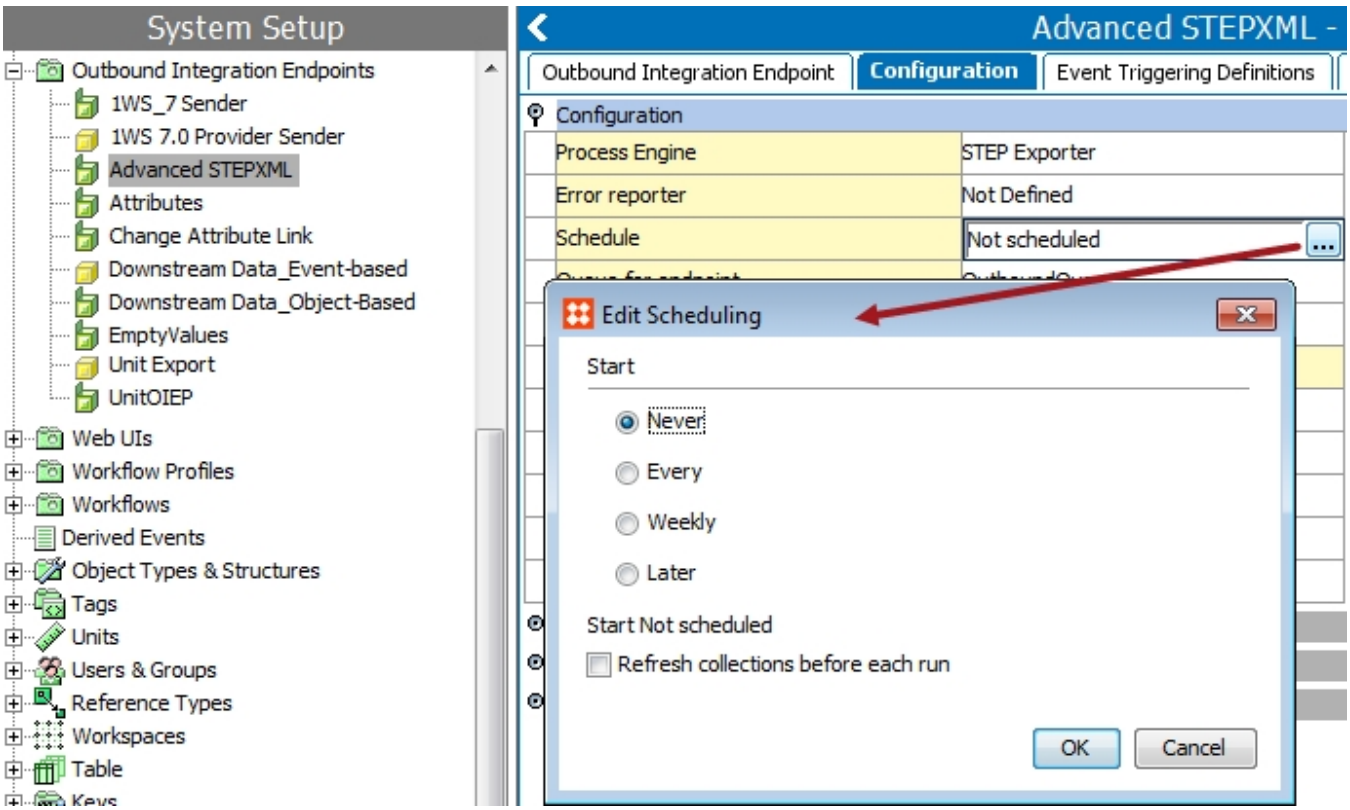
2. In **Hostname**, select the host name to be used for the delivery. Options are available based on the FTPDeliveryHostName property in the sharedconfig.properties file on the application server. The required format of the property is (square brackets not included): FTPDeliveryHostName=1=[host1],2=[host2] Contact Stibo Systems if you need assistance with configuration.
3. In **Username**, enter the username that will be used to log on to the FTP server.
4. In **Password**, enter the password that will be used to log on to the FTP server.
5. In **File Name Template**, enter the file naming template to be used for the deliveries. The default template is $filename-$timestamp.$extension. The following options are available:
   - **$filename** expands to 'exported' for CSV and Excel exports; expands to the Event ID(s) for event-based STEPXML. $filename can be replaced if a different file name should be used.
   - **$timestamp** expands to the time in milliseconds between January 1, 1970 and the time when the file is uploaded (e.g., 1438634758430). You can use $timestamp(ddMMyyyy) to format the timestamp as a date instead.
   - **$extension** expands to the extension of the output file

   Note: The File Name Template does not support conversions of file formats and can only be used to deliver files in the format specified on the Configuration tab of the outbound integration endpoint.
6. In **Zip before upload**, click **yes** to zip contents before uploading the files to the host.

Schedule Outbound Integration Endpoint

On the Configuration tab of an outbound integration endpoint, you can specify how often you want the endpoint to run.

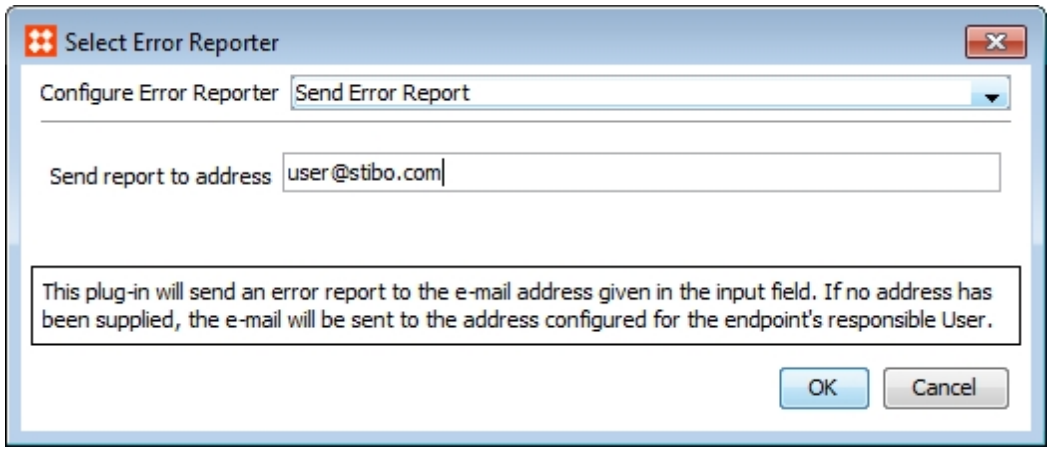
1. In **System Setup**, locate and select the relevant outbound integration endpoint.
2. Click the **Configuration** tab and navigate to the **Configuration** flipper. In the **Schedule** field, click the ellipsis button **(…)**, and then select one of the following:
   - **Never** if you want to invoke the endpoint manually.
   - **Every**, **Weekly**, or **Later** to specify how often you want the endpoint to run. Different options are available depending on the frequency you select.



3. Select **Refresh collections before each run** if the outbound integration endpoint contains collections that should be automatically refreshed before each run.

Outbound Integration Endpoint Error Reporter

On the **Configuration** tab, you can specify whether you want to activate an error reporter. If a background process fails, the error reporter can email information about the failed endpoint, including the background process, failed file, failing process step, cause of the error, and a copy of the file that triggered the error.



Note: On a standard STEP system without customizations, you can only use the **Send Error Report** plug-in.

Configure an Error Reporter

1. In **System Setup** locate and select the relevant outbound integration endpoint
2. Click the **Configuration** tab and navigate to the **Configuration** flipper. In the **Error Report** field, click the ellipsis button **(…)**.
3. In **Configure Error Reporter**, select **Send Error Report**.
4. In **Send report to address**, enter the email address of the error report recipient.

Note: If no email address is entered, the error report will be sent to the email address of the user who created the integration endpoint. If no email is defined for this user, or if no mail server is defined in the configuration, the error report will be written to the failed Background Process Execution Log.

Gateway Integration Endpoints

Gateway integration endpoints enable STEP access to external systems through business rules that make calls to the external system to fetch data or update status.

This is useful, for example, if you are creating products in STEP and need a product ID from an external ERP system to be able to create the object. Using the Gateway endpoint, STEP can access the ERP system and retrieve the ID.

The Gateway integration endpoint controls all access to a given system. It holds the information that is required to access the external system including the user name, password, and the server URL. This means the information does not have to be included in the JavaScript business action and that the duration and frequency of calls can be monitored and logged.

The business rule uses synchronous REST calls from JavaScript. For more information about REST and the REST API, see the STEP SDK documentation.

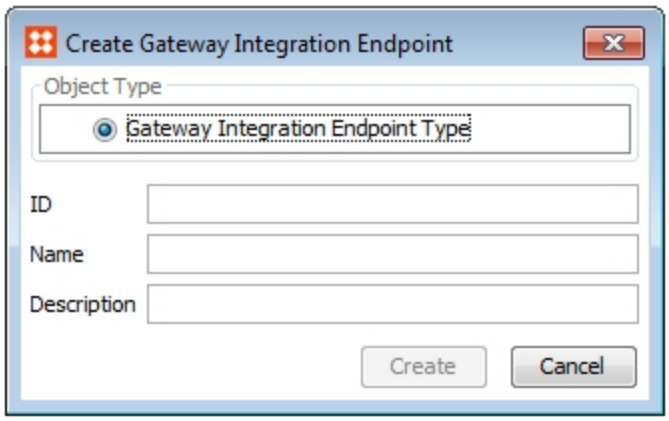
Note: Connecting to a REST destination via an external proxy is not supported. Keystore is not supported.

Create the Gateway Integration Endpoint

To create a Gateway Integration Endpoint, first verify that a setup group is created for integration endpoints.

For more information on how to do this see, Integration Endpoint Setup.

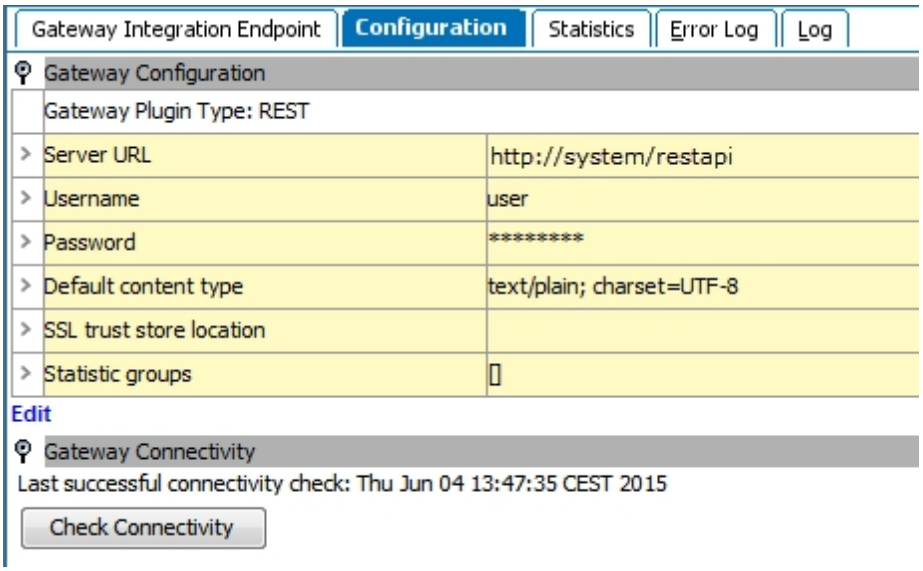
1. In **System Setup**, right-click the **Integration Endpoint** setup group, and then click **Create Gateway Integration Endpoint**.
2. Enter an **ID**, a **Name** and a **Description** of the integration endpoint, and then click **Create**.



When you have created the Gateway integration endpoint, you can edit it in the Gateway integration endpoint editor. The editor has five tabs:

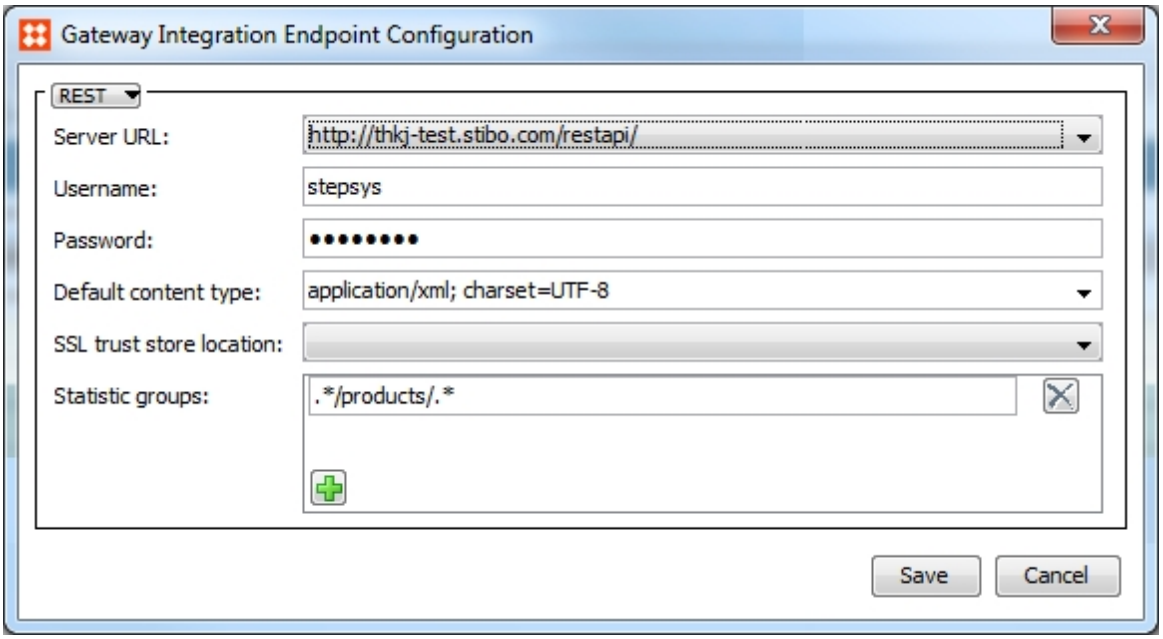
- **Gateway Integration Endpoint**: holds basic information like ID, Name and Enable status.
- **Configuration** tab: holds information about the external server, user credentials, etc. This tab also has a **Check Connectivity** button to check the connection to the external server.

- **Statistics** tab: provides statistics about frequency and timings of external calls using the gateway.
- **Error Log** tab: shows errors reported from external calls using the gateway.
- **Log** tab: provides information about who changed the gateway configuration and when.



Gateway Integration Endpoint Configuration

1. In **System Setup**, right-click the **Integration Endpoint** setup group, and then click the relevant Gateway integration endpoint.
2. On the **Configuration** tab, click **Edit**. **The Gateway Integration Endpoint Configuration** dialog appears. Here, you specify which external system that you want the Gateway integration endpoint to access.



3. In the **Gateway Integration Endpoint Configuration** dialog, specify the external system that you want the Gateway endpoint to access.

   On a standard STEP system no Server URLs are available. Please contact Stibo Systems to have required server URLs enabled on your system.

4. Enter the **Username** and **Password** required to establish the connection.
5. In **Default content type**, enter the data format to be used for the endpoint or select a content type from the list, which contains the most commonly used types.
6. In **SSL trust store location**, select the trust store location. The truststore holds certificates that verify that the system can be trusted. On a standard STEP system no trust store locations are available. If no trust store locations can be selected, the gateway connection is not configured to use SSL encryption.

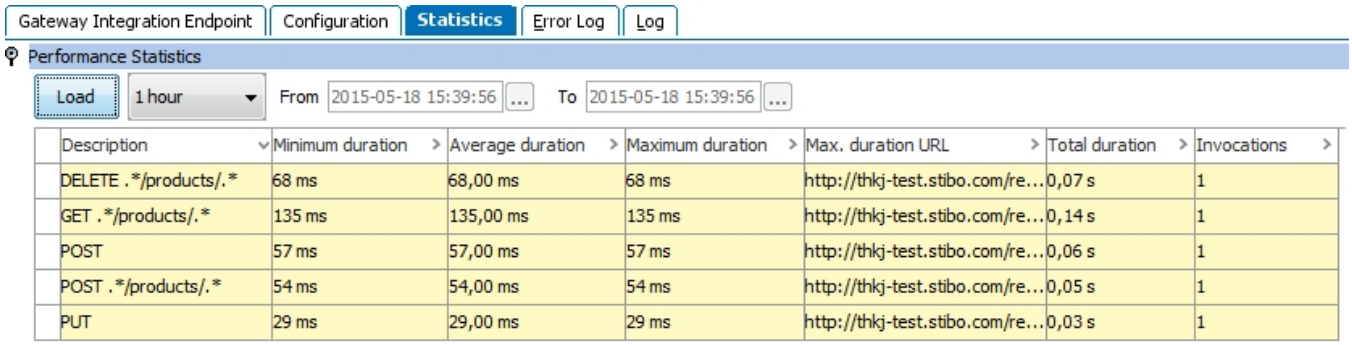
   If the field is empty, the connection will be established using a certificate signed by a recognized CA such as Verisign or Thawte.

   Please contact Stibo Systems if you need specific SSL trust store locations enabled on your system.

7. In **Statistic groups**, enter a regular expression to group executed REST methods in the overview on the **Statistics** tab. By specifying one or more regular expressions, the Statistics tab can display timings for

individual REST calls. Ensure that the regular expression you supply matches the entire URL you want singled out in the Statistics tab, for example, `*/products/.*`. Use the regular expression syntax used in Java in java.util.regex.Pattern class.

This field is optional. If no groups are supplied, the timings shown on the **Statistics** tab are grouped by the HTTP request types GET, POST, PUT, and so on.



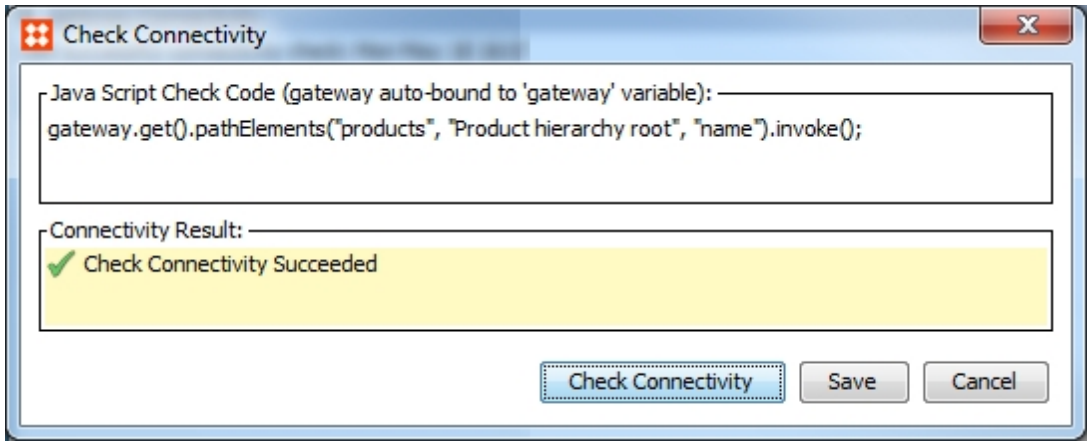
Description	Minimum duration	Average duration	Maximum duration	Max. duration URL	Total duration	Invocations
DELETE .*/products/.*	68 ms	68,00 ms	68 ms	http://thkj-test.stibo.com/re...	0,07 s	1
GET .*/products/.*	135 ms	135,00 ms	135 ms	http://thkj-test.stibo.com/re...	0,14 s	1
POST	57 ms	57,00 ms	57 ms	http://thkj-test.stibo.com/re...	0,06 s	1
POST .*/products/.*	54 ms	54,00 ms	54 ms	http://thkj-test.stibo.com/re...	0,05 s	1
PUT	29 ms	29,00 ms	29 ms	http://thkj-test.stibo.com/re...	0,03 s	1

8. When you have completed the configuration, click **Save**.

Check the Gateway Integration Endpoint Connection

When you have configured the Gateway, you can test the connection.

1. On the **Gateway Integration Endpoint** tab, enable the integration endpoint. It is enabled when the icon in the tree navigator is colored green.
2. On the **Configuration** tab, click **Check Connectivity**.
3. In the **Check Connectivity** dialog, use JavaScript to access the external system. In the following example, the JavaScript gets the name of the "Product Hierarchy root". Click **Check Connectivity** to view the connectivity result.

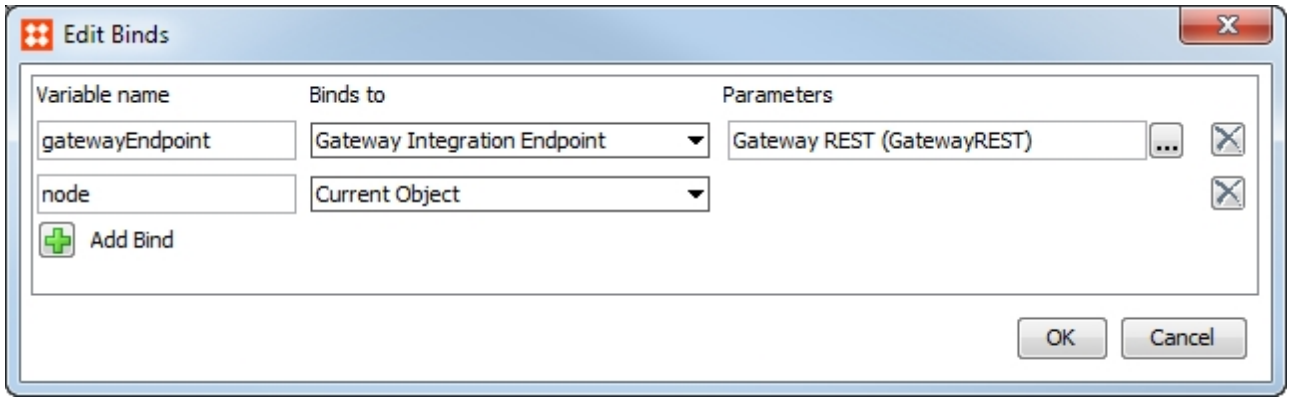


For information about the JavaScript syntax, see the STEP API documentation.

Business Action Accesses Gateway Integration Endpoint

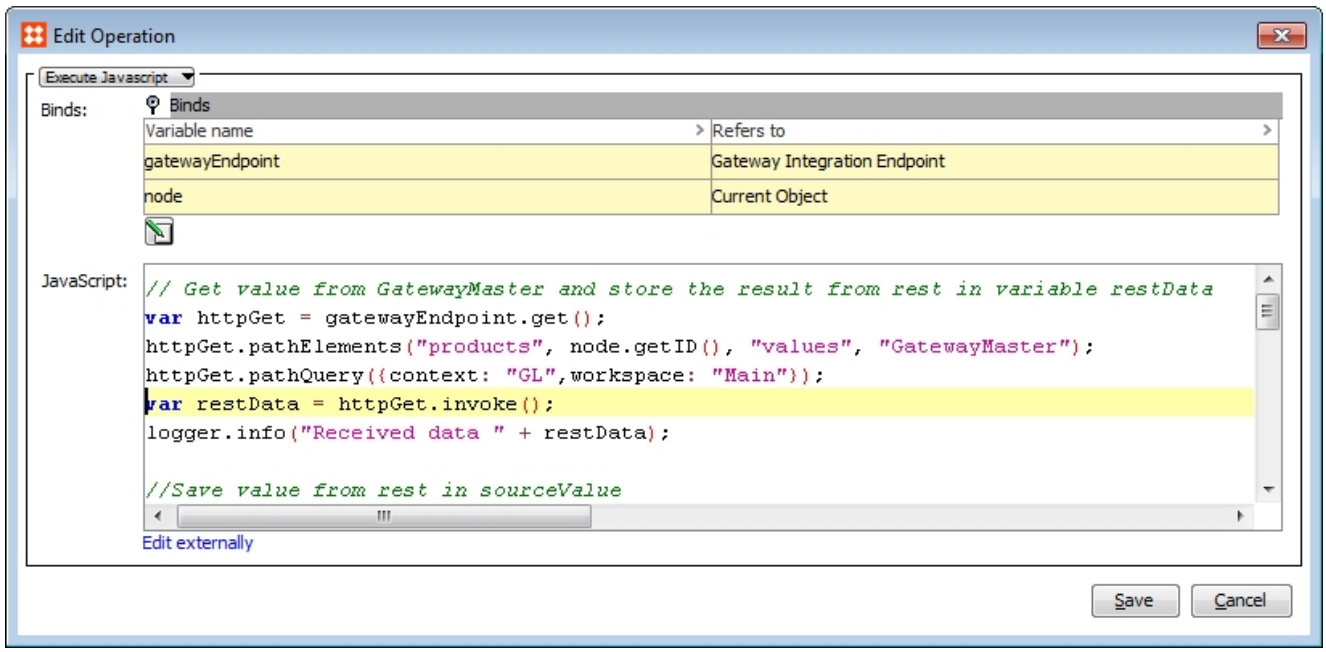
To set up a business action to access a Gateway endpoint, first bind the Gateway endpoint into the JavaScript code using variable names.

1. In **System Setup**, right-click the **Business Rules** setup group, and select **New Business Action**.
2. In the **Business Action** dialog, click **Add New Business Action**, and then select **Execute JavaScript**.
3. Click the Edit Bind button. In the **Edit Binds** dialog, bind the JavaScript to the relevant Gateway endpoint.



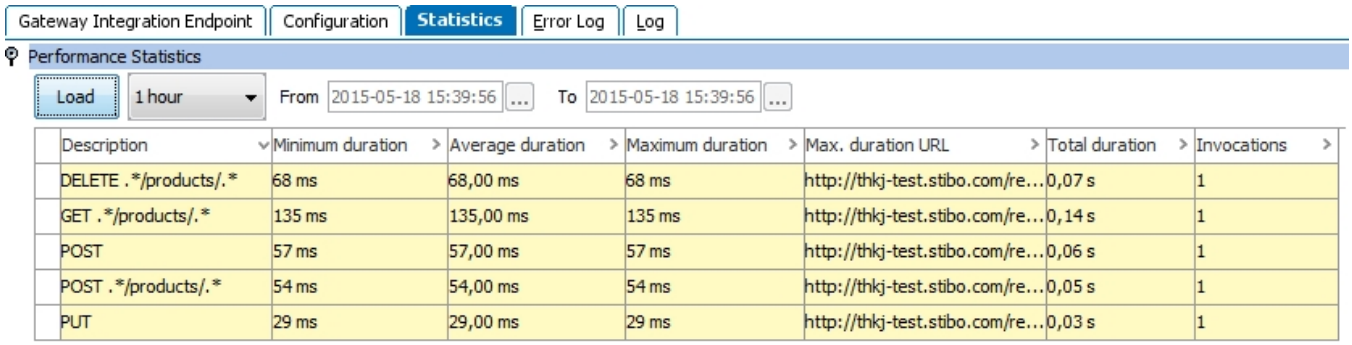
4. In **Edit Operation** dialog, enter the JavaScript to be used to access the Gateway, and then click **Save**.

   The following example uses a REST Gateway integration endpoint to access the external server for information.



5. In the **Edit Operations** dialog, click **Save**.
6. In the **Business Rules** editor, choose the object types where business actions are allowed to be executed.

When the business action is executed, the Gateway endpoint is accessed. The Gateway endpoint **Statistics** tab displays an overview of executed REST methods.



Description	Minimum duration	Average duration	Maximum duration	Max. duration URL	Total duration	Invocations
DELETE .*/products/.*	68 ms	68,00 ms	68 ms	http://thkj-test.stibo.com/re...	0,07 s	1
GET .*/products/.*	135 ms	135,00 ms	135 ms	http://thkj-test.stibo.com/re...	0,14 s	1
POST	57 ms	57,00 ms	57 ms	http://thkj-test.stibo.com/re...	0,06 s	1
POST .*/products/.*	54 ms	54,00 ms	54 ms	http://thkj-test.stibo.com/re...	0,05 s	1
PUT	29 ms	29,00 ms	29 ms	http://thkj-test.stibo.com/re...	0,03 s	1

Gateway Integration Endpoint Binds

Gateway integration endpoints are accessed from JavaScript in business rule conditions and actions. The following table includes the available methods and an example of the syntax.

REST Method	Syntax Example
delete	`delete().path("products").pathQuery({ workspace: "Approved", context: "GL"}).body("Test").invoke();`
get	`get().path("products").pathQuery({ workspace: "Approved", context: "GL"}).invoke();`
head	`head().path("products").pathQuery({ workspace: "Approved", context: "GL"}).invoke();`
options	`options().path("products").pathQuery({ workspace: "Approved", context: "GL"}).invoke();`
post	`post().path("products").pathQuery({ workspace: "Approved", context: "GL"}).invoke();`
put	`put().path("products").pathQuery({ workspace: "Approved", context: "GL"}).body("Test").invoke();`

For more information, see the **Business Action Accesses Gateway Integration Endpoint** section of the **Integration Endpoints** documentation.

For more information about the available methods, see the STEP API documentation.

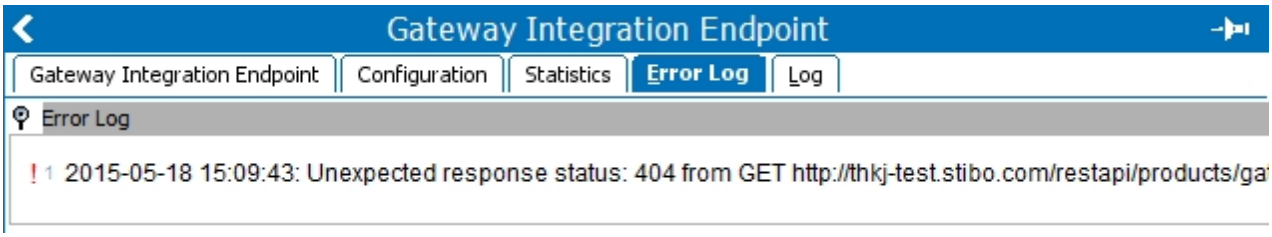
Gateway Integration Endpoint Errors

Errors may occur when business actions are used to access Gateway endpoints. Typical errors include:

- The external system could not be accessed
- The external system did not contain the data being requested in the business action

In these cases, an unexpected response status is received and logged in the in the **Error Log** tab on the Gateway integration endpoint. Here, you can get an overview of all logged errors on the Gateway integration endpoint.

In the following example, a method has been executed to retrieve data on an object which did not exist on the external system. An unexpected response status logged.



You can also monitor the gateway endpoint externally without logging on a STEP system to get an overview of unexpected errors.

This can be accomplished by going to:

- http://<appserver>/admin/monitoring to view a list of all Gateway endpoints
- http://<appserver>/admin/monitoring/<Gateway-ID> to monitor a specific endpoint , as shown in the following example.

Sensor status for GatewayIntegrationEndpointStatus-GatewayEndpointCheckConnection

Plugin	GatewayIntegrationEndpointStatus
Sensor	GatewayEndpointCheckConnection
Status	**Warning**
Created	Thu May 21 11:01:02 CEST 2015 (28 seconds ago)
TTL	30 seconds
Short message	Endpoint is enabled, and has errors - but has no recent errors since 2015-05-20 11:01:02
Long message	The endpoint has errors: 2015-05-18 15:09:43: Unexpected response status: 404 from GET http://thkj-test.stibo.com/restapi/products/gatewayproductdoesnotexist/values/Description?context=GL&workspace=Main http-equiv="Content-Type" content="text/html; charset=ISO-8859-1"/ Error 404 Not Found HTTP ERROR 404 Problem accessing /restapi/products/gatewayproductdoesnotexist/values/Description. Reason: Not Found / Powered by Jetty://

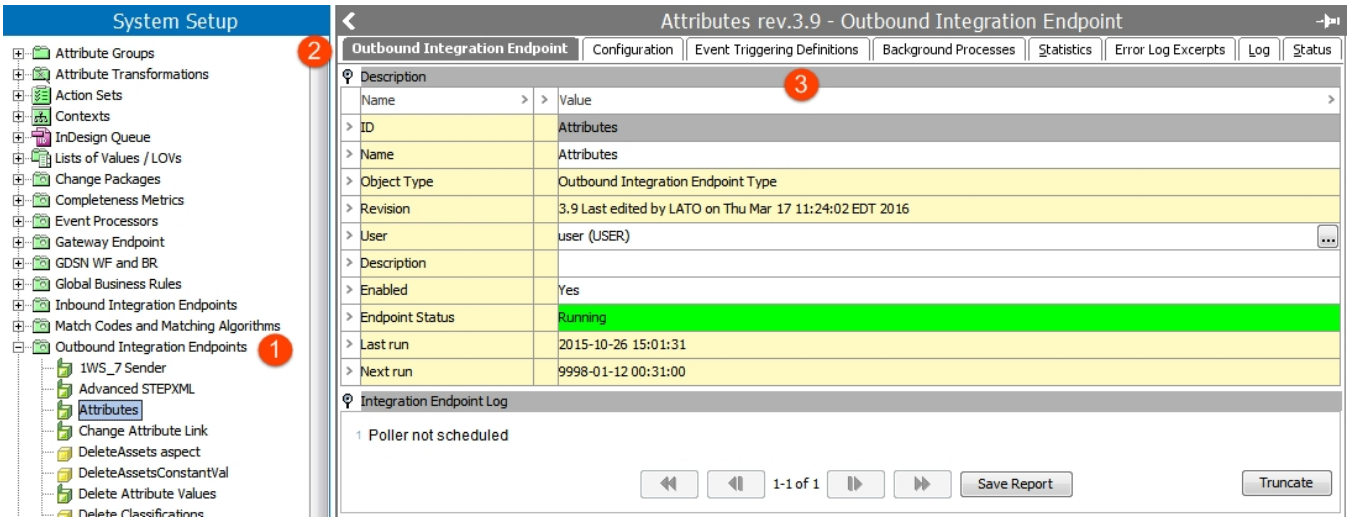
Integration Endpoint Editor

The editors for the inbound and outbound integration endpoints are very similar. However, the number of tabs that are available is different. The **Outbound Integration** editor has a **Configuration** tab and, if the data source is event queue, it also has a **Event Triggering Definitions** tab.

While most of the configuration settings of inbound integration endpoints are specified in the wizard (discussed in the **Inbound Integration Endpoint Wizard** section), for outbound endpoints, you can only specify some basic configuration settings in the wizard (discussed in the **Outbound Integration Endpoints** section).

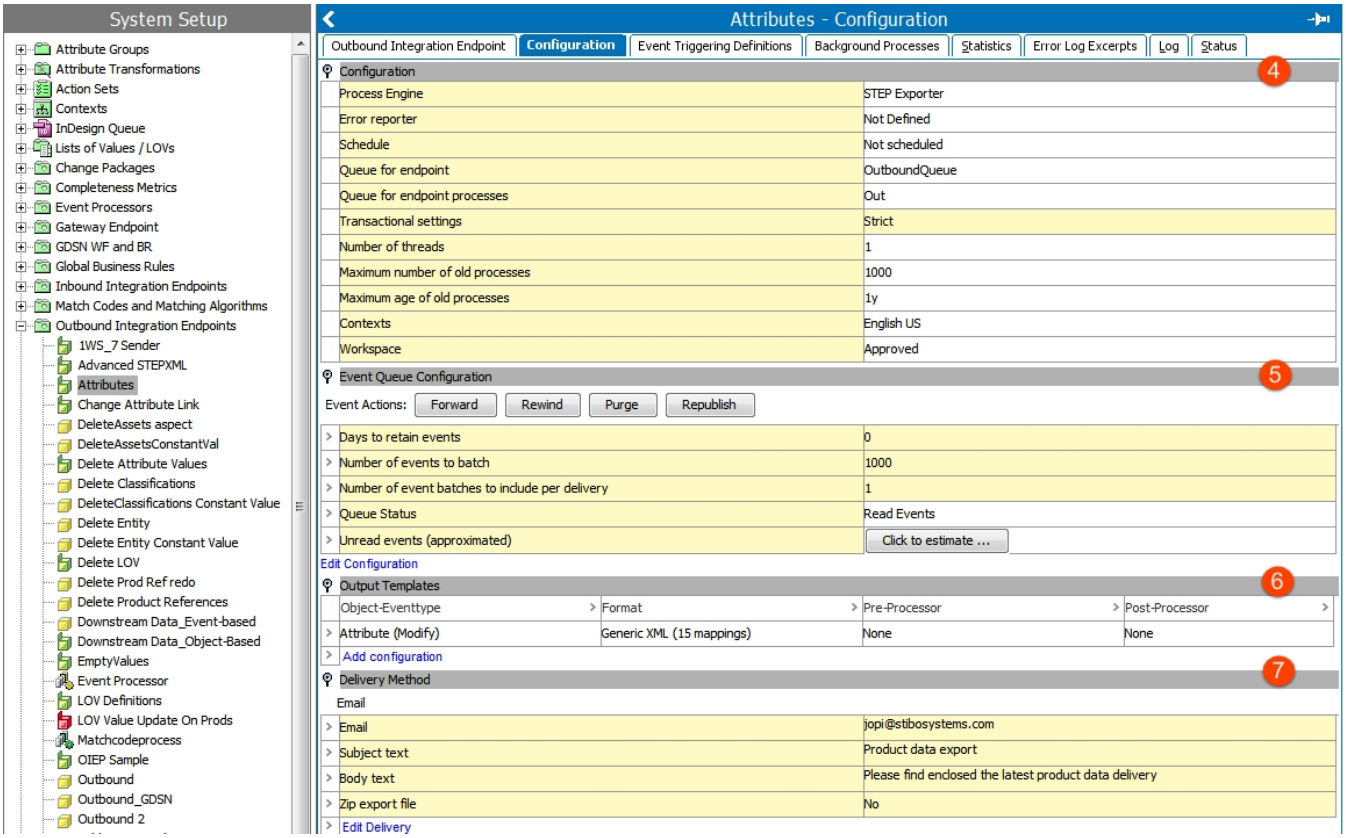
Outbound Editor

Since the majority of outbound integration endpoint settings must be specified in the **Outbound Integration Endpoint** editor, the following images show the Outbound Integration Endpoint editor.



Number	Area	Description
1	**Setup Groups**	Integration endpoints are created and maintained in Setup Groups in System Setup. Before you can create any Integration Endpoints, you must define the setup group(s) in which integration endpoints can be created. When the Setup Group hierarchy is expanded, the color of the Integration Endpoint icon indicates the status of the endpoint. For an explanation of the colored icons, see **Integration Endpoint Status**. The endpoint status is also shown in the endpoint editor on the Integration Endpoint tab > Description flipper > Endpoint Status field.
2	**Tabs**	Both the Outbound and the Inbound Integration Endpoint editor have a number of

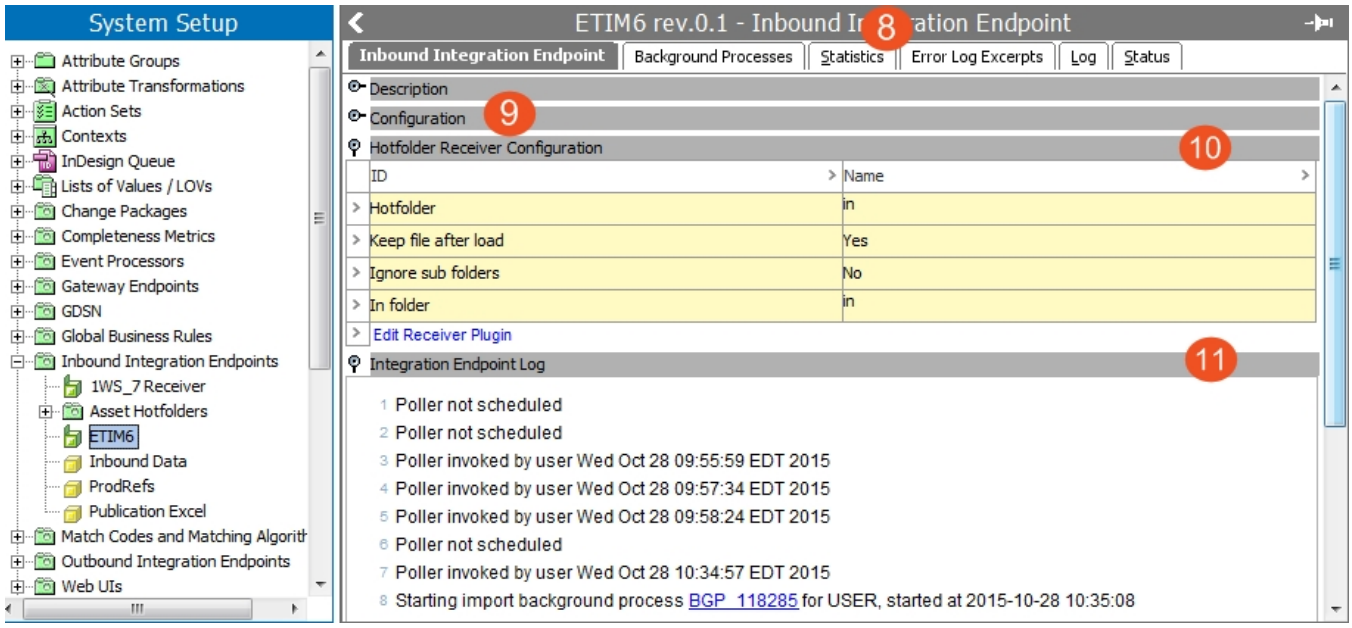
Number	Area	Description
		tabs available: • **Outbound Integration Endpoint**: Contains the description of the endpoint and the background process of the endpoint. • **Configuration**: Contains the configuration of the endpoint. • **Event Triggering Definitions**: Holds the configuration of the events triggering. • **Background Processes**: Provides an overview of background processes started by the endpoint. For more information, see the **Integration Endpoint Background Process** section. • **Statistics**: Provides an overview of processed data and data in queue to be processed. For more information, see the **Integration Endpoint Statistics** section. • **Error Log Excerpts**: Shows data from the main Java log file related to failed background processes with Log Level > 'Info'. Click a hyperlink to a failed background process and correct the cause of the failed background process. • **Log**: Shows information about when the endpoint was created or edited, along with the date, time, and user taking the action. Not all changes are tracked. • **Status**: Displays the revision history and any comments that were entered if a manual revision was created. The revisions allow a user to purge or revert to a past revision. For more information on revision functionality, see STEP Super User Guide / System Setup documentation.
3	**Description**	Displays high-level information about the endpoint. You can also see whether the endpoint is enabled or not.



Number	Area	Description
4	**Configuration**	Provides information about the processing engine used, when the endpoint is scheduled to start, which background process queue it is allowed to use, and how the endpoint should handle background processes.
5	**Event Queue Configuration**	Shows the status of the event queue and the number of unread events. You can also manually remove events from the queue.
6	**Output Templates**	Allows you to view, edit, and configure output templates. For more information on output templates, see the **Outbound Integration Endpoint Output Template** section of the **Outbound Integration Endpoint** documentation.
7	**Main Background Process**	The main background process log is shown in the integration endpoint main editor and shows information about background processes being started. Whenever you restart your STEP system, the main background process is automatically started again. The log remains, so you can always get an overview of started background processes.

Inbound Editor

The majority of inbound settings match those described above for outbound integration endpoint settings. The options that are different are shown below.



Number	Area	Description
8	**Statistics**	Key measures are Endpoint uptime, Waiting in data source, Number of failed requests (Background Processes with errors), and Mean run time (average processing time per import). **Note:** Receiver plugins poll the data source to check for waiting files / messages more frequently than the IIEPs are invoked.
9	**Configuration**	This flipper includes many of the same settings as are available on the Configuration tab for outbound integration endpoints.
10	**Receiver Settings**	This section differs depending on the selected Receiver plugin. Clicking 'Edit Receiver Plugin link opens the inbound endpoint wizard on step 2.
11	**Integration Endpoint Log**	The Execution Report for the endpoint background process. An entry is generated each time the endpoint is invoked, when background processes generated by the endpoint are started, and when processing errors occur. You can access the individual Importer Background Processes by clicking the links included in the log.

Integration Endpoint Status

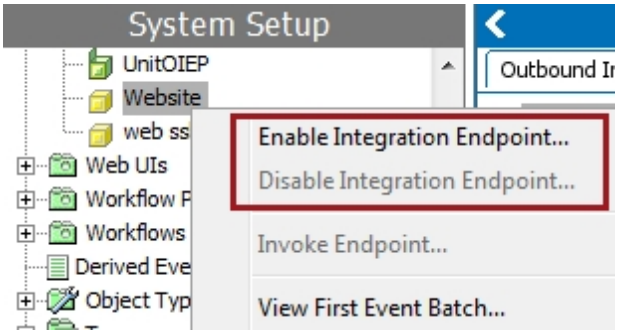
Expand the integration endpoints hierarchy to display the integration endpoint status icons.

Icon	Endpoint Status	Description
	Enabled	The integration endpoint is active and connected.
	Disabled	The integration endpoint has stopped and no data is being imported / exported. Newly created endpoints and endpoints that have been manually disabled will have this status.
	Failed	An error occurred during processing that caused the integration endpoint background process to fail. No data will be imported / exported until the endpoint has been manually resumed / reactivated. For more information, see **Correcting Failed Integration Endpoint Background Processes** in the **Integration Endpoint** documentation.
	Enabled	The Gateway integration endpoint is active and ready to use.
	Disabled	The Gateway integration endpoint has been disabled by user request.

For information about how the Endpoint Status and Queue Status settings work together on event-based endpoints, see the **Event-Based Outbound Integration Endpoint Status and Queue Status** section of the **Integration Endpoints** documentation.

Enable or Disable an Integration Endpoint

1. In **System Setup**, expand the relevant **Setup Group** and select the **Integration Endpoint** you want to enable or disable.
2. Right-click the endpoint, and then select **Enable Integration Endpoint** or **Disable Integration Endpoint**.



Integration Endpoint Background Processes

This is not relevant for Gateway Integration Endpoints.

An active integration endpoint uses an associated background process to handle the scheduled invocation of the endpoint. The execution report for this background process is integrated in the workbench endpoint editor and includes invocation information as well as messages logged from the plugins. When using the standard STEP Exporter Processing engine, the actual import / export is handled in separate background processes started by the endpoint, as illustrated below.

When configuring an integration endpoint that uses the STEP Importer processing engine, you will specify a background process queue to use for each of the two types of background processes:

- Queue for endpoint
- Queue for endpoint processes

Typically, all background processes of the same type (same process type ID) will use the same queue. However, integration endpoint processes can be tied to different integration endpoints and use different queues.

Selecting the Background Process Queue

The main background process uses a default background process queue: **InboundQueue** for inbound integration endpoints and **OutboundQueue** for outbound endpoints.

However, when you create an integration endpoint, you can specify that you want to use a different background process queue for the main integration endpoint background process queue.

- For IIEPs, specify the default background process in the Inbound Integration Endpoint wizard Step 3 - Configure Endpoint.
- For OIEPs, specify the default background process in the integration endpoint editor: Configuration tab > Configuration flipper > **Queue for Endpoint** field.

Background Process Queue Size Property

The queue size setting can have great impact on processing. The background process queue size property determines the number of background processes can be executed in parallel on the queue. If the size is 1, only one background process can run at a time. If the size is 2, two processes can run in parallel, and so forth.

Important: Although the integration endpoint configuration wizards suggest the number of queues to use, be aware that if you create multiple endpoints and do not change the default queue suggestions, your setup could include endpoints that block each other since the queues have a default size of 1.

This is not a big concern for the endpoint processes that, most of the time, are idle and only take up a slot on the queue briefly when the endpoint is invoked (such as when polling a hotfolder for new files). However, it can be an issue for the generated background processes that perform the actual export processing. The decision as to whether different endpoints should use the same or different processing queues and the size of the Queues should therefore be an informed one.

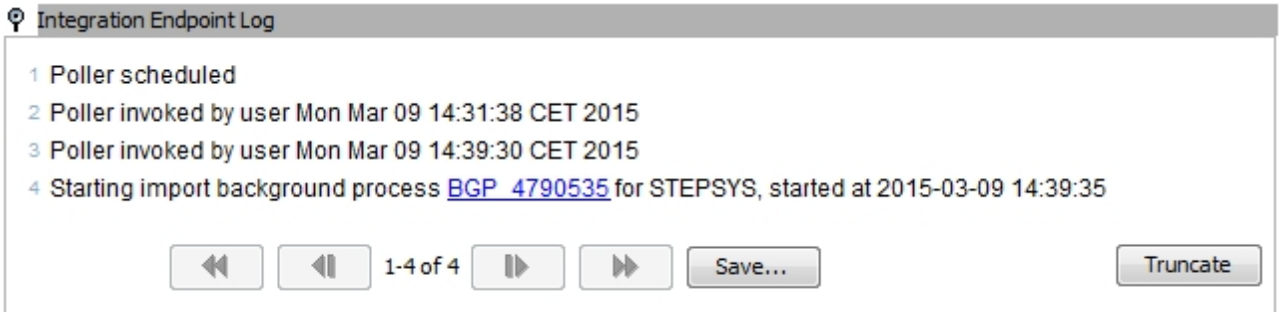
Use the following sharedconfig.properties entry to overwrite the default queue size value:

```
BackgroundProcess.Queue.[name of queue].Size = [number of allowed parallel processes]
```

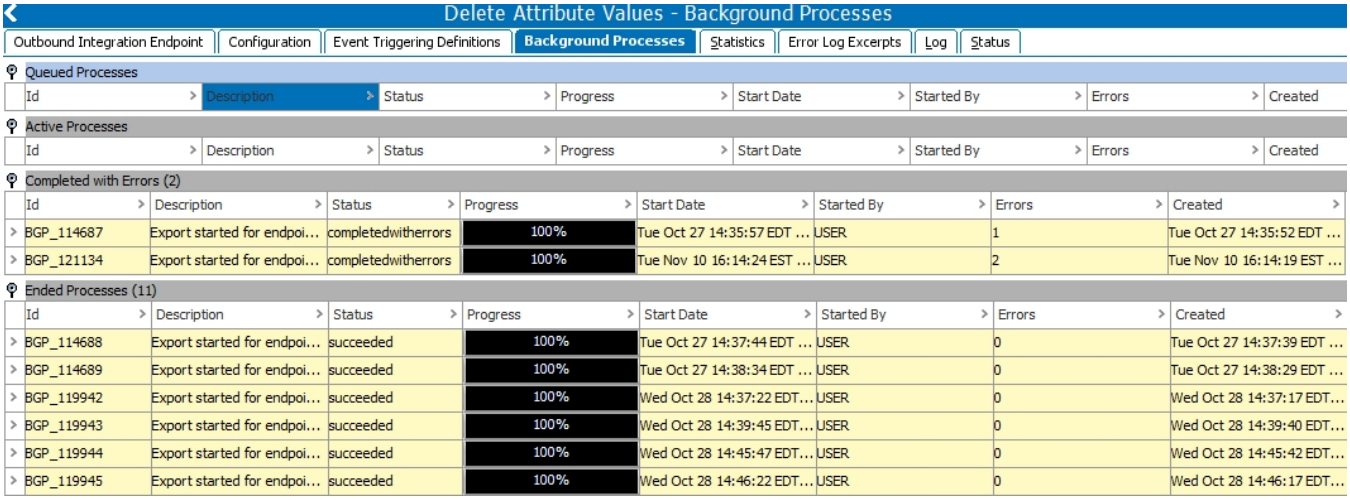
Monitoring Integration Endpoint Background Processes

Use the following steps to determine which background processes have been started. By default, the started integration endpoint background processes are not visible on the BG Processes tab.

1. In System Setup, select the relevant integration endpoint, and click the **Integration Endpoint** tab.
2. Open the **Integration Endpoint Log** flipper. A list of started background processes is displayed.



3. Click a background process link to display the actual background process and see detailed information.
4. Click the **Background Processes** tab to To view a list of the background processes of an integration endpoint, to view all background processes, the status, progress, and more.



Background Process Flippers

The background processes are divided into the following groups:

Background Processes	Description
Queued Processes	Lists all background processes that are started in the state "waiting". The maximum number of background processes that can be started in the state waiting is specified in the integration endpoint configuration.
Active or Failed Processes	Lists the background processes that are currently being processed. If data is processed as strict or chained, the endpoint only has one active background process at a time. Otherwise, it is possible to apply more background process queues to work in parallel. In this case the endpoint can have more active background processes at a time. For more information, see Correcting Failed Integration Endpoint Background Processes.
Completed with Errors	Lists succeeded background processes but where an error has been detected.
Ended Processes	Lists succeeded background processes. The number of ended processes is determined by the configuration.

Correcting Failed Integration Endpoint Background Processes

You can reactivate a failed integration endpoint may using Disable / Enable or by Resuming.

Note: This is not relevant for Gateway Integration Endpoints.

If a background process fails and the processing of data is set to **strict**, you have to restart the endpoint before the endpoint is able to start another background process.

If a background process fails and the processing of data is set to **chained**, the remaining background processes that belong to the chain will also fail. You have to correct the failed background process before the endpoint is able to start another background process that belong to the same chain. The endpoint will, however, continue to process data from other chained background processes.

Disable and Enable a Background Process

Setting an integration endpoint to **Disable** and then **Enable** *does clear* the log file.

- Right-click the endpoint and select **Disable Integration Endpoint**.
- Right-click the endpoint again, and select **Enable Integration Endpoint**.

Resume a Background Process

Resuming an integration endpoint *does not clear* the log file

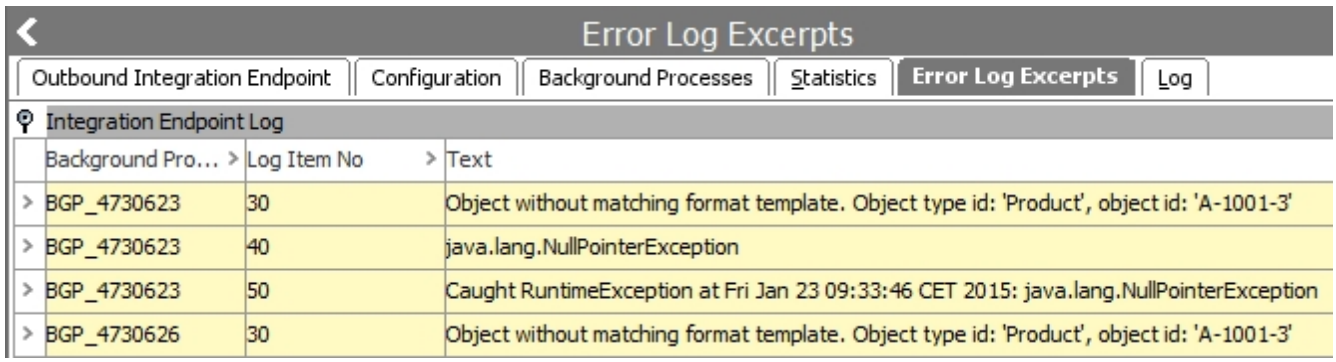
Note: Whenever the STEP system is patched or restarted, the main background process is automatically restarted. The integration endpoint log still exists, and continues to show an overview of the started background processes.

- Right-click the endpoint, and select **Resume Integration Endpoint**.

Resolving a Failed Background Process

1. In **System Setup** select the relevant integration endpoint, and then click the **Error Log Excerpts** tab to display a list of failed background processes and the description of the failure.

   You can also see if a process has been suspended because it is dependent on a failed process.



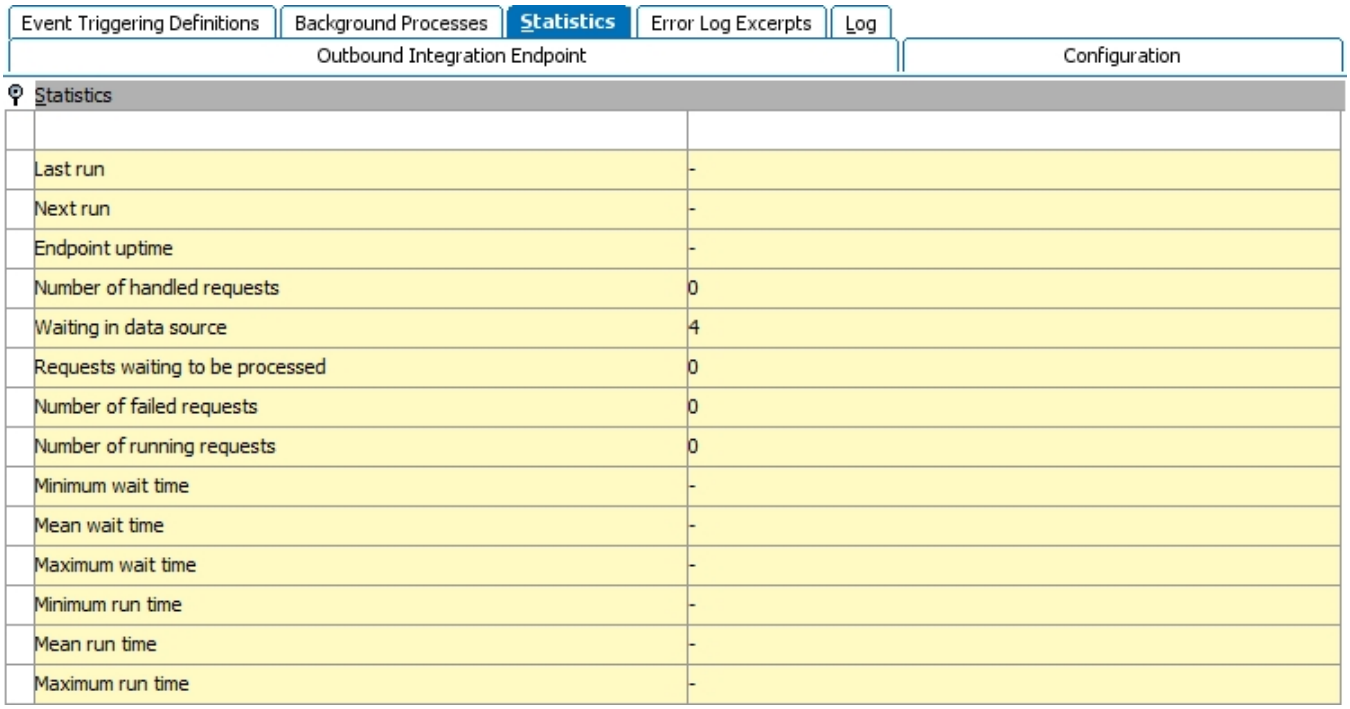
2. In the **Background Process** column, click the link for the background process to correct.
3. In the Background Process editor **Execution Report**, view the progress of the background process, and where it failed.
4. In the Properties area > **Value** column, click the Save button near the failed status, and save the file to your computer.
5. Open the file in a relevant editor and make the required changes.
6. After completing the changes, upload the file back to the same background process and restart the process.
7. In the Background Process editor > Properties area > **Value** column, click the Upload button near the failed status.
8. In the Properties area > **Value** column, click the Restart button near to the failed status. The background process is restarted.
9. If the background process is part of a batch of chained processes, you also have to restart the other processes that are part of the chain.

Integration Endpoint Statistics

The Statistics tab displays information about processed data and data waiting to be processed. It also provides information about the last time the endpoint processed data, and the next time the endpoint will process data as well as information about how long it takes to run the endpoint.

For Gateway integration endpoints, the statistics page enables you to query for frequency and timings of external calls using the gateway. For more information, see Gateway Integration Endpoints.

In **System Setup**, locate and click the relevant integration endpoint, and then click the **Statistics** tab.



Last run	-
Next run	-
Endpoint uptime	-
Number of handled requests	0
Waiting in data source	4
Requests waiting to be processed	0
Number of failed requests	0
Number of running requests	0
Minimum wait time	-
Mean wait time	-
Maximum wait time	-
Minimum run time	-
Mean run time	-
Maximum run time	-