ibi™ iWay® Service Manager

iWay Integration Solution for EDI X12 User Guide

Version 9.2.0 | November 2024

Contents

Introducing the iWay Integration Solution for EDI X12

The iWay Integration Solution for EDI X12 transforms Electronic Data Interchange (EDI) documents into standard XML format, or transforms XML representations into EDI format.

This section provides an overview of EDI and describes the features that are provided by the iWay Integration Solution for EDI X12.

A Brief History of Electronic Data Interchange

Electronic Data Interchange (EDI) is a set of standards for formatting information that is electronically exchanged between one business and another, or within a business. These standards describe how documents for conducting certain aspects of business—such as purchase orders and purchase order acknowledgements—are structured.

By specifying a standardized, computer-readable format for transferring data, EDI enables the automation of commercial transactions around the world. It provides a common, uniform language through which computers can communicate for fast and efficient transaction processing.

Early Standardization Efforts

Before the development of standards, many businesses used proprietary systems to exchange trading information such as purchase orders and invoices. However, they recognized the economic need for a faster, less costly way to process information in order to stay competitive in the business world. Business sectors such as transportation, grocery supply, and banking drove the creation of standards for the communication of data.

In 1968, the United States Transportation Data Coordinating Committee (TDCC) was formed to oversee the design and development of format standards for transportation documents. In 1975, the TDCC released its first standard, the Rail Transportation Industry Application.

The Rail Transportation Industry Application focused on the content of a message—rather than the means of transmission—through the use of transaction sets. A *transaction set* is a business document that consists of an arrangement of data segments. The data segments include data elements in an exact order. The concept of the transaction set is the basis of the EDI ANSI X12 standard created later and widely used today.

About the same time that the TDCC was formed, the United Kingdom started its own effort to develop standard transaction documents for trans-Atlantic trade. The U.K. Department of Customs and Excise, with the help of the British Simplification of Trade Procedures Board (SITPRO), developed a competitive document standard for international trade, named TRADACOMS.

The ANSI X12 and UN/EDIFACT Standard

Standards development progressed in 1979, when the American National Standards Institute (ANSI) chartered the Accredited Standards Committee (ASC) X12 to develop a uniform standard for electronic, inter-industry business transactions. The United States Electronic Data Interchange (EDI) ANSI X12 standard, which resulted from the efforts of the committee, extended and ultimately replaced the standards created by the TDCC.

In 1988, the United Nations chartered UN/EDIFACT (United Nations Electronic Data Interchange for Administration, Commerce, and Transport) to develop a worldwide, internationally approved standard structure for exchanging information among partners. The UN/EDIFACT standards are called United Nations Standard Messages (UNSM). They are comparable to the ANSI ASC X12 transaction sets.

EDI is the standardized data format used for the majority of the world electronic business transactions. Many companies use either the ANSI X12 or UN/EDIFACT standard, or both.

With over 275 transaction sets, the ANSI X12 standard is used to perform nearly every aspect of business operation such as order placement and processing, shipping and receiving, invoicing and payment, pricing and sales, and inventory. It streamlines the communication of data to and from a broad range of entities, including financial and education institutions, insurance providers, food and pharmaceutical suppliers, retailers, automotive manufacturers, and federal and state government.

Features of the iWay Integration Solution for EDI X12

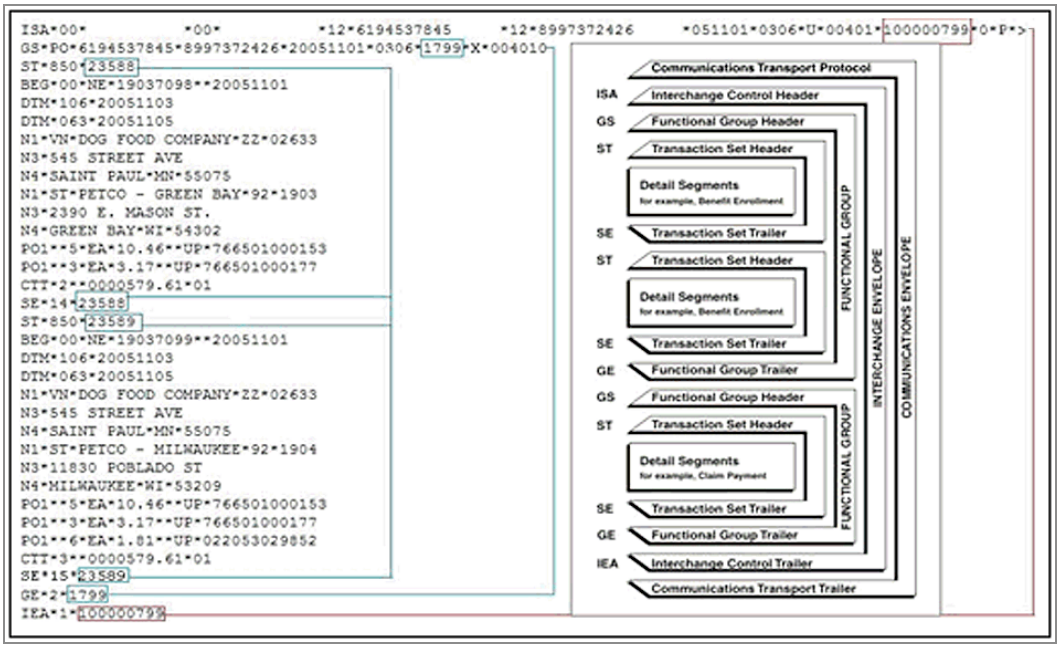
The standards-based iWay Integration Solution for EDI X12 reduces the amount of effort it takes to integrate Electronic Data Interchange (EDI) documents with your internal enterprise applications and third-party trading partners. It includes conversion and validation of documents from EDI to XML format, making it easy to include EDI documents in your XML-based integration projects.

Features of the iWay Integration Solution for EDI X12 include:

- Integration with ibi™ iWay® Service Manager to provide bi-directional conversion of EDI formats and XML.
- Integration with iWay TPM to provide routing, custom transformation by document, and other value-added features.
- Integration with more than 200 other information assets, including J2EE-based back-office systems; data structures such as DB2, IMS, VSAM, and ADABAS; and front-office systems based on Sybase.
- Integration with leading application servers, integration brokers, and development environments. Supported software platforms include BEA WebLogic, IBM WebSphere, Sun Java Enterprise System, and Oracle Application Server.
- Support for synchronous and asynchronous bi-directional interactions for EDI documents between application servers, integration brokers, third-party software packages, and messaging services.
- Support for EDI ANSI X12 transaction sets. For details on the supported transaction sets, see Ebix-Supported Transaction Sets.
- Reusable framework for parsing, transforming, and validating EDI documents without the need to write custom code.
- Data dictionary approach that facilitates EDI-to-XML transformations. The iWay Integration Solution for EDI X12 uses dictionaries to transform data from EDI format to any other format, or from any format to EDI format. It supports flat files, comma-delimited files, popular relational database formats, XML, and more.
- Pre-built data dictionaries, XML schemas, transformation templates, and rule files for automatic transformation and validation of input and output documents.

EDI X12 Transmission Envelope Structure

The following image illustrates a typical EDI X12 envelope structure that is used during a purchase order transmission. Syntax for an EDI X12 document containing two purchase orders is shown on the left and a graphical representation of the hierarchy is provided on the right.



An EDI X12 document must contain the following segments in its structure:

- **Interchange Control Header (ISA).** Indicates the start of the interchange. The ISA segment has a fixed length and consists of 106 characters. The fourth character, for example, an asterisk (*), is the segment delimiter that is used throughout the document.
- **Functional Group Header (GS).** Indicates the start of a group, which contains one or more transaction sets. The GS segment contains various sender and receiver codes for identification and control purposes.
- **Transaction Set Header (ST).** Indicates the start of a transaction set. The transaction set contains segments that make up the message data. All of the details that are required to process the transaction are available within the transaction set. A transaction set can contain one or more loops, which are required to repeat a collection of related segments.
- **Transaction Set Trailer (SE).** Indicates the end of a transaction set. The SE segment

provides a count of the data segments that includes the header and trailer segments.

- **Functional Group Trailer (GE).** Indicates the end of the group. The GE segment contains an element that indicates the number of transaction sets within the group.
- **Interchange Control Trailer (IEA).** Indicates the end of the interchange. The IEA segment contains an element that indicates the number of groups within the interchange.

Deployment Information for Your iWay Integration Solution

This topic describes the iWay products used with your iWay Integration Solution for EDI and provides a roadmap to full information on those products.

It also introduces the concept of a channel for the construction of a message flow in iWay® Service Manager.

iWay Products and Components

Your iWay integration solution works in conjunction with one or more of the following products and components:

- iWay Service Manager
- iWay Integration Tools Transformer
- iWay Integration Tools Designer
- iWay Correlation Facility

iWay Service Manager

iWay Service Manager is the heart of the Universal Integration Framework and is an open transport service bus. Service Manager uses graphical tools to create sophisticated integration services without writing custom integration code by:

- Using metadata from target applications
- Transforming and mapping interfaces
- Managing stateless processes

Its capability to manage complex integration interactions makes it ideally suited to be the foundation of a service-oriented architecture.

For more information, see the *ibi™ iWay® Service Manager User's Guide*.

iWay Integration Tools Transformer

iWay Integration Tools (iIT) Transformer (previously known as iWay Transformer) is a GUI tool that is delivered as a plugin with iIT. iIT Transformer is a rule based data transformation tool that converts an input document of one data format to an output document of another data format or structure. The easy-to-use graphical user interface and function tool set facilitate the design of transform projects that are specific to your requirements.

For more information, see the *iWay Integration Tools Transformer User's Guide*.

iWay Integration Tools Designer

iWay Integration Tools (iIT) Designer (previously known as iWay Designer) is a GUI tool that is delivered as a plugin with iIT.

The capability of graphically visualizing a business process is a powerful and necessary component of any e-Business offering. iWay Integration Tools Designer, a Windows-based design-time tool, provides a visual and user-friendly method of creating a business process, also called a process flow. Through a process flow, you control the sequence in which tasks are performed and the destination of the output from each task.

For more information, see the *iWay Integration Tools Designer User's Guide*.

iWay Correlation Facility

The iWay Correlation Facility (also known as the Correlation Manager) maintains records of anticipated activities occurring in the system. Correlation actions take the correlation from OPEN to CLOSED state, and allow history to be recorded. Agents are provided to implement Correlation Facility interactions within process flows, however, it is possible to use this API to accomplish this same purpose within your own exits.

For more information on using the iWay Correlation Facility, see the *ibi™ iWay® Service Manager User's Guide* and the *ibi™ iWay® Service Manager Programmer's Guide*.

Using a Channel to Construct a Message Flow

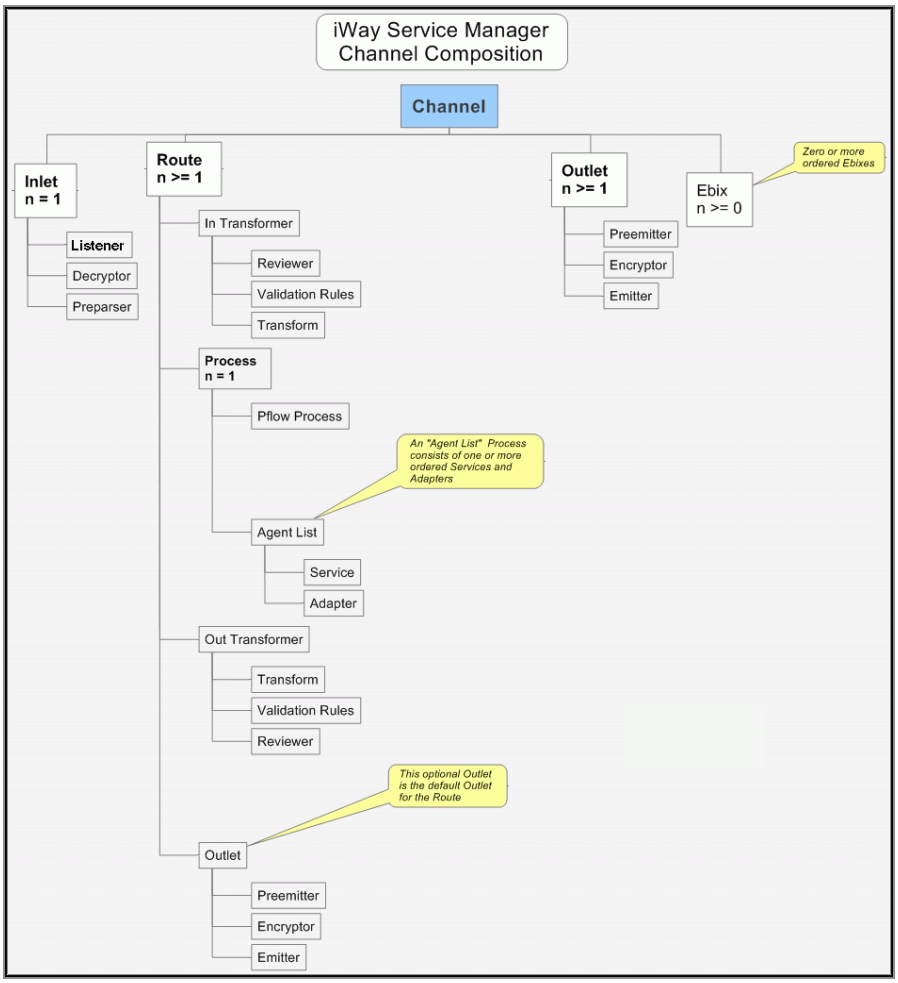
The use of iWay Service Manager is centered on a channel. A channel is a container for all the iWay business components used in an EDI message flow.

At a high level, a channel accepts input data through an **inlet**, processes the data using a **route**, and outputs the resulting data through an **outlet**. Another component in the process is an e-Business Information Exchange (**Ebix**).

The following diagram shows the channel components available in the construction of a message flow.

In the following diagram, the value **n** underneath a component name indicates how many instances of that component you can have in a channel configuration—zero, one, or more than one. For example, n = 1 for Inlet means that you can have only one inlet on the channel.

Required components are in boldface type.



Components of a Channel

A channel consists of:

- An inlet, which defines how a message enters a channel.
- A route, which defines the path a message takes through a channel.
- Outlets, which define how transformed messages leave a channel.
- An e-Business Information Exchange (Ebix), which is a collection of metadata that defines the structure of data.

iWay Service Manager provides a design-time repository called the Registry, where you assemble and manage the components in a channel.

An **inlet** can contain:

- A listener (required), which is a protocol handler responsible for picking up an incoming message on a channel.
- A decryptor, which applies a decryption algorithm to an incoming message and verifies the security of the message.
- A preparser, which is a logical process that converts an incoming message into a document that can be processed. The preparsed document then passes through the standard transformation services to reach the designated processing service.

A **route** can contain:

- An in transformer, which is an exit sequence that applies to a message before processing occurs.
    - A reviewer, which is either the first exit to receive a document after parsing (inbound), or the last exit to receive a document prior to the actual emit operation (outbound). These exits are intended for envelope handling but can be used for any desired purpose.
    - Validation rules, which apply validation using the rules validation engine. Rules are provided when the iWay Integration Solution for EDI is installed.
    - A transform, which is a transformation definition file that contains sets of rules, interpreted and executed by a transformation engine. Transformation is the process by which data is transformed from one structure/format to another.
- A process, which is a stateless, lightweight, short-lived microflow that is executed by iWay Service Manager on a message as it passes through the system. Processes that are published using iIT Designer are available in the Registry and can be bound to channels as routes.
    - A process flow.
    - An agent list.
        - A service, which is an executable Java procedure that handles the business logic of a message.
        - An adapter, which refers to a target that represents a specific instance of a connection to a back-end system.
- An out transformer, which is an exit sequence that applies to a message after processing occurs.
    - A transform, which is a transformation definition file that contains sets of rules,

interpreted and executed by a transformation engine. Transformation is the process by which data is transformed from one structure/format to another.

- Validation rules, which apply validation using the rules validation engine. Rules are provided when the iWay Integration Solution for EDI is installed.
- A reviewer, which is either the first exit to receive a document after parsing (inbound), or the last exit to receive a document prior to the actual emit operation (outbound). These exits are intended for envelope handling but can be used for any desired purpose.

- An outlet (optional), which is responsible for all aspects of preparing a document for emission and then emitting it.
  - A preemitter, which is a logical process that handles a document immediately before transmission. Normally it converts an XML document into non-XML format.
  - An encryptor, which can be called to encrypt an outgoing document.
  - An emitter, which is a transport protocol that sends a document to its recipient.

An **outlet** can contain:

- A preemitter.
- An encryptor.
- Multiple emitters.

For details on the preceding components, see the *ibi™ iWay® Service Manager User's Guide*.

Components of the iWay Integration Solution for EDI

iWay business components used in the construction of a message flow for EDI transactions include:

- An Ebix (e-Business Information Exchange)
- A preparser
- An acknowledgment service

- A preemitter

Ebix

iWay Software provides various e-Business Information Exchange (Ebix) files used in conjunction with the iWay integration solutions. In iWay Service Manager, the iWay Integration Solution for EDI contains several Ebix files, one for each supported EDI ANSI X12 transaction set.

An Ebix file for EDI-X12 s named X12_*transaction_set*.ebx, where *transaction_set* is the transaction set number. For example, the Ebix file for EDI X-12 transaction set 4050 is named X12_4050.ebx.

For details on the supported EDI X-12 transaction sets, see Ebix-Supported Transaction Sets.

An Ebix is a collection of metadata that defines the structure of data. The Ebix supplied with the iWay Integration Solution for EDI defines the structure of supported EDI messages.

Each Ebix includes:

- Pre-built data dictionaries. The structure of each EDI document is described by two data dictionaries:
  - Header dictionary, which describes the enveloping structure of the document.
  - Document dictionary, which describes the segments and elements that compose each document.

  The dictionaries from the Ebix are used to transform the structure of a document per EDI regulation.
- Pre-built XML schemas that define the structure and content of XML messages in detail.
- Pre-built EDI to XML transformation templates, and XML to EDI templates, for the supported EDI ANSI X12 transaction sets.
- Pre-built rule files for each message. The iWay Integration Solution for EDI uses these rule files to validate inbound and outbound documents.

Preparsers

A preparser is an iWay business component that converts incoming messages into processable documents.

Typically a preparser converts a non-XML document into XML format. The preparser for the iWay Integration Solution for EDI converts an incoming EDI-X12 formatted document to XML format.

The EDIX12SplitterPreParser is provided by iWay Software for the iWay Integration Solution for EDI.

EDIX12SplitterPreParser

The EDIX12SplitterPreParser (com.ibi.preparsers.EDISplitPP) parses an EDI input file that contains one or more interchanges (ISA) and multiple documents, and creates multiple XML output files. One XML output file is produced for each document.

For example, if the EDI input file contains three documents within one ISA, the EDIX12SplitterPreParser creates three XML output files, one per document.

Use the EDIX12SplitterPreParser for large files with multiple documents within one ISA; if there is a specific business requirement to create separate XML output files; or if you receive multiple documents within one ISA and want to separate each document for further business processing. You can also use the EDIX12SplitterPreParser if there is only one document within the ISA.

You can also run the EDIX12SplitterPreParser in a *splitter only* mode based on configuration. The output files are in X12 format, one document per file with a wrapper consisting of the original interchange. This option can be used to separate and route documents prior to transformation, in conjunction with several of the SREG values that are available. For example, you may receive text-based documents that you want to simply email rather than transform. Or you may want to separate your documents by document type, which would allow you to process all purchase orders immediately and all sales reports overnight in a batch mode if required.

EDIBatchSplitter

The EDIBatchSplitter (com.ibi.preparsers.XDEDIBatchSplitter) parses an EDI input file that contains one or more interchanges (ISA) and multiple documents. You must use this preparser with the EDIX12PreParser (com.ibi.preparsers.XDEDIpreParser). The EDIBatchSplitter should not be used as a standalone preparser. To successfully transform an inbound X12 input file using this preparser, you must also include the EDIX12PreParser in your channel Inlet.

One XML output file is produced for each document that is processed through this Inlet definition. For example, if your EDI input contains three documents within an ISA, the EDIBatchSplitter/EDIX12Preparser will create three XML output files, one for each document.

Acknowledgment Service

An acknowledgment service is an iWay business component used in inbound processing to create a functional acknowledgment (997) for inbound messages.

An acknowledgment indicates that an inbound document was received and validated for structure against the appropriate standard. An acknowledgment does not indicate that a document was processed.

An acknowledgment is typically routed back to the originator of the inbound document or to the next step in the integration process. It is a best business practice to send an acknowledgment to the originator of the inbound document.

The acknowledgment service for the iWay Integration Solution for EDI is called EDIX12AckAgent (com.ibi.agents.XDX12AckAgent). The iWay Integration Solution for EDI creates one acknowledgment for each interchange that is received.

Preemitter

A preemitter is a logical process that handles a document immediately before transmission.

Typically a preemitter is used to convert an XML document to non-XML format. The XML document is created from EDI input data in inbound processing. The iWay Integration

Solution for EDI uses a preemitter in outbound processing to convert the XML-formatted EDI document to an EDI-X12 formatted document.

The XML structure must be compliant with the schema supplied in the Ebix.

The preemitter for the iWay Integration Solution for EDI is called EDIX12PreEmitter (com.ibi.preemit.XDX12PreEmitter).

Data Segments and Data Elements

The following example shows what an 850 purchase order looks like. Each line is called a **Data Segment** and begins with the **Segment Name**. For example, 'N1' represents name and address line 1 while 'PO1' represents purchase order line 1.

```
BEG*00*SA*A14578**20070112~
REF*VR*54863~
ITD*01*3*1**15**16~
DTM*002*20070131~
N1*BT*Buy Snacks Inc.*9*3456 Main St.~
N4*Temple*TX*76503~
N1*ST*Buy Snacks Inc.*9*1000 Highway 27 N.~
N3*Regional Distribution Center~
N4*Athens*GA*30603~
PO1**16*CA*12.34*  *CB*000111111*UA*002840022222~
PID*F****Crunchy Chips~
PO4*48*7.89*LB~
PO1**13*CA*12.34*  *CB*000555555*UA*002840033333~
PID*F****Nacho Chips~
PO4*48*8.9*LB~
PO1**32*CA*12.34*  *CB*000666666*UA*002840044444~
PID*F****Potato Chips~
PO4*72*6.78*LB~
PO1**51*CA*12.34*  *CB*000874917*UA*002840055555~
PID*F****Corn Chips~
PO4*48*8.9*LB~
PO1**9*CA*12.34*  *CB*000874958*UA*002840066666~
PID*F****BBQ Chips~
PO4*48*4.5*LB~
PO1**85*CA*12.34*  *CB*000874990*UA*002840077777~
PID*F****Large Bag Chips~
PO4*48*4.56*LB~
PO1**1*CA*12.34*  *CB*000875088*UA*002840088888~
PID*F****Small Bag Chips~
PO4*48*4.56*LB~
CTT*7~
SE*32*000000001~
```

Following the Segment Name are a number of **Data Elements**. In the N1 segment, the code 'BT' means it's a bill-to name and address. Data elements are separated by a single

character, usually an asterisk (*). A segment ends with a single character-- in this example a tilde (~).

Other EDI documents such as an 835 Health Care Claim will have their own sets of data segments and data elements. Segments such as the N1 overlap many transaction sets, but an 835 will have its own segments and elements that are unique to health care.

Any number of data segments come together to form a transaction set. In this example, there are 32, as shown in the control counter stored in the very last segment (SE). You will notice that the PO1, PID and PO4 segments repeat multiple times, just as they would on a paper-based purchase order.

There is flexibility in how an industry or company uses the EDI standards. For example, a purchase order going from a retailer to its supplier will look very different from a purchase order going from a mining company to its supplier. The drawback is when one supplier receives purchase orders from five different customers and they each structure their 850s differently. The supplier is burdened with the task of handling the five different 850 layouts.

Configuring the EDI Activity Driver

This section describes how to configure the EDI Activity Driver using iWay Service Manager.

EDI Activity Driver Configuration Overview

The EDI Activity Driver is an extension of the Activity Facility in iWay Service Manager. It is used to log events as messages are processed. Logging can occur when:

- a message is acquired.
- a message is emitted.
- an error occurs.
- a component such as an agent or process flow is called.

For more information about the Activity Facility, see the *ibi™ iWay® Service Manager User's Guide*.

Using iWay Service Manager, you must first configure the EDI data provider and then the Activity Facility handler.

Configuring the EDI Data Provider Using iWay Service Manager

This section describes how to configure the EDI data provider.

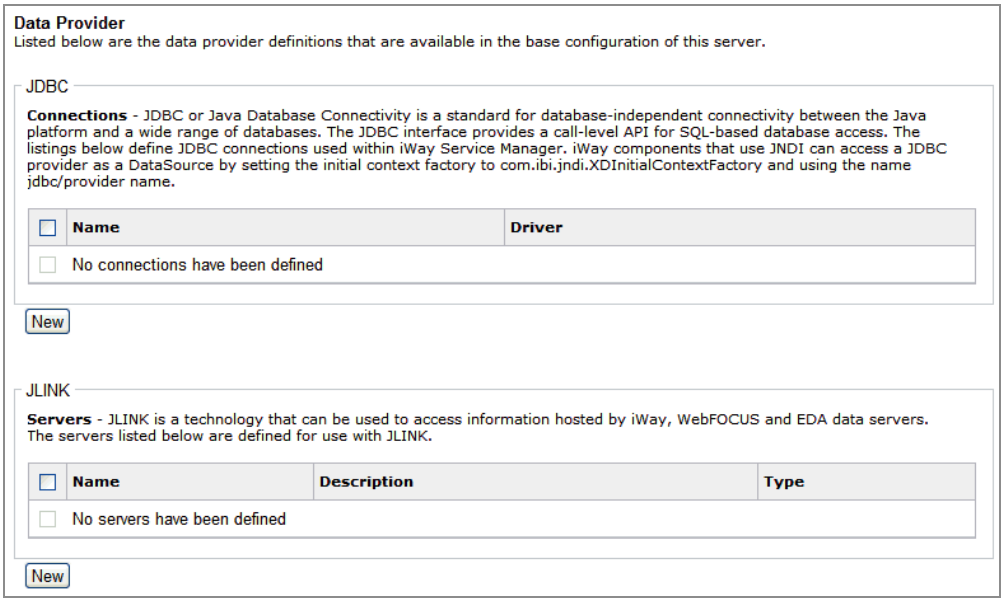
Configure the EDI Data Provider

To configure the EDI data provider:

Procedure

1. In the left console pane of the Server menu, select **Data Provider**.

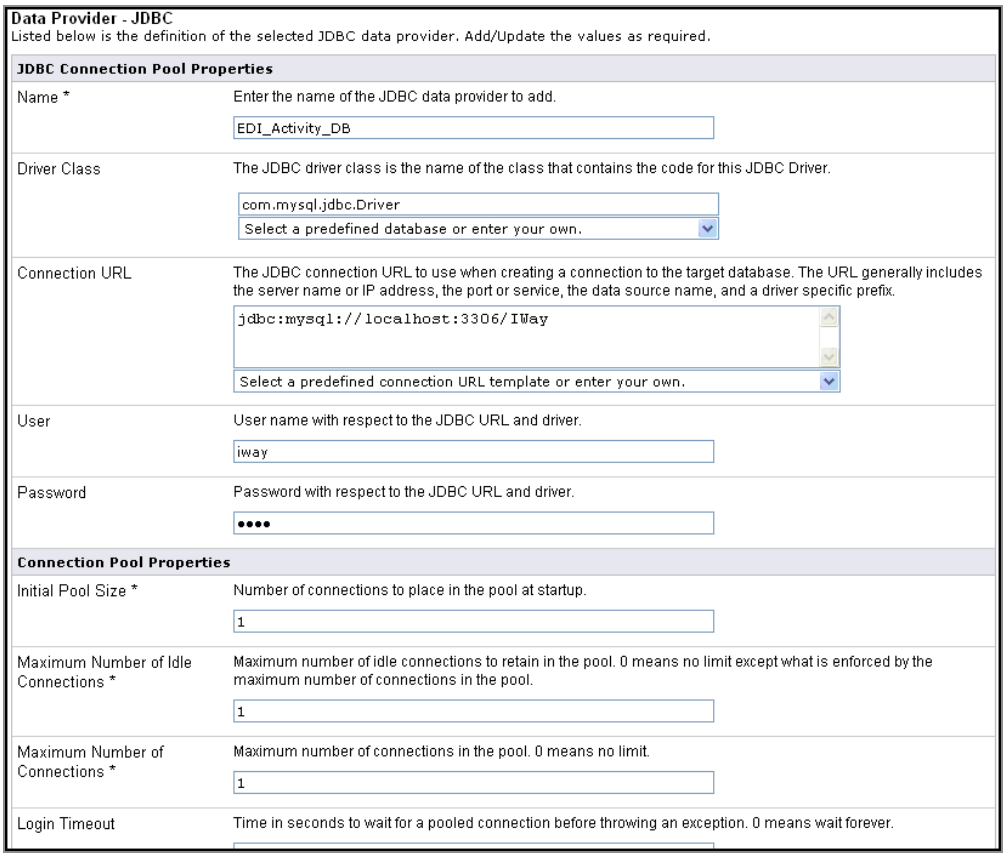
   The Data Provider pane opens.



   The tables that are provided list the configured JDBC and JLINK data providers that are available. By default, no data providers are configured.

2. In the JDBC area, click **New** to configure a new JDBC data provider.

   The configuration pane for the JDBC data provider opens.



3. In the Name field, enter a name for the new JDBC data provider, for example, EDI_Activity_DB.
4. From the Driver Class drop-down list, select an appropriate driver or enter the specific driver name (class) that you are using, for example:

```
com.mysql.jdbc.Driver
```

5. From the Connection URL drop-down list, select an appropriate connection URL or enter the specific driver connection URL that you are using, for example:

```
jdbc:mysql://localhost:3306/IWay
```

6. In the User field, enter a user name with respect to the JDBC URL and driver.
7. In the Password field, enter a password with respect to the JDBC URL and driver.
8. In the Initial Pool Size field, enter the number of connections to place in the connection pool during startup.

9. In the Maximum Number of Idle Connections field, enter the maximum number of idle connections to retain in the connection pool.

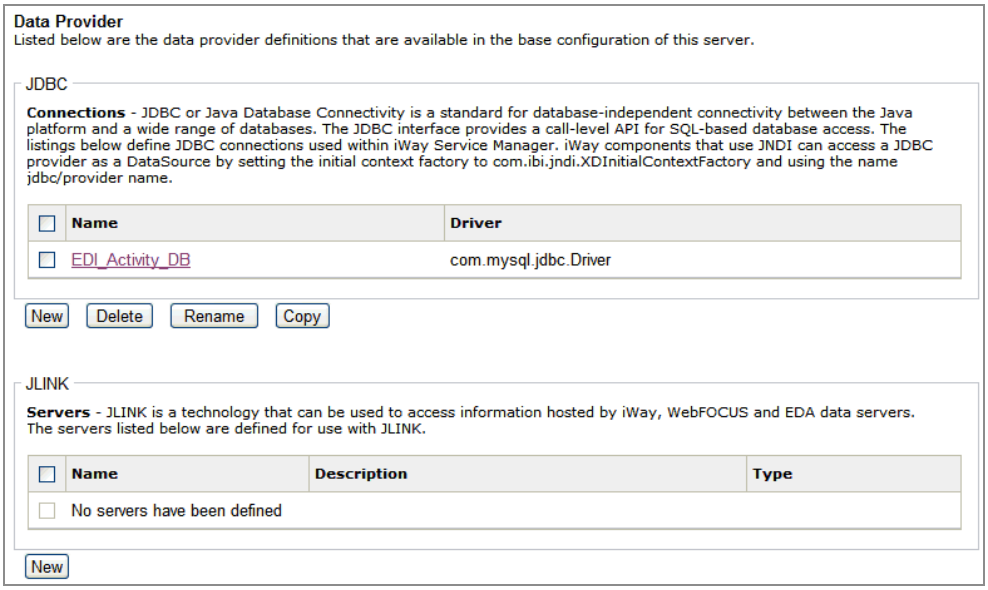
   A value of zero means that there is no limit, except what is enforced by the maximum number of connections in the connection pool.

10. In the Maximum Number of Connections field, enter the maximum number of connections in the connection pool.

    A value of zero means that there is no limit.

11. Click **Add**.

    The JDBC data provider that you configured is added to the JDBC Connections list, as shown in the following image.

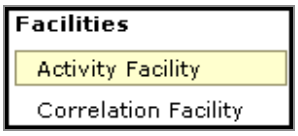


Configuring the EDI Activity Driver Using iWay Service Manager

This section describes how to configure the EDI Activity Driver.

Configure the EDI Activity Driver

To configure the EDI Activity Driver:



Procedure

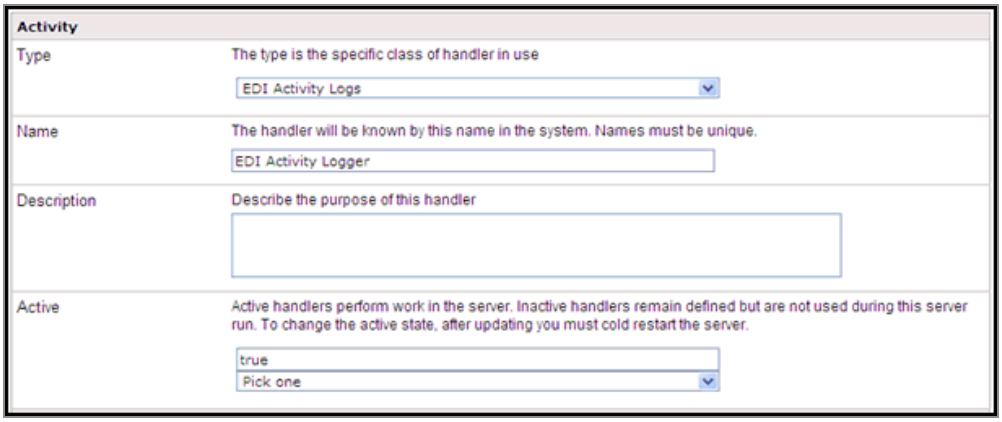
1. In the left console pane of the Server menu, select **Activity Facility**.

   The Activity Facility pane opens.

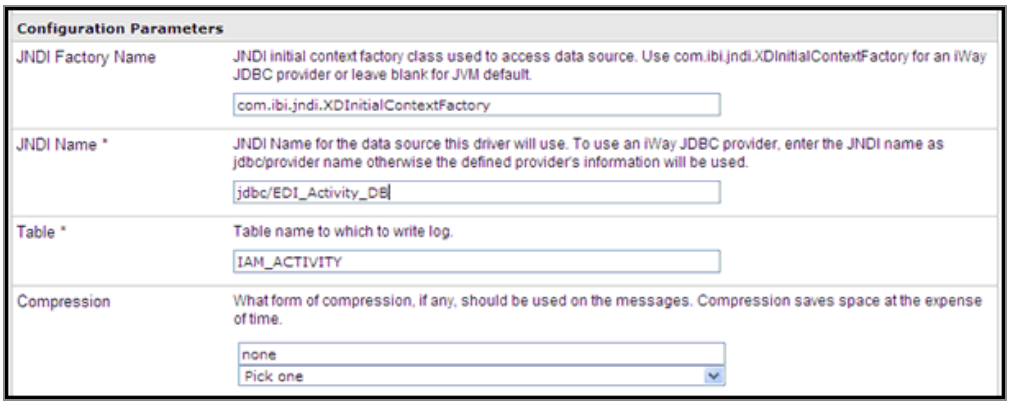
   The table that is provided lists the configured Activity Facility handlers. Initially, no handlers are shown.

2. Click **Add** to configure a new Activity Facility handler.

   The configuration pane for the Activity Facility handler opens.



3. From the Type drop-down list, select **EDI Activity Logs**.
4. Enter a unique name for the EDI Activity Driver and a brief description.
5. From the Active drop-down list, select **true**.
6. Configure the JDBC driver for the database you are using.



If the database tables do not exist, they will be automatically created when the iSM is restarted.

7. Provide values for the remaining parameters, as defined in the following table.

Parameter Name	Type	Description
JNDI Factory Name	String	The JNDI initial context factory class that is used to access a data source. Use `com.ibi.jndi.XDInitialContextFactory` for an iWay JDBC provider or leave this field blank for the JVM default.
JNDI Name	String	The JNDI name for the data source this driver will use. To use an iWay JDBC provider, enter the JNDI name as `jdbc/<data provider name>`, where *data provider name* is the name of the EDI Activity Driver that was specified in step 4. Otherwise the information for the defined provider will be used.
Table	String	Table name for the activity log. This must be a valid identifier in the database being used. If the table does not exist at startup, it will be created automatically.

Parameter Name	Type	Description
Compression	Drop-down list	Specify whether the messages are to be compressed. Values include: • none (default) • smallest • fastest • standard • Huffman
Start Events	Boolean Drop-down list	If set to **true** (default), the input messages will be recorded in the activity log. This values must be set to **true** for use of the audit reports in the console.
Internal Events	Boolean Drop-down list	If set to **true**, system events are included in the activity log. System events include activities such as parsing and transformations (optional). False is selected by default.
Security Events	Boolean Drop-down list	If set to *true* (default), security events are recorded. This includes digital signature, and so on. However, console activity is not recorded.
Business Error Events	Boolean Drop-down list	If set to **true**, business errors are recorded, such as rules system violations. False is selected by default.
Emit Events	Boolean Drop-down list	If set to **true** (default), output messages from emitter services will be recorded. This is required for use of the audit log reports in

Parameter Name	Type	Description
		the console.
End Events	Boolean Drop-down list	If set to **true** (default), the end of message processing will be recorded in the activity log. This is required for use of the audit log reports in the console.
Notes Table	String	Table name for the notes table, which contains log annotations. If the table does not exist at startup, it will be created automatically.
MAC Algorithm	String Drop-down list	The Message Authentication Code (MAC) algorithm. None (default) indicates a MAC should not be computed.
MAC Provider	String Drop-down list	The Message Authentication Code (MAC) provider. Not Specified indicates the default provider should be used. The remaining available value is **SunJCE**.
MAC Secret Key	String	The Message Authentication Code (MAC) secret key to use.

8. Click **Update**.

   If necessary, start the database services.

9. Restart iSM to start the EDI Activity Driver and begin logging.

   The EDI Activity Driver inserts records into the configured activity database. The records are designed for fast writing rather than for ease of later analysis. A set of inquiry service agents suitable for use in a process flow is available to assist during the analysis of the log. Users are cautioned that iWay does not guarantee the layout of the record from release to release, and this should be checked against the actual schema.

Database Field	Description
recordkey	Unique record identifier.
recordtype	Type of this record - the event being recorded. • 101 - Message start. • 131 - Entry to event (see subtype codes below). • 132 - Normal exit from event. • 133 - Failed exit from event. • 151 - Ancillary message (usually rules violation). • 181 - Emit. • 191 - Message end.
signature	Encoding of the listener name and protocol.
protocol	Name of the protocol.
address	Address to which an emit is to be issued. The format depends on the protocol.
tstamp	Timestamp of record.
correlid	ISA13
tid	Transaction ID assigned to this message.
msg	Message appropriate to this record type. For example, an input message contains the original message received, if possible. Streaming input does not contain a record.
context	Serialized special registers that were in the context at the time the record was written.

Database Field	Description
text	Message text for business errors (rules system violations).
status	Status code recorded. • 0 - Success • 1 - Success, message end (191 record) • 10 - Rules error
subtype	Event code for event records. • 1 - Preparser • 2 - Parser • 3 - In reviewer • 5 - In validation • 6 - In transform • 7 - Agent or flow • 8 - Out transform • 9 - Out validation • 11 - Preemitter • 1000 - input record written to table before transformation
partner_to	ISA06
partner_from	ISA08
encoding	Encoding of the listener that obtained the document.
mac	Not used in this version.
Driver version	1.0 in 8.0 SM

Working With EDI X12 Inbound and Outbound Applications Using iWay Integration Tools (iIT)

This chapter describes how to work with EDI X12 inbound and outbound applications using iWay Integration Tools (iIT).

EDI X12 Inbound and Outbound Application Overview

This chapter provides instructions to create, import, export, and work with EDI X12 inbound and outbound applications using iWay Integration Tools (iIT). In addition, you will learn how to create an iWay Integration Application (iIA) for deployment based on the sample data.

What will the Application do?

The iIAs will be used to transform EDI X12 to XML for inbound processing and XML to EDI X12 for outbound processing.

The inbound application channel creates an XML representation of a EDI X12 (ANSI X12N formatted) inbound message, a functional acknowledgment (997), and an XML-formatted validation report. The documents are routed to designated folders based on the success or failure results of the transformation and EDI X12 validation.

The outbound application channel creates an ANSI X12N formatted EDI X12 message from XML and a XML-formatted validation report. The documents are routed to designated folders based on success or failure of transformation and EDI X12 validation.

EDI X12 Inbound and Outbound Application Prerequisites

Before you continue, ensure that the following prerequisites are met:

- You have a working knowledge of iWay Service Manager (iSM) and iWay Integration Tools (iIT).
- iSM Version 8.0 or higher is installed.
- iWay EDI X12 Adapter is installed.
- iIT Version 8.0 or higher is installed.
- System and channel Special Registers (SREGs) are updated to match your directory structure, as shown in Downloading and Extracting EDI X12 User Samples.

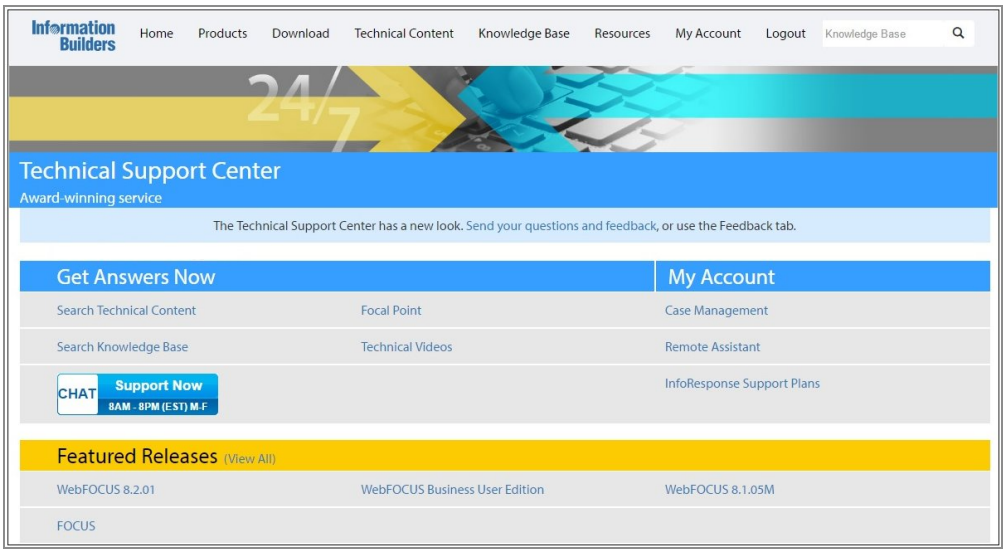
Downloading and Extracting EDI X12 User Samples

This section describes how to download and extract user samples for EDI X12.

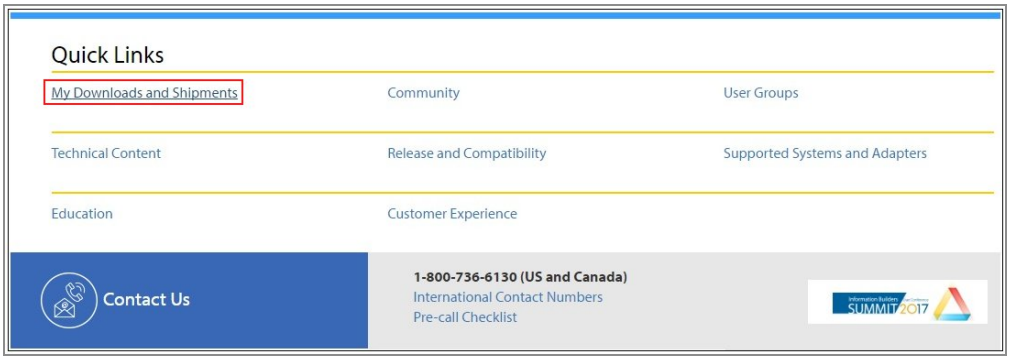
Download and Extract User Samples for EDI X12

Procedure

1. Enter the following URL in your browser to access the Information Builders Technical Support Center:

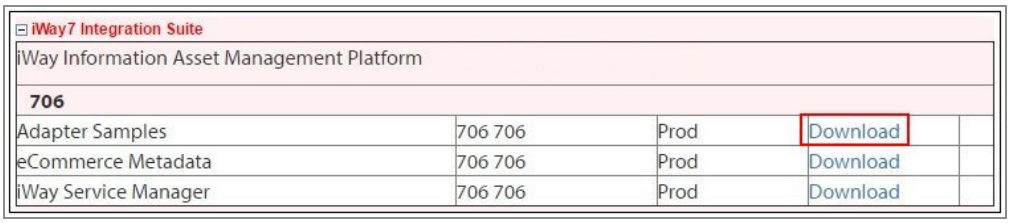


2. Scroll down and click **My Downloads and Shipments** in the Quick Links area, as shown in the following image.



The Software Downloads/Shipments page opens. Scroll down and click **Personal Downloads**, as shown in the following image.

From the list of available software categories that is displayed, expand **iWay7 Integration Suite** and then click **Download** in the Adapter Samples row, as shown in the following image.

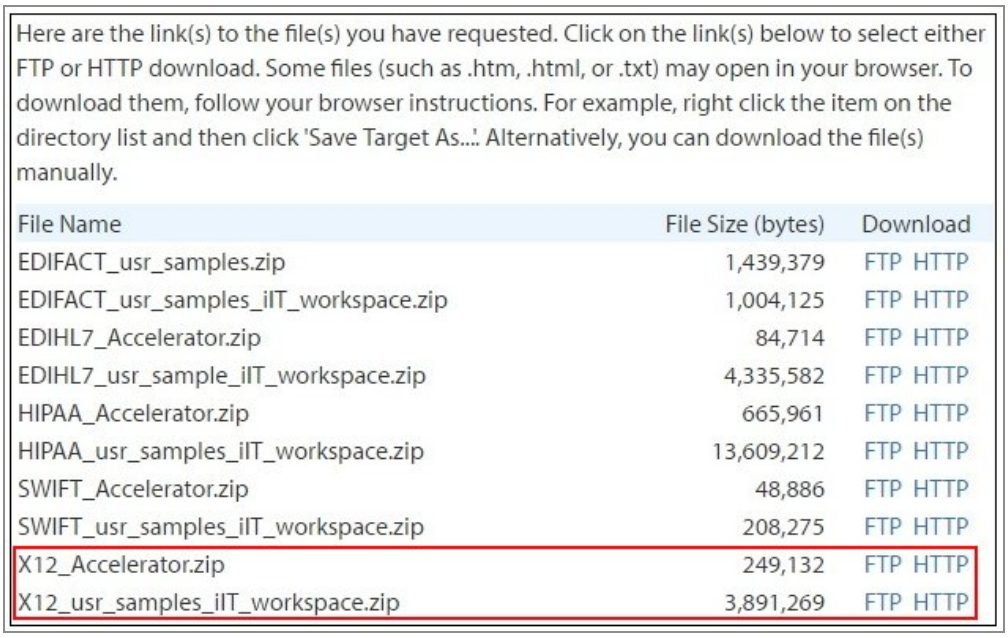


iWay7 Integration Suite			
iWay Information Asset Management Platform			
706			
Adapter Samples	706 706	Prod	Download
eCommerce Metadata	706 706	Prod	Download
iWay Service Manager	706 706	Prod	Download

You are prompted with a download registration form and then a license agreement form.

3. Provide the requested information and accept the license agreement.

   A list of sample files that are available for download is displayed, as shown in the following image.



Here are the link(s) to the file(s) you have requested. Click on the link(s) below to select either FTP or HTTP download. Some files (such as .htm, .html, or .txt) may open in your browser. To download them, follow your browser instructions. For example, right click the item on the directory list and then click 'Save Target As...'. Alternatively, you can download the file(s) manually.

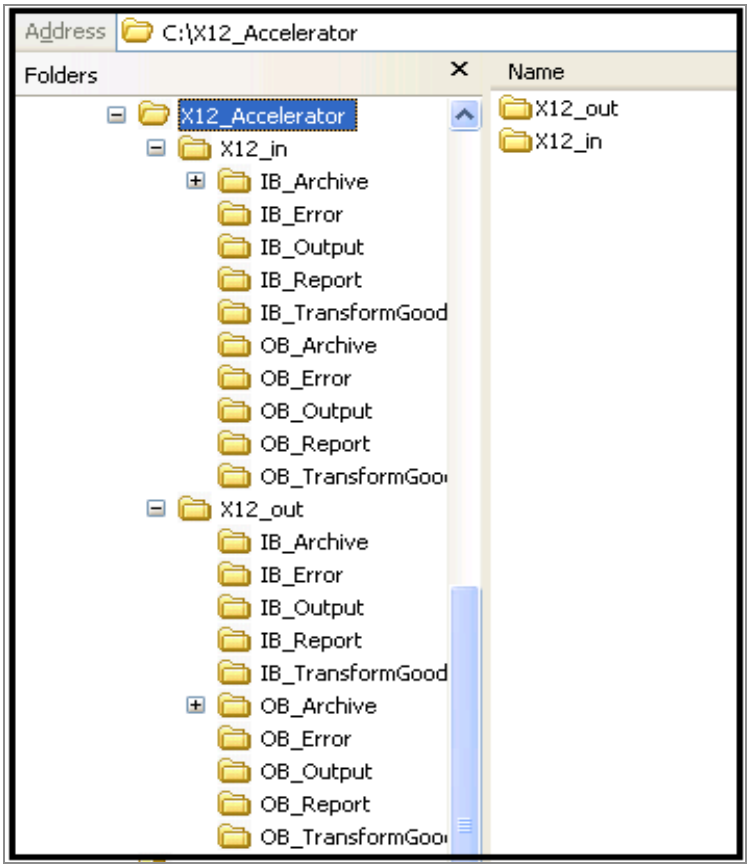
File Name	File Size (bytes)	Download
EDIFACT_usr_samples.zip	1,439,379	FTP HTTP
EDIFACT_usr_samples_iIT_workspace.zip	1,004,125	FTP HTTP
EDIHL7_Accelerator.zip	84,714	FTP HTTP
EDIHL7_usr_sample_iIT_workspace.zip	4,335,582	FTP HTTP
HIPAA_Accelerator.zip	665,961	FTP HTTP
HIPAA_usr_samples_iIT_workspace.zip	13,609,212	FTP HTTP
SWIFT_Accelerator.zip	48,886	FTP HTTP
SWIFT_usr_samples_iIT_workspace.zip	208,275	FTP HTTP
X12_Accelerator.zip	249,132	FTP HTTP
X12_usr_samples_iIT_workspace.zip	3,891,269	FTP HTTP

4. Download the following .zip files:

   - **X12_Accelerator.zip.** Contains sample data and a pre-configured folder structure that is used by the sample channel during inbound and outbound processing.
   - **X12_usr_samples_iIT_workspace.zip.** Contains a sample workspace, which includes a pre-configured project that you must import into iWay Integration Tools (iIT).

5. Save the *X12_usr_samples_iIT_workspace.zip* file to a folder on your local drive.

6. Save and extract the *X12_Accelerator.zip* file to a location where you want to store your data, as shown in the following image.

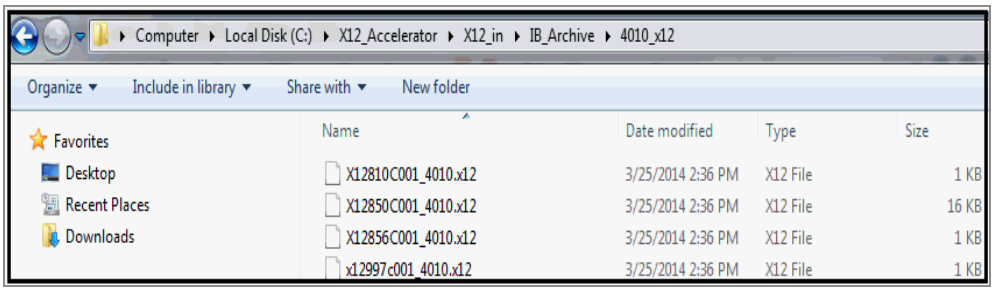


7. The X12_Accelerator.zip file contains sample input and output data that you can use.
   - Inbound test data is located in the following folder:

     ```
     \X12_Accelerator\X12_in\IB_Archive
     ```

     There are two subfolders, 4010_x12 and 5010_x12.

     For example:

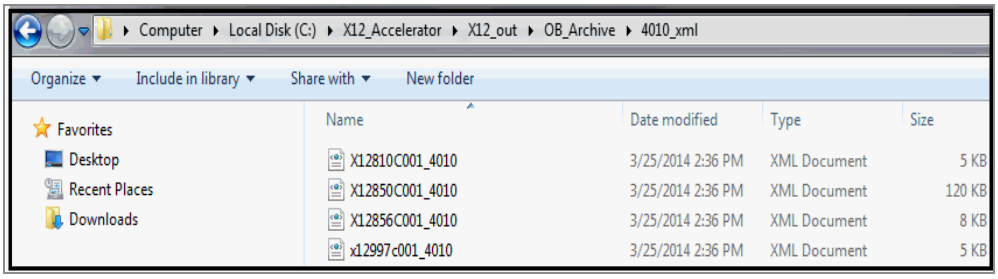


- Outbound test data is located in the following folder:

  ```
  \X12_Accelerator\X12_out\OB_Archive
  ```

  There are two subfolders, 4010_xml and 5010_xml.

  For example:



Importing EDI X12 User Samples to iWay Integration Tools as a Workspace

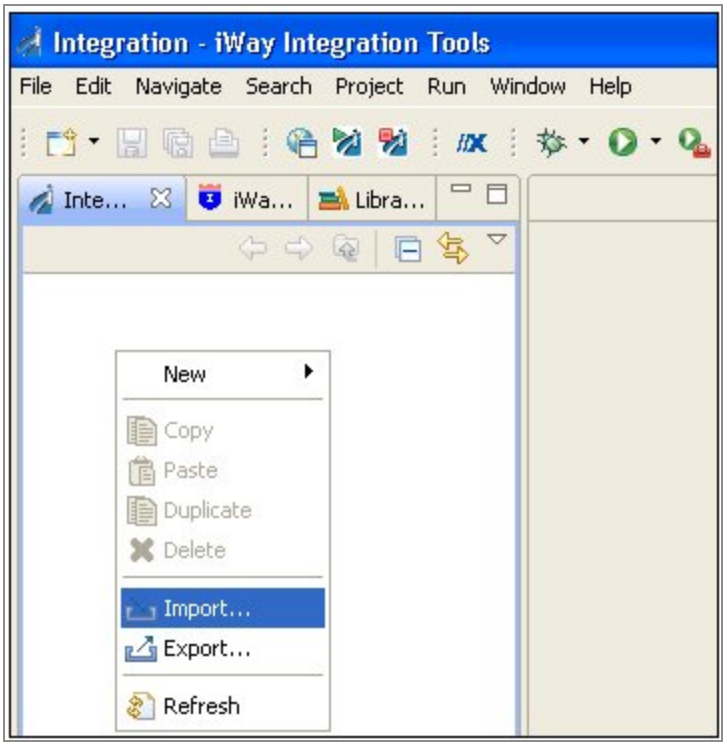
This section describes how to import EDI X12 user samples to iWay Integration Tools (iIT) as a workspace.

Import EDI X12 User Samples to iWay Integration Tools as a Workspace

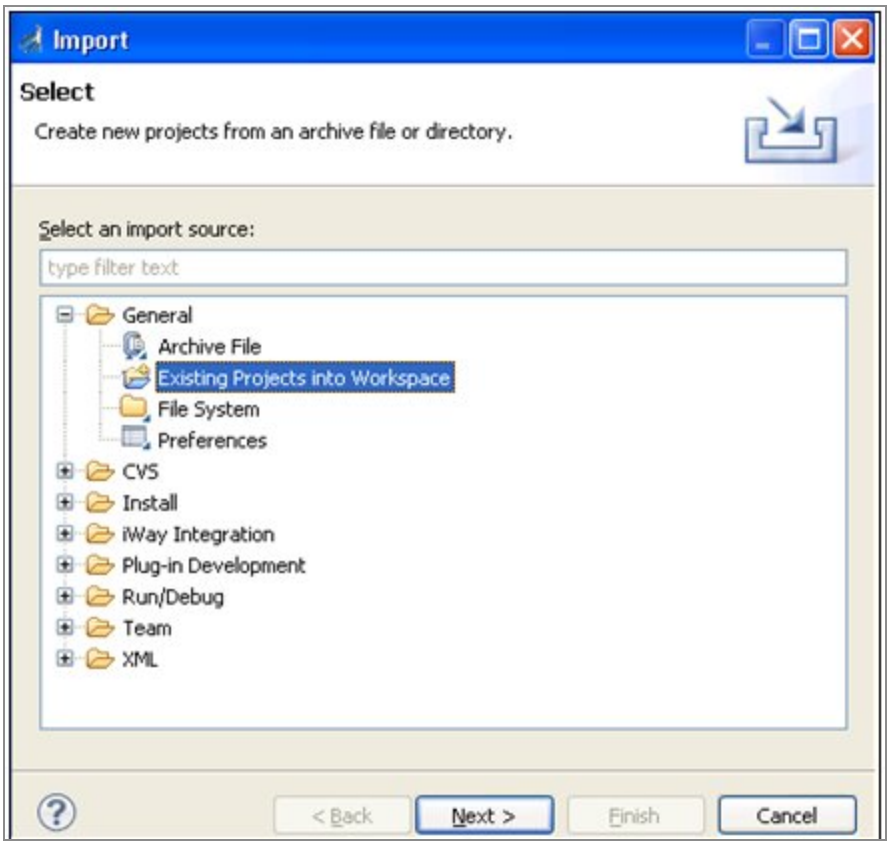
Procedure

1. Start iWay Integration Tools (iIT).

2. Right-click anywhere inside the Integration Explorer tab and select **Import...** from the context menu, as shown in the following image.

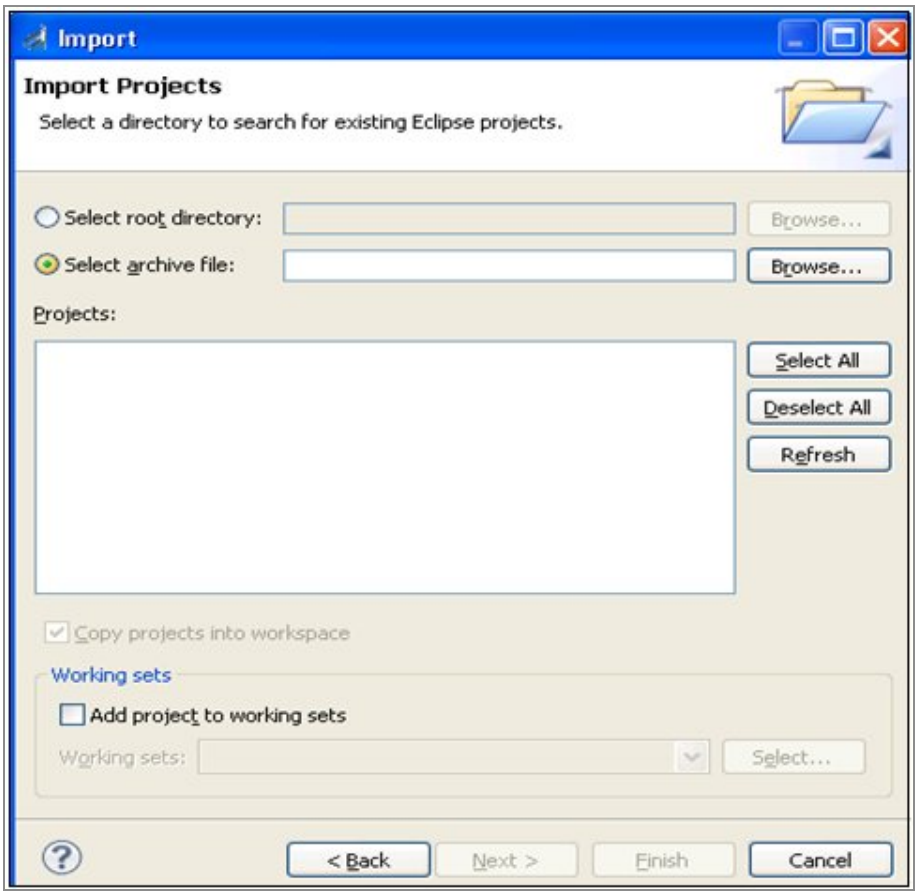


The Import dialog opens, as shown in the following image.



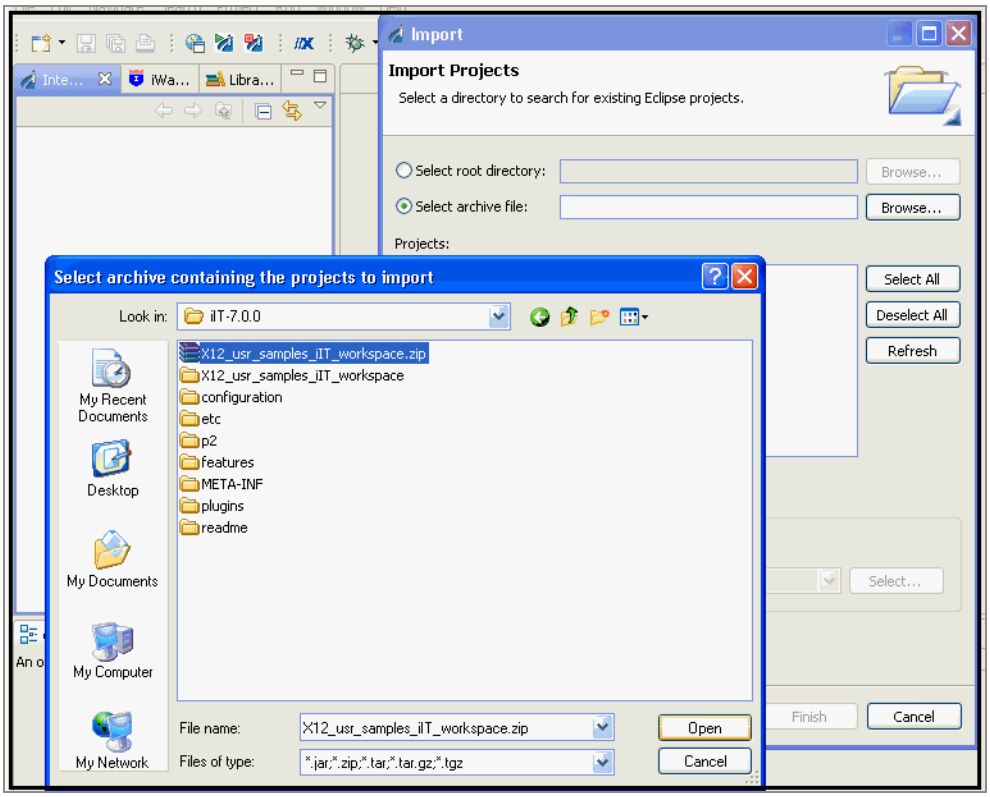
3. Expand the **General** folder, select **Existing Projects into Workspace**, and then click **Next**.

   The Import Projects pane opens, as shown in the following image.



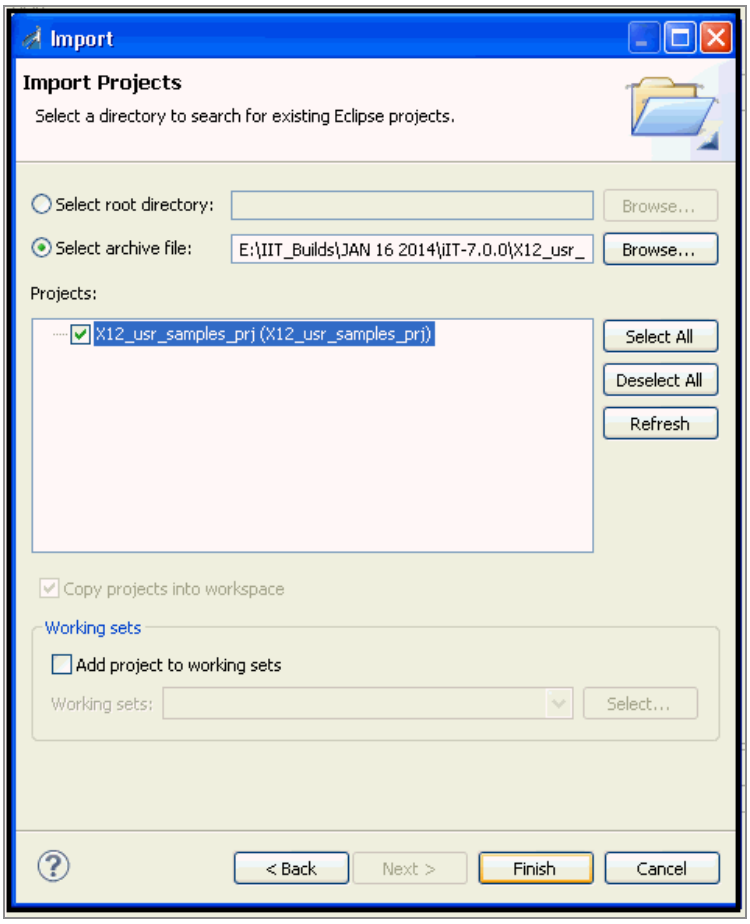
4. Click **Select archive file** and then click **Browse**.

   The Select archive containing the projects to import pane opens, as shown in the following image.



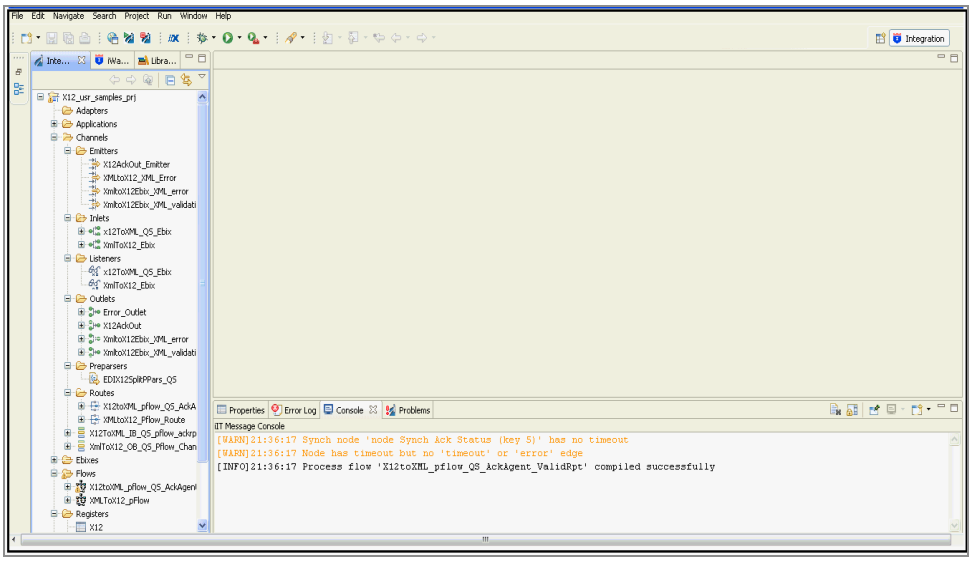
5. Select the *X12_usr_samples_iIT_workspace.zip* file and click **Open**.

   You are returned to the Import Projects pane, as shown in the following image.



6. Click **Finish**.

   The EDI X12 user samples are loaded into your iIT workspace, as shown in the following image.



The Integration Explorer tab on the left pane displays a hierarchy of all the imported channel components (for example, Ebixes, listeners, outlets, preparsers, routes, process flows, and so on). The Console tab on the bottom provides a status as each channel component is imported.

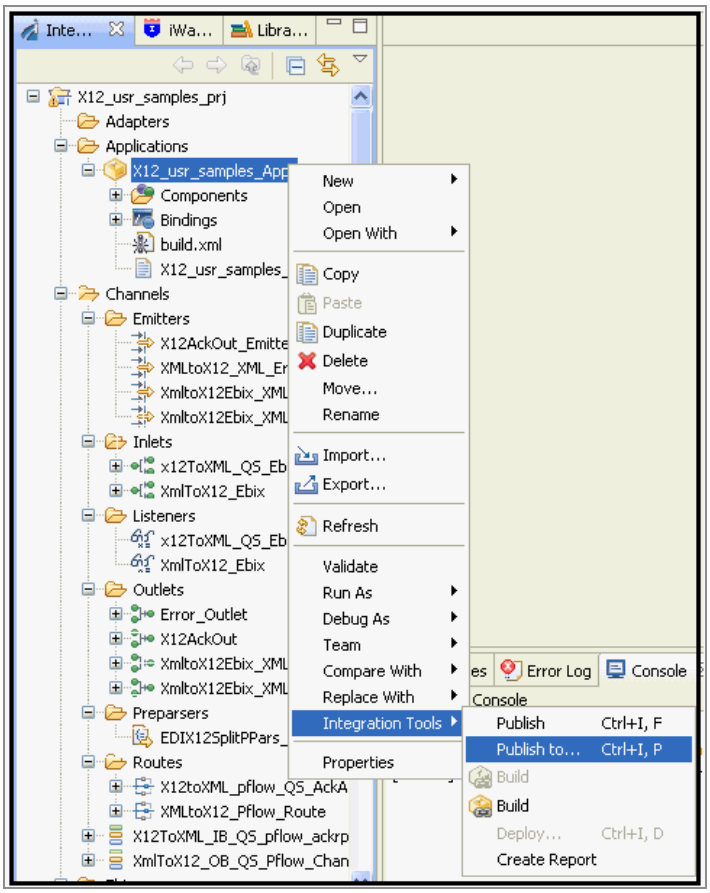
Publishing iWay Integration Applications to the iWay Service Manager Registry

This section describes how to publish iWay Integration Applications (iIAs) to the iWay Service Manager (iSM) Registry.

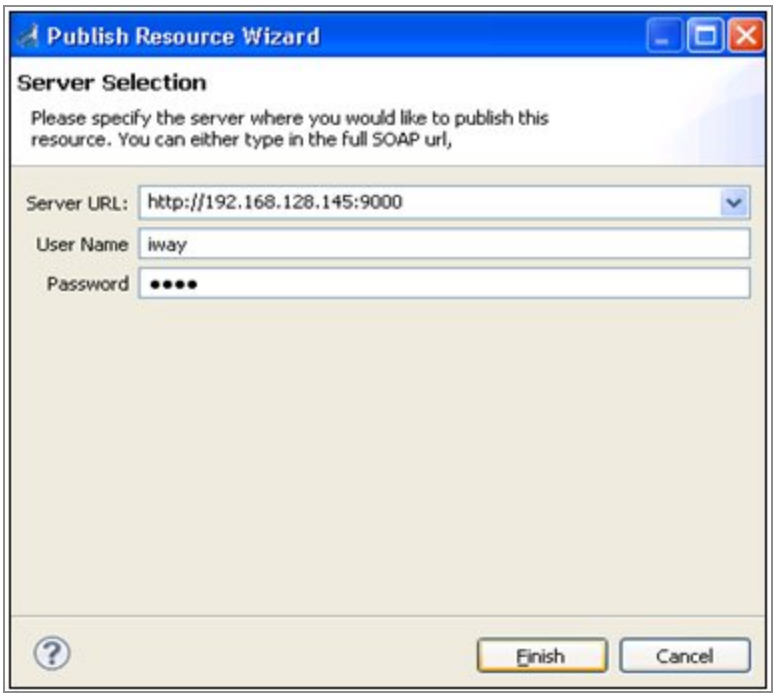
Publish iWay Integration Applications to the iWay Service Manager Registry

Procedure

1. In the Integration Explorer tab, right-click **X12_usr_samples_App**, select **Integration Tools** from the context menu, and then click **Publish to**..., as shown in the following image.



The Publish Resource Wizard dialog opens, as shown in the following image.



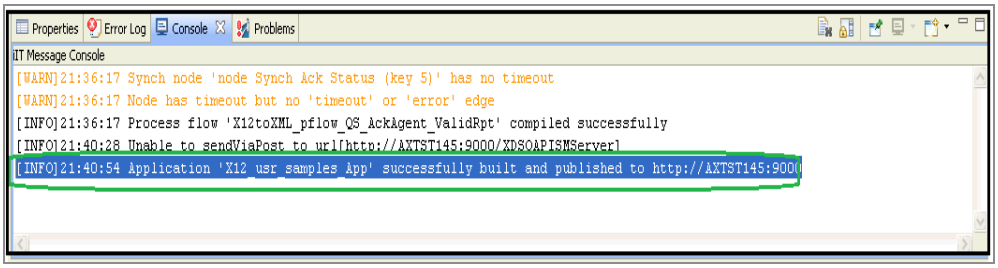
2. In the Server URL field, type the server IP number or computer name and then the port number (default port is 9000). For example:

```
http://111.111.111.000:9000
```

   Type the iSM credentials (for example, user name: *iway*, password: *iway*).

3. Click **Finish**.

   The Console tab on the bottom provides a status log that you can use for verification purposes, as shown in the following image.



Deploying iWay Integration Applications to iWay Service Manager

This section describes how to deploy iWay Integration Applications (iIAs) to iWay Service Manager (iSM).

Deploy iWay Integration Applications to iWay Service Manager

Procedure

1. Enter the following URL to access the iSM Administration Console:

   ```
   http://[host]:[port]/ism
   ```

   where:

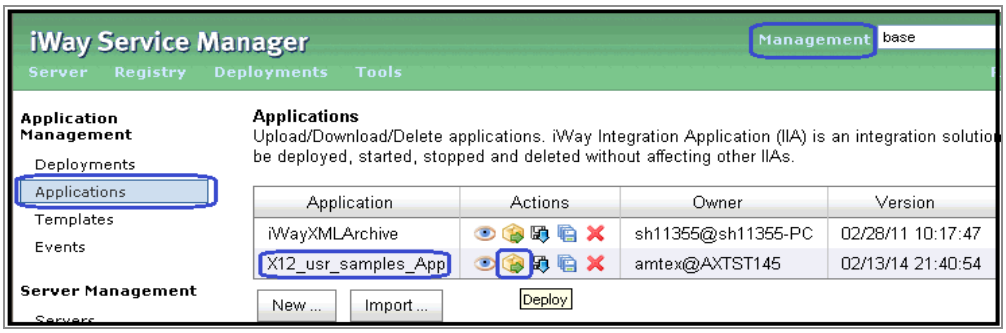
   host

   Is the host machine where iSM is installed. The default value is *localhost*.

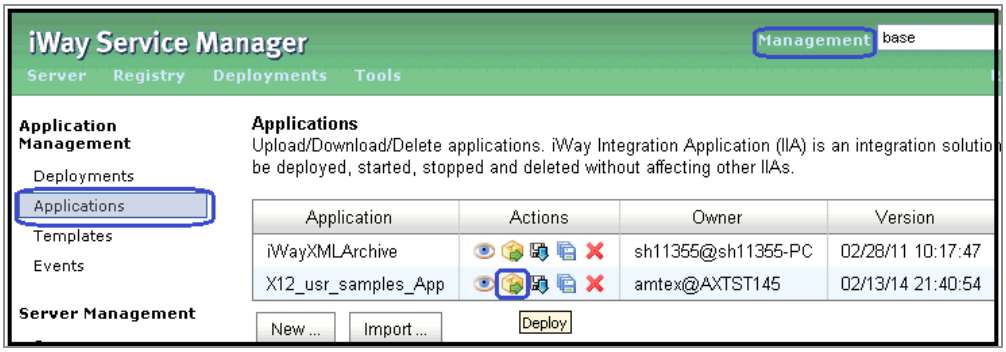
   port

   Is the port where iSM is listening. The default port is *9999*.

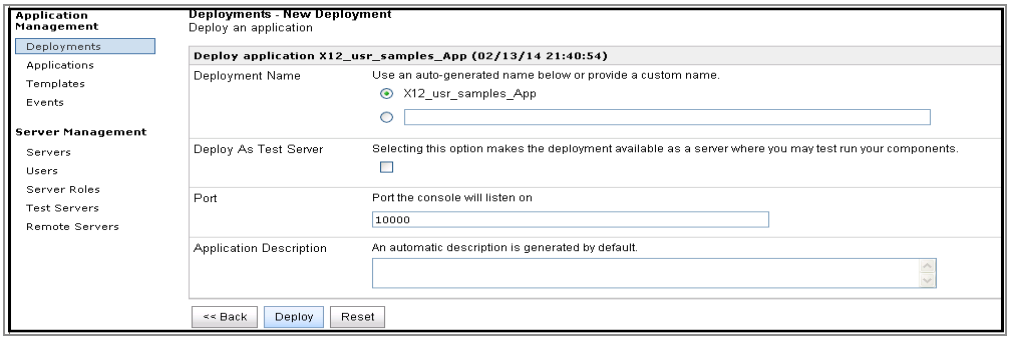
2. After publishing the iWay Integration Application (X12_usr_samples_App), you can find this iIA under the Management\Applications link in the iSM Administration Console, as shown in the following image.



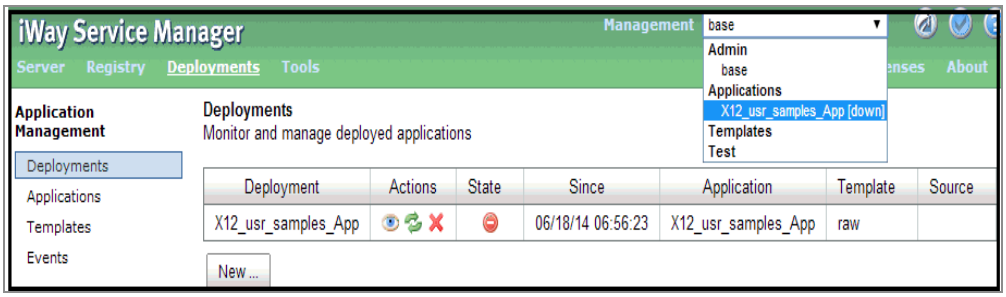
3. Click the **Deploy** icon  next to the application name under the Actions column, as shown in the following image.



The Deployments pane opens, as shown in the following image.



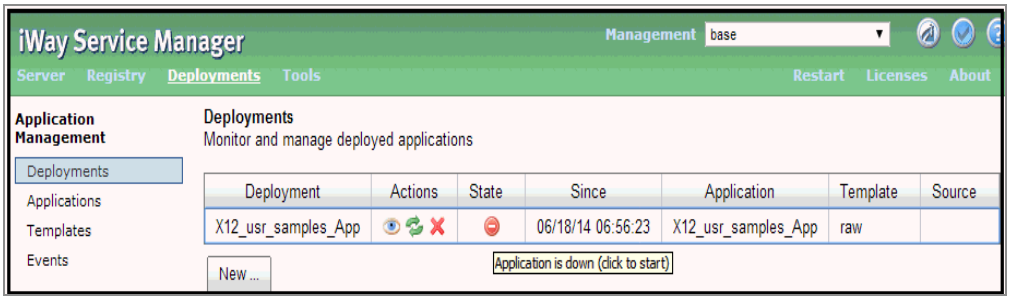
4. Click **Deploy**.
5. From the Management drop-down list, select your deployed application (for example, **X12_usr_samples_App [down]**), as shown in the following image.



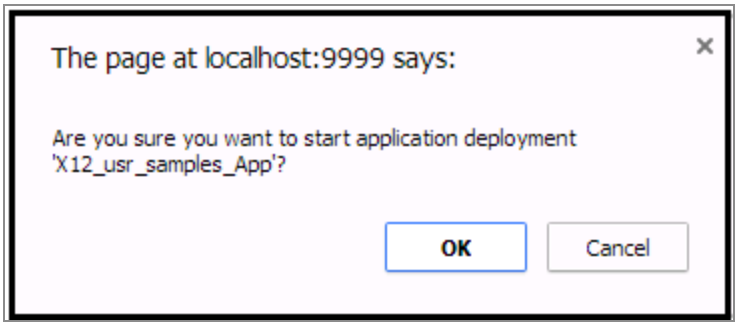
6. Click **Server** in the top menu and then **Register Settings** in the left pane for the *X12_usr_samples_App [down]* application.
7. Click **Add** to create all required registers (*X12_Installdir*, *X12_Input*, *X12_Output*, and *ValidateX12*) for the *X12_usr_samples_App [down]* application.

   For more information, see Setting Registers in the iWay Service Manager Administration Console.

8. In the State column, click the **Deployment State** icon to start the deployed Application.

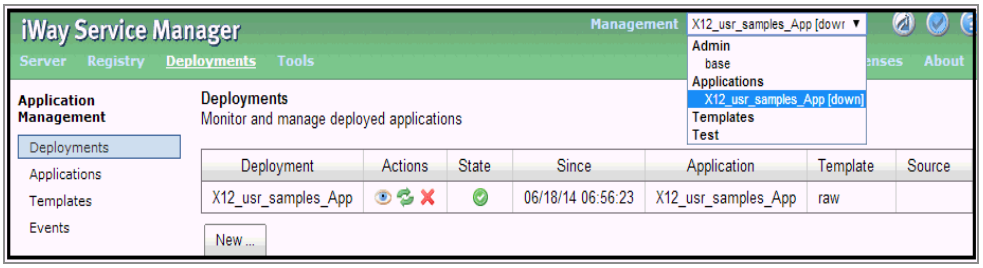


9. When the *Message from webpage* window appears, click **OK** to proceed.

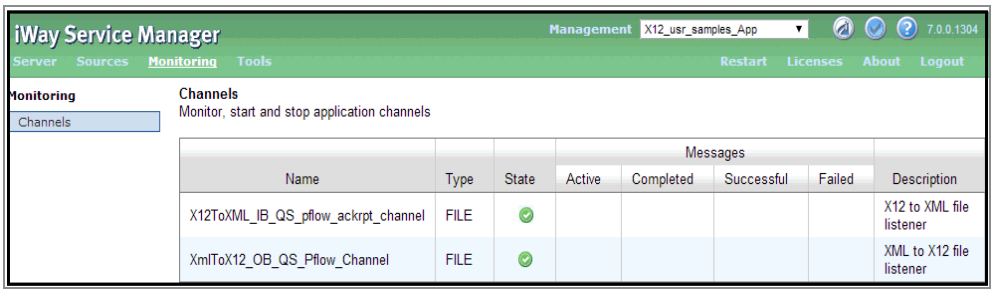


10. Once the application has successfully started, place your input data into the input location that is configured for the application.

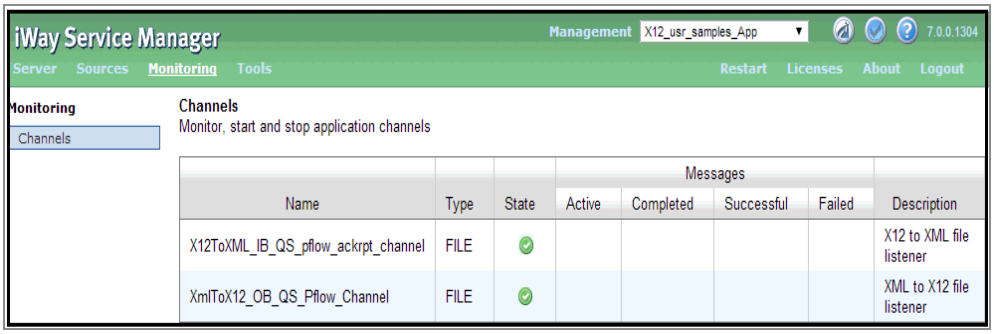
11. Select the *X12_usr_samples_App [down]* application from the Management drop-down list.



12. Click the **Monitoring** link and observe the page. The deployed application channels *X12ToXML_IB_QS_AckRpt_Pflow_Channel* and *xmlToX12_QA_Channel* are displayed, as shown in the following image.



The following image shows the inbound and outbound channels that are running in iSM. You can stop either channel and have only one channel running at a time as required.



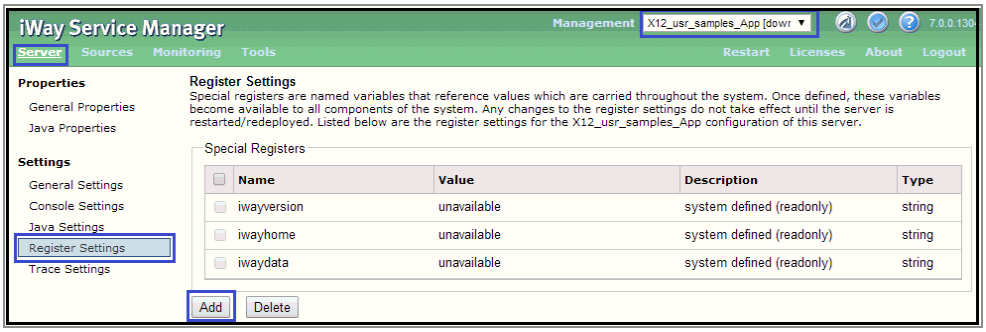
Setting Registers in the iWay Service Manager Administration Console

This section describes how to set Registers in the iWay Service Manager (iSM) Administration Console.

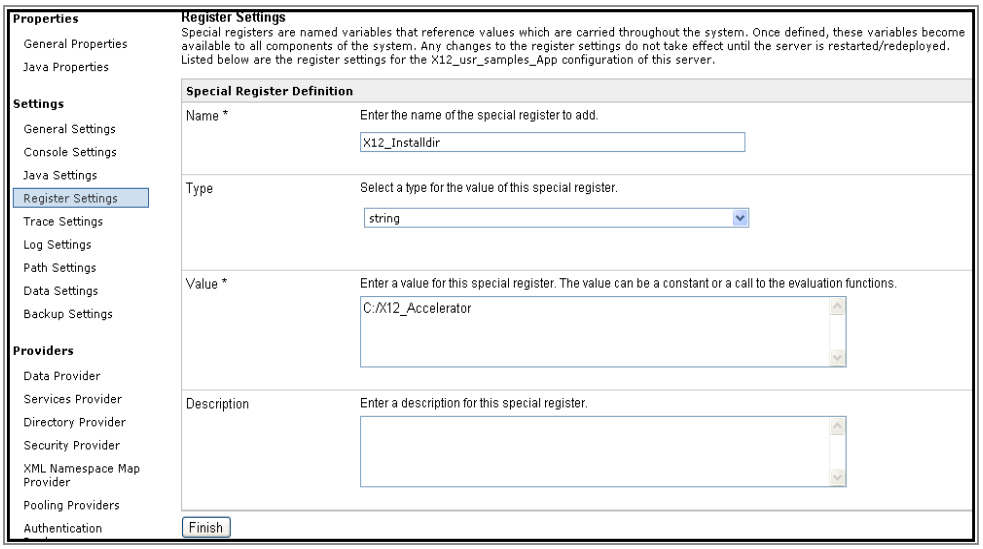
Set Registers in the iWay Service Manager Administration Console

Procedure

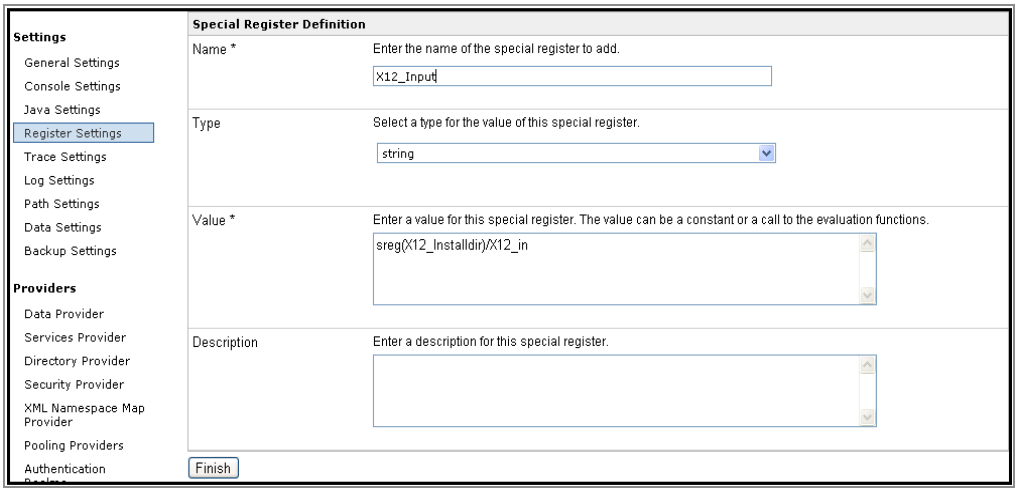
1. From the iSM Administration Console, select the *X12_usr_samples_App [down]* application from the Management drop-down list. Click **Server** in the top menu and then **Register Settings** in the left pane.



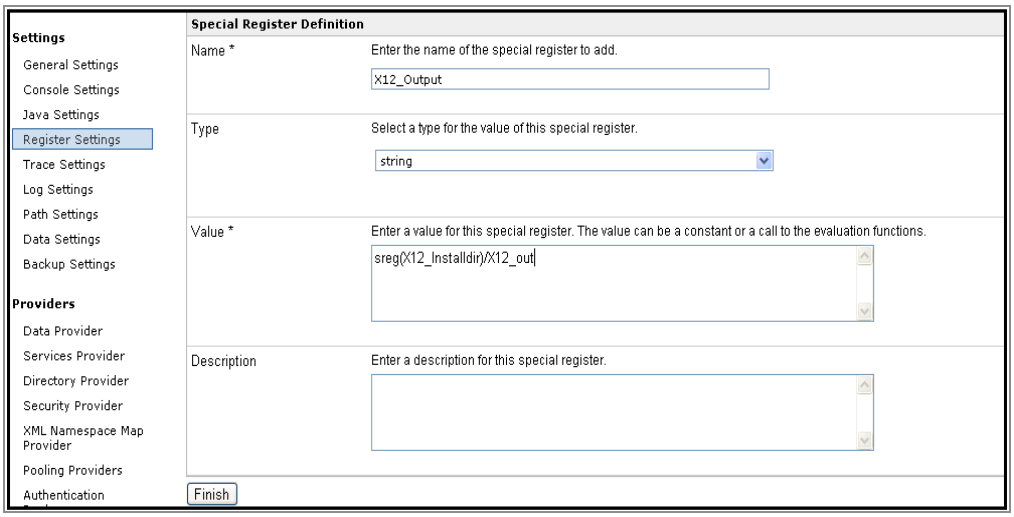
2. Click **Add**.
3. Add *X12_Installdir* and provide the appropriate values in the fields, as shown in the following image. Click **Finish**.



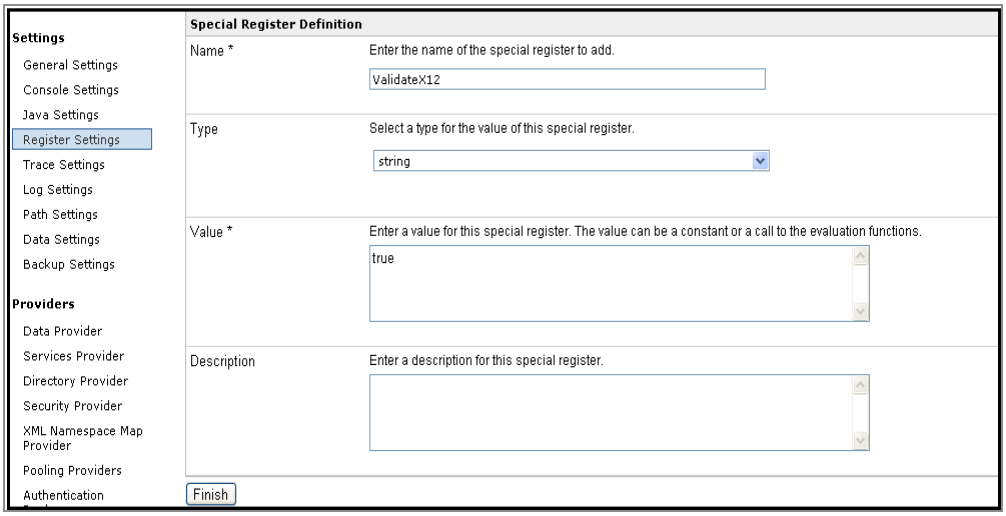
4. Add *X12_Input* and provide the appropriate values in the fields, as shown in the following image. Click **Finish**.



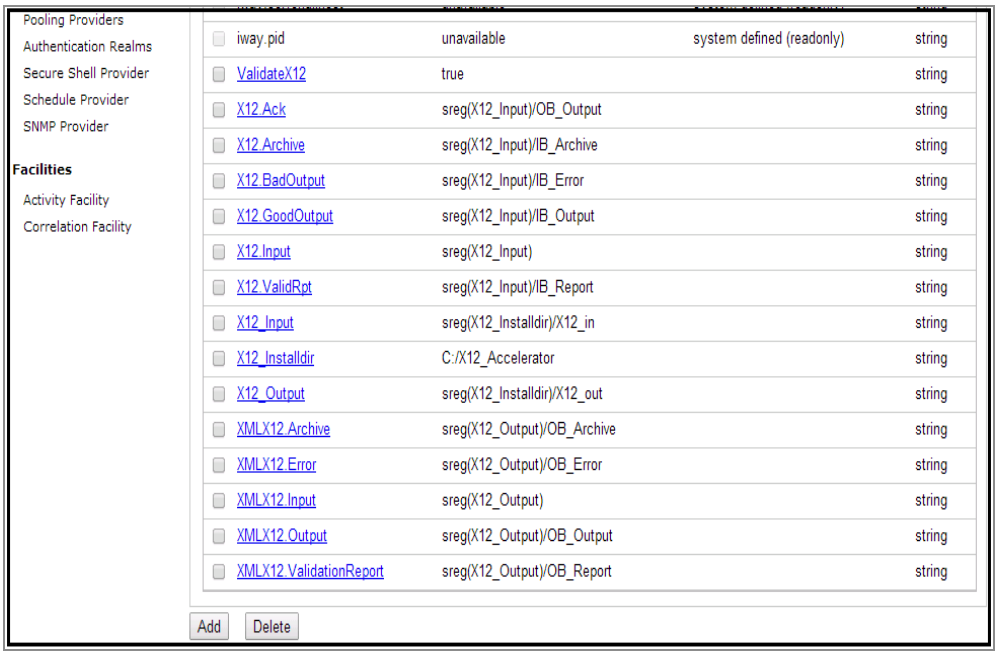
5. Add *X12_Output* and provide the appropriate values in the fields, as shown in the following image. Click **Finish**.



6. Add *ValidateX12* and provide the appropriate values in the fields, as shown in the following image. Click **Finish**.



The following image shows the summary of defined Registers.



Note: If any changes are made to Registers after an application has started, you must restart that application for these changes to be applied.

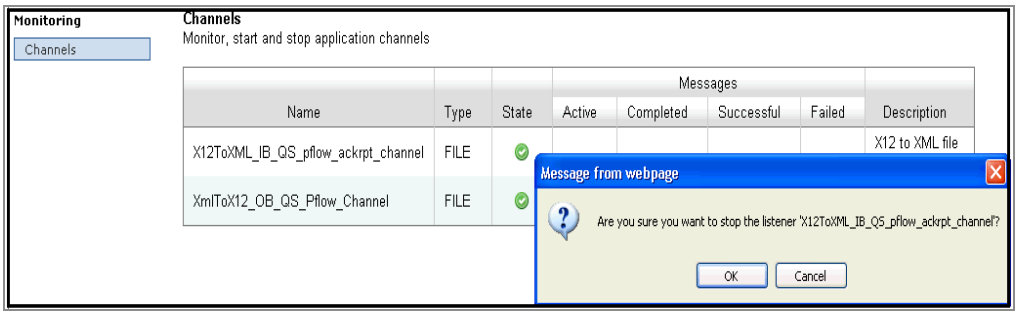
Stopping Inbound (EDI X12 to XML) and Outbound (XML to EDI X12) Processing

This section describes how to stop inbound (EDI X12 to XML) and outbound (XML to EDI X12) processing.

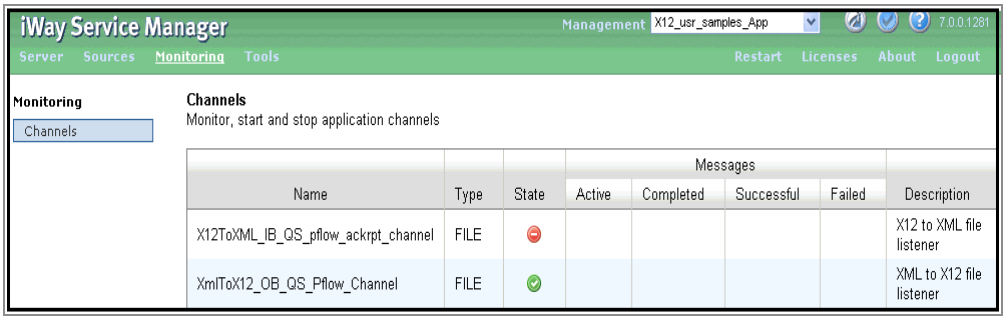
Stop Inbound (EDI X12 to XML) Processing

Procedure

1. Click the **State** icon  adjacent to the inbound application channel (X12ToXml_IB_QA_AckRpt_Pflow_Channel) under Management\Monitoring and click **OK**, as shown in the following image.



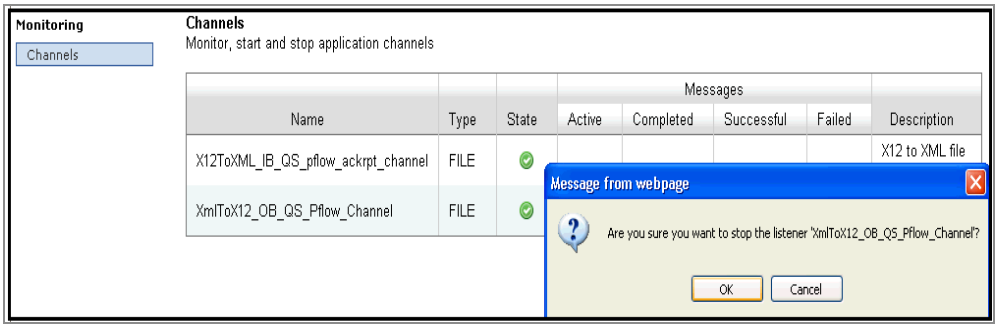
The inbound application channel will be stopped, as shown in the following image.



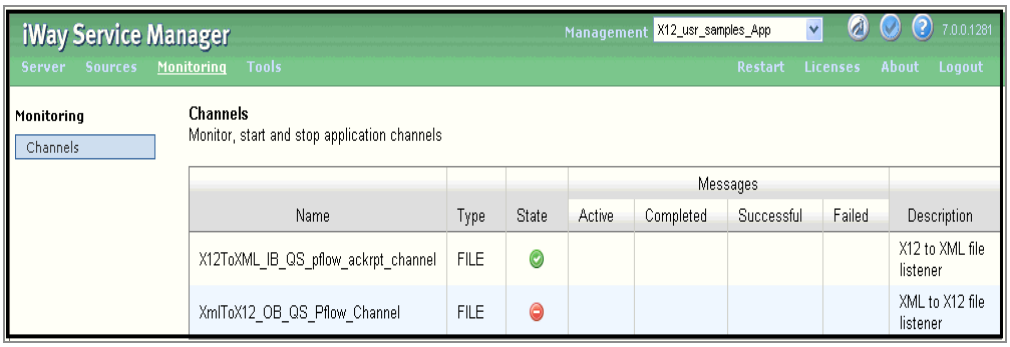
Stop Outbound (XML to EDI X12) Processing

Procedure

1. Click the **State** icon  adjacent to the outbound application channel (XmlToX12_QS_Channel) under Management\Monitoring and click **OK**, as shown in the following image.



The outbound application channel will be stopped, as shown in the following image.



Testing the Sample EDI X12 Applications

This section describes how to test the sample inbound (EDI X12 to XML) and outbound (XML to EDI X12) applications.

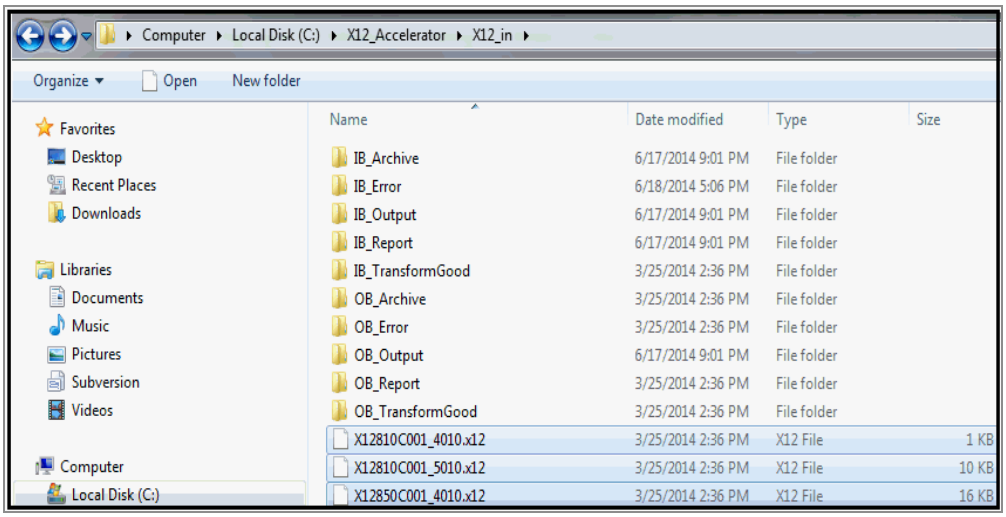
Test the Sample Inbound (EDI X12 to XML) Application

Procedure

1. Copy the input test data to the following directory:

   ```
   X12_Accelerator\X12_in
   ```

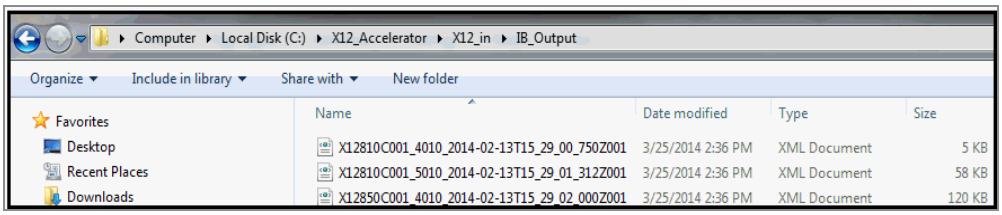
   For example:



2. Observe the transformed XML output in the following directory:

```
X12_Accelerator\X12_in\IB_Output
```

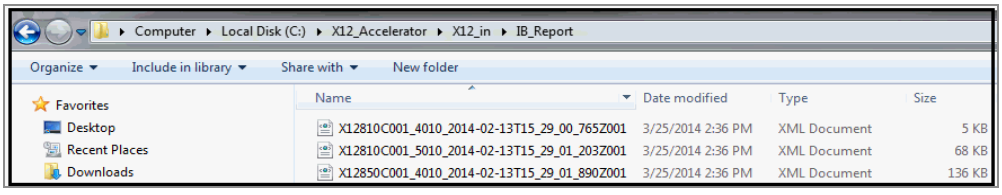
For example:



3. Observe the Reports in the following directory:

```
X12_Accelerator\X12_in\IB_Report
```

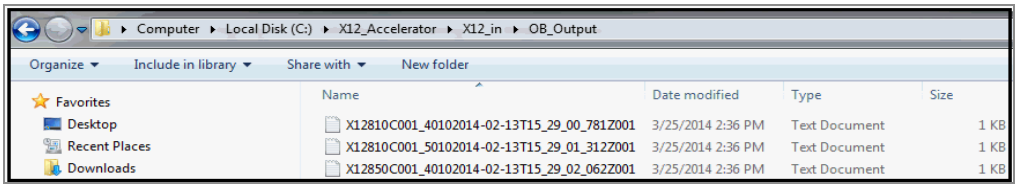
For example:



4. Observe the Acknowledgment in the following directory:

```
X12_Accelerator\X12_in\OB_Output
```

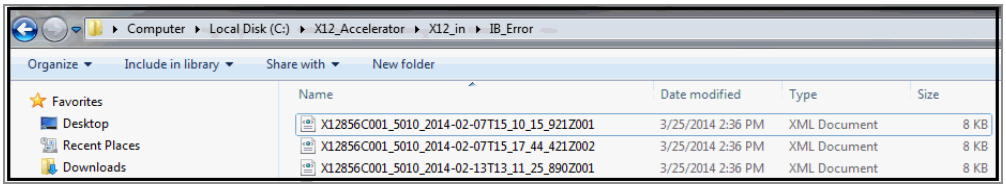
For example:



5. If any Error occurs in the input test data then observe Error data in the following directory:

```
X12_Accelerator\X12_in\IB_Error
```

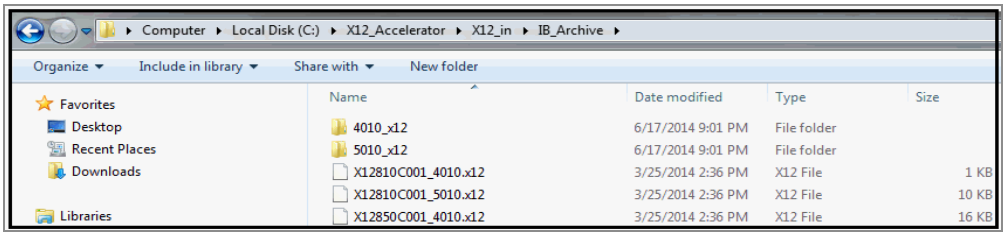
For example:



6. After processing the input data that you place for transformation, a copy of input data will get stored in the following directory:

```
X12_Accelerator\X12_in\IB_Archive
```

For example:



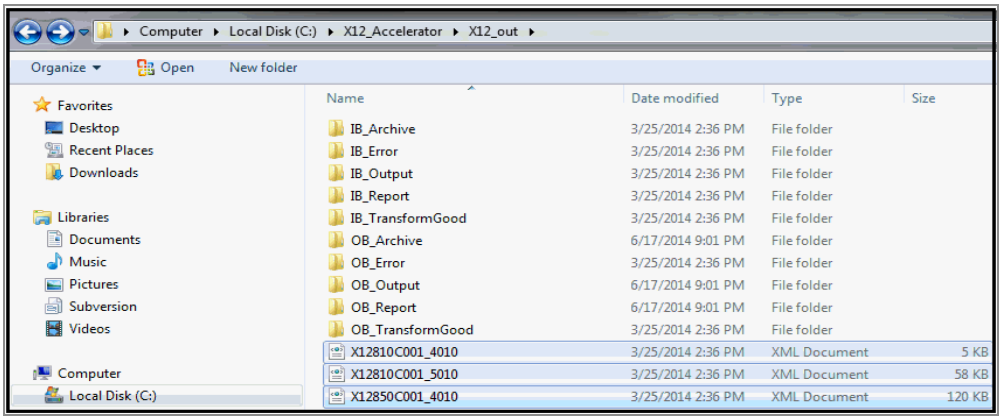
Test the Sample Outbound (XML to EDI X12) Application

Procedure

1. Copy the input test data to the following directory:

   ```
   X12_Accelerator\X12_out
   ```

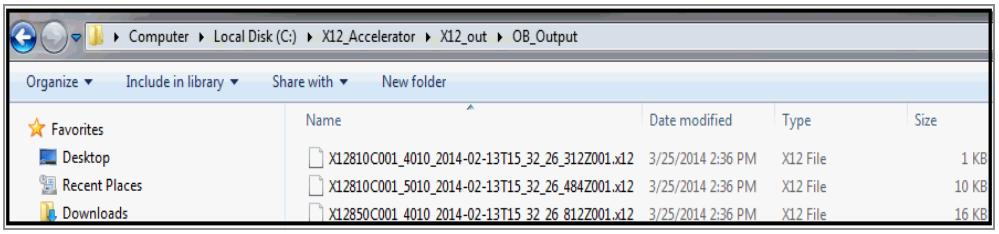
   For example:

2. Observe the transformed XML output in the following directory:

   ```
   X12_Accelerator\X12_out\OB_Output
   ```

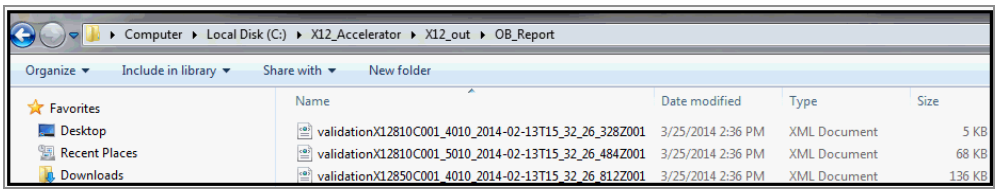
   For example:

3. Observe the Reports in the following directory:

   ```
   X12_Accelerator\X12_out\OB_Report
   ```

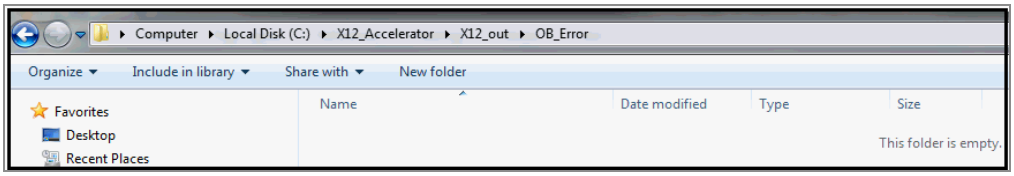
   For example:



4. If any Error occurs in the input test data then observe Error data in the following directory:

```
X12_Accelerator\X12_out\OB_Error
```

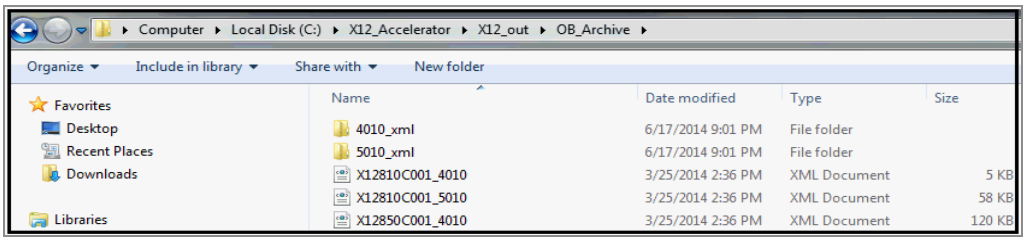
For example:



5. After processing the input data that you place for transformation, a copy of input data will get stored in the following directory:

```
X12_Accelerator\X12_out\OB_Archive
```

For example:



Inbound Processing: EDI X12 to XML

The iWay Integration Solution for EDI X12 includes iWay Service Manager. iWay Service Manager converts a document from Electronic Data Interchange (EDI) X12 format to XML format, and validates it based on EDI published implementation guides.

This chapter provides the information you need to understand and implement a basic inbound message flow.

- The **inbound processing overview** describes the iWay business components and the processing steps in the basic inbound message flow.
- The **sample configuration** contains detailed instructions for configuring the basic inbound message flow. This topic guides you through each step of the configuration procedure.

EDI X12 Inbound Processing Overview

The inbound process converts an EDI X12 formatted document to an XML document.

In a basic message flow, inbound processing consists of the following components and steps. For an illustration of the components available in the construction of a message flow, see Using a Channel to Construct a Message Flow. You will define the components in the configuration instructions in Sample Configuration for Inbound Processing: EDI to XML.

Inlet

- The **listener** picks up the incoming EDI document.
- The **preparser** obtains the message type and version from the EDI document, in order to select the appropriate transformation template name. The transformation template converts the original EDI document to an XML representation of that document.

  The preparser ensures that the document is converted to a structurally correct EDI XML document. The transformation templates that are provided in the Ebix are used to transform the structure of the document.

  The iWay Integration Solution for EDI X12 supports one preparser

(EDIX12SplitterPreParser), which is described in Components of the iWay Integration Solution for EDI.

Validation

- The inbound EDI document is validated for structure and content. The published EDI standards and user implementation guides define element types (for example, numeric, alpha, or date) and describe business rules to apply for validation.

  For example, here is a typical date segment in an inbound EDI document:

  ```
  DTM*001*20080701
  ```

  The value in DTM01 ("001") is validated against an allowed code list. The value in DTM02 ("20080701") is validated as a properly formatted date.

  In addition, the following business rule is applied: DTM02 is required if DTM01 is present (if there is a qualifier, there must be data).

Route

- In our basic message flow example, the route will redirect the transformed document to a designated folder that is dependent on rules validation. After validation, you can apply any additional business logic to the document. You can use a single service or multiple services, passing the output of one service as the input of the next.

  For details on available services, see the *ibi™ iWay® Service Manager User's Guide*.

- The **acknowledgement service** creates a functional acknowledgment (997) for the inbound document. The acknowledgment indicates that the document was received and validated for structure.

- The **validation report service** creates a validation report in XML format, which is routed to a reports folder. This validation report indicates a success or failure result based on X12 validation rules.

Outlets

Outlets define how messages leave a channel at the end of a process. In our basic example, two outlets are configured in the route and one outlet is configured in the channel.

- The two outputs defined in the route will be XML documents. Documents will be placed into their appropriate folders dependent on the results of the EDI X12 rules validation.

- The report outlet contains the validation report in XML format. This document

contains the inbound data as well as the output transformed XML.

- The output defined in the channel is the functional acknowledgment. A functional acknowledgment is typically returned to the sender of the document.

Sample Configuration for Inbound Processing: EDI to XML

This topic provides step-by-step instructions on how to configure a basic inbound message flow for the iWay Integration Solution for EDI X12. The message flow represents the movement and tasks in the conversion of a message from Electronic Data Interchange (EDI) format to XML format and acknowledgment of the message.

Accessing the iWay Service Manager Administration Console

To access the iWay Service Manager Administration Console, you must first ensure that the iWay Service Manager service is running.

For instructions on starting iWay Service Manager, see the *ibi™ iWay® Service Manager User's Guide*.

Access the iWay Service Manager Administration Console on Windows

Procedure

1. From the Windows desktop, select **Start**, **All Programs**, **iWay 8.0 Service Manager**, and **Console**.

   or,

from a browser such as Microsoft Internet Explorer, enter the following URL,

```
http://host:port
```

where:

host

Is the host machine on which iWay Service Manager is installed. The default value is localhost.

port

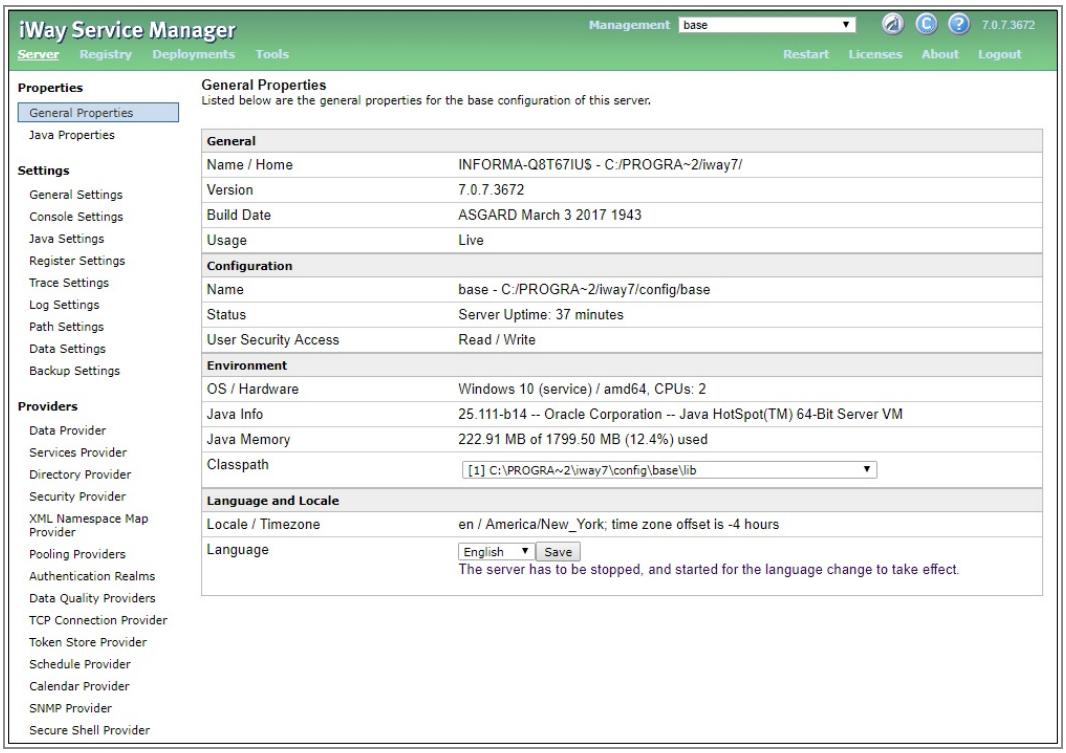
Is the port number on which iWay Service Manager is listening. The default value is 9999.

The following image shows the URL with the default values.

2. When prompted, enter your user name and password, and click OK.

   Note: The default user name and password is *iway*.

   The iWay Service Manager Administration Console opens, as shown in the following image.



Adding an Ebix to the Registry

The iWay e-Business Information Exchange (Ebix) framework supplies several Ebix files for the iWay Integration Solution for EDI X12.

An Ebix file for EDI-X12 is named X12_*transaction_set*.ebx, where *transaction_set* is the transaction set number. For example, the Ebix file for EDI X-12 transaction set 4050 is named X12_4050.ebx.

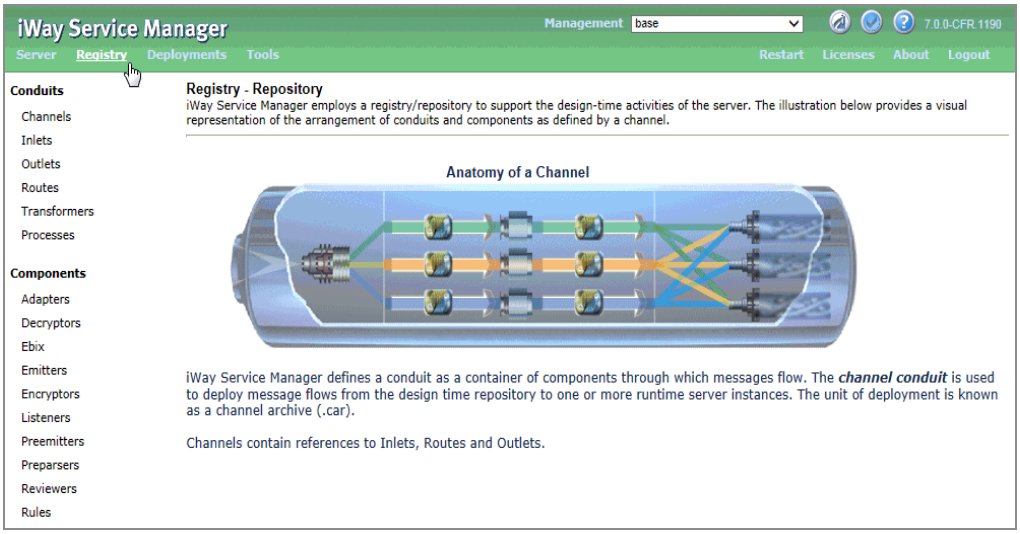
For details on the supported EDI-X12 transaction sets, see Ebix-Supported Transaction Sets.

This topic describes how to add an Ebix to the Registry on Windows and UNIX.

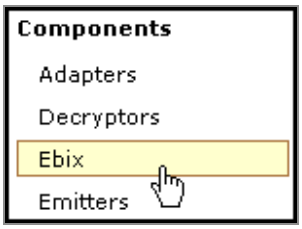
Add an Ebix to the Registry on Windows

Procedure

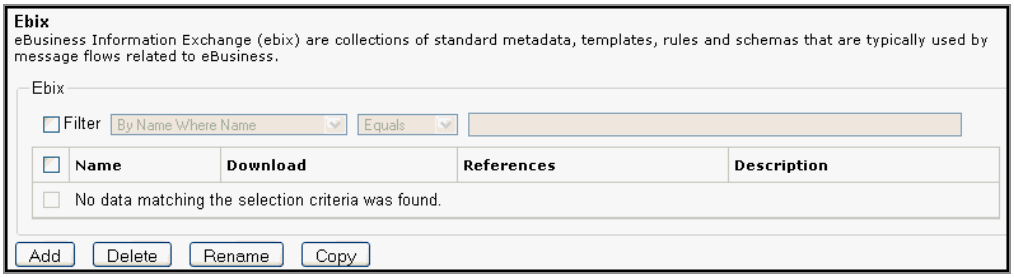
1. To access the Registry, select the **Registry** option in the green shaded area below the iWay Service Manager banner, as shown in the following image.



2. Under Components in the left pane of the Registry, select **Ebix**.

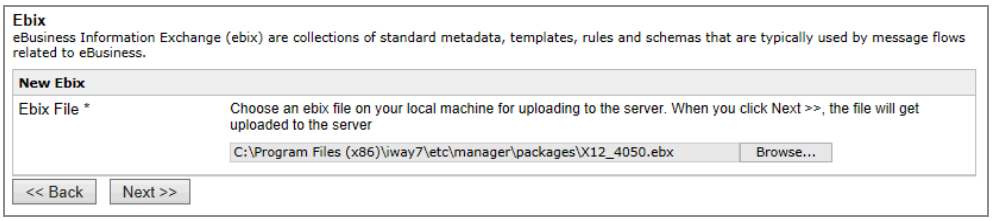


The Ebix pane opens, as shown in the following image.



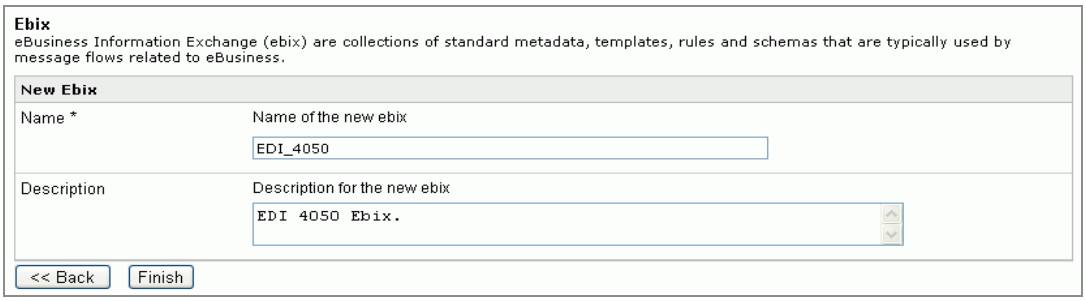
3. Click **Add** to add a new Ebix.

The New Ebix pane opens.



4. Browse to the directory in which the Ebix is located and select the name of the file, for example, **X12_4050_pipeline.ebx**.
5. Once you have selected the Ebix, click **Next**.

   You are prompted for the name of the Ebix and an optional description.



6. Enter a name for the Ebix, for example, **EDI_4050**, and an optional description, such as **EDI 4050 Ebix**.

   Note: This step must be repeated for each Ebix X12 message set that is added to the Registry.
7. Click **Finish**.

   On the Ebix pane, you will see that the Ebix was successfully added. Later you will associate it with the channel for inbound processing.

Add an Ebix to the Registry on UNIX

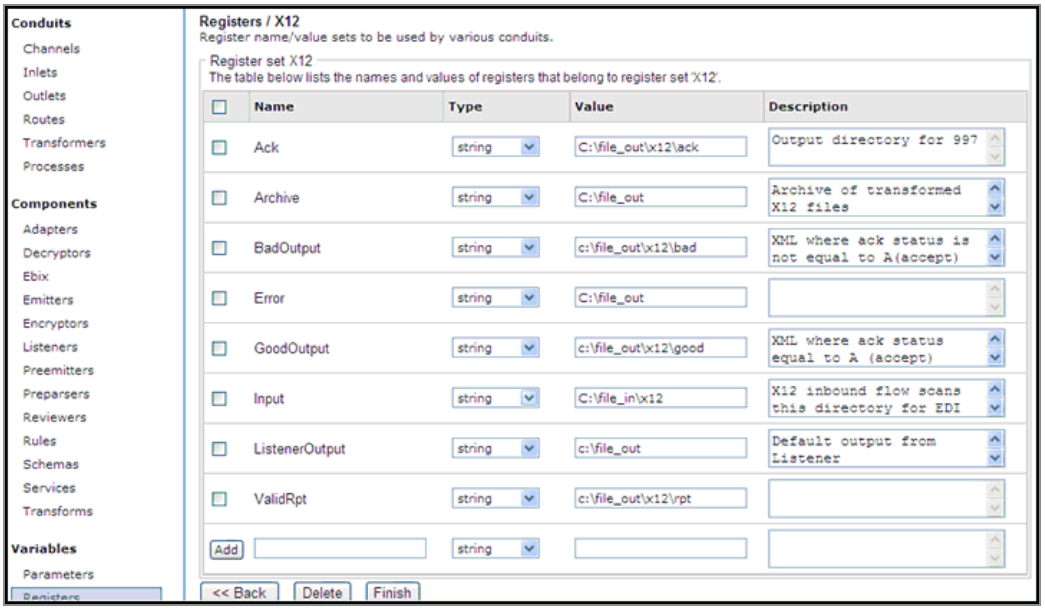
Depending on your system configuration, there are two methods that you can use to add an Ebix to the Registry on UNIX.

- If you have a web browser on the UNIX machine, follow the instructions for Windows.
- Use FTP to download the Ebix from the *iway7/etc/manager/packages* directory to your Windows machine and follow the instructions for Windows.

Adding Special Register Sets

In iWay Service Manager, a special register is a name-value pair that defines a variable that is carried throughout the system. Once defined, this variable is available to all components of the system. Within the EDI components, a Best Practice is to use special registers to define inputs and outputs. When packages containing channels are migrated between systems, the only changes required to deploy in the new location is to modify these special registers and build the channel. Channels may have many locations and this practice will minimize the effort required to migrate. For a complete list of system special registers that are provided, see the *ibi™ iWay® Service Manager Programmer's Guide*. For more information on defining a special register of your own, see the *ibi™ iWay® Service Manager User's Guide*.

The sample inbound channel uses a set of special registers defined as X12. For example:



Add a Special Register Set to Your Channel

To add a special register set to your channel:

Procedure

1. In the left console pane of the Registry menu, select **Channels**.

The Channels pane opens.

2. In the row for your channel, click **Regs** for the channel you want to modify.

   The Assign register pane opens.

3. Select a register and click **Finish**.
4. Click **Back** to return to the Channels pane.

Defining an Inlet

An inlet defines how a message enters a channel. It typically contains a:

- **Listener.** A listener is a component that picks up input on a channel from a configured end point.
- **Decryptor.** A decryptor is a component that applies a decryption algorithm to an incoming message and verifies the security of the message. The configuration example in this topic does not include a decryptor, which is an optional component.
- **One or more preparsers.** A preparser is a component that converts incoming messages into processable documents. Typically a preparser converts a document into XML format. Other preparsers may perform data decryption or reformatting.

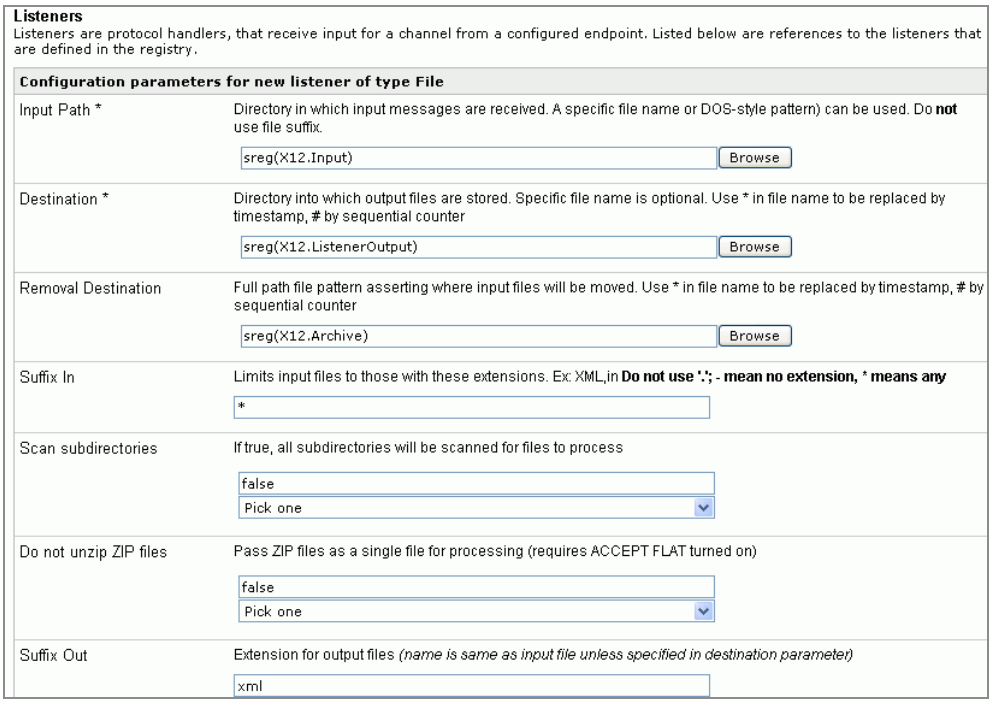
Add a Listener

Procedure

1. From the Registry menu options on the left pane, select **Listeners** under Components.
2. On the Listeners pane on the right, click **Add** to add a new listener.
3. For the purpose of this example, we will show the configuration with a File listener. For details on supported protocols, see the *ibi™ iWay® Service Manager Protocol Guide*.

   Select **File** from the Type drop-down list and click **Next**.

   The configuration parameters pane opens.



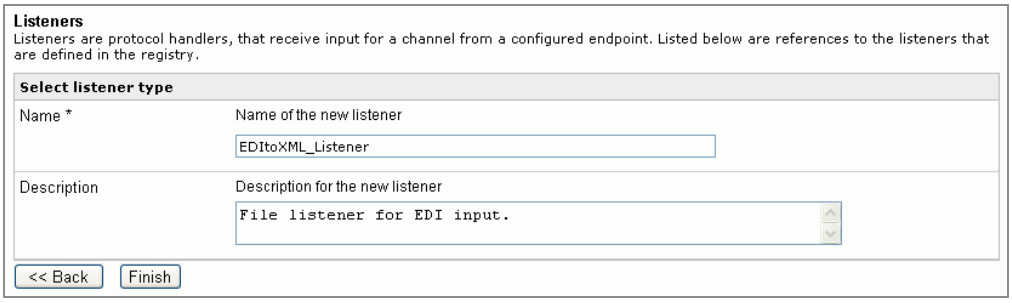
4. Supply configuration parameters for the new File listener as follows. An asterisk indicates that a parameter is required. For parameters not listed in the following table, accept the default value.

Parameter	Value
Input Path *	sreg(X12.Input) This value is a special register that uses a defined directory in which input messages are received. Make sure that you have created this directory; otherwise, errors will occur during deployment.
Destination *	sreg(X12.ListenerOutput) This value is a special register that uses a defined directory in which output files are stored after transformation.

Parameter	Value
	Make sure that you have created this directory; otherwise, errors will occur during deployment.
Removal Destination	sreg(X12.Archive) This value is a special register that uses a defined directory to which input messages are moved if they fail during transformation. Make sure that you have created this directory; otherwise, errors will occur during deployment. It is recommended to configure a removal destination when you are constructing a basic channel.
Suffix In	* Input files with any file extension are allowed.
Suffix Out	xml The extension for output files is .xml.

5. Click **Next**.

   You are prompted for the name of the listener and an optional description.



6. On the Listeners pane, enter the name of the new listener, **EDItoXML_Listener**, and

an optional description. Then click **Finish** to add the listener.

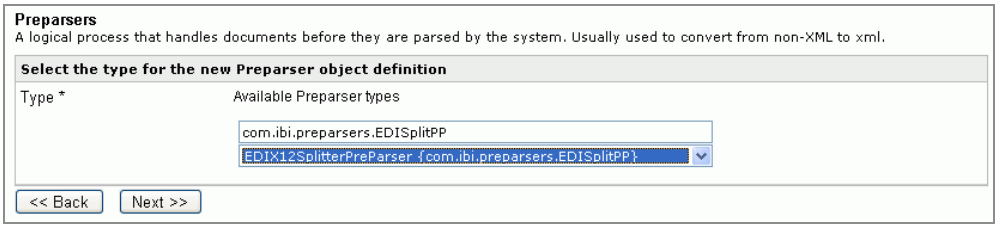
In a later step, you will associate this listener with the inlet.

Add a Preparser

Procedure

1. From the Registry menu options, select **Preparsers** under Components.
2. On the Preparsers pane, click **Add** to add a new preparser.

   You are prompted for the type of preparser.



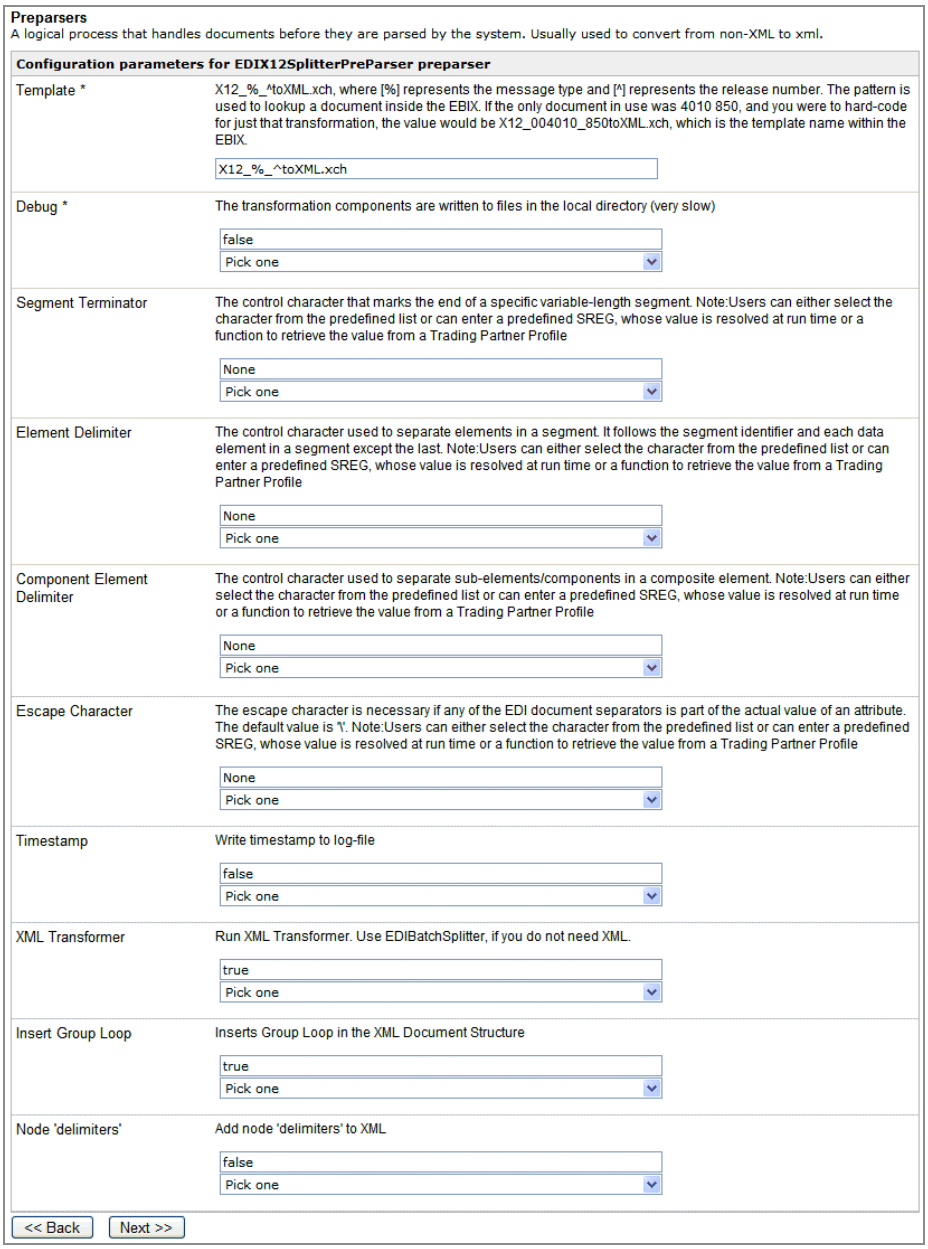
3. Select **EDIX12SplitterPreParser (com.ibi.preparsers.EDISplitPP)** from the Type drop-down list.

   The **EDIX12SplitterPreParser** parses an EDI input file with one or more ISAs and multiple transaction sets (STs), and creates *multiple* XML output files. One XML output file is produced for each transaction set. You can also use the EDIX12SplitterPreParser if there is only one transaction set in an ISA.

   For details on the supported EDI-X12 transaction sets, see Ebix-Supported Transaction Sets.

4. Click **Next**.

   The Preparsers configuration parameters pane opens.



The following table lists and describes the available configuration parameters for the preparser:

Parameter	Description
Template	Used to locate the template in the Ebix used in the transformation from EDI format to XML format.
Debug	If enabled, the transformation components are written to files in the local directory. This parameter is set to False by default.
Segment Terminator	The control character that marks the end of a specific variable-length segment. To view a list of segment terminator characters, see Using EDI X12 Separators and Terminators.
Element Delimiter	The control character used to separate elements in a segment. It follows the segment identifier and each data element in a segment except the last. To view a list of element delimiter characters, see Using EDI X12 Separators and Terminators.
Component Element Delimiter	The control character used to separate sub-elements/components in a composite element. To view a list of component element delimiter characters, see Using EDI X12 Separators and Terminators.
Escape Character	The escape character is necessary if any of the EDI document separators is part of the actual value of an attribute.
Timestamp	Disabled by default, this option writes a

Parameter	Description
	timestamp to the log file. When enabled, the log file will display the start and end time of the file transformation and the input file name that is used. This feature is useful in development or debugging environments when processing batches of files. When the transaction log is not in use (for example, in a production mode) then this information is available in the Activity Log. **Note:** To use this feature, logging must be enabled in the *Log Settings* section of the iWay Service Manager Administration Console.
XML Transformer	Enabled by default, this parameter sets the EDIX12SplitterPreParser to transform the individual documents that are split from the incoming message into XML format. **Note:** Use the standalone EDI batch splitter preparser (com.ibi.preparsers.XDEDIBatchSplitter) if you do not require an XML transformation to be called.
Insert Group Loop	Inserts a group loop tag in the XML document. Group loop tags are displayed in activity logs and validation processing reports. **Note:** Ensure that this parameter is set to *false*. By default, this parameter is set to *true*.
Node 'delimiters'	If set to *true*, node delimiters are added to the generated XML document. By default, this parameter is set to *false*.

5. In the Template field, enter **X12_%_^toXML.xch**.

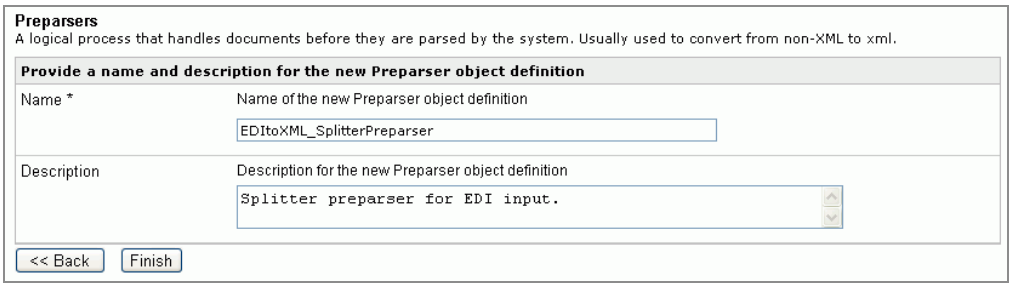
   The preparser obtains the message type and version information from the EDI input

document. In the parameter, the character "%" represents the message type, and the character "^" represents the version.

For example, if the message type of the EDI input document is 810, and the version is 004050, the constructed template name is X12_810_004050toXML.xch.

6. Click **Next**.

   You are prompted for a name and optional description for the new preparser.



7. Enter a name for the new preparser, for example, **EDItoXML_SplitterPreparser**, and an optional description.
8. Click **Finish** to add the preparser.

   In the next procedure, you will associate this preparser with an inlet.

Define an Inlet

Now that you have added a File listener and splitter preparser to the Registry, you are ready to add and define an inlet. You will associate the previously created listener and preparser with the inlet.

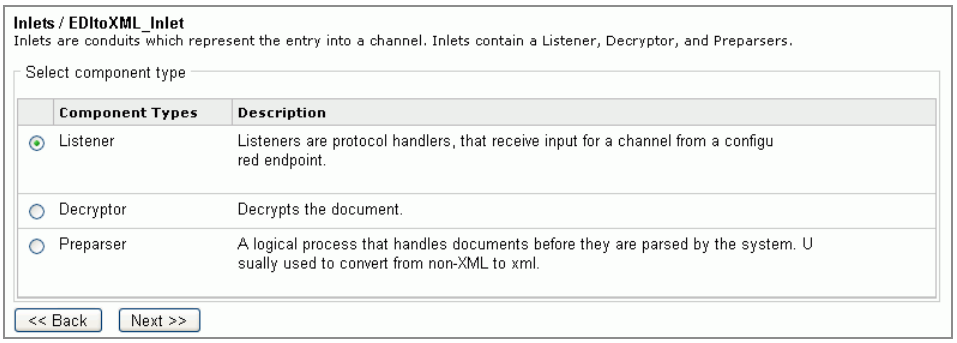
Procedure

1. From the Registry menu options, select **Inlets** under Conduits.
2. On the Inlet Definitions pane, click **Add** to add an inlet.
3. On the New Inlet Definition pane, enter the name of the new inlet and an optional description, as shown in the following table. Then click **Finish** to add the inlet.

Parameter	Value
Name *	EDItoXML_Inlet
Description	Inlet for EDI to XML

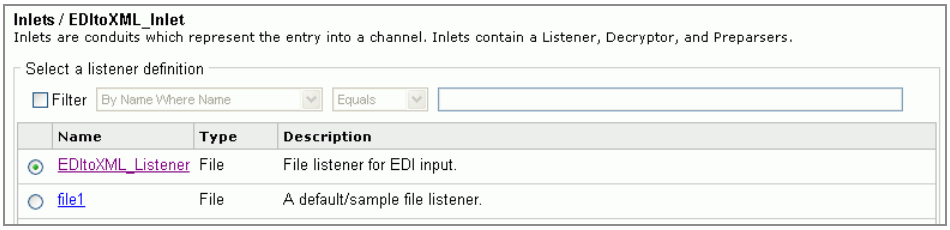
4. On the Construct Inlet pane, click **Add** to associate the listener and preparser with the inlet.

   The next pane prompts you for the component type.



5. Select **Listener** and click **Next**.

   The next pane prompts you to select a listener.

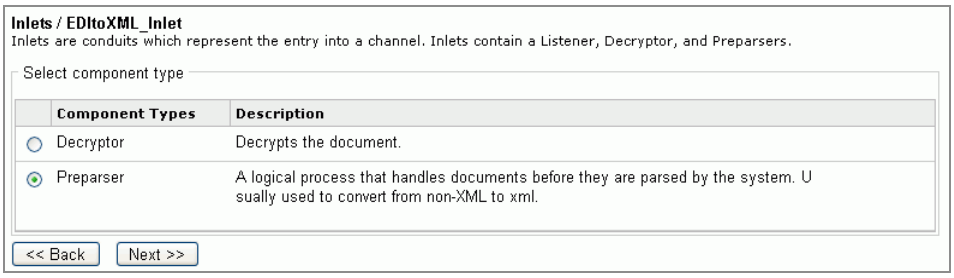


6. Select **EDItoXML_Listener**, which is the listener you added earlier, and click **Finish**.

   The listener is associated with the inlet. Now you need to associate the preparser created earlier with the inlet.

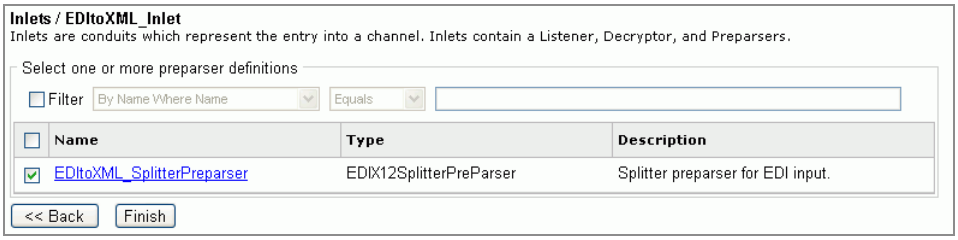
7. On the Construct Inlet pane, click **Add**.

   The next pane prompts you for the component type.



8. Select **Preparser** and click **Next**.

   On the next pane, you are prompted to select a preparser.



9. Select **EDItoXML_SplitterPreparser**, which is the preparser you added earlier, and click **Finish**.

   You have now successfully completed definition of the inlet.

Defining a Route

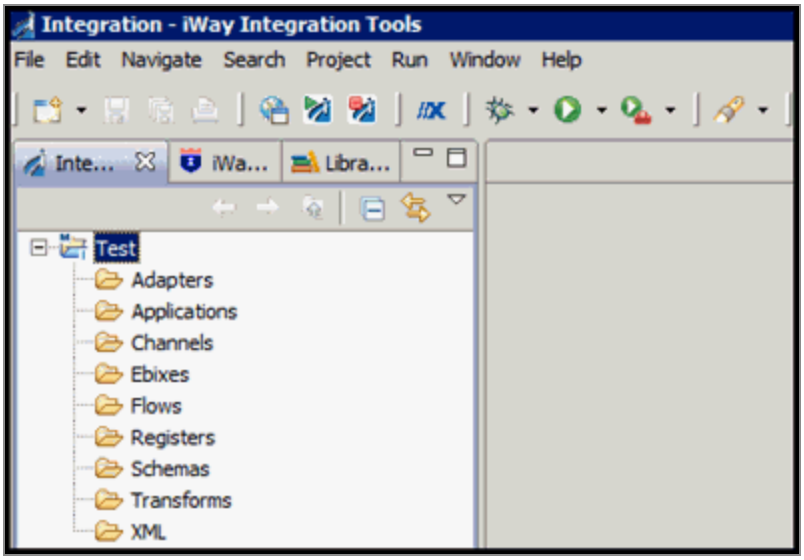
For this sample channel configuration, you will define a route that will invoke the X12 to XML validation process flow. The outcome of the validation process flow will place valid transformed XML data in a defined output folder. Invalid transformed data will be routed to an errors folder. An X12 functional acknowledgment and a validation report will be sent to their designated output folder defined in the sample channel. This section describes how to create a validation process flow using iWay Integration Tools and bind it to a sample inbound channel as a route.

Create a New Project and Start the Process Flow

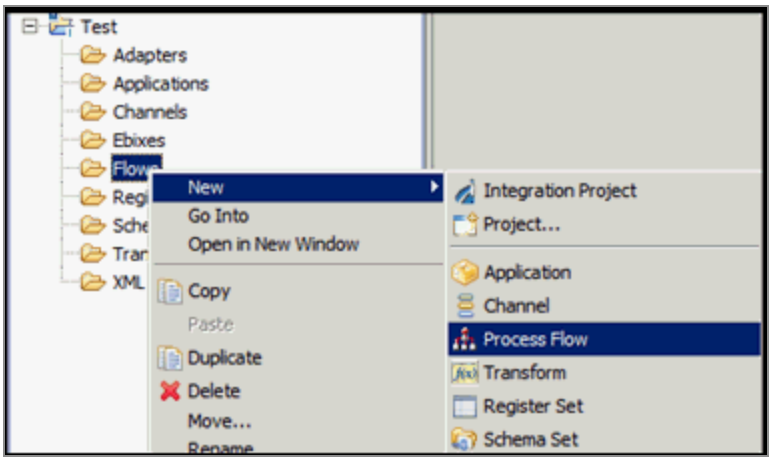
To create a new project and start the process flow using iWay Integration Tools:

Procedure

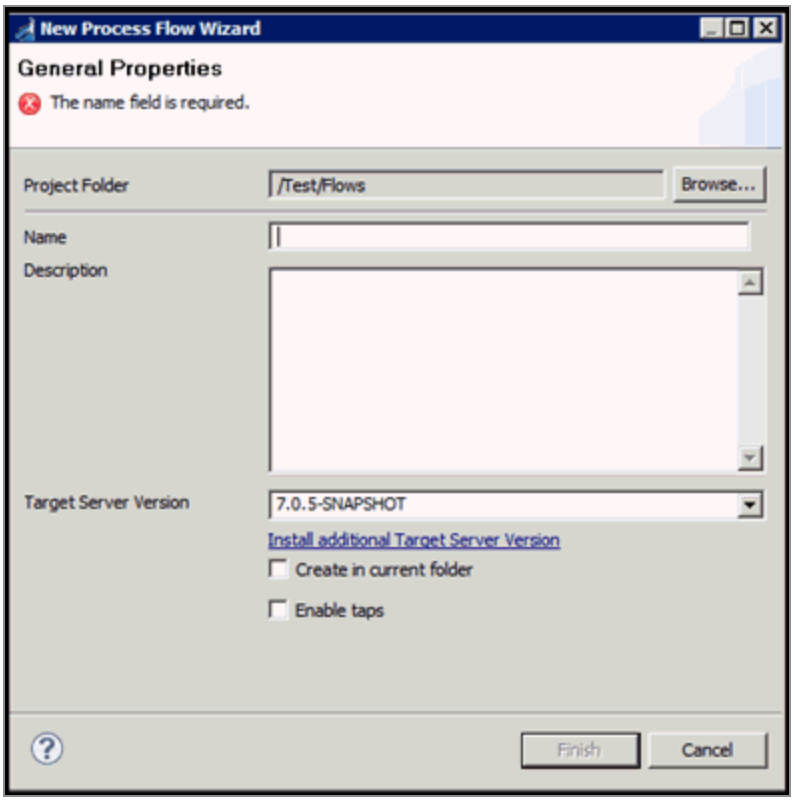
1. Open iWay Integration Tools.
2. Connect to the repository from which you want to work, for example, iWay.
3. Right-click the integration explorer window, select **New**, and then click Integration Project from the context menu.
4. In the Name field, provide a valid integration name, for example, **Test**, and then click **Finish**.



5. Right-click the Flows folder, select **New**, and then click **Process Flow** from the context menu, as shown in the following image.



The New Process Flow Wizard opens, as shown in the following image.



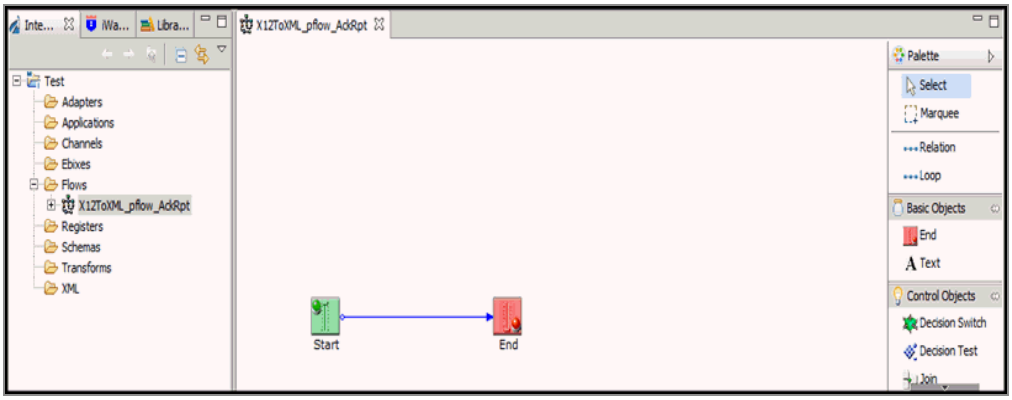
6. In the Name field, type **x12toXML_pflow_AckRpt** as the process flow name.

   In the Description field, type a brief description (optional).

7. Click **Finish**.

   The new x12toXML_pflow_AckRpt node appears under the Flows folder, and the

workspace displays a Start and End object with a relation established in between, as shown in the following image.



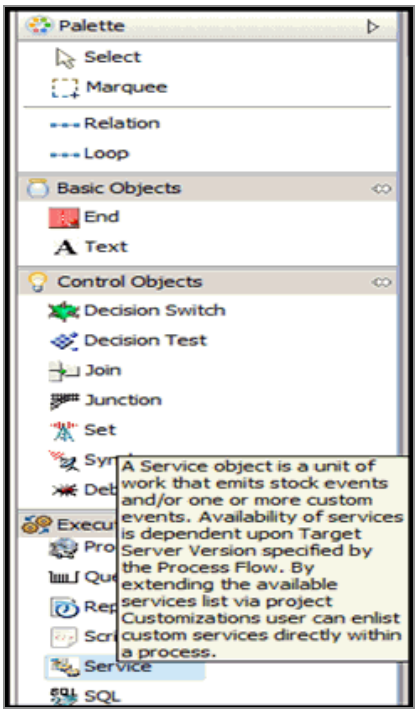
You are ready to build the x12toXML_pflow_AckRpt validation process flow by configuring objects to it and specifying their relationships.

Configure Objects for the Process Flow

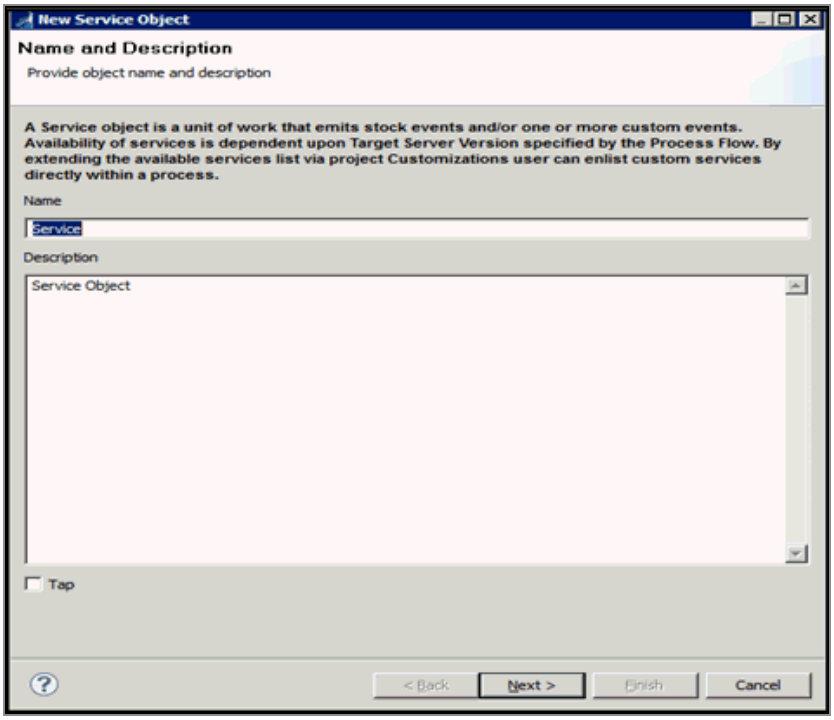
To configure objects for the process flow using iWay Integration Tools:

Procedure

1. Drag and drop the Service object from the toolbar to the workspace, as shown in the following image.



The New Service Object dialog box opens.

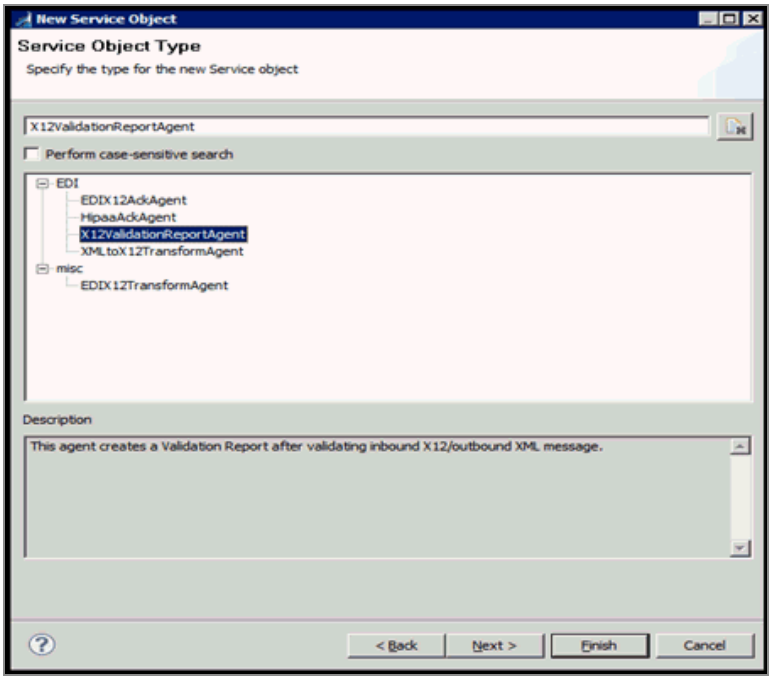


2. In the Name field, type **X12_Validation_Rpt**, and a brief description (optional) in the

Description field and click **Next**.

The Service Object Type wizard opens.

3. Select **Class Name** and enter **com.ibi.agents.XDX12ValidationReportAgent** and click **Next**, as shown in the following image.

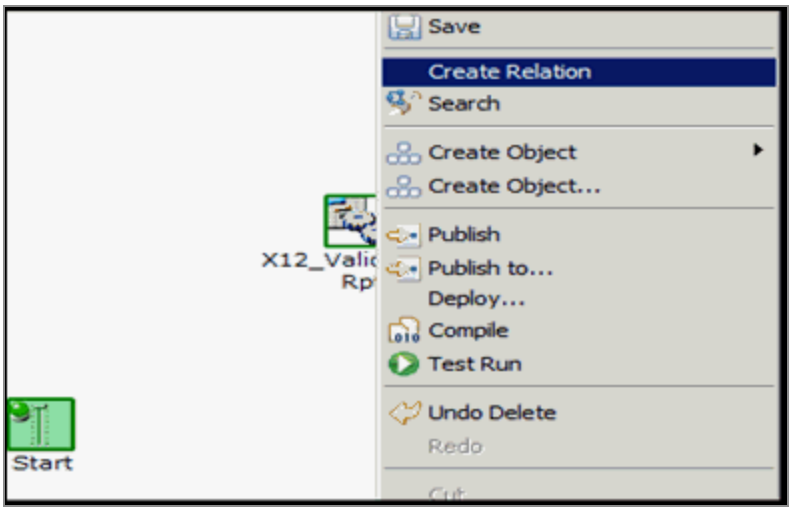


4. Click **Finish**.

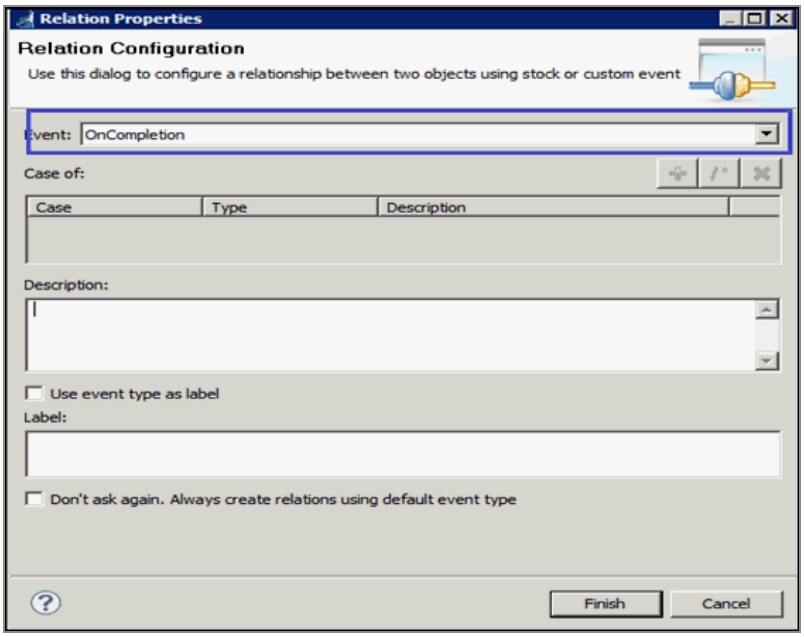
   The new Service object (X12_Validation_Rpt) appears in the workspace.

5. Remove the relation (link) between the Start and End objects.

6. Establish new relation from the Start object to X12_Validation_Rpt by selecting the Start object, right-clicking on the X12_Validation_Rpt object, and then selecting **Create Relation** from the context menu, as shown in the following image.



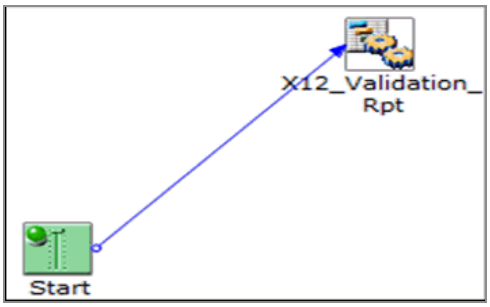
The Relation Properties wizard opens.



7. From the Event drop-down list, select **OnCompletion** and then click **Finish**.

   This option indicates that there are no conditions that affect the path, and that the path between the two objects will always be followed.

   A line appears between the objects to indicate that a relationship has been established, as shown in the following image.



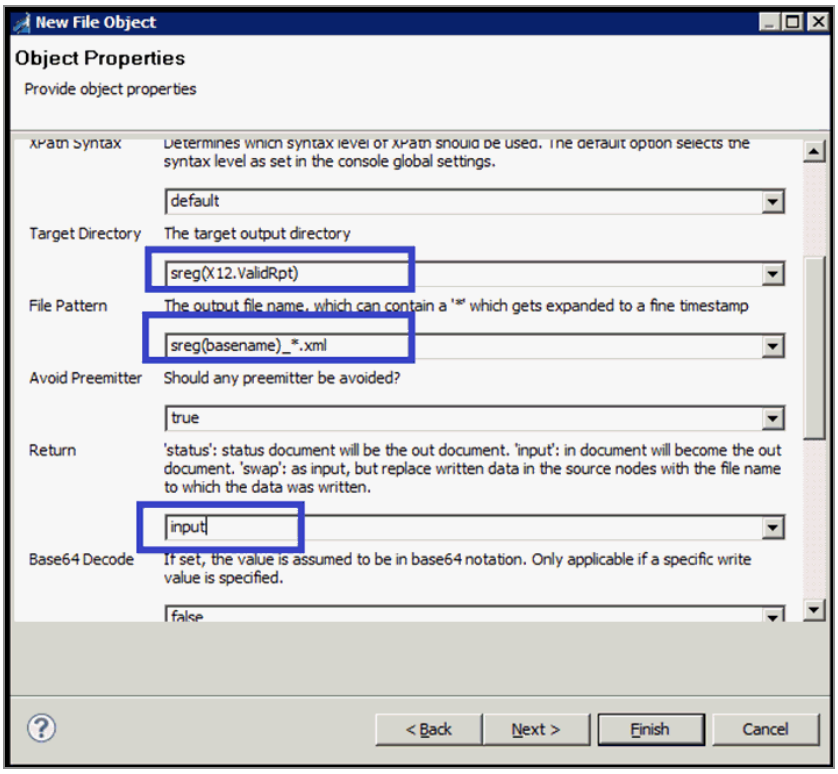
8. Drag and drop the File object from the object palette to the workspace.

   The File Type dialog box opens.

9. In the Name field, type **Write_Validation_rpt**, and a brief description (optional) in the Description field and click **Next**.

   The File Type dialog box opens.

10. From the Type drop-down list, select **File Emit Agent** and click **Next**.
11. For the Target Directory parameter, enter a location where validation report data will be written, for example, **sreg(X12.ValidRpt)**.
12. For the File Pattern parameter, enter **sreg(basename)_rpt.xml**.
13. For the Return parameter, select **input** from the drop-down list and click **Finish**, as shown in the following image.



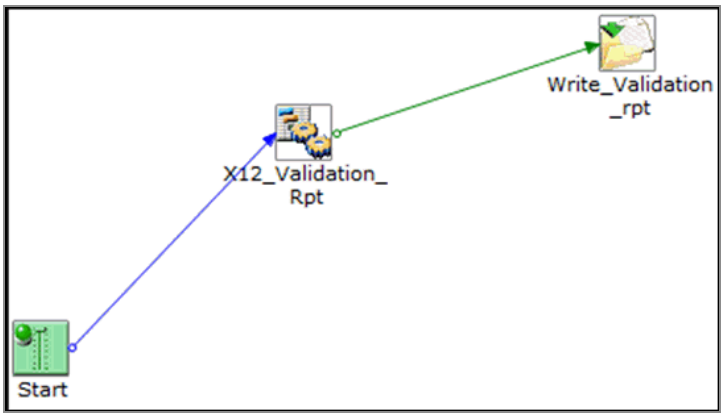
The new File object (Write_Validation_rpt) appears in the workspace.

14. Select the **X12_Validation_Rpt** object, right-click the **Write_Validation_rpt** object, and select **Create Relation** from the context menu.

    The Line Configuration dialog box opens.

15. From the Event drop-down list, select **OnSuccess** and click **OK**.

    A line appears between the objects to indicate that a relationship has been established.



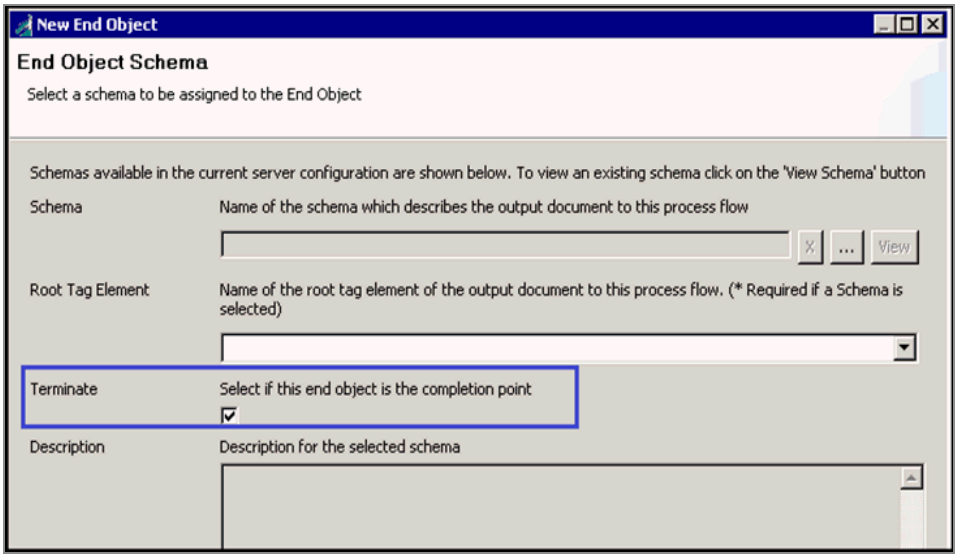
16. Drag and drop the End object from the Object palette to the workspace.

    The End Name and Description dialog box opens.

17. In the Name field, type **End3**, and a brief description (optional) in the Description field and click **Next**.

    The End Name Schema dialog box opens.

18. From the Terminate parameter, select the check box for **Select if this end object is the completion point**, as shown in the following image.

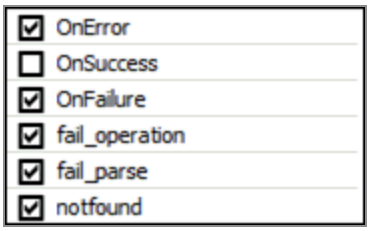


19. Click **Finish**.

20. Select the **Write_Validation_rpt Dir** object, right-click the **End3** object, and select **Create Relation** from the drop-down list.

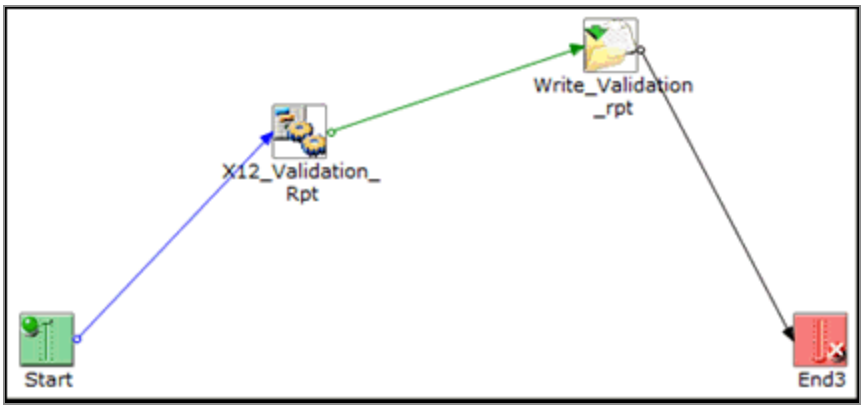
The Relation Configuration wizard opens.

21. From the Event drop-down list, select **OnCustom** and then select the following check boxes:



22. Click **Finish**.

    A line appears between the objects to indicate that a relationship has been established, as shown in the following image.

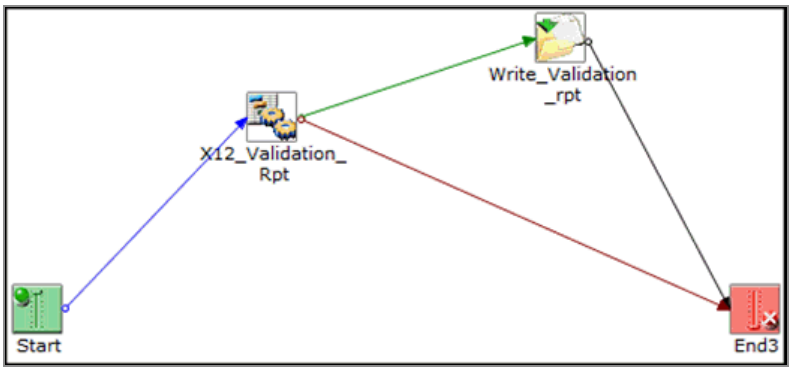


23. Select the **X12_Validation_Rpt** object, right-click the **End3** object, and then select **Create Relation** from the drop-down list.

    The Relation Configuration wizard opens.

24. From the Event drop-down list, select **OnFailure** and click **Finish**.

    A line appears between the objects to indicate that a relationship has been established, as shown in the following image.



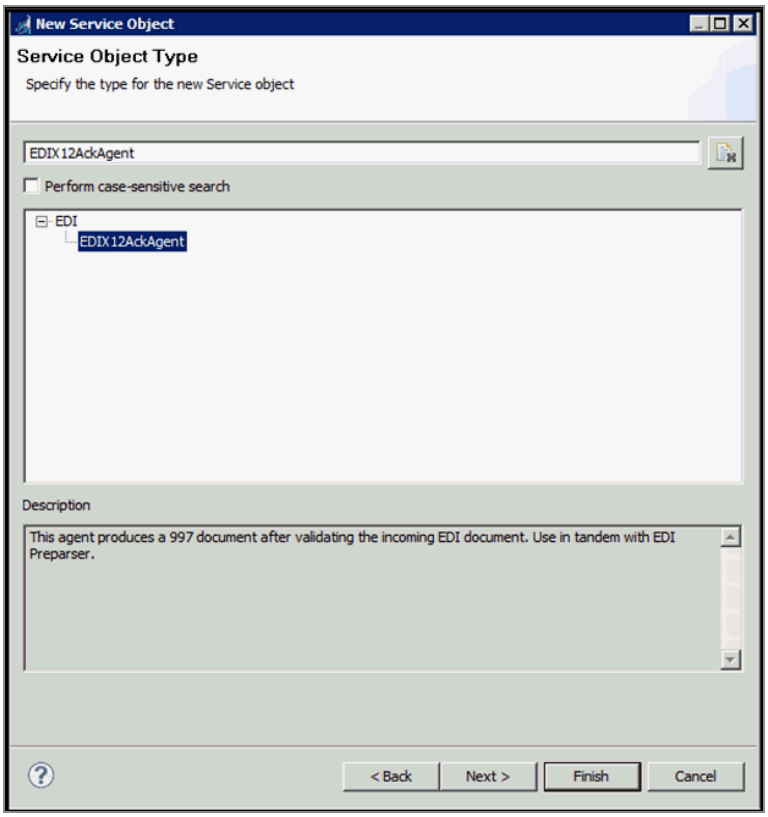
25. Drag and drop the Service object from the Object palette to the workspace.

    The New Service Object dialog box opens.

26. In the Name field, type **X12AckAgent**, and a brief description (optional) in the Description field and click **Next**.

    The Service Type dialog box opens.

27. Select **Class Name** and enter **com.ibi.agents.XDX12AckAgent** and click **Next**.



The Properties dialog box opens. The configuration parameters for EDIX12AckAgent are displayed. The following table lists and describes the configuration parameters.

Parameter	Description
Protocol	Protocol on which to make acknowledgment copies. Select one of the following options from the drop-down list: • NONE • FILE
Location	Location for acknowledgment copies.
End Tag	The surrounding XML tag.

Parameter	Description
Preemitter	Determines whether the preemitter should be run on acknowledgment output.
Error	Determines whether to send an error.
ISA Control Number	Element location of ISA control number. Select one of the following locations from the drop-down list: • Input Document • _SReg(edi.ICN)
GS Control Number	Element location of GS control number. Select one of the following locations from the drop-down list: • Input Document • _SReg(edi.GCN)
ST Control Number	Element location of ST control number. Select one of the following locations from the drop-down list: • Input Document • _SReg(edi.MCN)
Stream Acknowledgment	Determines the level of acknowledgment information to return. Select one of the following acknowledgment levels from the drop-down list: • **Group.** Returns acknowledgment information at the Group level.

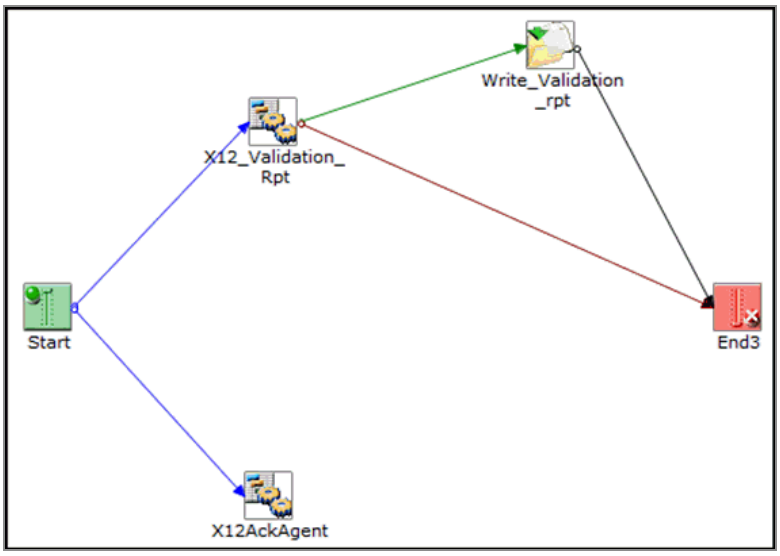
Parameter	Description
	• **Interchange.** Returns acknowledgment information at the Interchange level. • **Transaction.** Returns acknowledgment information at the Transaction level.

28. Configure the available parameters according to your requirements.
29. Click **Finish**.

    The new Service object (X12AckAgent) appears in the workspace.
30. Select the **Start** object, right-click the **X12AckAgent** object, and select **Create Relation** from the context menu.

    The Line Configuration dialog box opens.
31. From the Event drop-down list, select **OnCompletion** and click **OK**.

    A line appears between the objects to indicate that a relationship has been established.



32. Drag and drop the File object from the Object palette to the workspace.

The New File Object wizard opens.

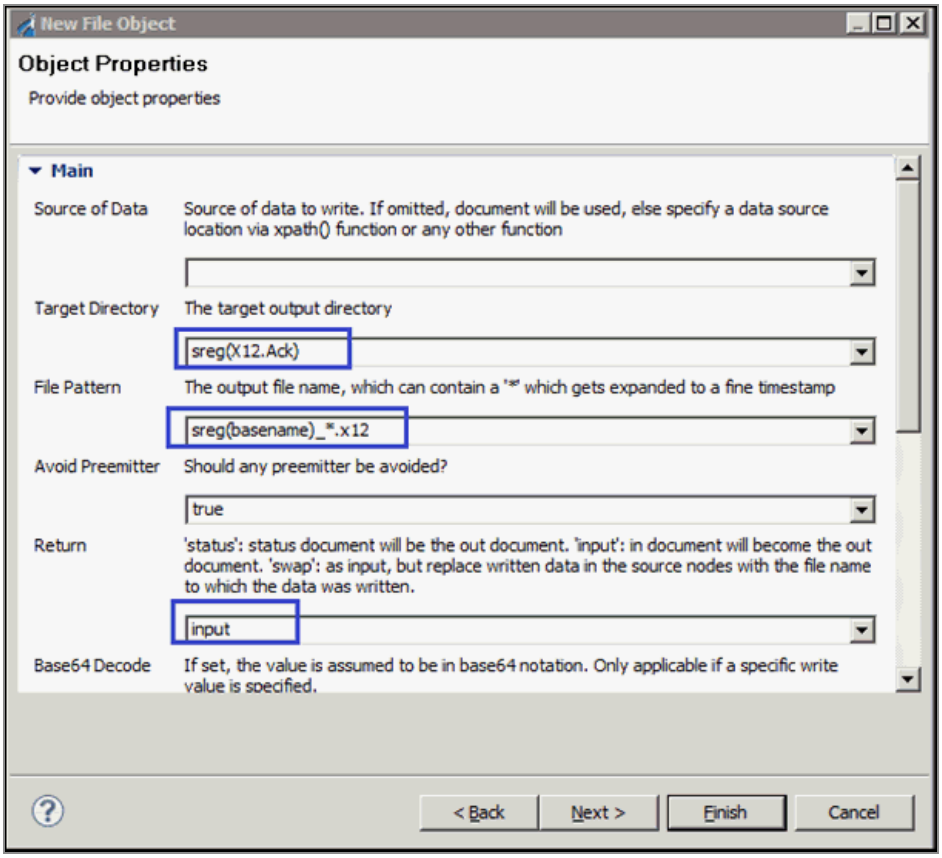
33. In the Name field, type **Write_Ack**, and a brief description (optional) in the Description field and click **Next**.

    The File Object Type wizard opens.

34. From the Type drop-down list, select **File Emit Agent {com.ibi.agents.XDFileEmitAgent}** and then click **Next**.

    The Object properties wizard opens.

35. In the Target Directory field, enter a valid physical folder location to write Acknowledgments data, for example, **sreg(X12.Ack)**, as shown in the following image.



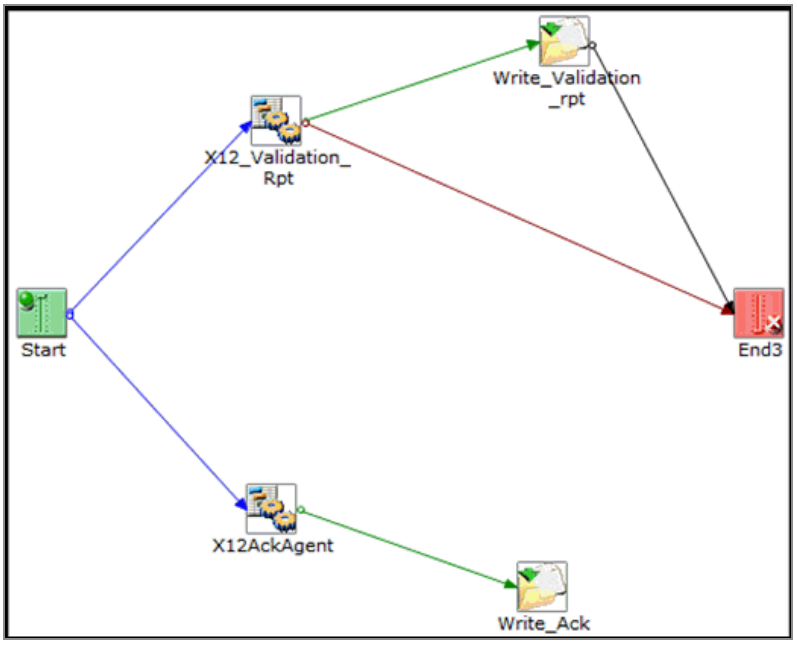
36. In the File Pattern parameter, enter **sreg(basename)_*.x12)**.

37. For the Return parameter, select **input** from the drop-down list and click **Finish**.

    The new File object (Write_Ack) appears in the workspace.

38. Select the **X12AckAgent** object, right-click the **Write_Ack** file object, and select **Create Relation** from the context menu.
39. From the Event drop-down list, select **OnSuccess** and click **Finish**.

    A line appears between the objects to indicate that a relationship has been established, as shown in the following image.



40. Drag and drop the End object from the Object palette to the workspace.

    The End Name and Description dialog box opens.
41. In the Name field, type **End_Ack**, and a brief description (optional) in the Description field and click **Next**.

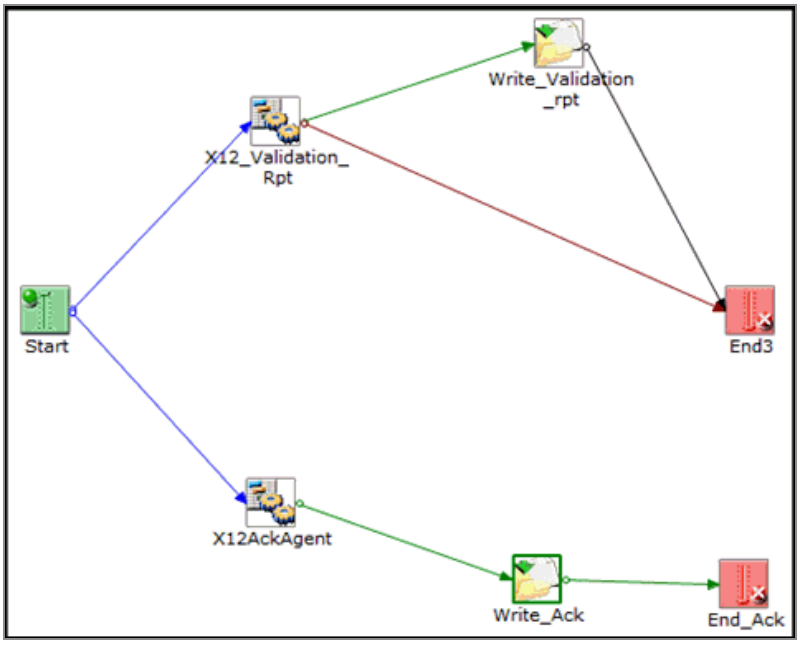
    The End Object Schema wizard opens.
42. From the Terminate parameter, select the check box for **Select if this end object is the completion point**.
43. Click **Finish** to accept the default values and close the dialog box.

    The new End_Ack object appears in the workspace.
44. Select the **Write_Ack** object, right-click the **End_Ack** object, and select **Create Relation** from the context menu.

    The Relation Configuration wizard opens.

45. From the Event drop-down list, select **OnSuccess** and click **Finish**.

    A line appears between the objects to indicate that a relationship has been established.



46. Select the **X12AckAgent** object, right-click the **End3** object (which is already linked with the X12_Validation_rpt and Write_Validation_rpt objects), and select **Create Relation** from the context menu.

    The Relation Configuration wizard opens.

47. From the Event drop-down list, select **OnCustom** and select the following cases from the case parameter list:

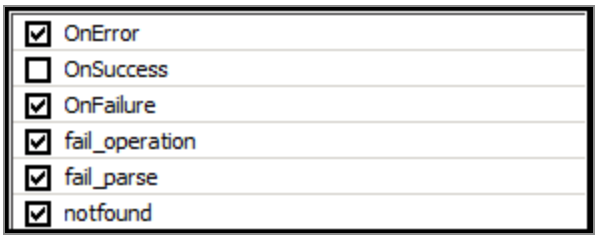
    - OnError
    - OnFailure

48. Click **Finish**.

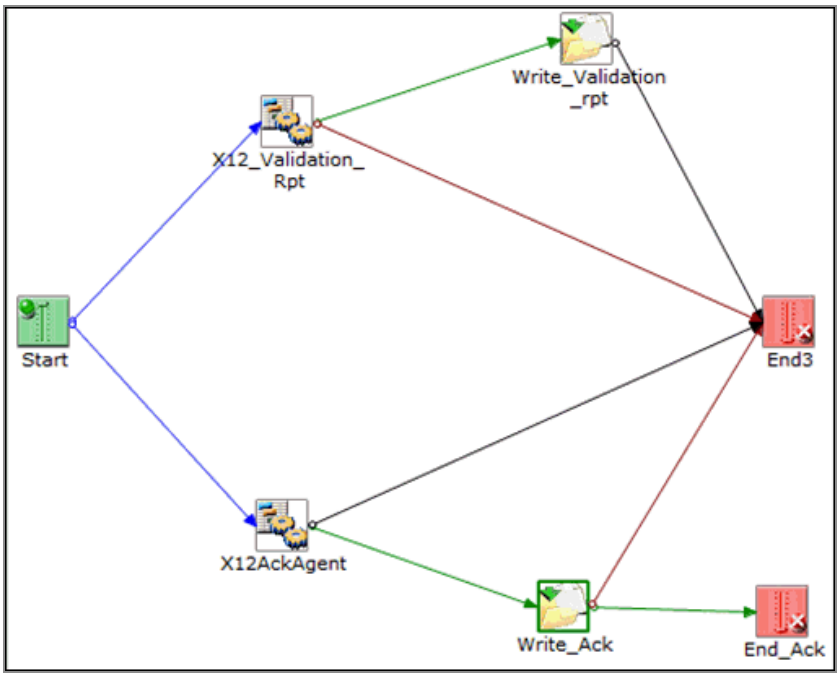
49. Select the **Write_Ack** object, right-click the **End3** object (which is already linked with the X12_Validation_rpt and Write_Validation_rpt objects), and select **Create Relation** from the context menu.

    The Relation Configuration wizard opens.

50. From the Event drop-down list, select **OnCustom** and select the cases from the case parameter list, as shown in the following image:



A line appears between the objects to indicate that a relationship has been established.



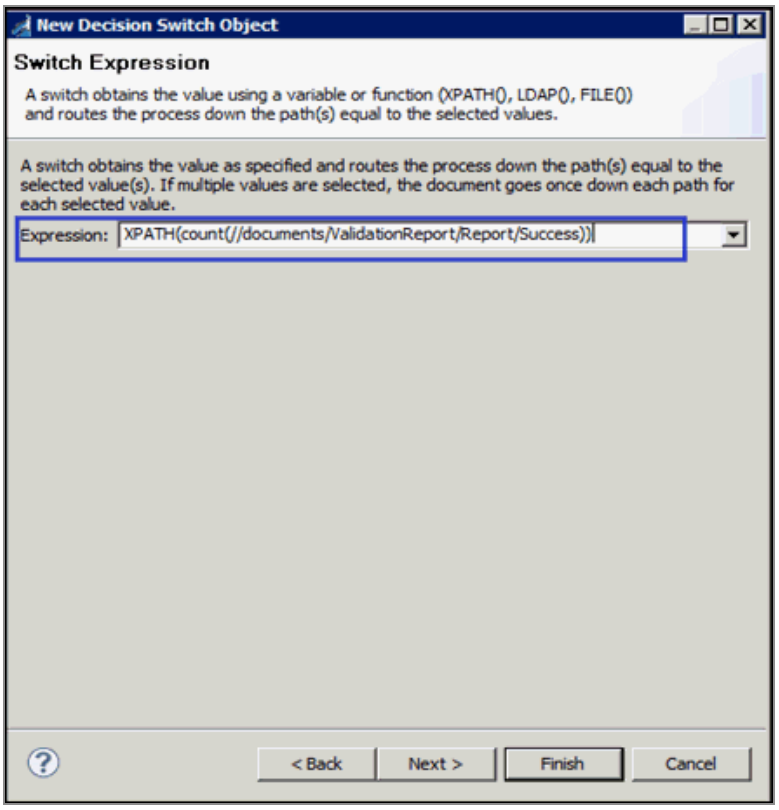
51. Drag and drop the Decision Switch object from the Objects palette to the workspace.

    The New Decision Switch Object wizard opens.

52. In the Name field, type a relevant name (for example, **Decision_switch**), and a brief description (optional) in the Description field and click **Next**.

53. Type the following expression in the Expression parameter:

```
XPATH(count(//documents/ValidationReport/Report/Success))
```

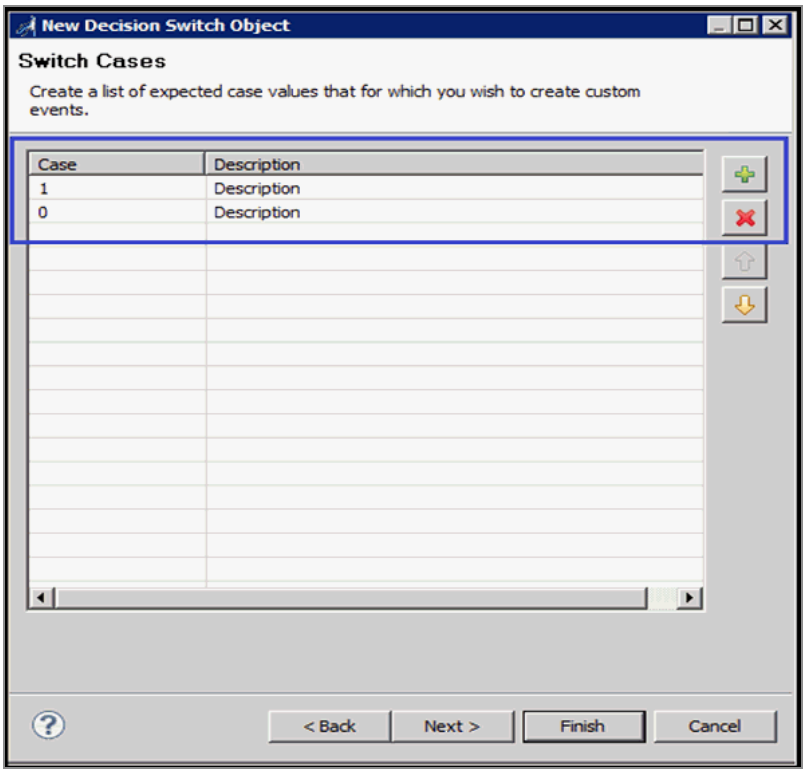


This will check if the validation report is successful or not. If the validation report returns a success node, then the incoming data will go to the Good File write. Otherwise, it will go with the error route to write the error file write.

54. Click **Next**.
55. Add new Switch Cases (1 and 0) by clicking on the Add (**+**) button shown in the Switch Cases wizard.

    You can delete default **empty** and **null** cases from the cases list.

    The following image shows the Switch Cases configuration wizard showing the two cases (1 and 0) in the switch cases list.



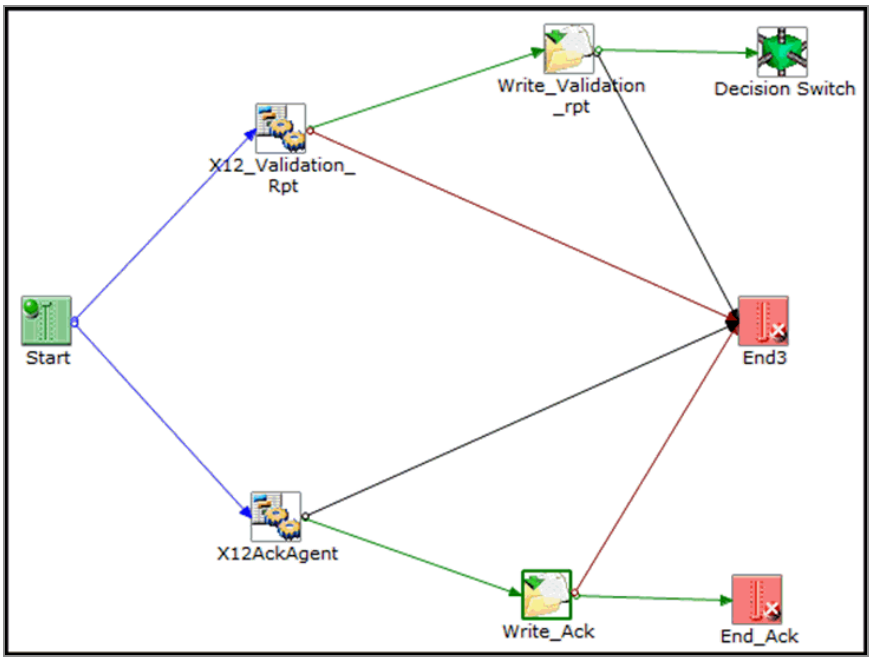
56. Click **Finish** to complete creating the Switch Cases.

    The new Decision Switch case object appears in the workspace.

57. Select the **Write_Validation_rpt** object, right-click the switch case (**Decision Switch**) object, and select **Create Relation** from the context menu.

58. From the Event drop-down list, select **OnSuccess** and click **Finish**.

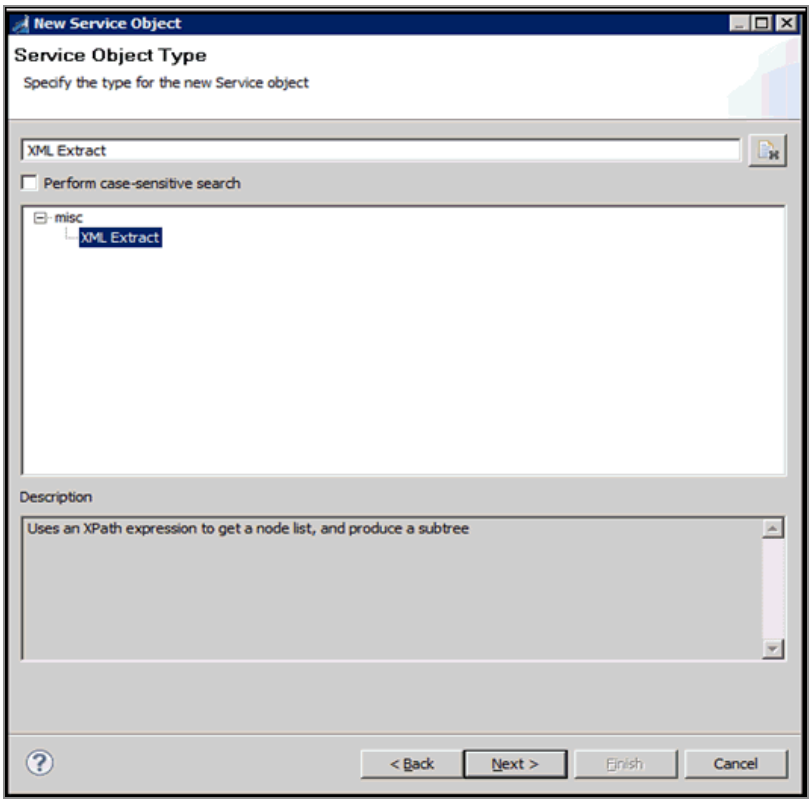
    A line appears between the objects in the workspace to indicate that a relationship has been established.



59. Drag and drop the Service object from the Object palette into the workspace.
60. In the Name field, type **XDXMLExtract**, and a brief description (optional) in the Description field and click **Next**.

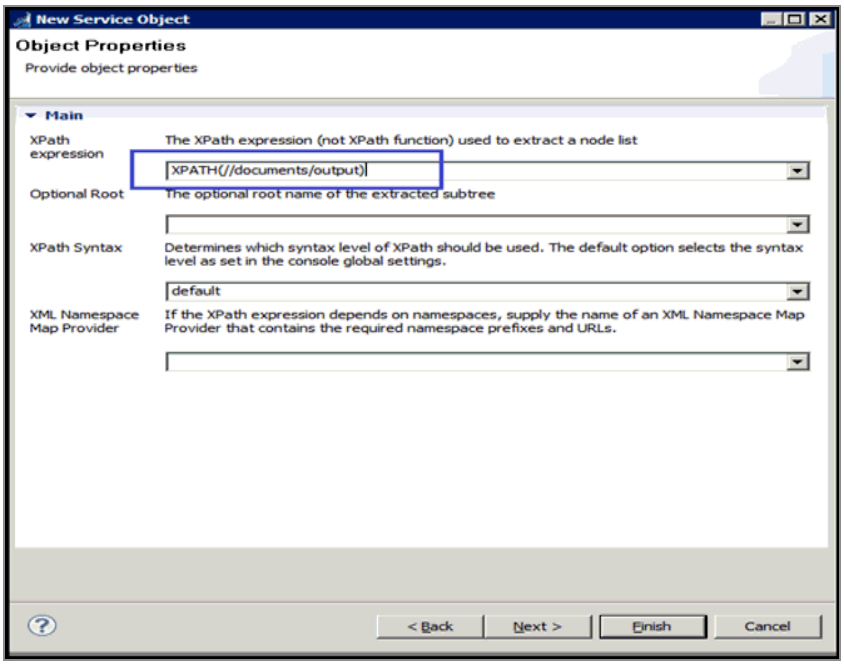
    The Service Object Type wizard opens.
61. In the Service Object Type dynamic search box, enter the following class name:

    ```
    com.ibi.agents.XDXMLExtract
    ```
62. Select the **XML Extract** agent object and click **Next**, as shown in the following image.



63. Provide the XPATH expression (for example, *XPATH(//documents/output*) to extract the node lists from the XML, and click **Finish**.

    Note: This is not the xpath function.



64. Create a copy of XDXMLExtract and name it as *XDXMLExtract1* for non-success validation reports (error route).
65. Select **Decision Switch**, right-click on **XDXMLExtract**, and select **Create Relation** from the context menu.

    The Relation Configuration wizard opens.
66. From the Event drop-down list, select **OnCustom**.
67. In the *Case of* section in the Relation Configuration wizard, select case **1** and then click **Finish**.

    A line appears between the Decision Switch and XDXMLExtract objects to indicate that a relationship has been established.
68. Select **Decision Switch**, right-click on **XDXMLExtract1**, and select **Create Relation** from the context menu.

    The Relation Configuration wizard opens.
69. From the Event drop-down list, select **OnCustom**.
70. In the *Case of* section in the Relation Configuration wizard, select the following cases:

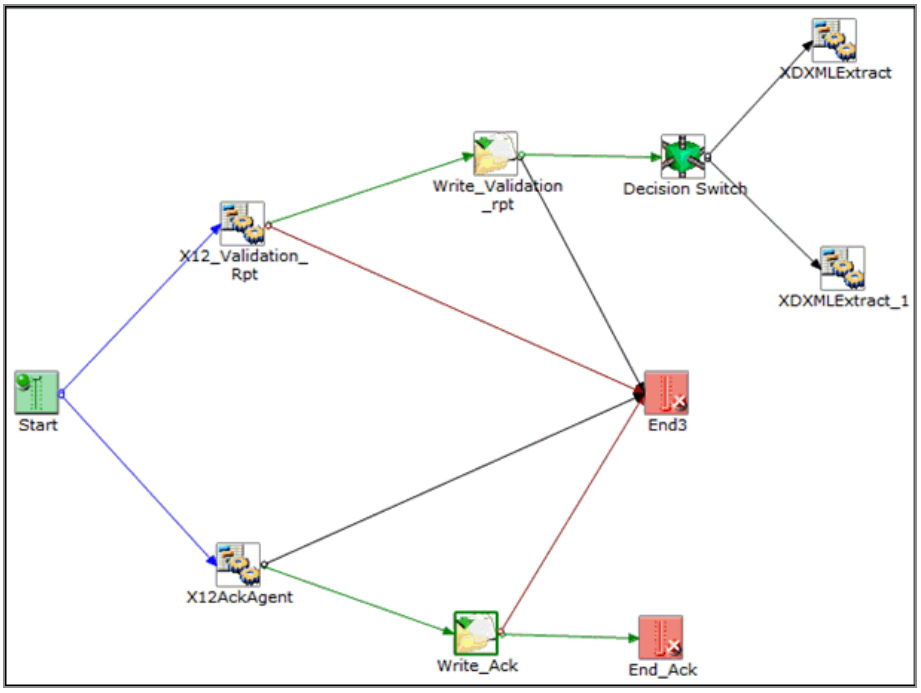
    - OnError

- OnSuccess
- OnDefault
- 0 (Zero)

All cases except case 1 should have been selected.

71. Click **Finish**.

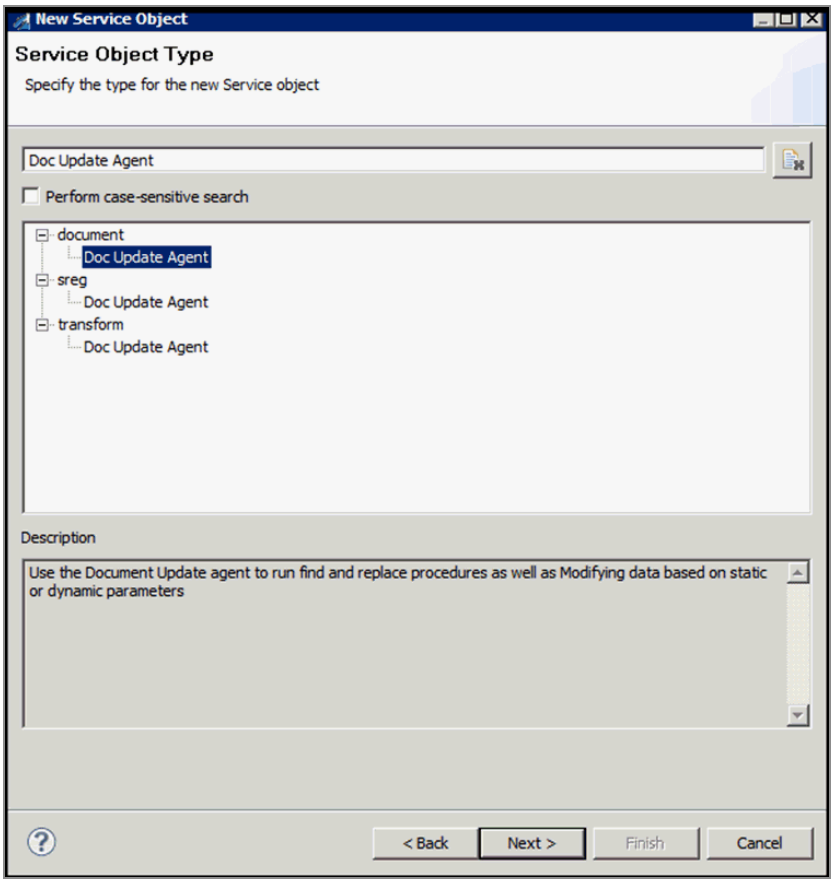
    A new relation line appears between the Decision Switch and XDXMLExtract1 objects to indicate that a relationship has been established, as shown in the following image.



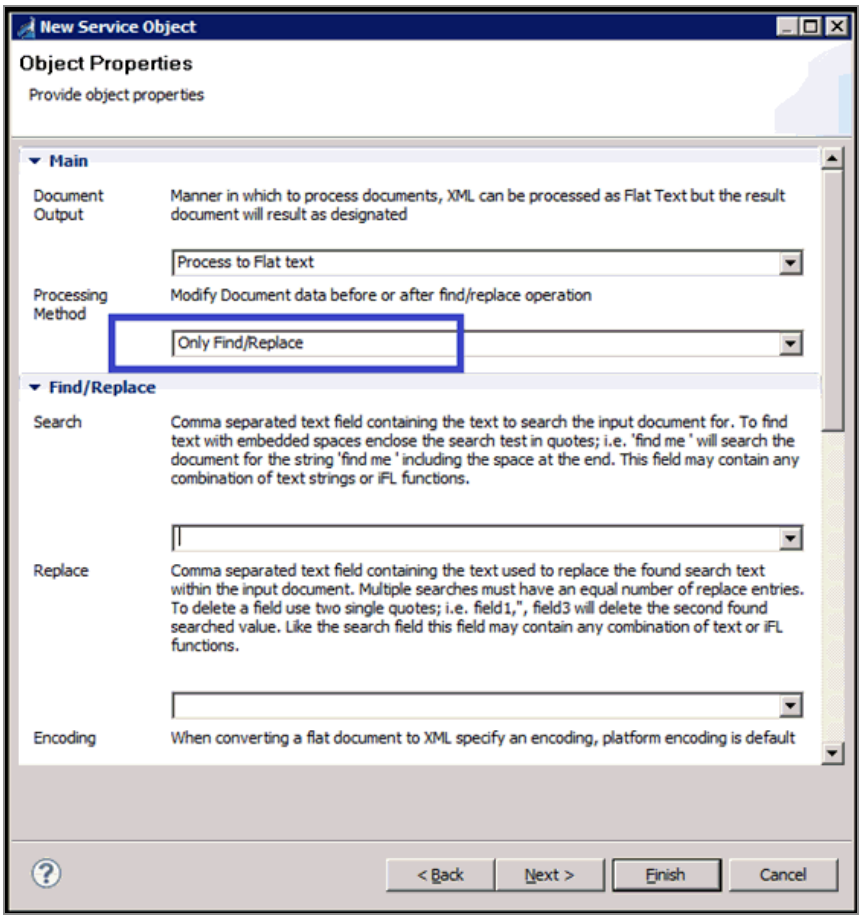
72. Drag and drop the Service object from the Object palette into workspace.
73. In the Name field, type **XDDocUpdate**, and a brief description (optional) in the Description field.
74. Click **Next**.

    The Service Object Type wizard opens.

75. In the dynamic search box, enter the class name (for example, *com.ibi.agents.XDDocUpdateAgent*), select any one of the **Doc Update Agent** options and then click **Next**, as shown in the following image.



76. In the Object properties wizard, select **Only Find/Replace** from the drop-down list for the Processing Method parameter in the Object properties wizard, as shown in the following image.

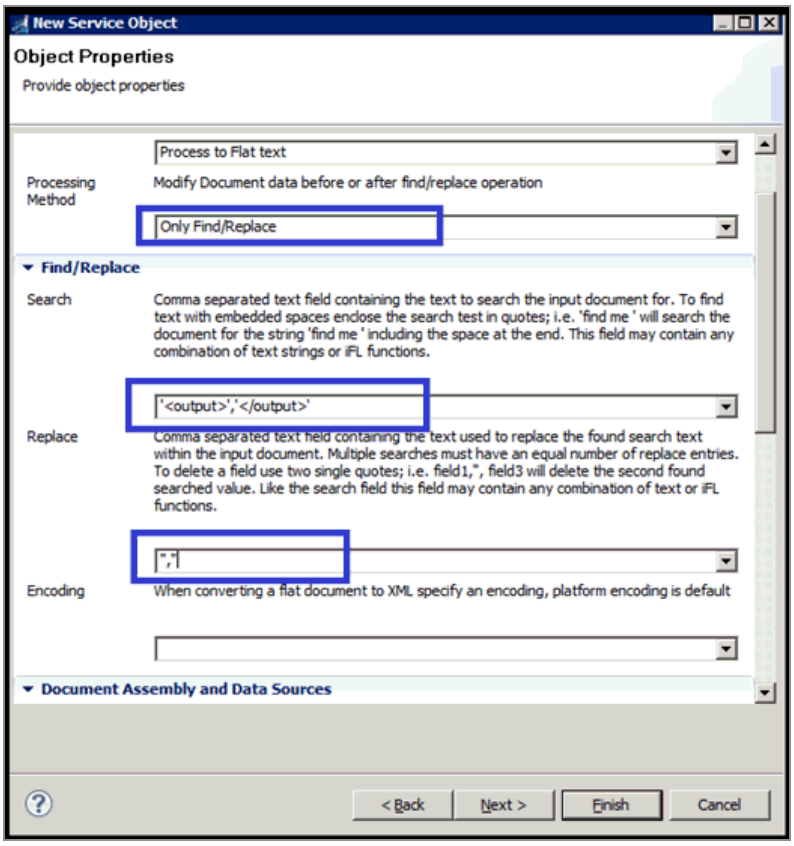


77. For the Search parameter, enter the following string with single quotes:

```
'<output>','</output>'
```

78. For the Replace parameter, enter the following string with double quotes:

```
“ ‘ ”
```



79. Click **Finish** to complete creating the Doc Update Agent.

    The XDDocUpdate object appears in the workspace.

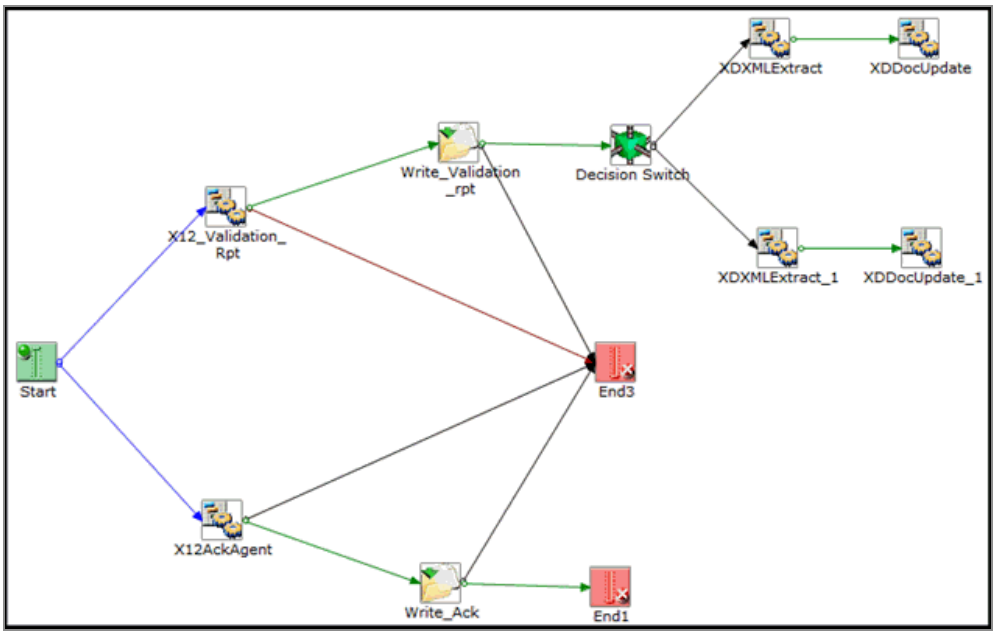
80. Create a copy of the XDDocUpdate agent and name it as *XDDocUpdate1* for the non-success validation report (error route).

81. Select **XDXMLExtract**, right-click the **XDDocUpdate** agent, and select **Create Relation** from the context menu.

82. From the Event drop-down list, select **OnSuccess** and click **Finish**.

83. Select the **XDXMLExtract1** object, right-click on the **XDDocUpdate1** object, and select **Create Relation** from the context menu.

84. From the Event drop-down list, select **OnSuccess** and click **Finish**.



85. Drag and drop the File object from the object palette to the workspace.

    The New File Object wizard opens.

86. In the Name field, type *Good File*, and a brief description (optional) in the Description field, and then click **Next**.

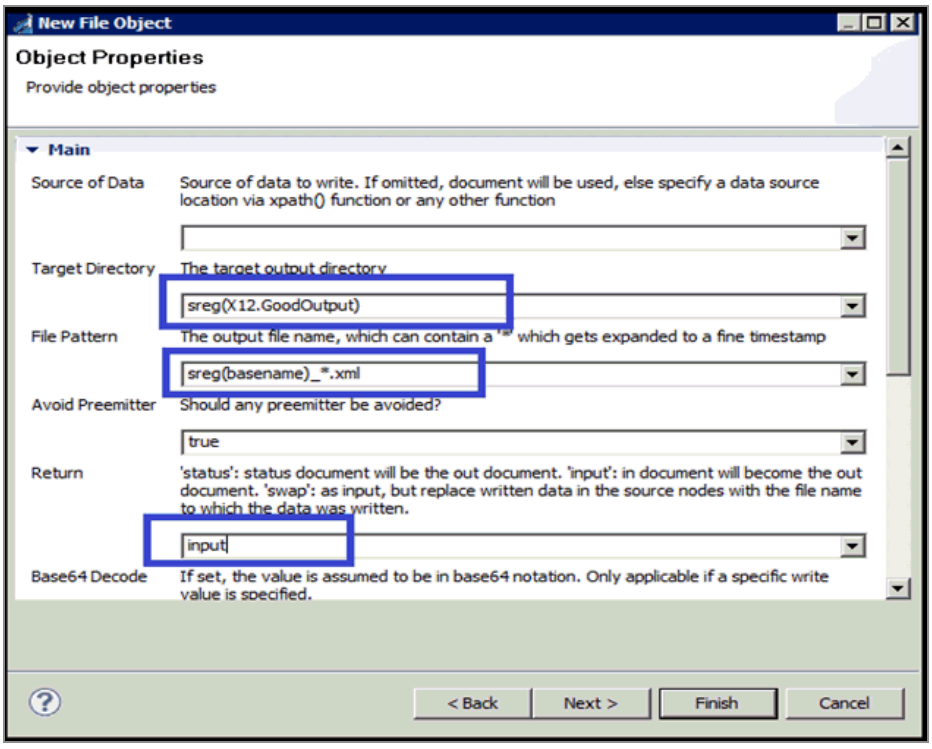
    The File Object Type wizard opens.

87. From the Type drop-down list, select **File Emit Agent {com.ibi.agents.XDFileEmitAgent}** and click **Next**.

    The Object properties wizard opens.

88. For the target directory, enter a valid physical folder location to write Acknowledgments data, for example:

```
sreg(X12.GoodOutput)
```



89. In the File Pattern parameter, enter the following:

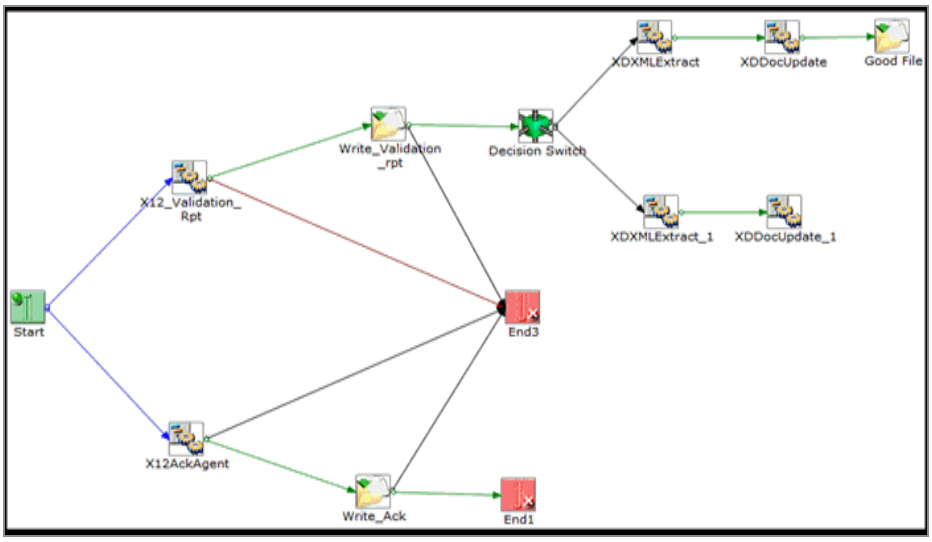
```
sreg(basename)_*.xml
```

90. In the Return parameter, select **input** from the drop-down list, and then click **Finish**.

    The new File object (Good File) appears in the workspace.

91. Select the **XDDocUpdate** object, right-click the **Good File** object, select **Create Relation** from the context menu, and then click **Next**.

92. From the Event drop-down list, select **OnSuccess** and then click **Finish**.



93. Drag and drop the File object from the Object palette to the workspace.

    The New File Object wizard opens.

94.

95. In the Name field, type *Bad File*, then enter a brief description (optional) in the Description field and click **Next**.

    The File Object Type wizard opens.

96. From the Type drop-down list, select **File Emit Agent {com.ibi.agents.XDFileEmitAgent}** and click **Next**.

    The Object Properties wizard opens.

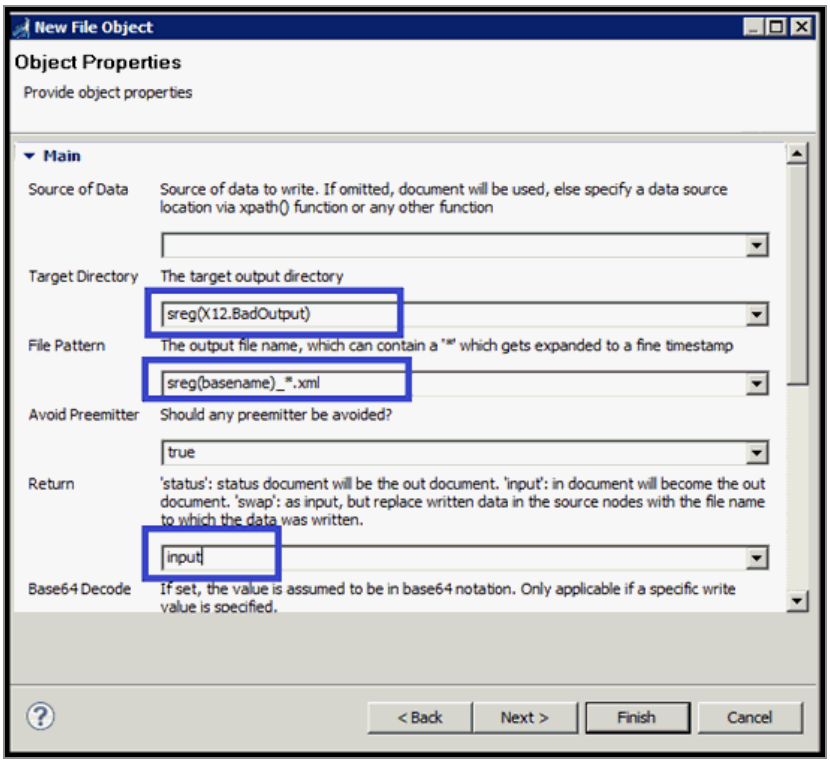
97. In the Target Directory field, enter a valid physical folder location to write Acknowledgments data. For example:

    ```
    sreg(Hipaa.BadOutput)
    ```

98. In the File Pattern paremeter, enter the following:

    ```
    sreg(basename)_*.xml
    ```

99. In the Return parameter, select **input** from the drop-down list and then click **Finish**, as shown in the following image.

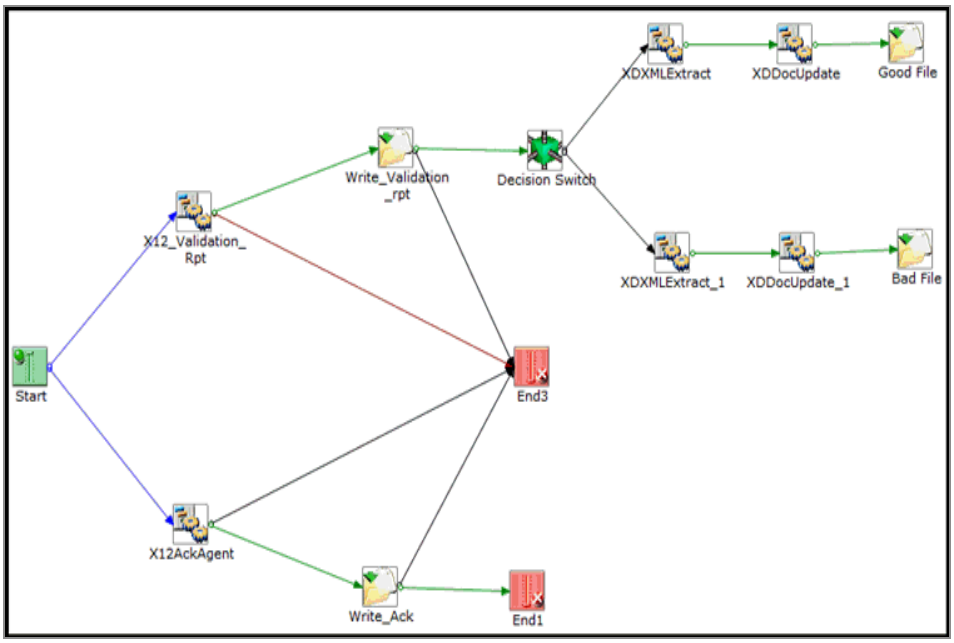


The new File object (Bad File) appears in the workspace.

100. Select the **XDDocUpdate1** object, right-click the **Bad File** object, and select **Create Relation** from the context menu.

101. Click **Next**.

102. From the Event drop-down list, select **OnSuccess** and then click **Finish**.



103. Drag and drop the Junction object into workspace from the Objects palette.
104. In the Name field, enter *Junction* (by default it will show Junction), and then click **Finish**.

     The Junction object appears in the workspace.
105. Create a copy of the Junction object, name it as *Junction1*, and then click **Finish**.

     A copy of the Junction object (junction1) appears in the workspace.
106. Select the **Good File** object, right-click the **Junction** object, and select **Create Relation** from the drop-down list.

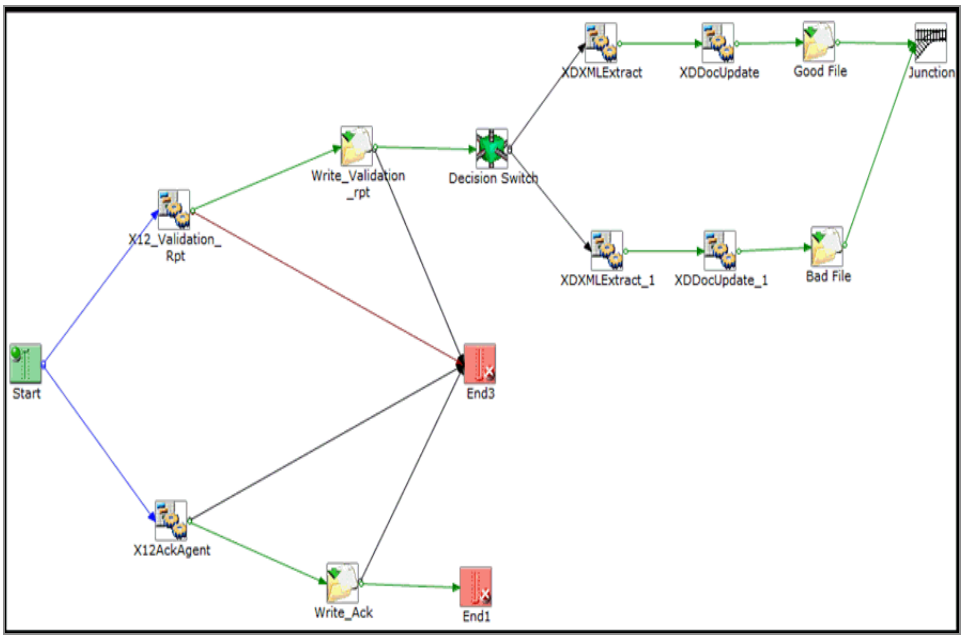
     The Relation Configuration wizard opens.
107. From the Event drop-down list, select **OnSuccess** and click **Finish**.

     A relation (line) appears between Good File and the Junction object.
108. Select the **Bad File** object, right-click the **Junction** object, and select **Create Relation** from the context menu.

     The Relation Configuration wizard opens.
109. From the Event drop-down list, select **OnSuccess** and click **Finish**.

     A line appears between the objects to indicate that a relationship has been

established, as shown in the following image.



110. Select the **Good File** object, right-click on the **Junction1** object, and then select the **Create Relation** option from the context menu.

     The Relation Configuration wizard opens.

111. From the Event drop-down list, select **OnCustom** and then select following cases from the case parameter list:

     - OnError
     - OnFailure
     - Fail_operation
     - Fail_parse
     - notfound

     A relation (line) appears between Good File and the Junction1 objects.

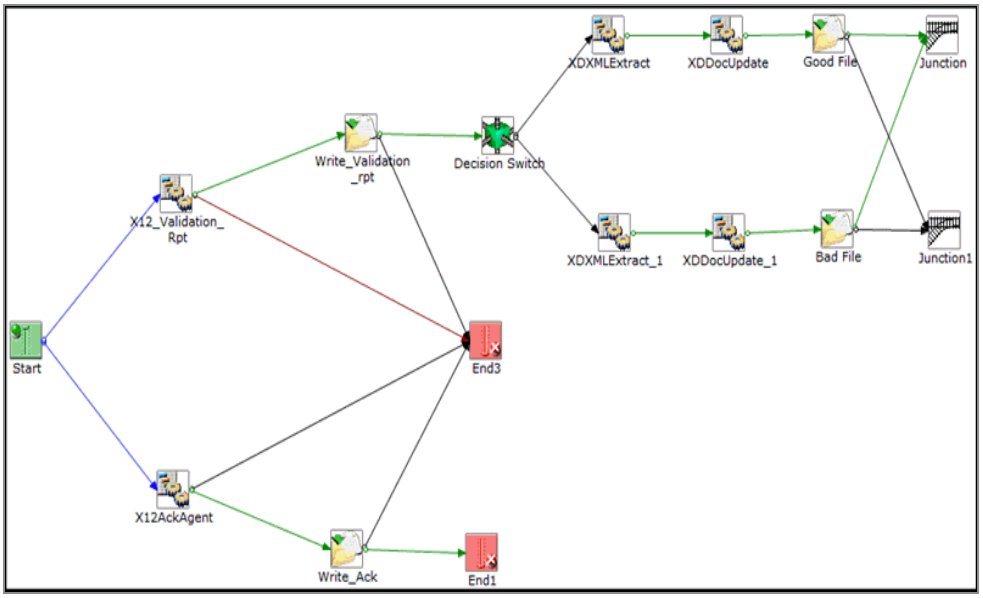
112. Select the **Bad File** object, right-click on the **Junction1** object, and then select the **Create Relation** option from the context menu.

     The Relation Configuration wizard opens.

113. From the Event drop-down list, select **OnCustom** and then select following cases from the case parameter list:

- OnError
- OnFailure
- Fail_operation
- Fail_parse
- notfound

A relation (line) appears between Bad File and the Junction1 objects, as shown in the following image.



114. Drag and drop the End object from the Object palette to the workspace.

     The End Name and Description properties wizard opens.

115. In the Name field, type *End*, then enter a brief description (optional) in the Description field and click **Next**.

     The End Object Schema wizard opens.

116. From the Terminate parameter, select the check box for **Select if this end object is the completion point**.

117. Click **Finish**.

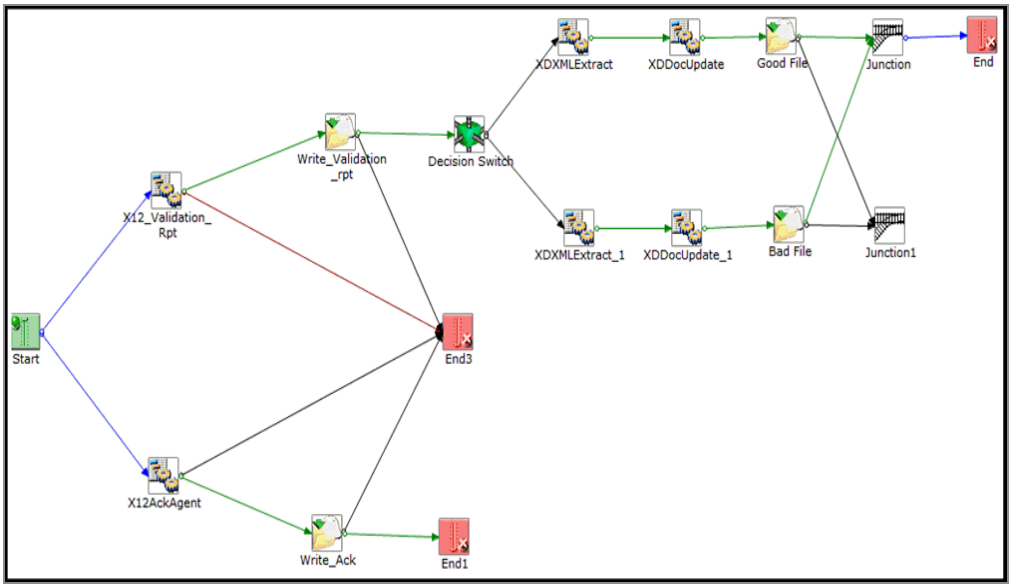
     The new End object appears in the workspace.

118. Select the **Junction** object, right-click on the **End** object, and then select the **Create**

Relation option from the context menu.

119. From the Event drop-down list, select **OnCompletion** and then click **Finish**.

     A new Relation (line) appears between the Junction object and the End object, as shown in the following image.



120. Drag and drop the End object from the Object palette to the workspace.

     The End Name and Description properties wizard opens.

121. In the Name field, type *End2*, then enter a brief description (optional) in the Description field and click **Next**.

     The End Object Schema wizard opens.

122. From the Terminate parameter, select the check box for **Select if this end object is the completion point**.

123. Click **Finish**.

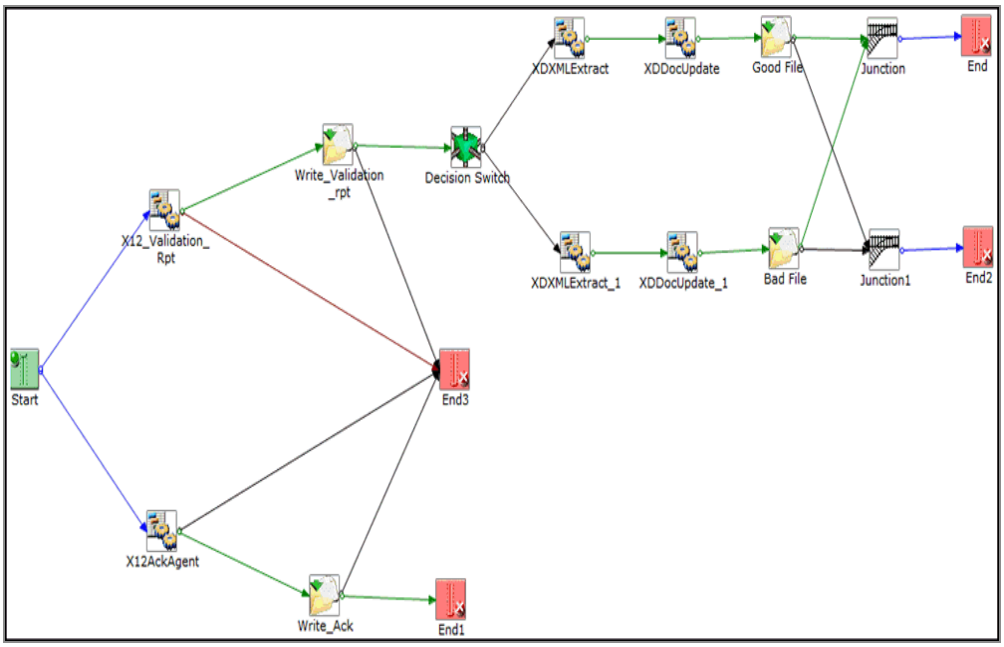
     The new End2 object appears in the workspace.

124. Select the **Junction1** object, right-click on the **End2** object, and then select the **Create Relation** option from the context menu.

125. From the Event drop-down list, select **OnCompletion** and then click **Finish**.

     A new Relation (line) appears between the Junction1 object and the End2 object, as

shown in the following image.



The process flow is now complete.

126. To save the process flow, click the File menu and then select **Save** from the File menu options.

    Now you need to validate the process flow and publish it to the Registry of the iWay Service Manager Administration Console for use in the route of a channel for outbound processing.

    Validating a process flow ensures that its structure is correct. Publishing a process flow makes it available in the Registry for use in a channel configuration. For instructions on validating and publishing the process flow, see the *iWay Integration Tools Designer User's Guide*.

127. Close iWay Integration Tools.

    Your next step is to add a new route to the Registry using the iWay Service Manager Administration Console and associate the process flow with it.

Define a Route and Associate the Process Flow With the Route

To define a route and associate the process flow with It:

Procedure

1. From the Registry menu options in the iWay Service Manager Administration Console, click **Routes**.
2. On the Route Definitions pane, click **Add** to add a route.
3. On the New Route Definition pane, enter a name for the route and an optional description, as shown in the following table.

Parameter	Value
Name *	EDItoXML_Route
Description	This route will invoke the X12 to XML validation process. The outcome of this process will place valid X12 transformed data in your valid inbound folder. Invalid X12 transformed data will be routed to its appropriate folder. A validation report will also be generated and sent to its appropriate folder.

4. Click **Finish**.
5. On the Construct Route pane, click **Add**.

   You are prompted for the type of component to associate with the route.
6. Select **Process** and click **Next**.
7. The next pane prompts you to select a process. Select the process flow you created earlier with iWay Integration Tools, **x12toXML_pflow_AckRpt**, and click **Finish**.

   The route, with its associated process flow, has been successfully defined.

Defining the Outlets

An outlet defines how a message leaves a channel. An emitter is a transport protocol that sends a document to its recipient. In the sample configuration, we will use a File emitter. For details on supported protocols, see the *ibi™ iWay® Service Manager Protocol Guide*.

For the channel in this example, you will add one emitter to the Registry. Then you will define one outlet and associate the emitter with this outlet.

When you associate the outlet with the channel in later steps, you will apply a condition to dynamically direct the flow of the output document based on its content.

In the example, you will add an emitter for the acknowledgment data. In the example, the data for the functional acknowledgment (transaction 997) is in EDI flat file (non-XML) format. When you add the acknowledgment outlet to the channel, you will set the condition _isFLAT(). This condition tests the output data for flat file (non-XML) format. If the data is in flat file (non-XML) format, it is routed to the specified destination.

Add an Emitter for Acknowledgment Output

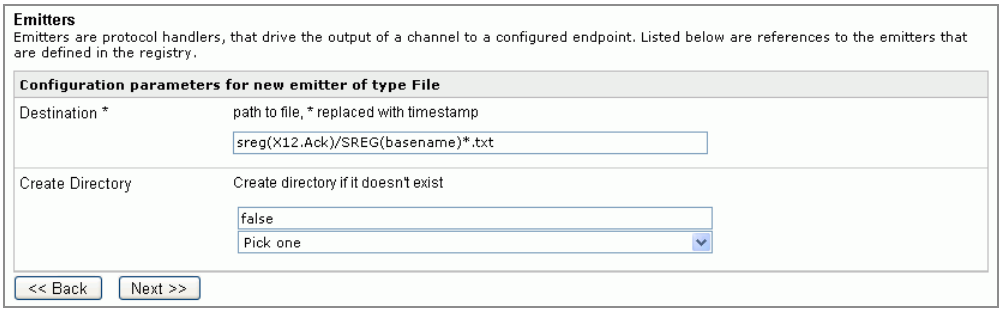
Procedure

1. On the Emitters pane, click **Add** to add another emitter.

   The next pane prompts you for the emitter type.

2. For this example, select **File** from the drop-down list and click **Next**.

   The configuration parameters pane opens.



3. Supply configuration parameters for the second File emitter as follows, then click **Next**.

Parameter	Value
Destination *	sreg(X12.Ack)/SREG(basename)*.txt

Parameter	Value
	This value is the directory where the acknowledgment output is placed. You can use an extension other than .txt, for example, .edi or .data. sreg(X12.Ack) is a special register value that uses a defined directory in which output files are stored after transformation. Make sure that you have created this directory; otherwise, errors will occur during deployment. On output, an asterisk (*) in the destination file name is replaced by a date and time stamp. For details on the special register (SREG) used in the preceding file name, see the *ibi™ iWay® Service Manager User's Guide*.
Create Directory	false

4. On the Emitters pane, enter the name of the new emitter and an optional description, as shown in the following table. Then click **Finish** to add the emitter.

Parameter	Value
Name *	Ack_Out_Emitter
Description	Emitter for acknowledgment output for EDI.

Define an Outlet for Acknowledgment Output

Procedure

1. From the Registry menu options, select **Outlets**.
2. On the Outlet Definitions pane, click **Add** to add an outlet.

3. On the New Outlet Definition pane, enter the name of the new outlet and an optional description, as shown in the following table. Then click **Finish** to add the outlet.

Parameter	Value
Name *	EDI_Ack_Outlet
Description	acknowledgment outlet for EDI.

4. On the Construct Outlet pane, click **Add** to associate the acknowledgment emitter with the acknowledgment outlet.

   The next pane prompts you for the component type.

5. Select **Emitter** and click **Next**.

   The next pane prompts you to select an emitter.

6. Select **Ack_Out_Emitter**, which is the acknowledgment emitter you added earlier, and click **Finish**.

   Now you have defined the two outlets.

Defining a Channel

Now that you have defined the inlet, route, and outlets for the channel, you are ready to add the channel to the Registry and associate the conduits with it.

Define a Channel

Procedure

1. From the Registry menu options, select **Channels** under Conduits.
2. On the Channel Definitions pane, click **Add** to add a channel.
3. On the New Channel Definition pane, enter the name of the new channel and an optional description, as shown in the following table. Then click **Finish** to add the channel.

Parameter	Value
Name *	EDItoXML_Channel
Description	Channel for EDI to XML inbound processing.

4. On the Construct Channel pane, click **Add** to associate the inlet, route, and outlets defined previously with the channel.

   You are prompted to associate components with the channel.

5. Select **Inlet** and click **Next**.

   The next pane prompts you to select an inlet.

6. Select **EDItoXML_Inlet**, which is the inlet you defined earlier, and click **Finish**.

   The inlet is added to the channel. Now you need to associate the route defined earlier with the channel.

7. On the Construct Channel pane, click **Add**.

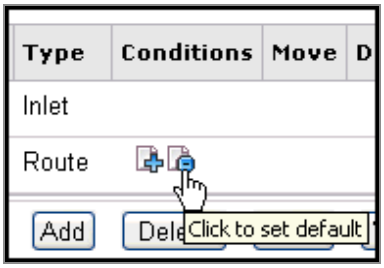
   The next pane prompts you for the component type.

8. Select **Route** and click **Next**.

   On the next pane, you are prompted to select a route.

9. Select **EDItoXML_Route**, which is the route created earlier, and click **Finish**.

10. On the Construct Channel pane, click the minus sign (-) under Conditions next to the name of the route to set it as the default.



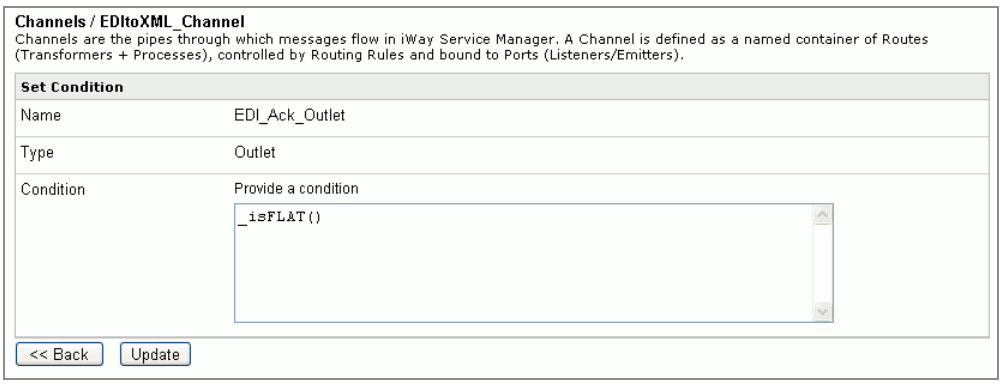
11. On the Construct Channel pane, click **Add** to add the outlets.

12. On the next pane, select **Outlet** and click **Next**.

13. Select the outlet defined earlier, **EDI_Ack_Outlet** and click **Finish**.

14. To set a condition for the EDI_Ack_Outlet, on the Construct Channel pane, click the plus sign (+) under Conditions for the EDI_Ack_Outlet.

    The Set Condition pane opens.



15. In the Condition input field, enter **_isFLAT()**, and click **Update**.

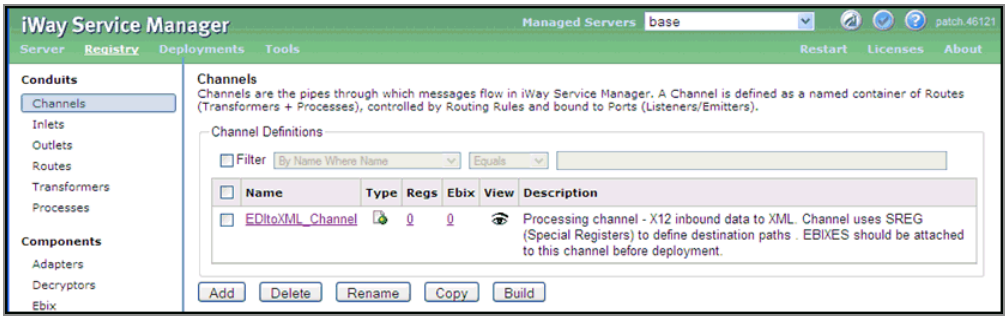
    This condition tests the output data for EDI flat file (non-XML) format. If the data is in EDI flat file (non-XML) format, it is routed to the destination specified when you added the emitter for acknowledgment output.

Add a Special Register Set to the Channel

Procedure

1. From the Registry menu options, select **Channels**.

   The Channel Definitions pane opens.



2. Click the link in the **Regs** column for EDItoXML_Channel.
3. On the next pane, which prompts you to add special register (SREG) sets, click **Add** to add the SREG set to the channel.

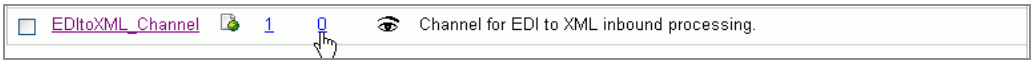
4. On the next pane, select **X12**, which is the name of the SREG set you created previously, and click **Finish**.

Add the Ebix to the Channel

Procedure

1. From the Registry menu options, select **Channels**.

   The Channel Definitions pane opens.



2. Click the link in the Ebix column for the EDItoXML_Channel.
3. On the next pane, which prompts you to add Ebix components, click **Add** to add the Ebix to the channel.
4. On the next pane, select **EDI_4050**, which is the name of the Ebix you added previously, and click **Finish**.

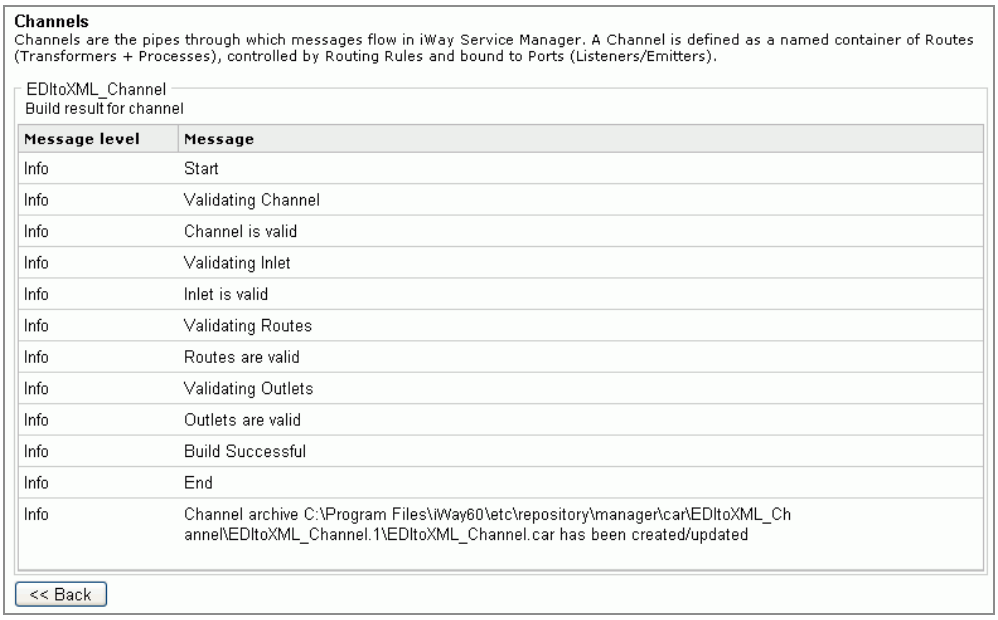
   Now that you have associated all the components with the channel, you are ready to build it.

Build the Channel

Procedure

1. From the Registry menu options on the left pane, select **Channels** under Conduits.
2. On the Channel Definitions pane, select the channel defined previously, **EDItoXML_Channel**, and click **Build**.

   The results of the build are displayed on the right pane.



3. Review the results of your build and then click **Back**.

   If an error or errors are displayed in the Message column, take the appropriate action as instructed.

Deploy the Channel

Deployment is the mechanism by which a channel moves from being stored in the Registry to becoming active in iWay Service Manager. For more information on deployment, see the *ibi™ iWay® Service Manager User's Guide*.

Procedure

1. Select the **Deployments** option in the green shaded area below the iWay Service Manager banner.
2. On the Channel Management pane, click **Deploy**.
3. On the Available Channels pane, select the channel you defined previously, **EDItoXML_Channel**, and click **Deploy**.

   The Channel Management pane reopens.
4. Select **EDItoXML_Channel** and click **Start**.

   The red X under Status changes to a green check mark to indicate that the channel

has been started. If an error or errors are displayed, take the appropriate action as instructed.

Verify the Channel

To ensure that the channel is working as expected, perform the following steps.

For more information on obtaining EDI X12 sample files for testing purposes, see Downloading and Extracting EDI X12 User Samples.

Procedure

1. Place an EDI document as test data in the C:\File_in directory. This is the path in which EDI messages are received, which you specified for the listener associated with the inlet for the channel.
2. Check for the XML file and the functional acknowledgment in the C:\File_out\EDI directory. This is the destination path you specified for the emitters associated with the outlets for the channel. The listener will detect the presence of the file in the input directory, and the copy service will copy it to the output directory, replacing the asterisk in the file name with a time stamp.

   For example, if you specified the destination file name for the XML emitter as _SREG (basename)_*.xml per the configuration example, an EDI input file named X12856C001_4050.x12 is named _X12856C001_4050_2008-03-03T19_33_26.684Z.xml on output.

Reusing Your Channel Configuration

Using the Archive Manager feature of iWay Service Manager, you can archive your channel configuration with its associated components and import them into another Registry. They will then be available from that Registry for modification or reuse.

For details on this feature, see the *ibi™ iWay® Service Manager User's Guide*.

Outbound Processing: XML to EDI X12

The iWay Integration Solution for EDI X12 includes iWay Service Manager. iWay Service Manager validates an XML document based on EDI X12 published implementation guides and converts it to a document in Electronic Data Interchange (EDI) X12 format.

This chapter provides the information you need to understand and implement a basic outbound message flow.

- The **outbound processing overview** describes the iWay business components and the processing steps in the basic outbound message flow.
- The **sample configuration** contains detailed instructions for configuring the basic outbound message flow. This topic guides you through each step of the configuration procedure.

EDI X12 Outbound Processing Overview

The standard outbound process converts an XML document to an EDI-formatted document.

The input document that is sent to the channel may not be in XML format. It can be any input document that first will be processed by the channel and transformed to an EDI document.

In a basic message flow, outbound processing consists of the following components and steps. For an illustration of the components available in the construction of a message flow, see Using a Channel to Construct a Message Flow. You will define the components in the configuration instructions in Sample Configuration for Outbound Processing: XML to EDI.

Inlet

- The **listener** picks up the input document.

Route/Process Flow

- A process flow guides the XML-formatted EDI document through the next stages of the process.

Rules processing runs against the XML-formatted EDI document to validate its structure and content. The published EDI standards and user implementation guides define element types (for example, numeric, alpha, or date) and describe business rules to apply for validation.

The *XMLToX12TransformationAgent* obtains the message type and version from the XML-formatted EDI document. The appropriate transformation template is applied from the Ebix. The transformation converts the XML-formatted EDI document to EDI X12 format.

The *XDX12ValidationReportAgent* creates a report (an XML document) containing the XML-formatted EDI document and resulting EDI X12 formatted data, as well as the validation status.

If the EDI X12 document did not contain any errors during the rules processing stage, it is emitted and continues to its next destination. The validation report is always emitted. In the sample process flow that is described later in this chapter, good validation reports are written with a file name prefix of *validation*. All other validation reports are written with a file name prefix of *error*. Information in the *error* validation reports can be routed accordingly for repair and reprocessing.

Outlet

- The EDI document is passed to the next step in the integration process.

Sample Configuration for Outbound Processing: XML to EDI

This topic provides step-by-step instructions for configuring a basic outbound message flow for the iWay Integration Solution for EDI X12. The message flow represents the movement and tasks in the conversion of a message from XML to EDI.

If you plan to modify the message flow presented here and would like more information on the supported iWay business components that you can use in channel construction, see the *ibi™ iWay® Service Manager User's Guide*.

Accessing the iWay Service Manager Administration Console

For instructions, see Sample Configuration for Inbound Processing: EDI to XML.

Adding an Ebix to the Registry

The iWay e-Business Information Exchange (Ebix) framework supplies several Ebix files for the iWay Integration Solution for EDI X12.

An Ebix file for EDI-X12 is named X12_*transaction_set*.ebx, where *transaction_set* is the transaction set number. For example, the Ebix file for EDI X-12 transaction set 4050 is named X12_4050.ebx.

For details on the supported EDI X-12 transaction sets, see Ebix-Supported Transaction Sets.

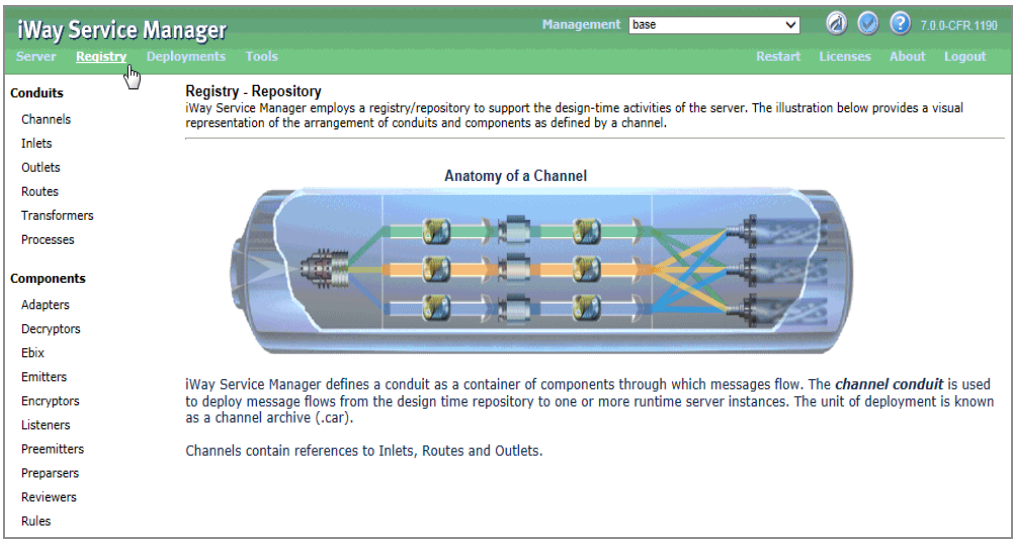
This topic describes how to add an Ebix to the Registry on Windows and UNIX.

Tip: If you already added an Ebix to the Registry as described in Sample Configuration for Inbound Processing: EDI to XML, you do not need to add it again for outbound processing. You can go directly to Sample Configuration for Outbound Processing: XML to EDI.

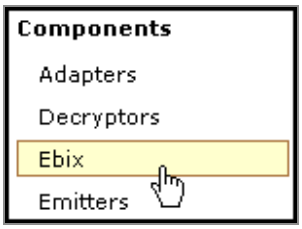
Add an Ebix to the Registry on Windows

Procedure

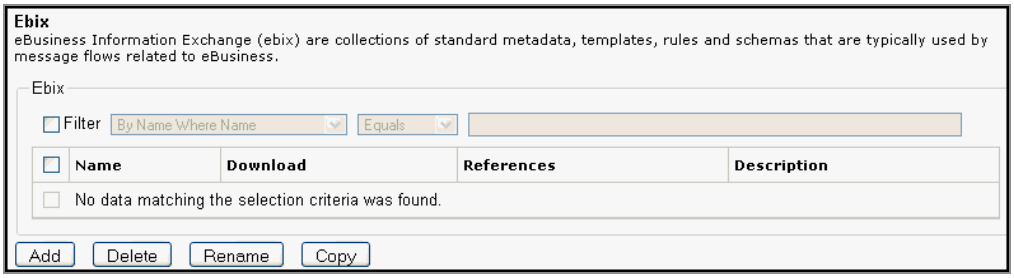
1. To access the Registry, select the **Registry** option in the green shaded area below the iWay Service Manager banner, as shown in the following image.



2. Under Components in the left pane of the Registry, select **Ebix**.

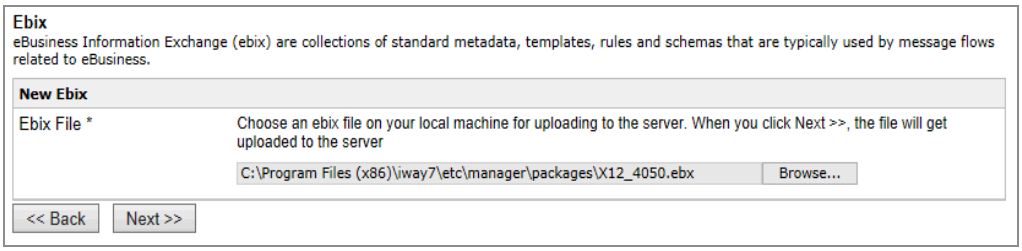


The Ebix pane opens, as shown in the following image.



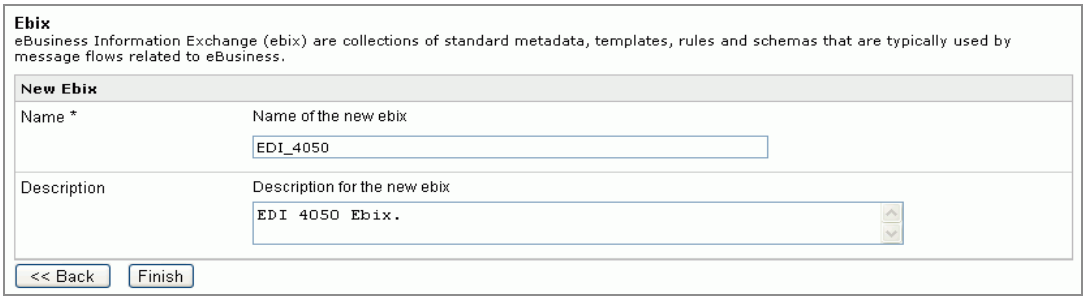
3. Click **Add** to add a new Ebix.

The New Ebix pane opens.



4. Browse to the directory in which the Ebix is located and select the name of the file, for example, **X12_4050_pipeline.ebx**.
5. Once you have selected the Ebix, click **Next**.

   You are prompted for the name of the Ebix and an optional description.



6. Enter a name for the Ebix, for example, **EDI_4050**, and an optional description, such as **EDI 4050 Ebix**.

   Note: This step must be repeated for each Ebix X12 message set that is added to the Registry.
7. Click **Finish**.

   On the Ebix pane, you will see that the Ebix was successfully added. Later you will associate it with the channel for inbound processing.

Add an Ebix to the Registry on UNIX

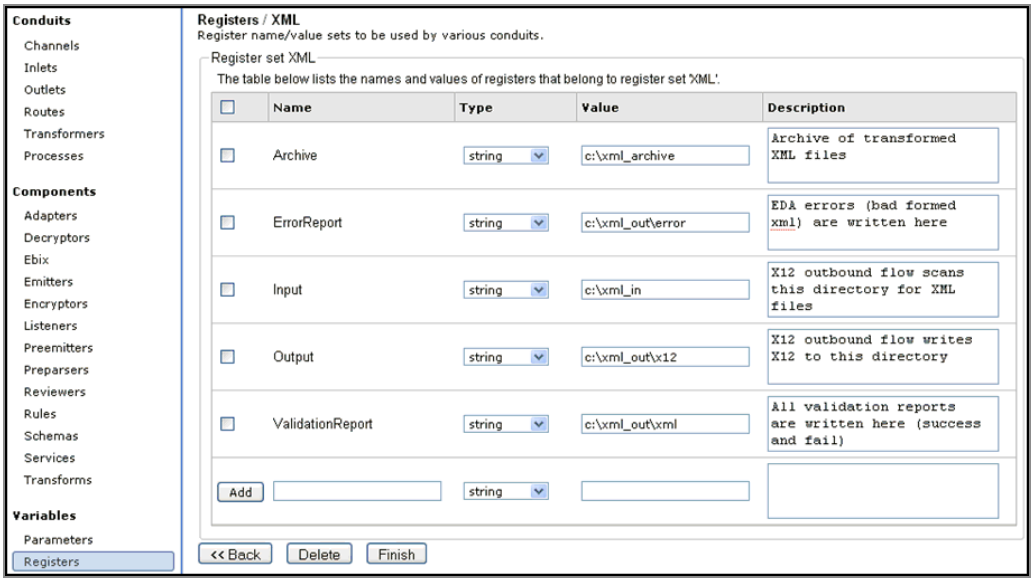
Depending on your system configuration, there are two methods that you can use to add an Ebix to the Registry on UNIX.

- If you have a web browser on the UNIX machine, follow the instructions for Windows.
- Use FTP to download the Ebix from the *iway7/etc/manager/packages* directory to your Windows machine and follow the instructions for Windows.

Adding Special Register Sets

In iWay Service Manager, a special register is a name-value pair that defines a variable that is carried throughout the system. Once defined, this variable is available to all components of the system. Within the EDI components, a best practice is to use special registers to define inputs and outputs. When packages containing channels are migrated between systems, the only changes required to deploy in the new location is to modify these special registers and build the channel. Channels may have many locations and this practice will minimize the effort required to migrate. For a complete list of system special registers that are provided, see the *ibi™ iWay® Service Manager Programmer's Guide*. For more information on defining a special register of your own, see the *ibi™ iWay® Service Manager User's Guide*.

The sample outbound channel uses a set of special registers defined as XML. For example:



Add a Special Register Set to Your Channel

To add a special register set to your channel:

Procedure

1. In the left console pane of the Registry menu, select **Channels**.

The Channels pane opens.

2. In the row for your channel, click **Regs** for the channel you want to modify.

   The Assign register pane opens.

3. Select a register and click **Finish**.
4. Click **Back** to return to the Channels pane.

Defining an Inlet

You will add a listener to the Registry, then associate that listener with a new inlet.

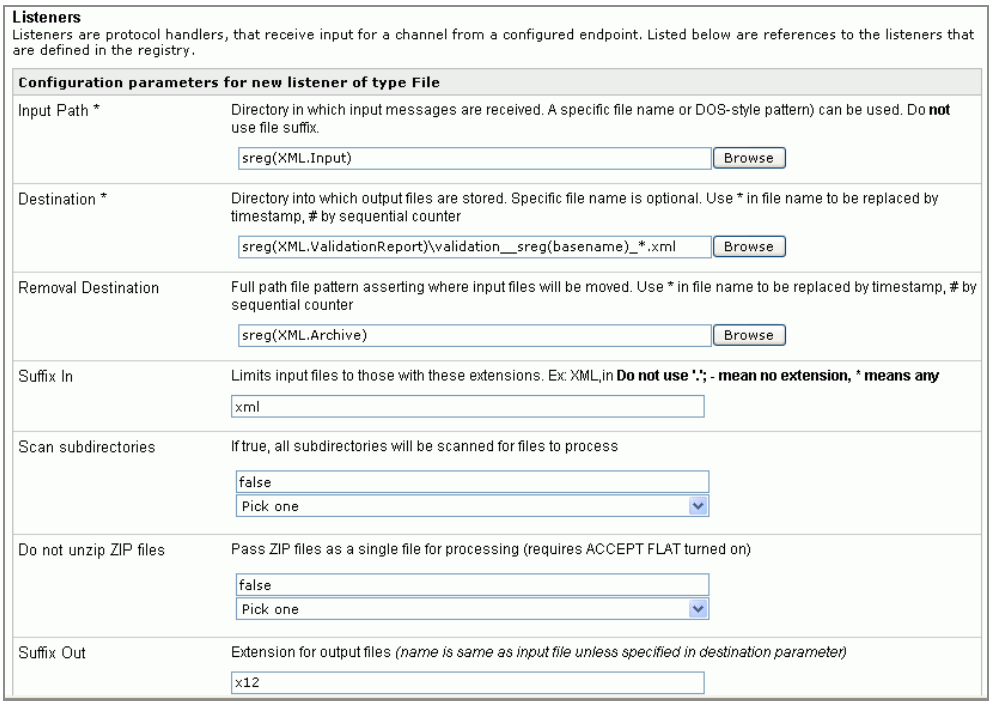
Add a Listener

Procedure

1. From the Registry menu options, select **Listeners**.
2. On the Listeners pane, click **Add** to add a new listener.
3. For the purpose of this example, we will show the configuration with a File listener. For details on supported protocols, see the *ibi™ iWay® Service Manager Protocol Guide*.

   Select **File** from the Type drop-down list and click **Next**.

   The configuration parameters pane opens.



4. Supply configuration parameters for the new File listener as follows. An asterisk indicates that a parameter is required. For parameters not listed in the following table, accept the default value.

Parameter	Value
Input Path *	sreg(XML.Input) This value is a special register that uses a defined directory in which input messages are received. Make sure that you have created this directory, otherwise, errors will occur during deployment.
Destination *	sreg(XML.ValidationReport)\validation__sreg(basename)_*.xml This value is a special register that uses a defined directory in which output messages are received. **Note:** The double underscore characters are used in the

Parameter	Value
	destination to escape the underscore. Make sure that you have created this directory, otherwise, errors will occur during deployment.
Removal Destination	sreg(XML.Archive) This value is a special register that uses a defined directory to which output messages are moved if they fail during transformation. Make sure that you have created this directory, otherwise, errors will occur during deployment. It is recommended to configure a removal destination when you are constructing a basic channel.
Suffix In	xml Input files with the extension .xml are allowed.
Suffix Out	x12 In this example, the extension for output files is .x12.

5. Click **Next**.
6. On the Listeners pane, enter the name of the new listener and a brief description, as shown in the following table.

Parameter	Value
Name *	XmlToX12_Ebix
Description	XML to X12 file listener

7. Click **Finish** to add the listener.

Define an Inlet

Procedure

1. From the Registry menu options, select **Inlets**.
2. On the Inlet Definitions pane, click **Add** to add an inlet.
3. On the New Inlet Definition pane, enter the name of the new inlet and an optional description, as shown in the following table.

Parameter	Value
Name *	XmlToX12_Ebix
Description	The file inlet contains a file listener for XML to X12 processing.

4. Click **Finish** to add the inlet.
5. On the Construct Inlet pane, click **Add** to associate the listener with the inlet.

   The next pane prompts you for the component type.
6. Select **Listener** and click **Next**.

   The next pane prompts you to select a listener.
7. Select **XmlToX12_Ebix**, which is the listener you added earlier for outbound processing, and click **Finish**.

   The listener is added to the inlet.

Defining a Route

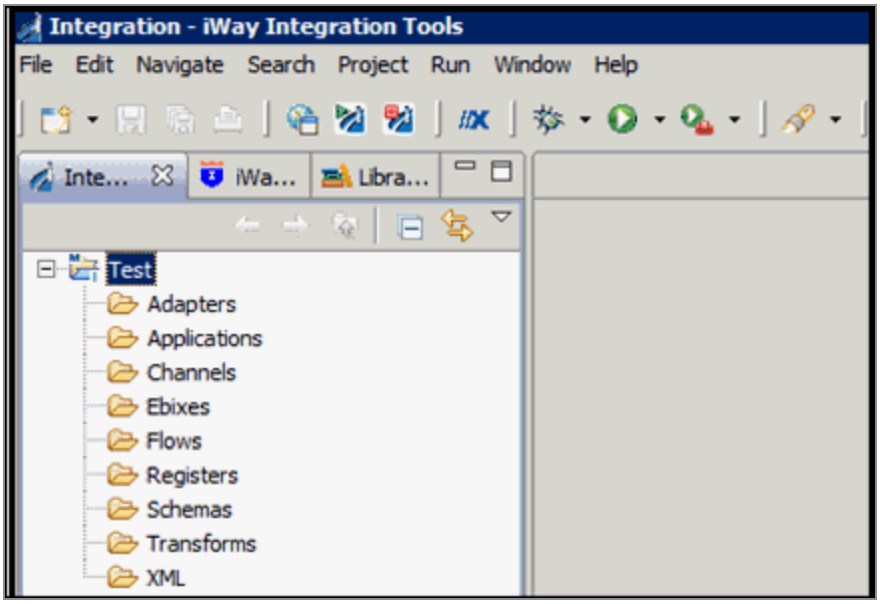
For this sample channel configuration, you will define a route that will invoke the XML to X12 validation process flow. The outcome of the validation process flow will place valid X12 data in a defined output folder. Invalid X12 data will be routed to an errors folder. A validation report will also be sent to the appropriate folder. This section describes how to create a validation process flow using iWay Integration Tools and bind it to a sample outbound channel as a route.

Create a New Project and Start the Process Flow

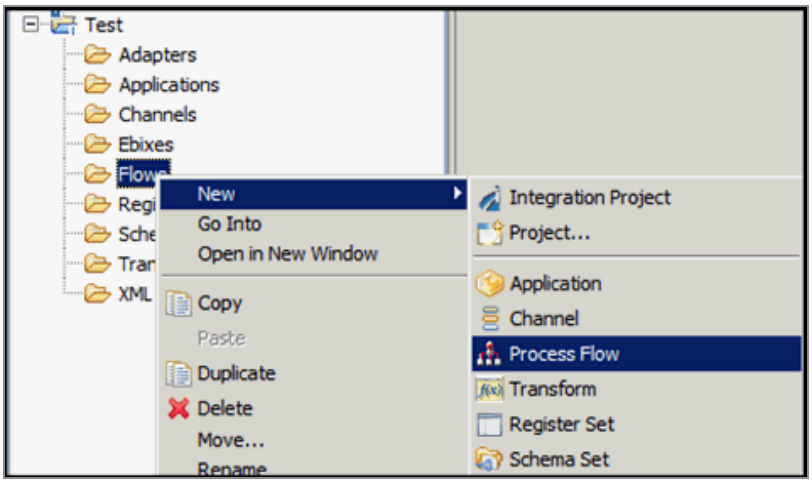
To create a new project and start the process flow using iWay Integration Tools:

Procedure

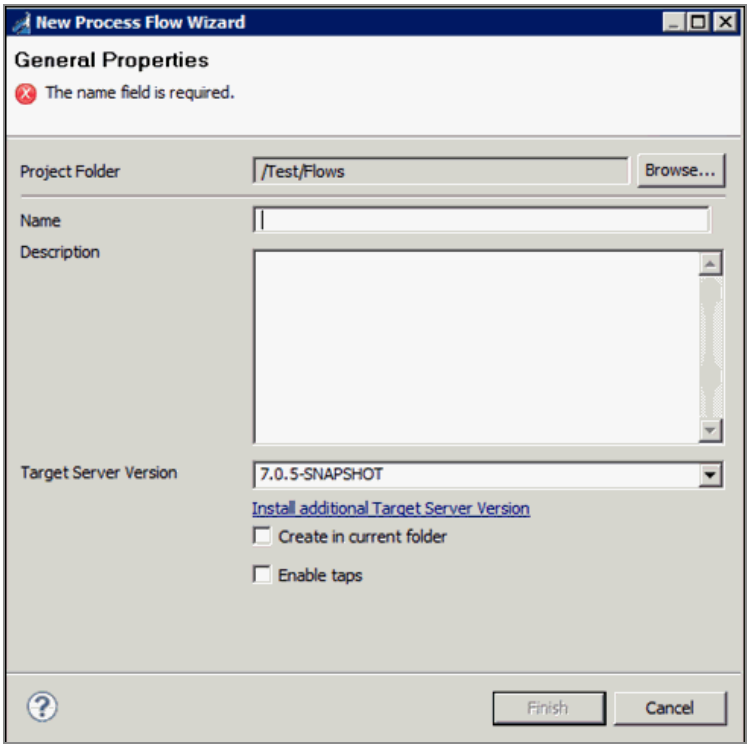
1. Open iWay Integration Tools.
2. Create a new project by right-clicking on the Integration Explorer window, selecting **New**, and then clicking **Integration Project**.
3. In the Name field, type a project name, for example, **Test**.
4. Click **Finish**.



5. Right-click the **Flows** folder, select **New**, and then click **Process Flow** from the context menu, as shown in the following image.



The New Process Flow Wizard opens, as shown in the following image.



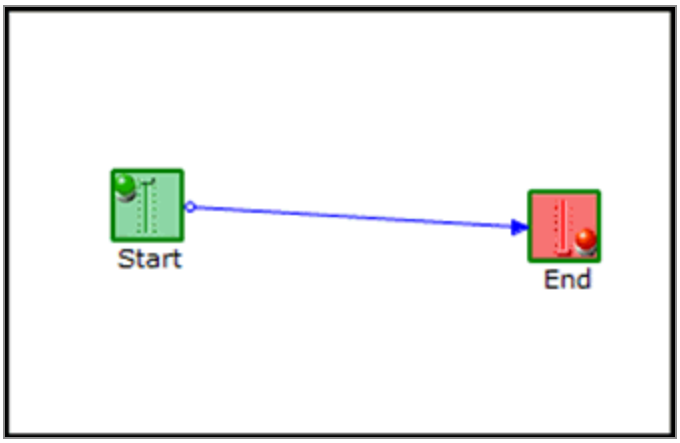
6. In the Name field, type **XMLToX12_pFlow** as the process flow name.

   In the Description field, type a brief description (optional).

7. Click **Finish**.

   The new XMLToX12_pFlow node appears under the Processes folder, and the

workspace displays a Start object, as shown in the following image.



You are ready to build the XMLToX12_pFlow validation process flow by configuring objects to it and specifying their relationships.

Configure Objects for the Process Flow

To configure objects for the process flow using iWay Integration Tools:

Procedure

1. Drag and drop the Service object from the Objects palette to the workspace.

   The New Service Object dialog box opens.

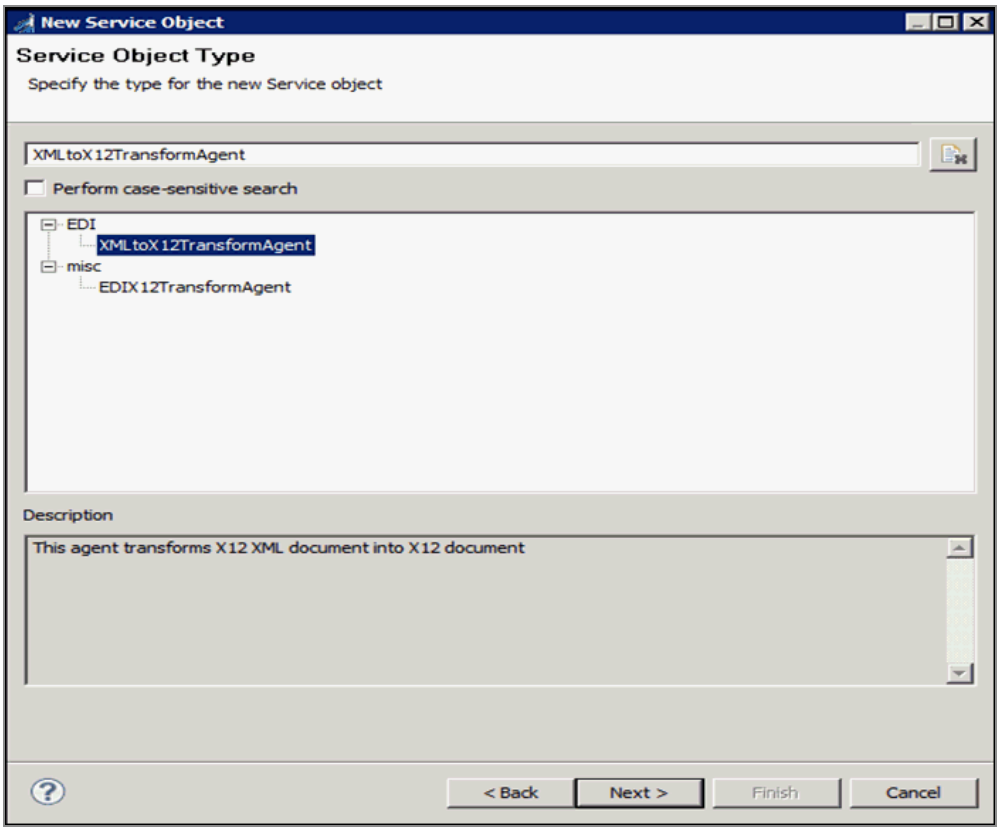
2. In the Name field, type **XMLtoX12TransformAgent**, and a brief description (optional) in the Description field.

3. Click **Next**.

   The Service Object Type dialog box opens.

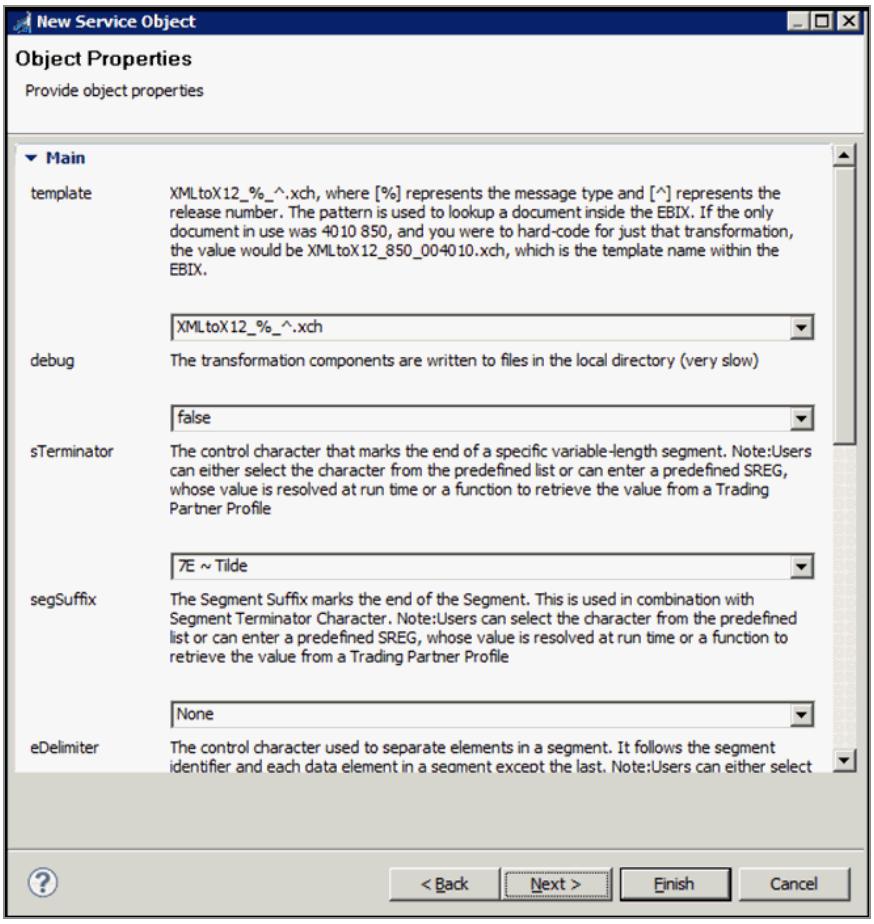
4. Select **Class Name** and enter **com.ibi.agents.XMLToX12TransformAgent**.

5. In the Search result, expand **EDI**, select **XMLtoX12TransformAgent**, and click **Next**, as shown in the following image.



6. Click **Next**.

   The Object Properties dialog box opens, as shown in the following image.



7. Set the InsertGroupLoop property to **false** (the default setting is already set to **false**).
8. For the debug parameter, select **false** from the drop-down list (the default setting is already set to **false**).
9. Click **Finish**.

   The new Service object (XMLtoX12TransformAgent) appears in the workspace.
10. Select the **Start** object, right-click the **XMLtoX12TransformAgent** object, and select **Create Relation** from the context menu.

    The Line Configuration dialog box opens.
11. From the Event drop-down list, select **OnCompletion** and click **OK**.

    This option indicates that there are no conditions that affect the path, and that the path between the two objects will always be followed.

    A line appears between the objects to indicate that a relationship has been

established, as shown in the following image.

12. Drag and drop the File object from the Object palette to the workspace.

    The New File Object dialog box opens.

13. In the Name field, type **X12_Error**, and a brief description (optional) in the Description field.

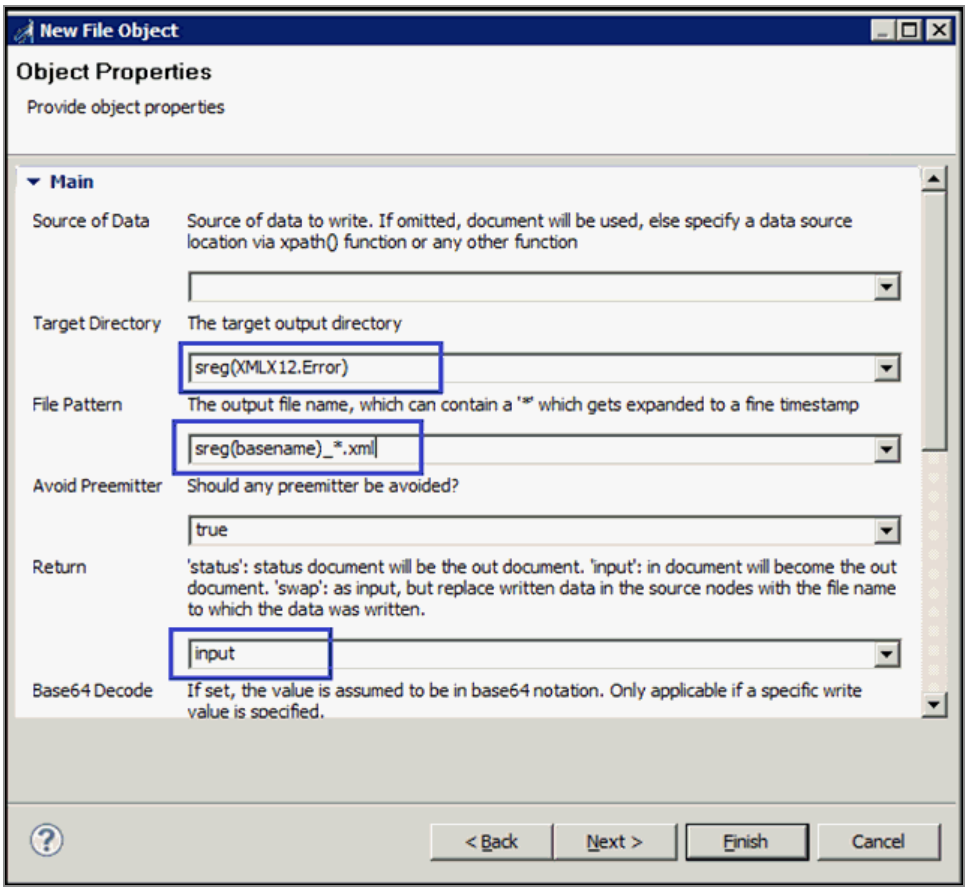
14. Click **Next**.

    The File Type dialog box opens.

15. From the Type drop-down list, select **File Emit Agent**.

16. Click **Next**.

    The Object Properties wizard opens, as shown in the following image.

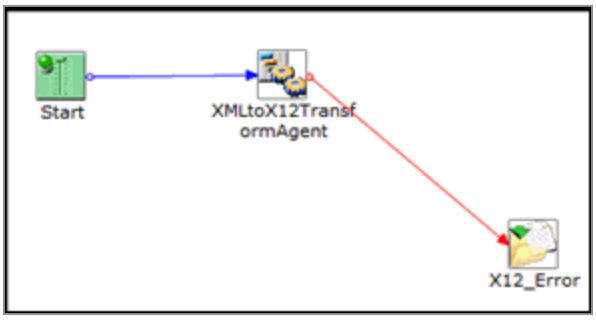


17. For the Target Directory parameter, enter a location where error data will be written, for example, **sreg(XML12.Error)**.
18. For the File Pattern parameter, enter **error__sreg(basename)_*.xml**.
19. For the Return parameter, select **input** from the drop-down list.
20. Click **Finish**.

    The new File object (X12_Error) appears in the workspace.
21. Select the **XMLtoX12TransformAgent** object, right-click the **X12_Error** file object, and select **Create Relation** from the context menu.

    The Line Configuration dialog box opens.
22. From the Event drop-down list, select **OnFailure** and click **OK**.

    A line appears between the objects to indicate that a relationship has been established, as shown in the following image.



23. Drag and drop the End object from the object palette to the workspace.

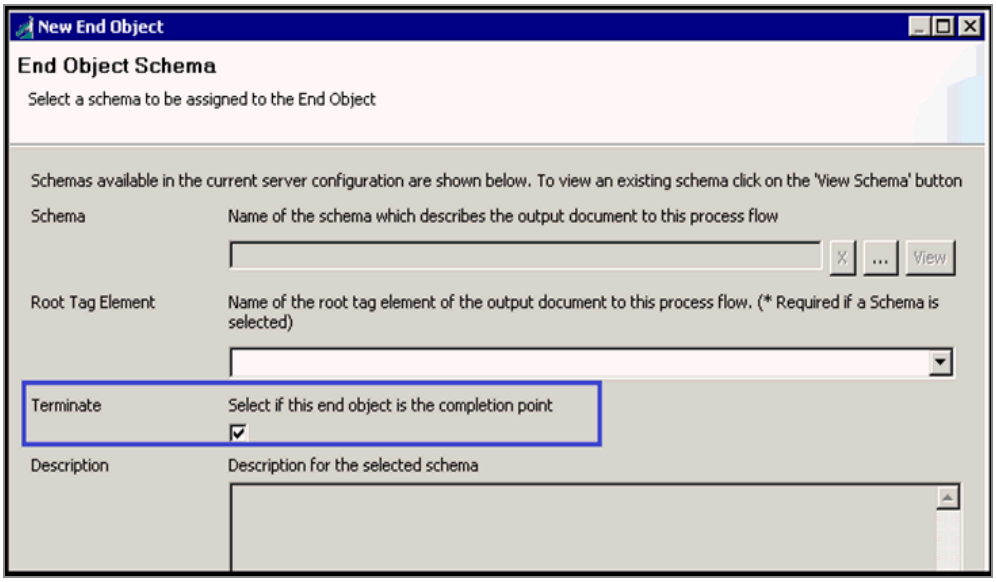
    The End Name and Description dialog box opens.

24. In the Name field, type **X12_End**, and a brief description (optional) in the Description field.

25. Click **Next**.

    The End Name Schema dialog box opens.

26. In the Terminate parameter, select the check box for **Select if this end object is the completion point**, as shown in the following image.



27. Click **Finish**.

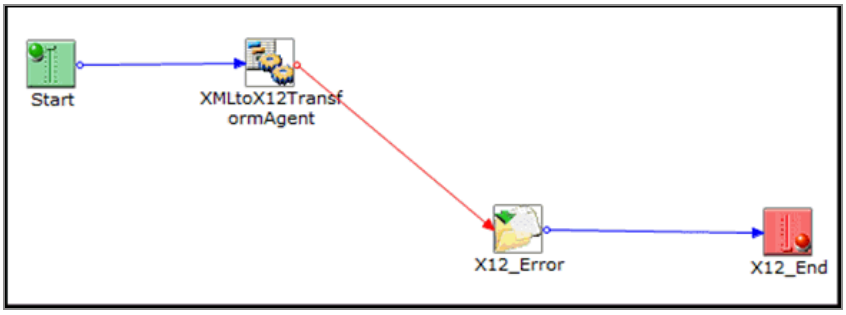
    The new X12_End object appears in the workspace.

28. Select the **X12_Error** file object, right-click the **X12_End** object, and select **Create Relation** from the drop-down list.

The Line Configuration dialog box opens.

29. From the Event drop-down list, select **OnCompletion** and click **Finish**.

    A line appears between the objects to indicate that a relationship has been established, as shown in the following image.



30. Drag and drop the Service object from the Object palette to the workspace.

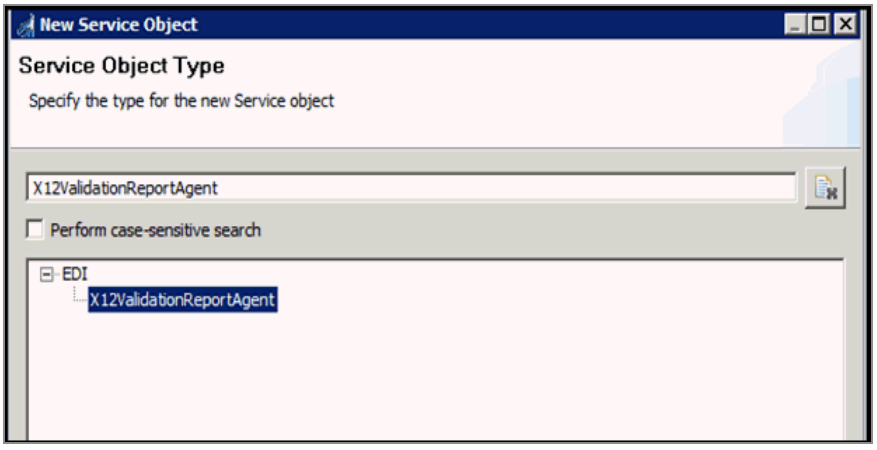
    The New Service Object dialog box opens.

31. In the Name field, type **XDX12ValidationReportAgent**, and a brief description (optional) in the Description field.

32. Click **Next**.

    The Service Type dialog box opens.

33. Select **Class Name** and enter **com.ibi.agents.XDX12ValidationReportAgent**, as shown in the following image.



34. Click **Next**.

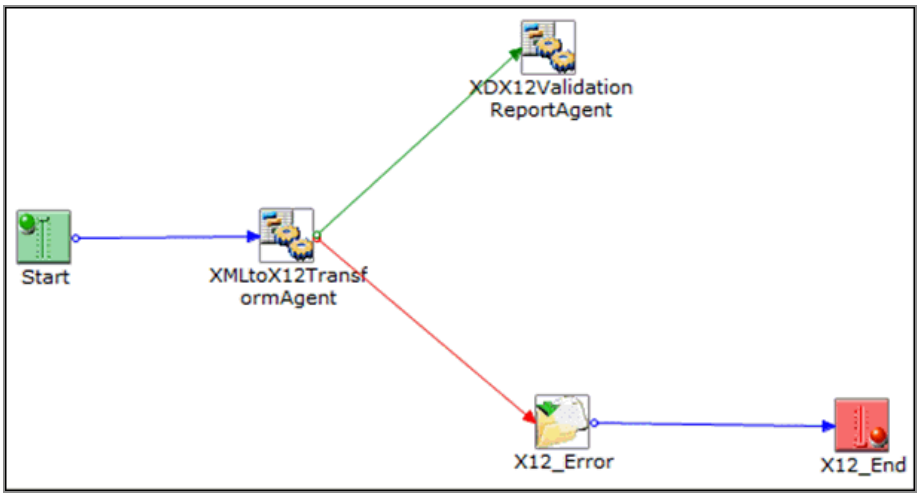
    The Properties dialog box opens.

35. Configure the available parameters according to your requirements.
36. Click **Finish**.

    The new Service object (XDX12ValidationReportAgent) appears in the workspace.
37. Select the **XMLtoX12TransformAgent** object, right-click the **XDX12ValidationReportAgent** object, and select **Create Relation** from the context menu.

    The Line Configuration dialog box opens.
38. From the Event drop-down list, select **OnSuccess** and click **OK**.

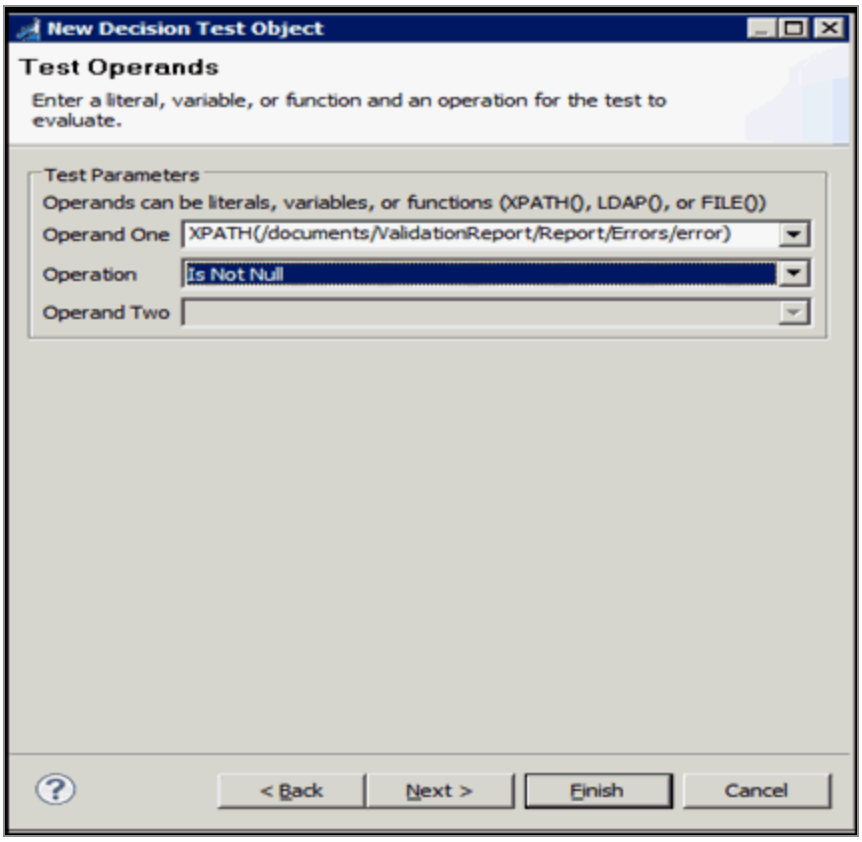
    A line appears between the objects to indicate that a relationship has been established, as shown in the following image.



39. Drag and drop the Decision Test object from the Object palette to the workspace.

    The New Test Object dialog box opens.
40. In the Name field, type **Decision Test**, and a brief description (optional) in the Description field.
41. Click **Next**.

    The Test Operands dialog box opens.

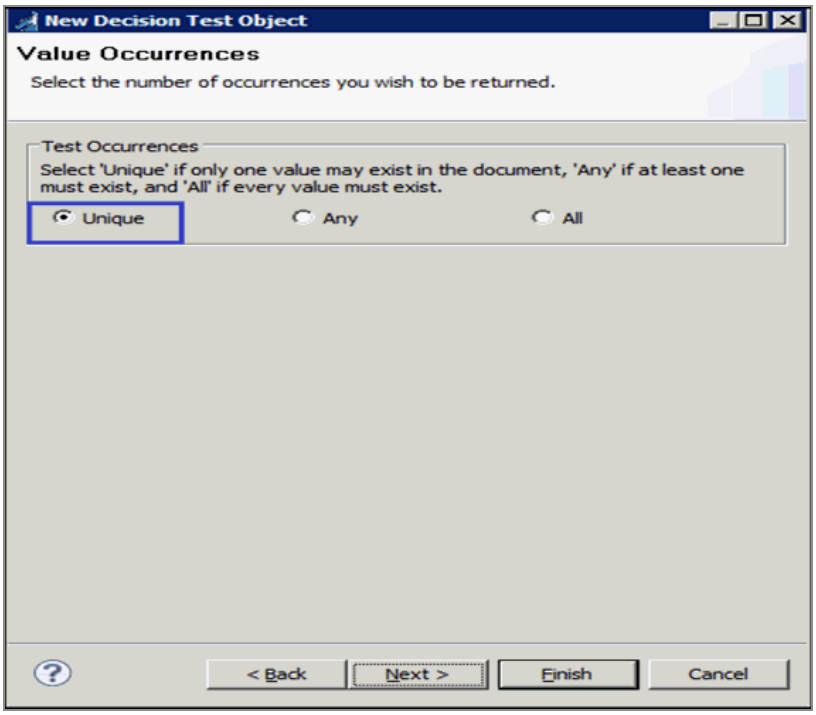


42. In the Operand One field, enter the following:

```
XPATH(/documents/ValidationReport/Report/Errors/error)
```

43. From the Operation drop-down list, select **Is Not Null**.
44. Click **Next**.

    The Value Occurrences dialog box opens.

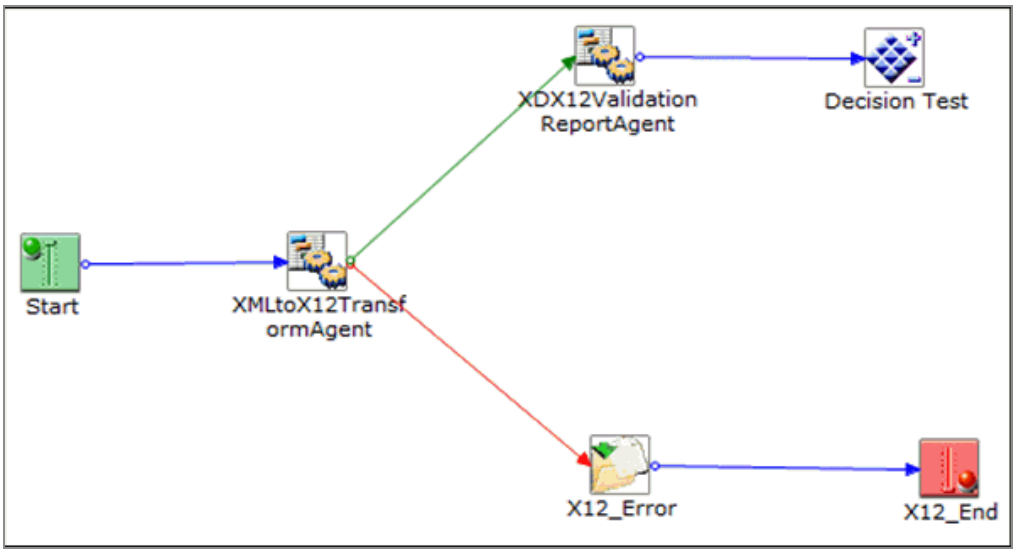


45. Ensure that **Unique** is selected from the available options.
46. Click **Finish**.

    The new Decision Test object appears in the workspace.
47. Select the **XDX12ValidationReportAgent** object, right-click the **Decision Test** object, and select **Create Relation** from the context menu.

    The Line Configuration dialog box opens.
48. From the Event drop-down list, select **OnCompletion** and click **OK**.

    A line appears between the objects to indicate that a relationship has been established.



49. Drag and drop the File object from the toolbar to the workspace.

    The New File Object dialog box opens.

50. In the Name field, type **Write Good File**, and a brief description (optional) in the Description field.

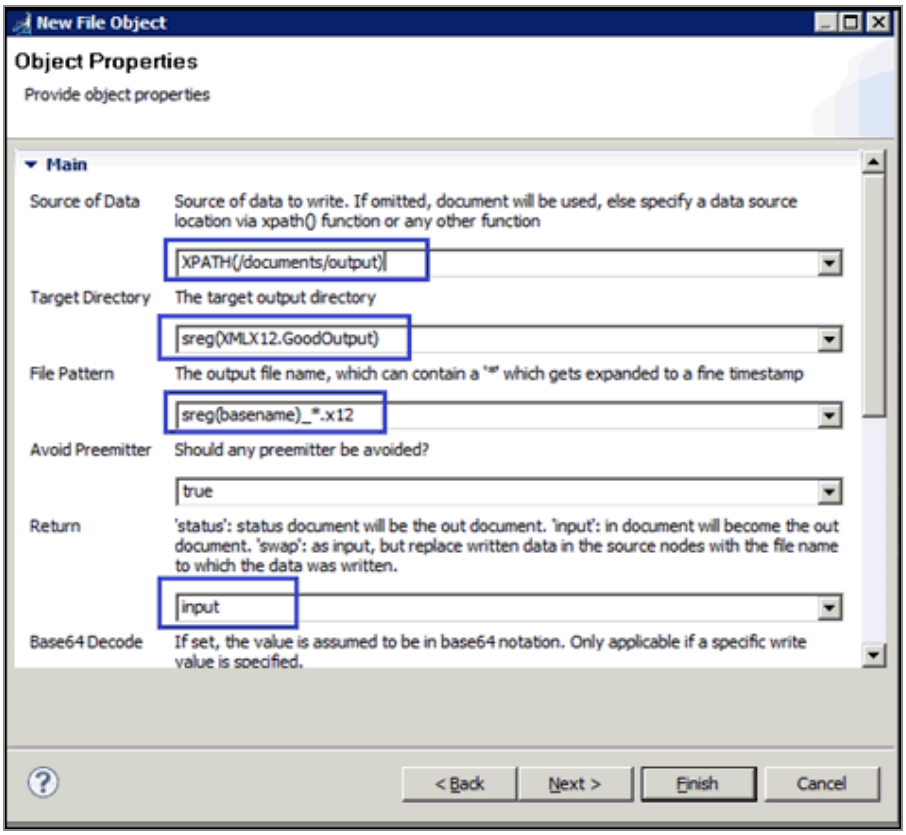
51. Click **Next**.

    The File Type dialog box opens.

52. From the Type drop-down list, select **File Emit Agent**.

53. Click **Next**.

    The Properties dialog box opens.



54. For the Source of Data parameter, enter the following:

```
XPATH(/documents/output)
```

55. For the Target Directory parameter, enter the following location where valid data will be written:

```
sreg(XMLX12.GoodOutput)
```

56. For the File Pattern parameter, enter the following:

```
sreg(basename)_*.x12
```

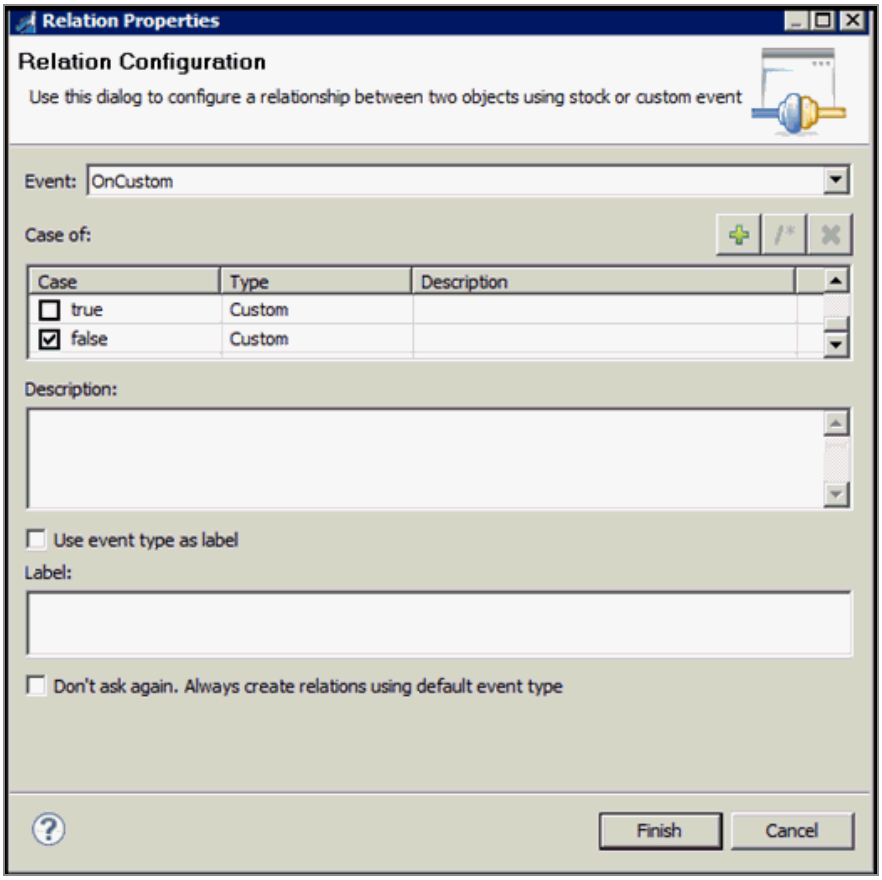
57. For the Return parameter, select **input** from the drop-down list.
58. Click **Finish**.

    The new File object (Write Good File) appears in the workspace.

59. Select the **Decision Test** object, right-click the **Write Good File** object, and select **Create Relation** from the context menu.

    The Line Configuration dialog box opens.

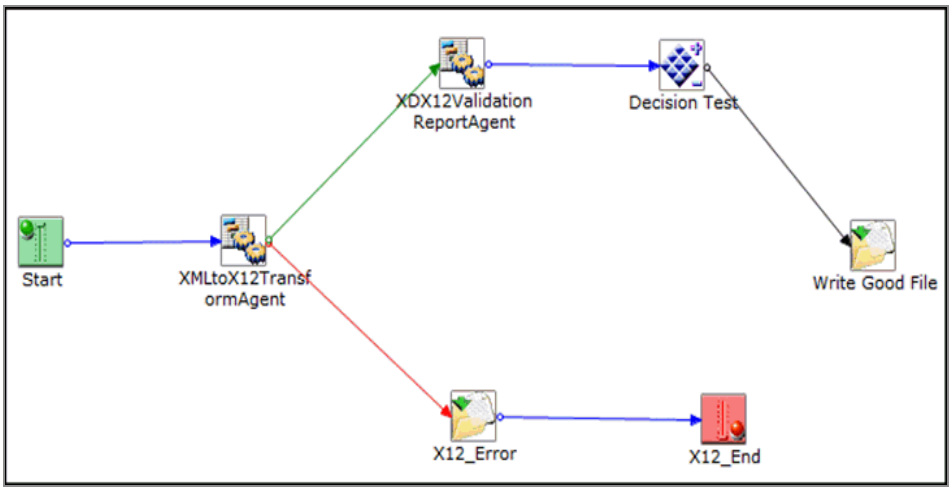
60. From the Event drop-down list, select **OnCustom**.



61. In the Case of section, select **false**.

62. Click **Finish**.

    A line appears between the objects to indicate that a relationship has been established.



63. Drag and drop the End object from the toolbar to the workspace.

    The End Name and Description dialog box opens.

64. In the Name field, type **End**, and a brief description (optional) in the Description field.

65. Click **Next**.

    The End Name Schema dialog box opens.

66. In the Terminate parameter, select the check box for **Select if this end object is the completion point**.

67. Click **Next**.

68. Click **Finish**.

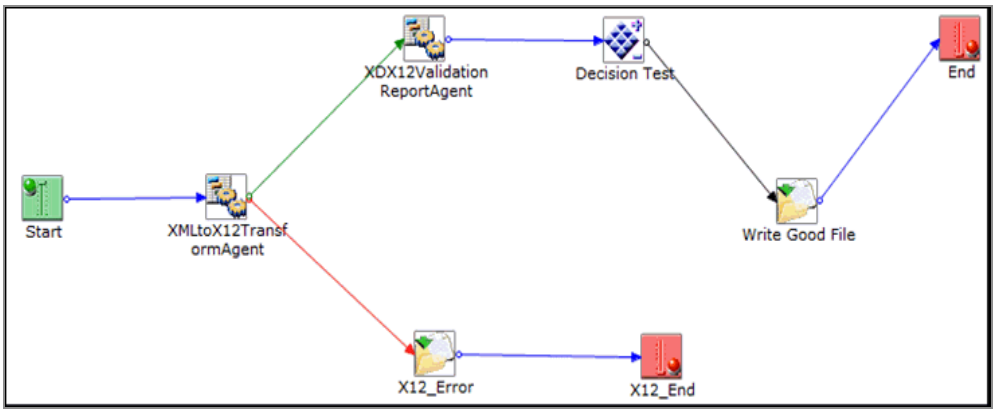
    The new End object appears in the workspace.

69. Select the **Write Good File** object, right-click the **End** object, and select **Create Relation** from the drop-down list.

    The Line Configuration dialog box opens.

70. From the Event drop-down list, select **OnCompletion** and click **OK**.

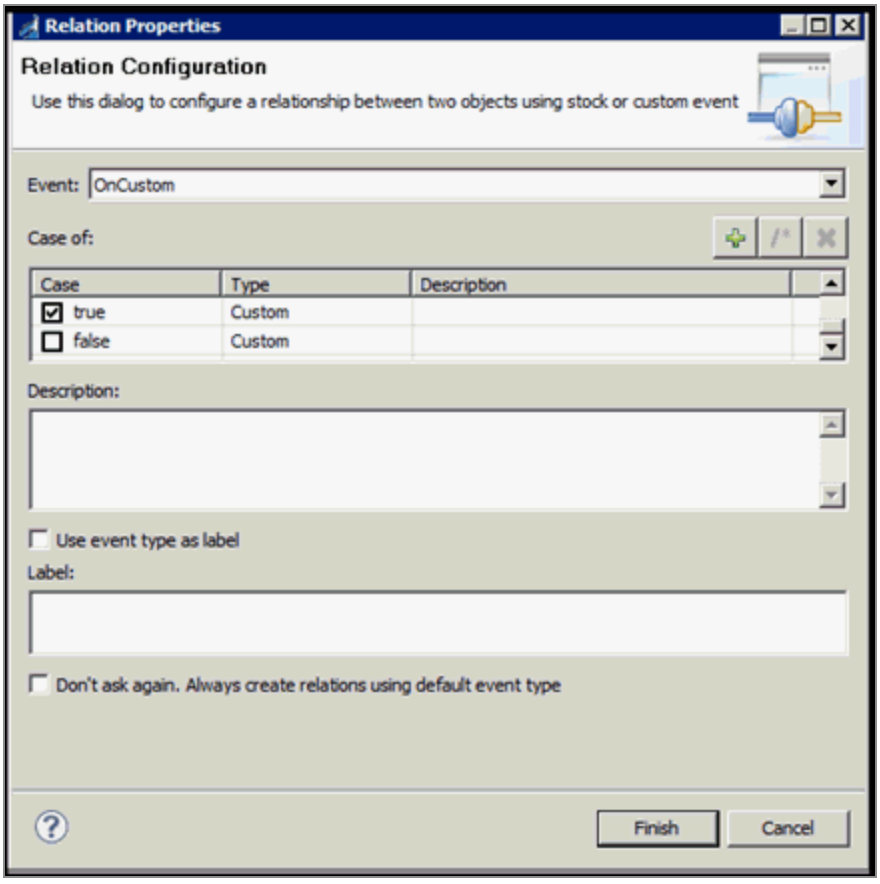
    A line appears between the objects to indicate that a relationship has been established.



71. Select the **Decision Test** object, right-click the **End_Success** object, and select **Relation** from the context menu.

    The Line Configuration dialog box opens.

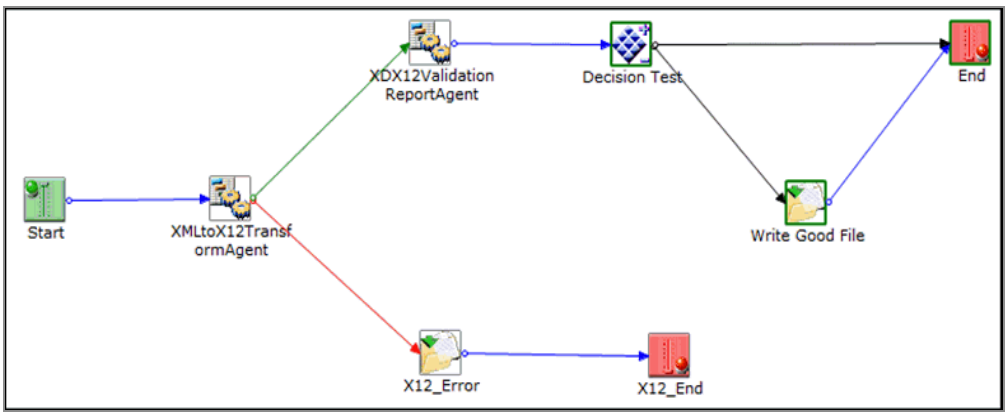
72. From the Event drop-down list, select **OnCustom**.



73. In the Case of section, select **true**.

74. Click **Finish**.

    A line appears between the objects to indicate that a relationship has been established.



    The process flow is now complete.

75. To save the process flow, click **File** from the toolbar menu and select **Save**.

    You can also click the **Save** icon available in iIT, just below the View menu.

    You must now validate the process flow and publish it to the Registry in the iWay Service Manager Administration Console, for use in the route of a channel for outbound processing.

    Validating a process flow ensures that its structure is correct. Publishing a process flow makes it available in the Registry for use in a channel configuration. For more information on validating and publishing the process flow, see the *iWay Integration Tools Designer User's Guide*.

76. Close iWay Integration Tools.

    Your next step is to add a new route to the Registry using the iWay Service Manager Administration Console and associate the process flow with it.

Define a Route and Associate the Process Flow With It

To define a route and associate the process flow with It:

Procedure

1. From the Registry menu options in the iWay Service Manager Administration Console, click **Routes**.
2. On the Route Definitions pane, click **Add** to add a route.
3. On the New Route Definition pane, enter a name for the route and an optional description, as shown in the following table.

Parameter	Value
Name *	XMLToX12
Description	This route will invoke the XML to X12 validation process. The outcome of the validation process will place valid X12 data in your valid outbound folder. Invalid X12 will be routed to an errors folder. A validation report will also be sent to the appropriate folder.

4. Click **Finish**.
5. On the Construct Route pane, click **Add**.

   You are prompted for the type of component to associate with the route.
6. Select **Process** and click **Next**.
7. The next pane prompts you to select a process. Select the process flow you created earlier with iWay Integration Tools, **XMLToX12_Ebix**, and click **Finish**.

   The route, with its associated process flow, has been successfully defined.

Defining an Outlet

For the iWay Integration Solution for EDI X12, you will add an emitter to the Registry, then associate it with a new outlet.

Add an Emitter for an Error Validation Report

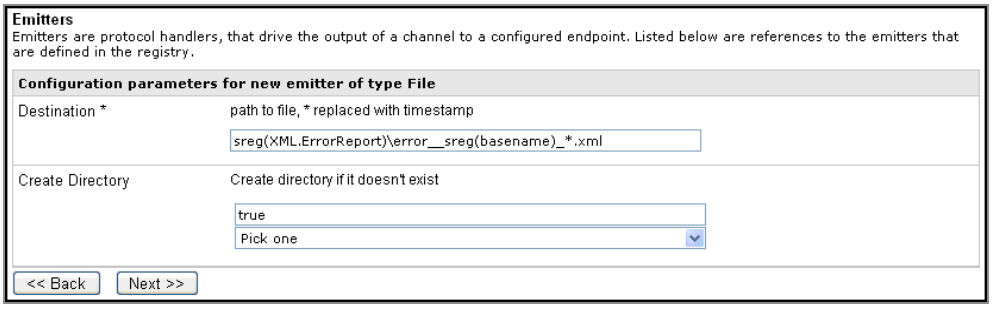
To add an emitter that will emit an error validation report and error file due to the XML to X12 validation process:

Procedure

1. From the Registry menu options, select **Emitters**.
2. On the Emitters pane, click **Add** to add an emitter.

   The next pane prompts you for the emitter type.
3. Select **File** from the drop-down list and click **Next**.

   The File Emitter configuration parameters pane opens.



4. In the Destination field, enter the following:

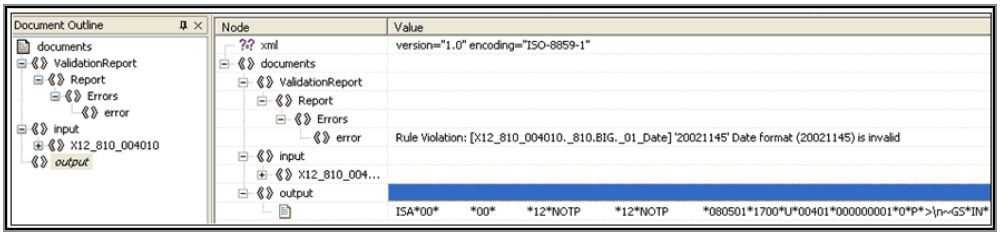
```
sreg(XML.ErrorReport)\error__sreg(basename)_*.xml
```

5. From the Create Directory drop-down list, select **true**.
6. Click **Next**.
7. On the Emitters pane, enter the name of the new emitter and an optional description, as shown in the following table:

Parameter	Value
Name *	XmltoX12Ebix_XML_error
Description	XmltoX12Ebix_XML

8. Click **Finish** to add the emitter.

The following example shows the structure of an error validation report that is returned if the X12-formatted input XML document is invalid.



Add an Emitter for a Valid Validation Report

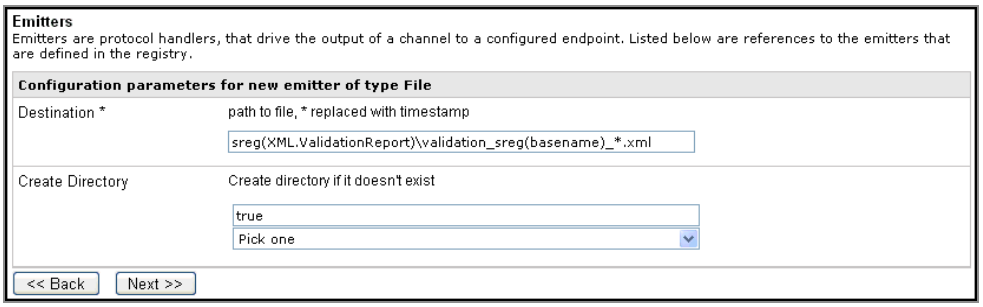
To add an emitter that will emit a valid validation report due to the XML to X12 validation process:

Procedure

1. From the Registry menu options, select **Emitters**.
2. On the Emitters pane, click **Add** to add an emitter.

   The next pane prompts you for the emitter type.
3. Select **File** from the drop-down list and click **Next**.

   The File Emitter configuration parameters pane opens.



4. In the Destination field, enter the following:

```
sreg(XML.ValidationReport)\validation_sreg(basename)_*.xml
```

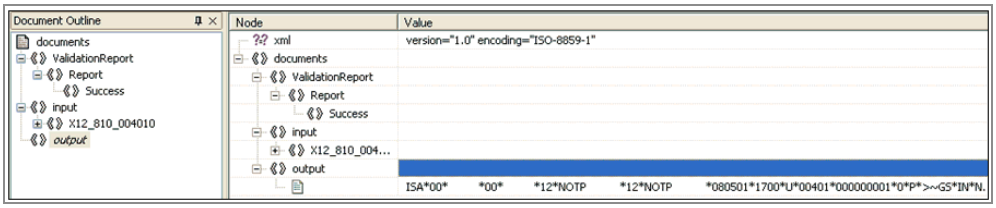
5. From the Create Directory drop-down list, select **true**.
6. Click **Next**.

7. On the Emitters pane, enter the name of the new emitter and an optional description, as shown in the following table:

Parameter	Value
Name *	XmltoX12Ebix_XML_validation
Description	XmltoX12Ebix_XML

8. Click **Finish** to add the emitter.

   The following example shows the structure of a valid validation report that is returned if the X12-formatted input XML document is valid.



Define the Outlets

Now that you have added two emitters to the Registry, you are ready to define the required outlets. Each emitter will be associated with a corresponding outlet.

Procedure

1. From the Registry menu options, select **Outlets**.
2. On the Outlet Definitions pane, click **Add** to add the first outlet.
3. On the New Outlet Definition pane, enter the name of the first new outlet and an optional description, as shown in the following table.

Parameter	Value
Name *	XmltoX12Ebix_XML_error

Parameter	Value
Description	Outlet which will contain error validation reports and error files due to the XML to X12 validation process.

4. Click **Finish** to add the outlet.
5. On the Construct Outlet pane, click **Add** to associate the emitter with the outlet.

   The next pane prompts you for the component type.
6. Select **Emitter** and click **Next**.

   The next pane prompts you to select an emitter.
7. Select **XmltoX12Ebix_XML_error**, which is the first emitter you added earlier, and click **Finish**.
8. On the Outlet Definitions pane, click **Add** to add the second outlet.
9. On the New Outlet Definition pane, enter the name of the second outlet and an optional description, as shown in the following table.

Parameter	Value
Name *	XmltoX12Ebix_XML_validation
Description	Outlet which will contain valid validation reports produced by the validation process.

10. Click **Finish** to add the outlet.
11. On the Construct Outlet pane, click **Add** to associate the emitter with the outlet.

    The next pane prompts you for the component type.
12. Select **Emitter** and click **Next**.

    The next pane prompts you to select an emitter.
13. Select **XmltoX12Ebix_XML_validation**, which is the second emitter you added earlier.

14. Click **Finish**.

Defining a Channel

Now that you have defined the required components for the outbound channel, you are ready to add the channel to the Registry and associate the conduits with it. At this time you will also add the route to the channel.

Define a Channel

To define a channel:

Procedure

1. From the Registry menu options, select **Channels**.
2. On the Channel Definitions pane, click **Add** to add a channel.
3. On the New Channel Definition pane, enter the name of the new channel (for example, **XmlToX12_Ebix**) and an optional description. Then click **Finish** to add the channel.
4. On the Construct Channel pane, click **Add** to associate the inlet, route, and outlets with the channel.

   You are prompted to associate components with the channel.
5. Select **Inlet** and click **Next**.

   The next pane prompts you to select an inlet.
6. Select **XmlToX12_Ebix**, which you defined earlier, and click **Finish**.

   The inlet is associated with the channel. Now you need to associate a route with the channel and set it to the default.
7. On the Construct Channel pane, click **Add**.

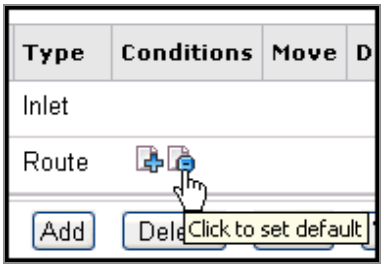
   The next pane prompts you for the component type.
8. Select **Route** and click **Next**.

   On the next pane, you are prompted to select a route.

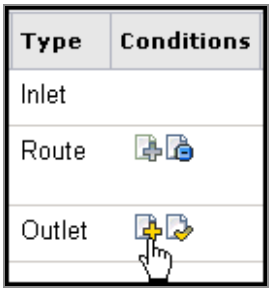
9. Select **XmltoX12Ebix**, which you defined earlier, and click **Finish**.

   The Construct Channel pane reopens.

10. Click the minus sign (-) under Conditions to set this route as the default.



11. On the Construct Channel pane, click **Add** to add the next component.
12. When prompted for the component type, select **Outlet** and click **Next**.
13. Select the two outlets you defined earlier, **XmltoX12Ebix_XML_error** and **XmltoX12Ebix_XML_validation**.
14. Click **Finish**.
15. To set a condition for the outlets, on the Construct Channel pane, click the plus sign (+) under Conditions for the specific outlet.



   The Set Condition pane opens.

16. In the Condition input field, enter the appropriate conditional expression, and click **Update**.

   The following table lists the expression that must be entered for each outlet.

Outlet	Expression
XmltoX12Ebix_XML_ validation	`_isxml() and sreg (iwaf.validationSuccess) = true`
XmltoX12Ebix_XML_error	`_isxml() and sreg (iwaf.validationSuccess) != true`

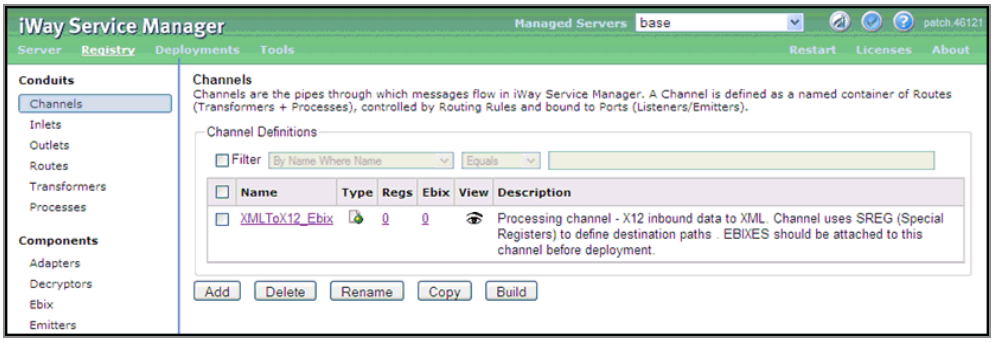
For details on supported conditions, see the topic on using functions in the *ibi™ iWay® Service Manager User's Guide*.

Add a Special Register Set to the Channel

Procedure

1. From the Registry menu options, select **Channels**.

   The Channel Definitions pane opens.



2. Click the link in the **Regs** column for the XMLtoX12_Ebix channel.
3. On the next pane, which prompts you to add special register (SREG) sets, click **Add** to add the SREG set to the channel.
4. On the next pane, select **XML**, which is the name of the SREG set you created previously, and click **Finish**.

Add the Ebix to the Channel

Procedure

1. From the Registry menu options, select **Channels**.

   The Channel Definitions pane opens.

2. Click the link in the Ebix column for the XmlToX12_Ebix channel.
3. On the next pane, which prompts you to add Ebix components, click **Add** to add the Ebix to the channel.
4. On the next pane, select **EDI_4050**, which is the name of the Ebix you added previously, and click **Finish**.

Build the Channel

Procedure

1. From the Registry menu options, select **Channels**.
2. On the Channel Definitions pane, select the channel for outbound processing defined previously, **XmlToX12_Ebix**, and click **Build**.

   The results of the build are displayed on the right pane.

3. Review the results of your build and then click **Back**.

   If an error or errors are displayed in the Message column, take the appropriate action as instructed.

Deploy the Channel

Deployment is the mechanism by which a channel moves from being stored in the Registry to becoming active in iWay Service Manager. For more information on deployment, see the *ibi™ iWay® Service Manager User's Guide*.

Procedure

1. Select the **Deployments** option.
2. On the Channel Management pane, click **Deploy**.
3. On the Available Channels pane, select the channel you defined previously, **XmlToX12_Ebix**, and click **Deploy**.

   The Channel Management pane reopens.
4. Select **XmlToX12_Ebix** and click **Start**.

   The red X under Status changes to a green check mark to indicate that the channel has been started. If an error or errors are displayed, take the appropriate action as instructed.

Verify the Channel

To ensure that the channel is working as expected, perform the following steps.

For more information on obtaining EDI X12 sample files for testing purposes, see Downloading and Extracting EDI X12 User Samples.

Procedure

1. Place an XML file as test data into the input directory. This is the path in which XML messages are received, which you specified for the listener associated with the inlet for the channel.
2. Check for the EDI output file in the output directory. This is the destination directory you specified for the listener.
3. Confirm that the output has been converted to EDI format.

Reusing Your Channel Configuration

Using the Archive Manager feature of iWay Service Manager, you can archive your channel configuration with its associated components and import them into another Registry. They will then be available from that Registry for modification or reuse.

For details on this feature, see the *ibi™ iWay® Service Manager User's Guide*.

Batching for Outbound Documents

This chapter describes the Outbound Batching feature for the iWay Integration Solution for EDI X12.

Overview

In previous releases, outbound EDI X12 documents were being sent as single units of work. Each document was wrapped in an interchange wrapper, a group wrapper, and a document wrapper.

As of iWay version 8.0.4, the Outbound Batching feature is available for the iWay Integration Solution for EDI X12, which enables you to send batches of outbound documents. This feature requires iWay TPM (TPM). New metadata tags were added to the TPAVALUES table, which are required by embedded TPA calls and extraction/concatenation. For example:

- ISA13 (X12 Interchange Control Number)
- GS06 (X12 Group Control Number)
- ST02 (X12 Transaction Set Control Number)

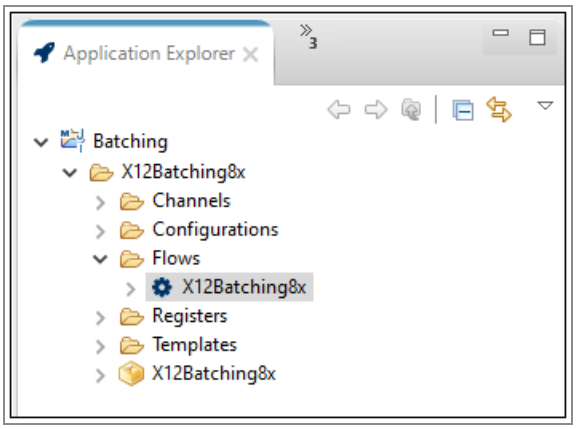
As a best practice, document headers are being populated from the trading partner table (s). The transformed standard formatted data already contains the values from these fields. This provides a user framework for creating documents rather than hard-coding within the transforms.

Packaging

The Outbound Batching feature is packaged with the sample EDI X12 channel that you can download from https://techsupport.informationbuilders.com/ and then import into iWay Integration Tools (iIT).

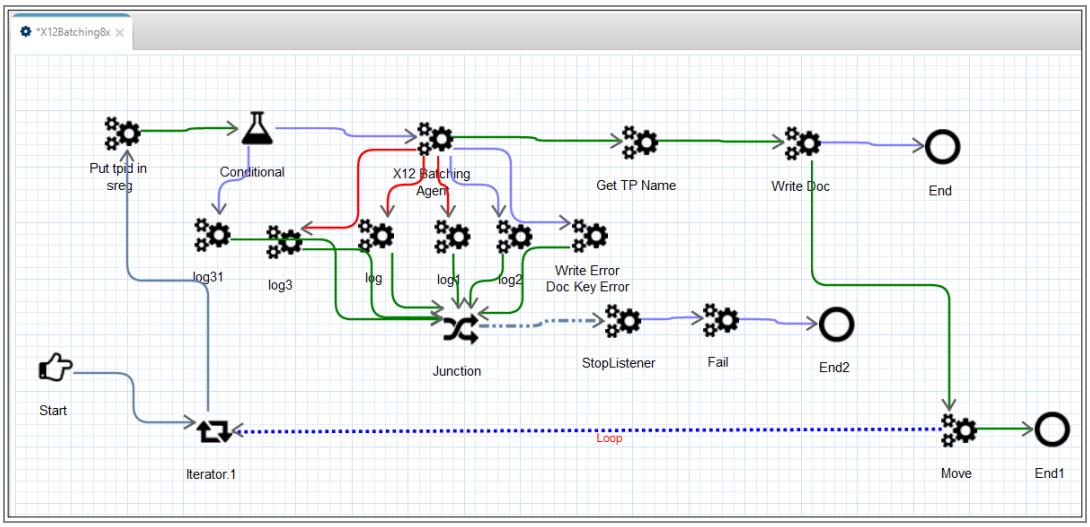
Outbound Batching Process Flow

After you import the sample EDI X12 channel that you downloaded into iWay Integration Tools (iIT), expand the **Flows** subfolder, as shown in the following image.



Double-click **X12Batching8x**.

The Outbound Batching process flow for EDI X12 (X12Batching8x) opens as a new tab in the iIT workspace area, as shown in the following image.



You can review this process flow in more detail by clicking on any object to view its properties.

Control Numbers

Control numbers are assigned at each Header/Trailer level break. The same number at the header is inserted into the trailer. The numbers are read from TPM using the TPAW function. This function reads the next number to use, increments by one, and then writes the next number to use to the database.

Transaction Set Control numbers for X12 can also be 0001, 0002, 0003, and so on.

A batch is defined as one header (ISA), one-to-many group headers (GS), and one-to-many documents (ST).

All documents in the batch should have the same delimiters and terminators. Delimiters or terminators are not changed within the documents. The first document selected in the batching process for the batch determines the delimiters and terminators for the batch header and the first group. The first document in each group determines the delimiters and terminators in each group. The delimiters and terminators in the group and batch trailers match their respective headers. X12 allows multiple documents in a group with different delimiters and terminators, so users are allowed to do this, even though it is not considered a best practice.

Assembly Details: Order of Operations

1. Outbound documents are processed from XML to EDI X12 format.
2. Header fields and document (as a Binary Large OBject (BLOB)) are extracted and persisted to the database table.
3. Database table is periodically swept. Documents with like trading partner credentials are selected and flagged.
4. Group is built from the inside out as follows:
    a. Each document header is constructed.
    b. Each document is appended to the header.
    c. Document trailer is constructed and appended to each document.
    d. Batches of documents (steps a-b-c) are concatenated together for a single group.
    e. Group header is constructed. Result of step d is appended to the header.

f. Group trailer is constructed and appended to the result of step e.

g. If there are more documents, then repeat steps a-b-c-d-e-f.

h. If there are no more documents, then the Interchange header is constructed. Result of step f is appended to the header.

i. Interchange trailer is constructed and appended to the output file.

Extracted Fields

Using XPATH, it is possible to define the header, group, and document elements to collect and persist to the database from the validation report. Additional elements to persist include the element and subelement delimiters and the segment terminator. These three fields must be persisted as hex values. Note that three different formats of segment terminators are allowed, and these must be preserved in the disassembly and reassembly processes.

Trading Partner Flags and Counters

A trading partner is defined to represent a document going to an interchange sender and receiver, group sender and receiver, version and document (*six key lookup*). For example, if you are sending invoices and advanced shipping notices to JCPenney, you will define two trading partners, as shown in the following example:

```
FROM_US*TO_JPC*SENT_FROM_US*TO_JCP_ACCT*4010*810 (invoice)
FROM_US*TO_JPC*SENT_FROM_US*TO_JCP_RECEIVING*4010*856 (ASN)
```

A combination of Interchange sender and receiver usually resolved to the mailbox address. Sometimes the group sender and receiver IDs are the same as the interchange. Other times it will resolve to the slot in the mailbox.

In order to use this feature:

- A metadata tagged field is added to TPM. This indicates if the partner receives data in the old (unbatched) or new (batched) presentation. Absence of the flag will default to “unbatched.” The stock channels require a decision point to read this tag. Unbatched documents continue to flow as they do now. Batched documents route to the new

agent that reads the data elements from each document and persists to a database table.

- Metadata tags must exist to contain the next interchange, group and document numbers to assign. TPA function TPAW reads this number, increments, and writes the new “next number to assign” to the table. If the document number is not present, then the default value of “0001” is assigned. The agent has three prompts to contain the name of the metadata tag for each of these. You can modify these via SREG or can insert the values that match your database.
- A metadata tag is added to assign the maximum number of documents to allow in a group. If this metadata tag and the corresponding agent tags are not present, then the batching, when run, will contain all available documents. If this metadata tag is present, then groups will be created of this size and a remainder group will be created if necessary.
- A metadata tag “Accumulation Threshold” is added to assign the minimum number of documents to assign to a batch. If this metadata tag is not present or this tag is zero (“0”), then the batching, when run, will contain all available documents. If this metadata tag is present, then batches will be created if the minimum number of documents available to batch exceeds this number.
- A metadata tag is added to determine the “Frequency To Batch” for a partner. Valid values for this tag can range from five minutes (“00:05”) up to an entire day (“23:59”). If the tag is missing, then the default is “23:59”. This ensures that the outbound documents are sent at least daily. In addition, the last date and time batched is written to a tag in the table. This is the value that is inspected and used to determine if data is ready to batch or not.
- A metadata tag is added to determine the *Age Threshold* for a partner. Valid values for this tag can range from five minutes (“00:05”) up to an entire day (“23:59”). If the tag is missing, then the default is “23:59”. This ensures that the outbound documents are not present on the table for more than one day. In addition, the last date and time batched are written to a tag in the table. This is the value that is inspected and used to determine if data is ready to batch or not. This value is optional and overrides the value set in the batching agent.
- A metadata tag is added to determine if multiple groups are allowed in an interchange. The default or missing tag value is true (“1”). This allows the user to send interchanges with only one group per interchange instead of multiple groups per interchange. This also allows the user to set a specific trading partner to batch with only one group per interchange.

Data Collection

Two new tables have been added to the TPM database for the iWay Integration Solution for EDI X12 in support of the Outbound Batching feature.

- Header Table: **X12_BATCHHEADER**
- Detail Table: **X12_BATCHDETAIL**

Each row in the X12_BATCHDETAIL table contains each individual document and its extracted metadata elements. In addition, the document is extracted and stored as a Binary Large OBject (BLOB). The *datetimestamp* that the row was persisted is recorded. When rows are selected for batching, a *datetimestamp* is also set and the active row count in the BATCHHEADER is updated. A cleanup SQL script is provided for this table, but does not delete data in real time.

The X12_BATCHHEADER table is a summary by six key lookup. It contains the active row count and total row count for each unique six key lookup that the sweep process will inspect. It contains the last written *datetimestamp* and last batched *datetimestamp*. These are used by the sweep process to determine if the partner has exceeded the age threshold, accumulation threshold, or maximum batch size from TPAVALUES.

Database Sweep (Data Selection) for Batching

The iWay Integration Solution for EDI X12 has its own selection process for batching and runs in its own iWay Integration Application (iIA). This allows user with X12 and EDIFACT to batch concurrently. Multithreading is not supported.

A standard SQL adapter is used to select headers with available details to batch. The adapter uses a stored procedure to allow users to modify the sort selection (for example, sort by date or invoice number). The user is prompted for the name and file location of the stored procedure to execute in this agent and TPAVALUES.

An option is provided to set the time to sleep between channel executions.

An option is provided to set the age threshold (maximum time) that a document can be on the work table. If the maximum time is exceeded, then that document must be selected and batched, regardless of any options set in the TPM table. This ensures that documents do not get *stuck* in the batching process and never emitted. Typically this will be “23:59” so documents are batched daily.

An option is provided to allow multiple groups in an interchange. This option defaults to *true*. The optional flag trading partner level takes precedence.

The selected headers are validated against TPM by checking the accumulation threshold (minimum batch size), maximum batch size, and age threshold to batch. If the partner is valid to batch, the header busy flag is set to “1”.

Building Each Batch

Follow the *Assembly Details: Order of Operations* as previously stated.

- For each header:
    - Select all of the details for that header.
    - Save the current *datetimestamp*.
    - Update *datetimestamp* with the value that was saved.
    - Count the number of detail records batched.
    - Create an interchange control from the first selected detail record in each group break. Replace the date and time in the header.
    - Use TPAW to get and set the control number.

      Note: The X12 ISA and IEA are fixed length.
- For each group:
    - Create a group header from the first selected detail record in each group break.
    - Use the same date and time as the interchange header.
    - Use TPAW to get and set the control number.
    - Create and append the group trailer. The trailer control number is the same as the group header.
- For each document:
    - Get the document BLOB.
    - Use TPAW to get and set the document control number.
    - Update the document header control number.
    - Append the document BLOB.

    - Update the document trailer control number.
- When there are no more document BLOBs:
    - Wrap up the last group and append a trailer.
    - Create and append the interchange trailer. The trailer control number is the same as the interchange header. Do not forget to update the number of groups.
    - Decrement the number of details to batch in the BATCHHEADER row.
    - Update BATCHHEADER last update *datetimestamp* with the value that was saved at the beginning of the process.
    - Output the batched file so it can be emitted by the process flow.
- Set the last processed *datetimestamp* in the header.
- Set the header busy flag to “0”.

Note: For EDI X12 the trailing segment delimiter is required.

Appendix A: Batch Agent Options

1. **Interchange Next Control Number** metadata tag name from TPAVALUES. Mandatory in agent and TPAVALUES. Used by TPAW. If not found in TPAVALUES, for EDI X12 default to:

   ```
   100000001
   ```

   Note: For EDI X12, value is fixed length 9 numeric, right padded with zeros (“0”). If you read “123” from the table, then “000000123” is the value to insert.

2. **Group Next Control Number** metadata tag name from TPAVALUES. Mandatory in agent and TPAVALUES. Used by TPAW. If not found in TPAVALUES, for EDI X12 default to:

   ```
   1
   ```

3. **Document Next Control Number** metadata tag name from TPAVALUES. Mandatory in agent and TPAVALUES. Used by TPAW. If not found in TPAVALUES, for EDI X12 default to:

```
0001
```

Note: For EDI X12, value is variable length 4-9 characters. If “123” is read from the table, then “0123” is the value to insert. If there is no tag in TPAVALUES, then the documents are numbered 0001, 0002, 0003, and so on in the group.

4. **Accumulation Threshold.** The minimum number of documents to batch in a Group. Optional in agent and TPAVALUES, TPAVALUES takes precedence. If this value is populated and this value is exceeded, then batch. Sort the oldest documents first. If there are 60 documents in the queue and this value is set to 50, create one batch of 60. If there are 110 documents in the queue, then create one batch of 110 documents each.
5. **Age Threshold.** Minimum age of an accumulated document required to trigger a new batch output. Minimum five minutes (“00:05”), maximum once a day (“23:59”). The document is included in the batching if value, compared to the timedate that it was added to the detail table, is exceeded. Mandatory in agent, optional in TPAVALUES. TPAVALUES takes precedence.
6. **Maximum number of documents in a Group.** Optional in agent and TPAVALUES, TPAVALUES takes precedence. If this value is populated and the document count is exceeded then create batches in multiples of this value. Sort the oldest documents first. Create a batch for the remainder of the documents. If there are 60 documents in the queue and this value is set to 50, create a group of 50 and a group of 10. If there are 110 documents in the queue, then create two output files of 50 documents each and one of 10 documents. This option might be typically used for someone creating price catalogs where the receiver can’t import large files. If you exceed the value, clear all documents waiting to batch. If the user sets both minimum and maximum, then minimum takes precedence and maximum is ignored.
7. **Frequency To Batch.** Time to sleep agent in minutes before waking up. There is a similar option in TPAVALUES that overrides. Minimum five minutes (“00:05”), maximum once a day (“23:59”). The agent runs when it starts. A restart of the iIA serves as a *run now* option.
8. **Allow Multiple Groups.** The value in TPAVALUES overrides this. If false or 0, then each interchange contains one group.

Appendix B: New TPAVALUES Metadata Tags for ANSI X12 Documents

- X12_Min – See *Appendix A*, #4.
- X12_Max – See *Appendix A*, #5.
- X12_Age – TPM specific. If not set, and the data sweep runs, and the min or max is met or not present, then batch. If set, and the data sweep runs, and the min or max is met or not present, and the time interval has passed, then batch. Set in minutes. For example, a setting of “60” indicates a batch every hour and not sooner. If the data sweep runs and an hour has not elapsed, then proceed to the next trading partner.
- X12_Last_timedate_batched. Updated by batch assembly process and then used by X12_Freq_to_Batch time.
- X12_Allow_Multiple_Groups. True or False or 1 or 0. Used by the batch assembly process.
- ISA_01_Authorization_Information_Qualifier
- ISA_02_Authorization_Information
- ISA_03_Security_Information_Qualifier
- ISA_04_Security_Information
- ISA_05_Interchange_ID_Qualifier
- ISA_06_Interchange_Sender_ID
- ISA_07_Interchange_ID_Qualifier
- ISA_08_Interchange_Receiver_ID
- [Do not need ISA_09_Interchange_Date]
- [Do not need ISA_10_Interchange_Time]
- ISA_11_Repetition_Separator [single character in HEX]
- ISA_12_Interchange_Control_Version_Number
- **ISA_13_Interchange_Control_Number*****
- ISA_14_Acknowledgement_Requested [0 or 1, implied 0 if not populated]

- ISA_15_Usage_Indicator [“P” if not populated]
- ISA_16_Component_Element_Separator [single character in HEX]
- ISA_17_Element_Seperator [single character in HEX]
- ISA_18_Segment_Terminator [single character in HEX]
- GS_01_Functional_Identifier_Code
- GS_02_Application_Senders_Code
- GS_03_Application_Receivers_Code
- [Do not need GS_04_Date]
- [Do not need GS_05_Time]
- **GS_06_Group_Control_Number*****
- GS_07_Responsible_Agency_Code
- GS_08_Version__Release__Industry_Identifier_Code
- ST_01_Transaction_Set_Identifier_Code
- **ST_02_Transaction_Set_Control_Number*****

Note: The *next numbers to assign* fields are indicated by ***.

Appendix C: Sample EDI X12 Batched Document

```
ISA*00* *00* *01*022463293 *ZZ*SPSCABELAS
*080425*0344*U*00401*000007480*0*P*>
GS*IN*022463293*SPSCABELAS*20080425*0344*3148*X*004010
ST*810*000003148
BIG*20080424*05629165*20080423*2020235***DR
REF*19*001
REF*IA*70067
N1*ST*CABELAS*UL*8095790000011
ITD******20080524******NET 30
DTM*011*20080424
FOB*PP
IT1**1*EA*335.31**IN*02510702*VN*52340M*UP*015813523400
```

```
PID*F*08***MOS JIC 500 MARINER 12 18.5
TDS*33531*33531*33531
CTT*1
SE*13*000003149
ST*810*000003148
BIG*20080424*05629165*20080423*2020235***DR
REF*19*001
REF*IA*70067
N1*ST*CABELAS*UL*8095790000011
ITD******20080524******NET 30
DTM*011*20080424
FOB*PP
IT1**1*EA*335.31**IN*02510702*VN*52340M*UP*015813523400
PID*F*08***MOS JIC 500 MARINER 12 18.5
TDS*33531*33531*33531
CTT*1
SE*13*000003149
ST*810*000003150
BIG*20080424*05629165*20080423*2020235***DR
REF*19*001
REF*IA*70067
N1*ST*CABELAS*UL*8095790000011
ITD******20080524******NET 30
DTM*011*20080424
FOB*PP
IT1**1*EA*335.31**IN*02510702*VN*52340M*UP*015813523400
PID*F*08***MOS JIC 500 MARINER 12 18.5
TDS*33531*33531*33531
CTT*1
SE*13*000003150
GE*3*3148
IEA*1*000007480
```

Ebix-Supported Transaction Sets

This topic describes the EDI ANSI X12 transaction sets supported by the iWay Integration Solution for EDI in the Ebix files supplied with the product.

Transaction Set and Acknowledgment Support

The iWay Integration Solution for EDI supports all documents in these versions.

X12:

- 2001
- 2002
- 2003
- 2040
- 3010
- 3020
- 3030
- 3040
- 3050
- 3060
- 3070
- 4010
- 4020
- 4030
- 4040
- 4050
- 4060

- 5010
- 5020
- 5030
- 5040
- 5050
- 6010
- 6020
- 6030
- 6040
- 6050

VICS:

- 3010
- 3020
- 3040
- 3050
- 4010
- 4030
- 4050
- 5010

UCS:

- 3040
- 4010
- 4030
- 5010

Using iWay Integration Tools to Configure an Ebix for EDI X12

This section describes how to use iWay Integration Tools (iIT) to configure an e-Business Information Exchange (Ebix) file for EDI X12.

Using iIT to Configure an Ebix File for EDI X12 Overview

You can use iWay Integration Tools (iIT) to import, edit, export, and work with e-Business Information Exchange (Ebix) files for EDI X12. The topics in this appendix describe how to:

- Import an X12 005010 856 Ebix into iIT.
- Add a qualifier at the 08 [Relationship Code] element under the SLN [Subline Item Detail] segment in the SG0 loop level, to the X12 005010 856 Ebix.
- Export the edited Ebix to a physical location.

The edited Ebix can be returned and then tested with the appropriate X12 005010 856 message.

Using iIT to Configure an Ebix File for EDI X12 Prerequisites

This section provides a list of prerequisites for using iWay Integration Tools (iIT) to configure an Ebix for EDI X12:

- Have a working knowledge of iIT and EDI X12.
- Ensure the iWay EDI X12 adapter is installed.
- Ensure iIT Version 8.0 or higher is installed.

Downloading and Extracting an Ebix File

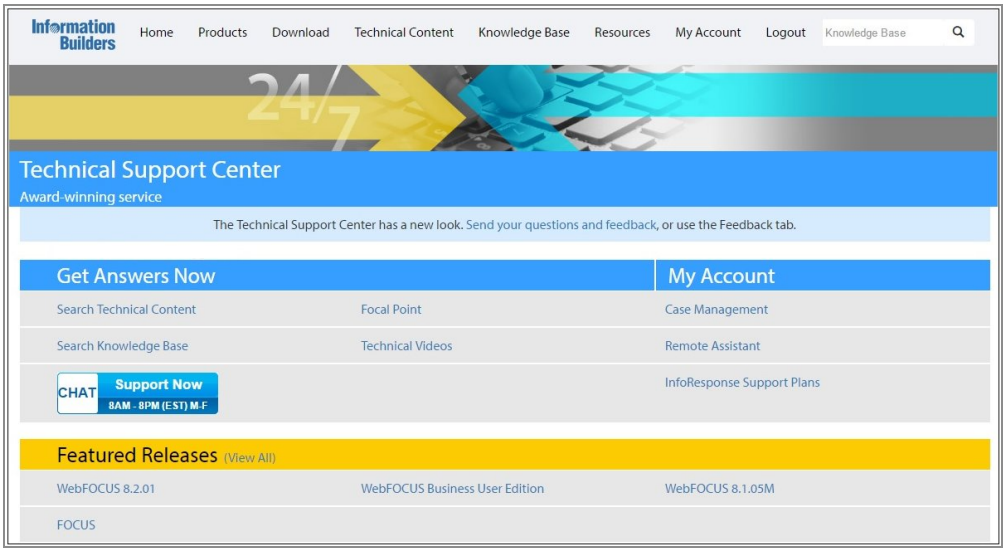
This section describes how to download and extract an Ebix file for EDI X12.

Download and Extract an Ebix File

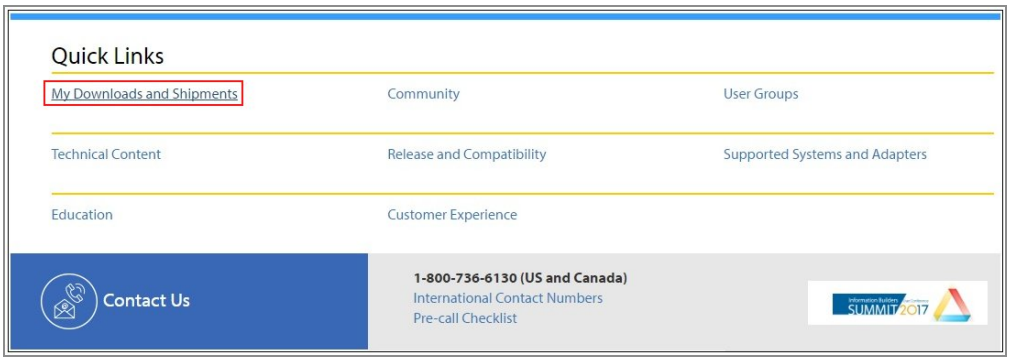
Procedure

1. Enter the following URL in your browser to access the Information Builders Technical Support Center:

   http://techsupport.ibi.com

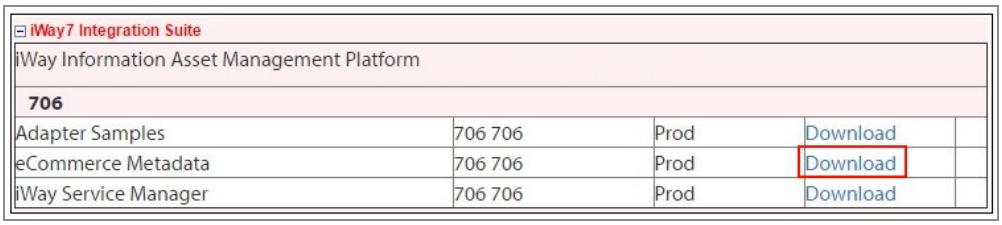


2. Scroll down and click **My Downloads and Shipments** in the Quick Links area, as shown in the following image.



The Software Downloads/Shipments page opens. Scroll down and click **Personal Downloads**, as shown in the following image.

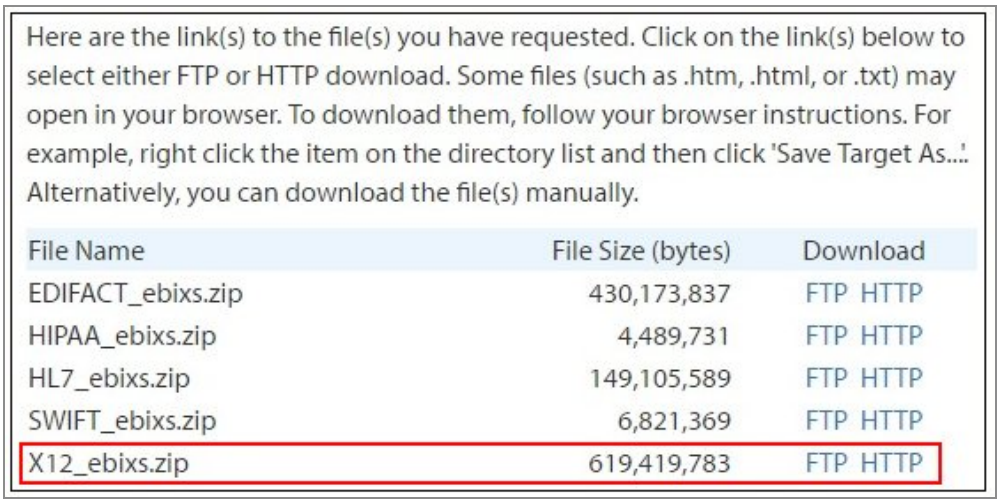
From the list of available software categories that is displayed, expand **iWay7 Integration Suite** and then click **Download** in the eCommerce Metadata row, as shown in the following image.



You are prompted with a download registration form and then a license agreement form.

3. Provide the requested information and accept the license agreement.

   A list of .zip archive files is displayed, as shown in the following image.

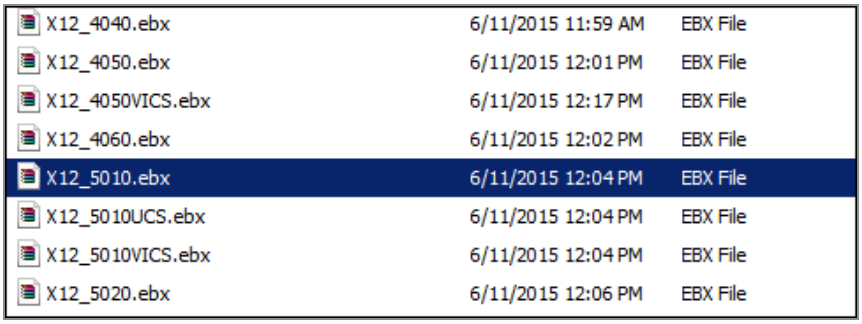


Here are the link(s) to the file(s) you have requested. Click on the link(s) below to select either FTP or HTTP download. Some files (such as .htm, .html, or .txt) may open in your browser. To download them, follow your browser instructions. For example, right click the item on the directory list and then click 'Save Target As...'. Alternatively, you can download the file(s) manually.

File Name	File Size (bytes)	Download
EDIFACT_ebixs.zip	430,173,837	FTP HTTP
HIPAA_ebixs.zip	4,489,731	FTP HTTP
HL7_ebixs.zip	149,105,589	FTP HTTP
SWIFT_ebixs.zip	6,821,369	FTP HTTP
X12_ebixs.zip	619,419,783	FTP HTTP

4. Download the *X12_ebixs.zip* file.
5. Unzip the downloaded *X12_ebixs.zip* file and save *X12_5010.ebx* into any physical location on your local drive.

   For example, this Ebix contains the X12 856 document.



X12_4040.ebx	6/11/2015 11:59 AM	EBX File
X12_4050.ebx	6/11/2015 12:01 PM	EBX File
X12_4050VICS.ebx	6/11/2015 12:17 PM	EBX File
X12_4060.ebx	6/11/2015 12:02 PM	EBX File
X12_5010.ebx	6/11/2015 12:04 PM	EBX File
X12_5010UCS.ebx	6/11/2015 12:04 PM	EBX File
X12_5010VICS.ebx	6/11/2015 12:04 PM	EBX File
X12_5020.ebx	6/11/2015 12:06 PM	EBX File

   Note: Ensure all folders used for the extracted X12_ebixs.zip file do not have any blank spaces in the folder name.

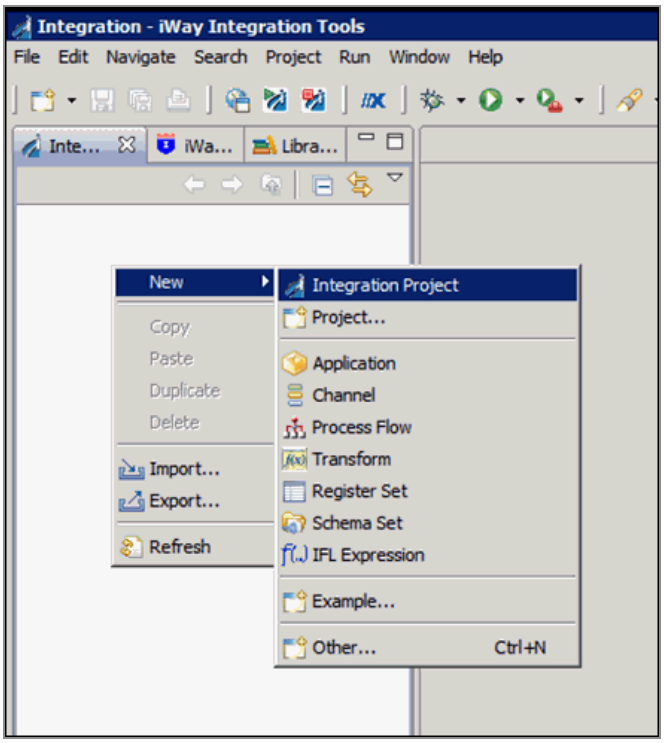
Working With iWay Integration Tools (iIT)

This section describes how to import, edit, and export an Ebix using iWay Integration Tools (iIT).

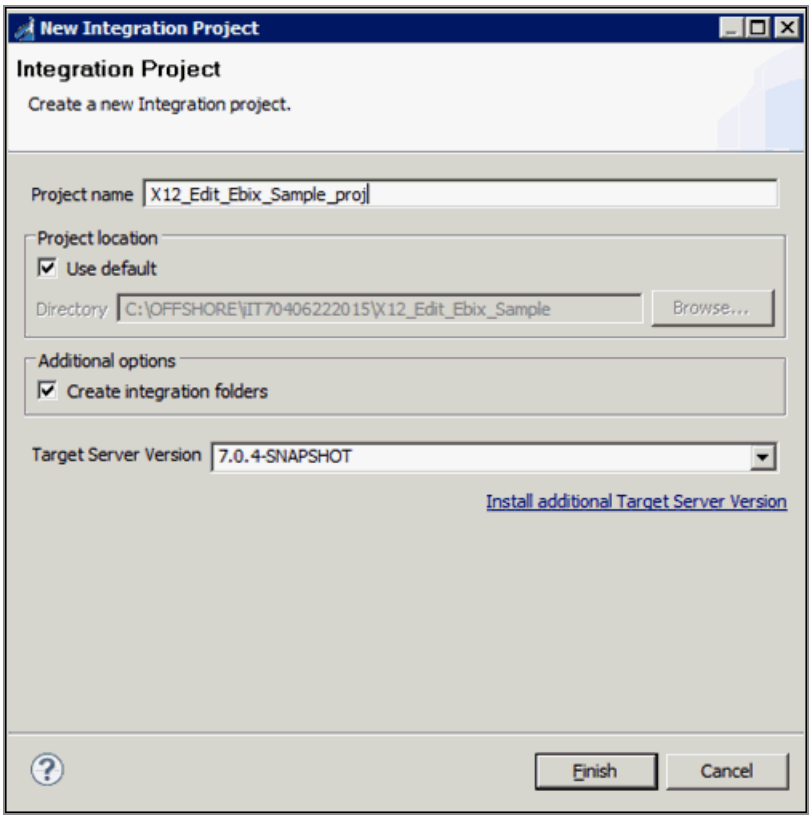
Import an Ebix

Procedure

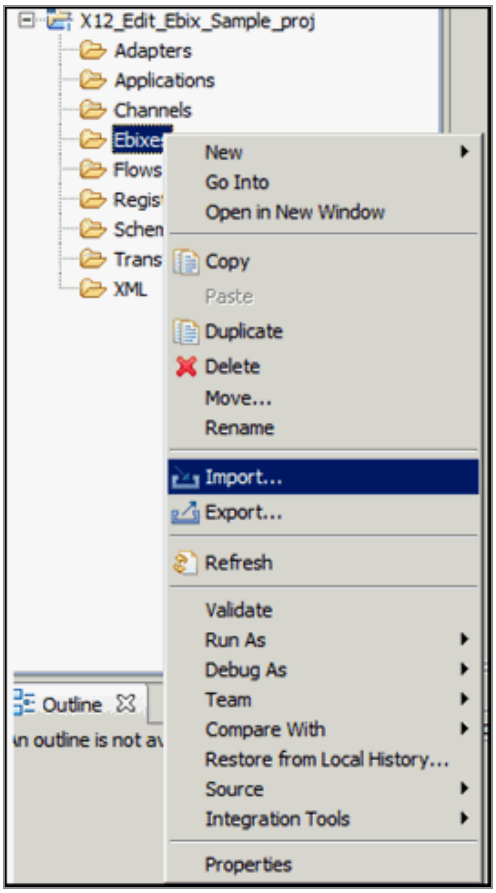
1. Start iWay Integration Tools (iIT).
2. Right-click the Integration Explorer pane, click **New**, and then select **Integration Project** from the context menu, as shown in the following image.



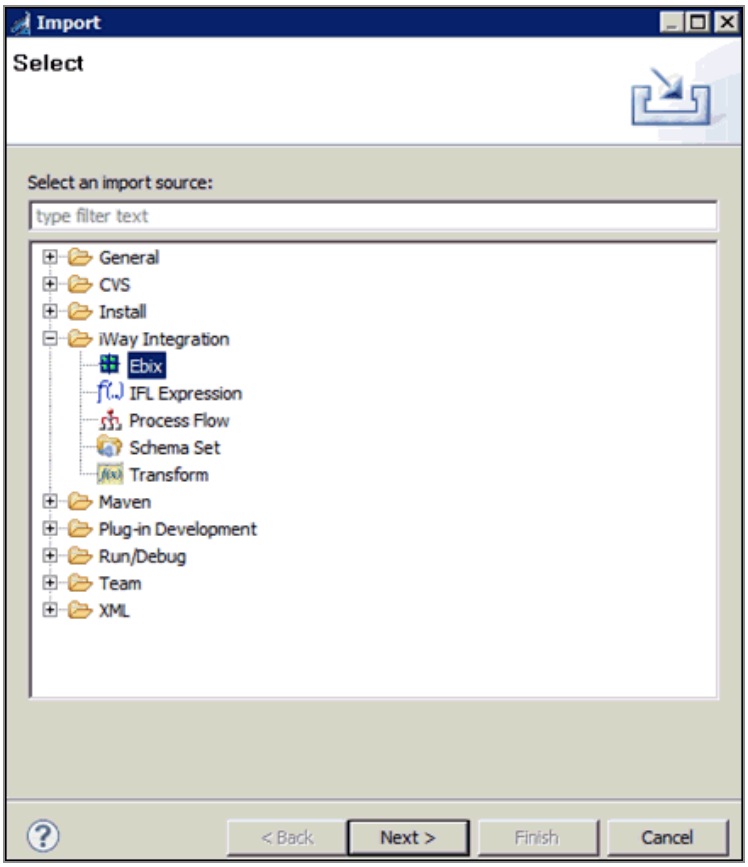
3. Enter a new Integration Project name, for example, *X12_Ebix_edit_sample_proj*, in the Project name field, and then click **Finish**, as shown in the following image.



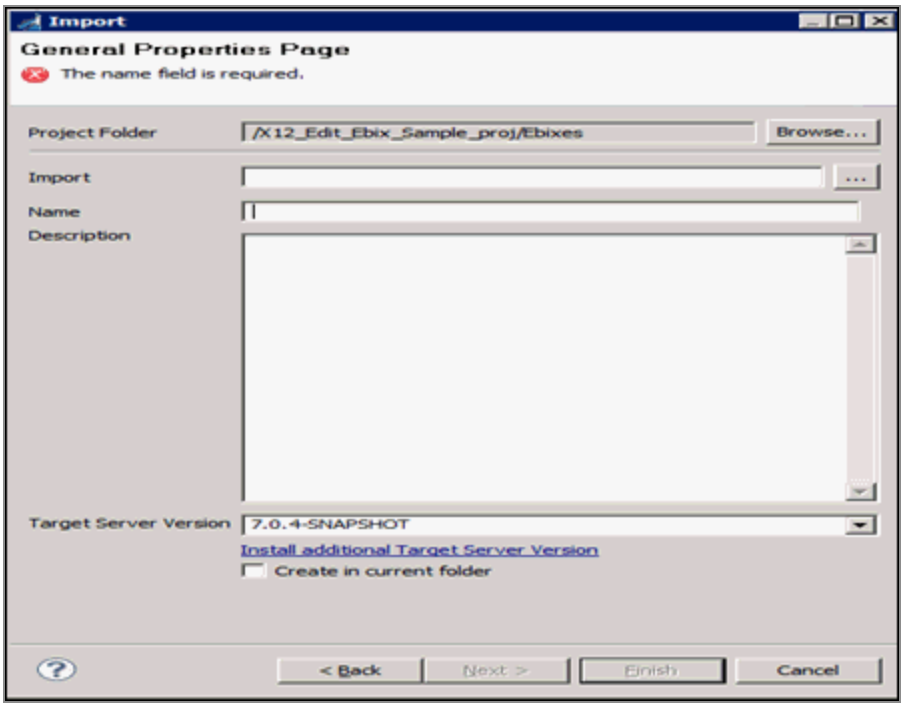
4. Right-click the Integration Explorer pane and select **Import** from the context menu, as shown in the following image.



5. In the Import wizard, expand **iWay Integration**, select **Ebix**, and then click **Next**, as shown in the following image.

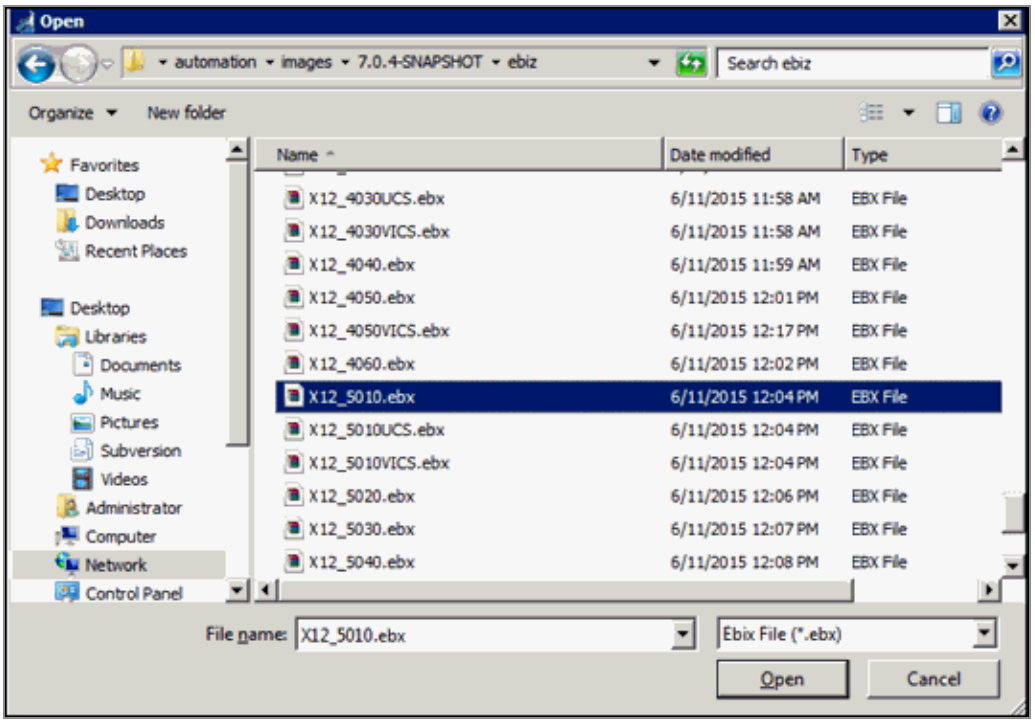


6. Click the **ellipsis** (…) button, as shown in the following image.

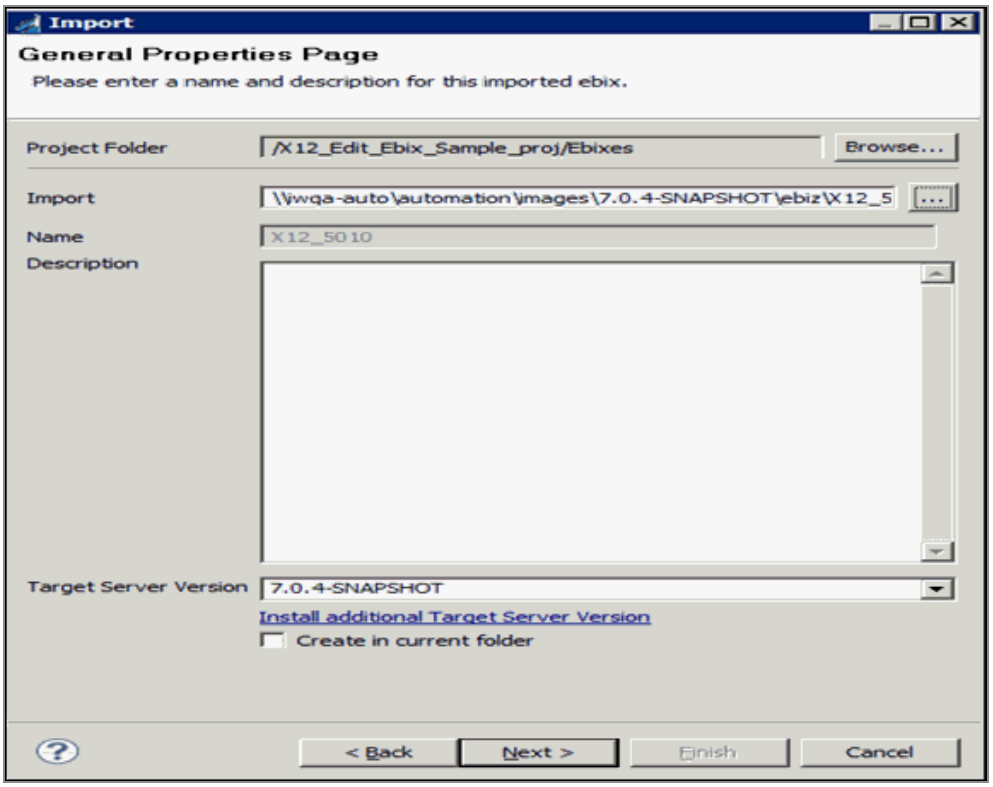


The Open dialog is displayed.

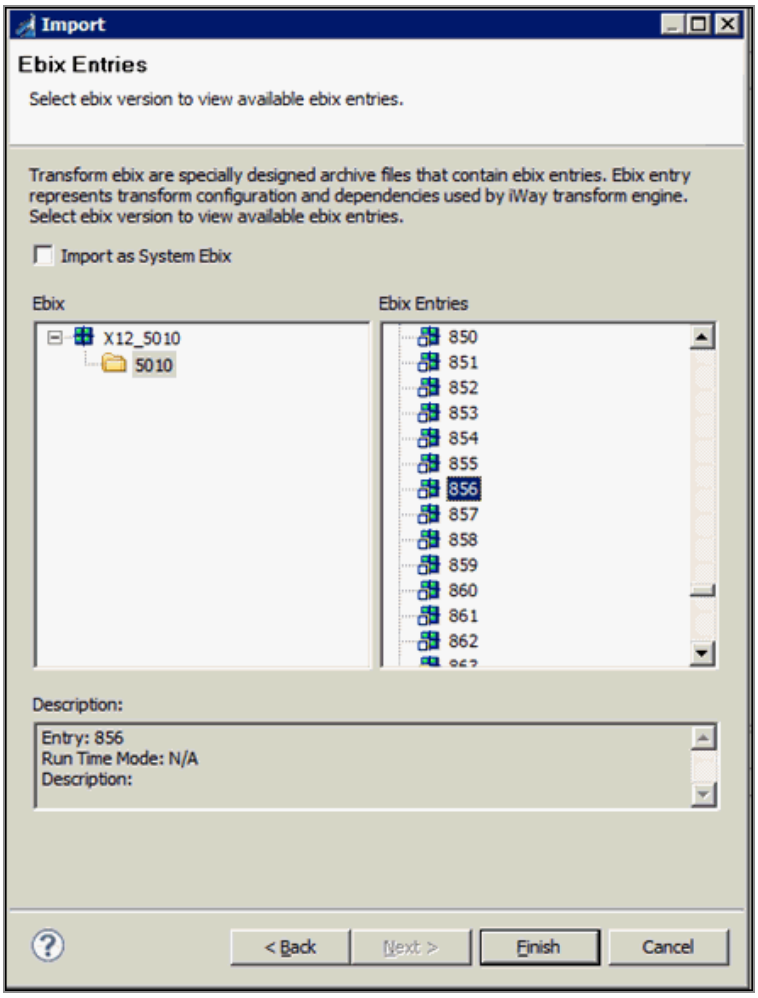
7. Select the downloaded *X12_5010.ebx* file from the physical drive location and then click **Open**, as shown in the following image.



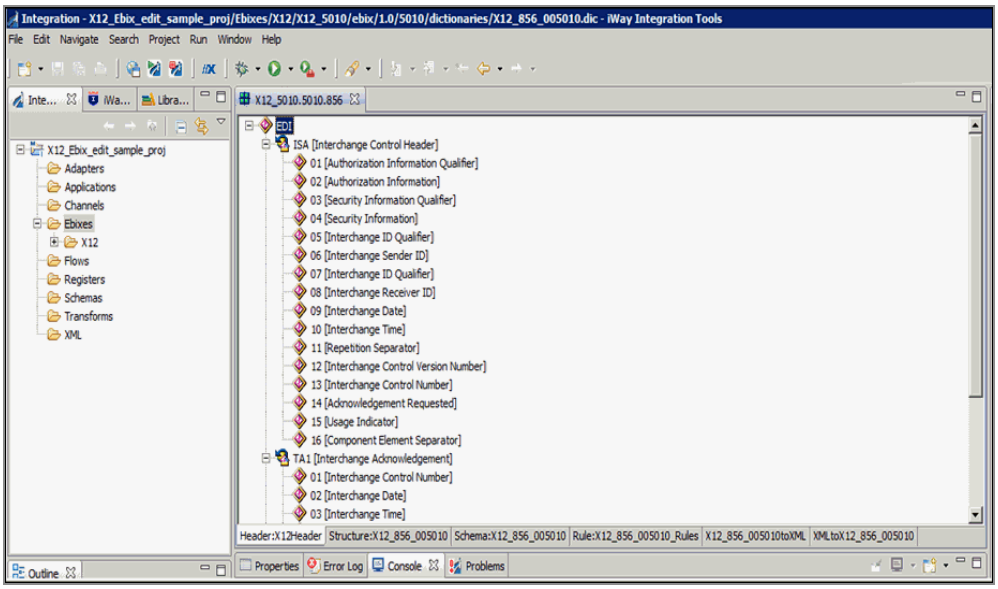
8. Click **Next**, as shown in the following image.



9. Expand **X12_5010** in the Ebix pane, click the **5010** folder, select **856** in the Ebix Entries pane, and then click **Finish**, as shown in the following image.



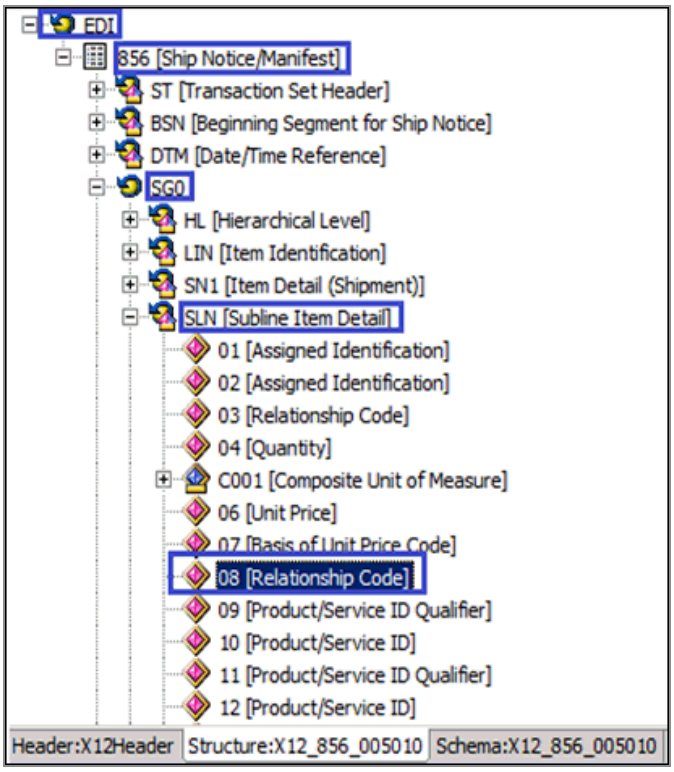
Your iIT interface should now resemble the following image:



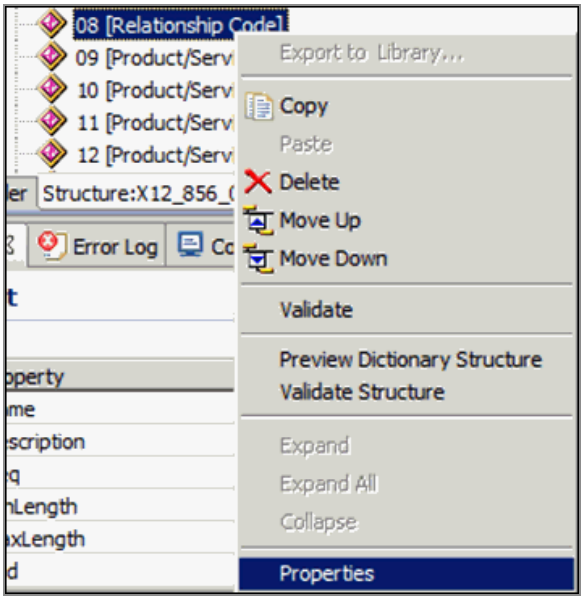
Edit an Ebix

Procedure

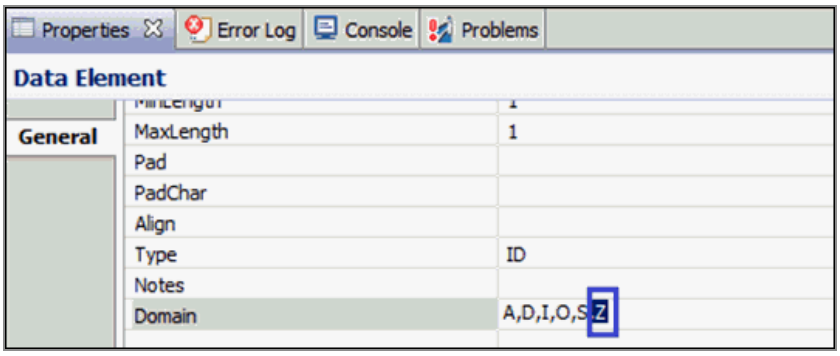
1. Click the **Structure:X12_856_005010** tab and navigate to the **08 [Relationship Code]** element by expanding **EDI**, **856 [Ship Notice/Manifest]**, **SG0**, and then **SLN [Subline Item Detail]**, as shown in the following image.



2. Right-click the **08 [Relationship Code]** composite element and then click **Properties** from context menu, as shown in the following image.

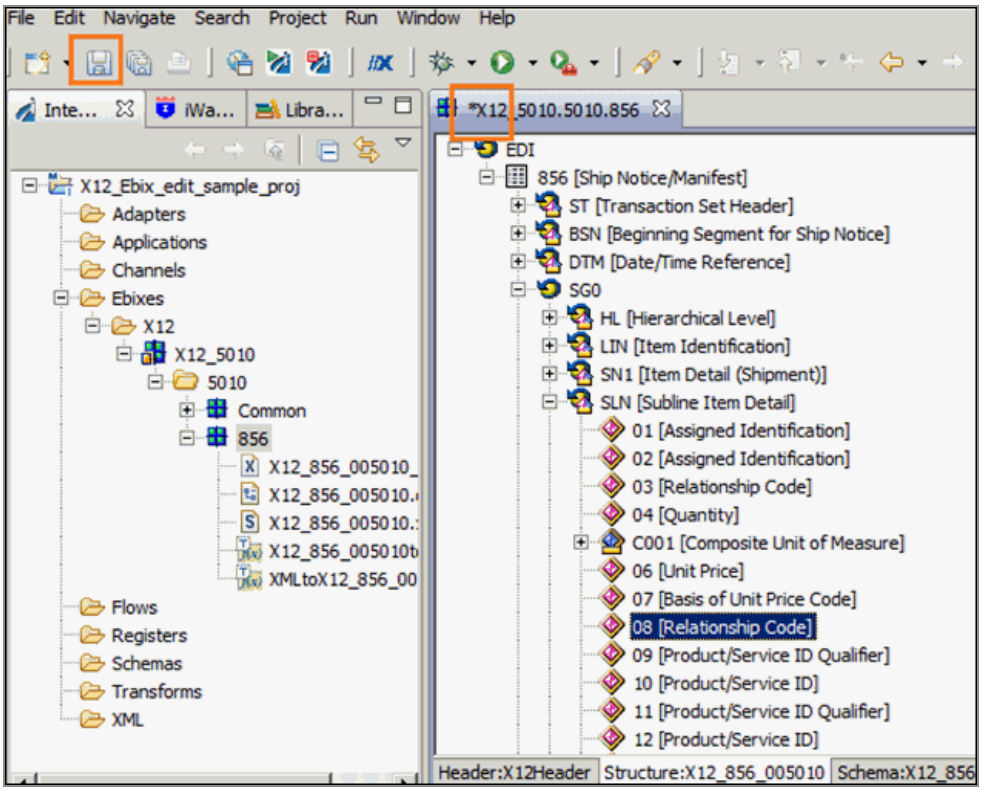


3. Scroll down to view the Domain value, and add **Z11** into the Domain value field in the properties window.

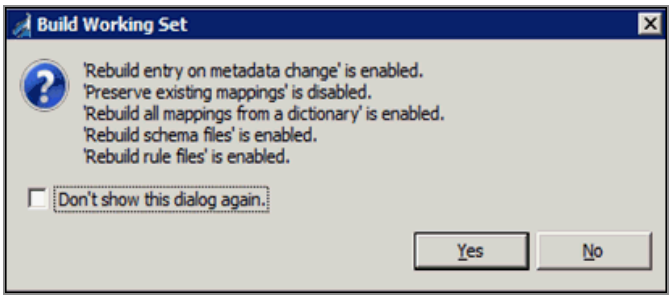


4. Save your edited Ebix by clicking the **Save** icon, which is located near the File menu. If you are using a Windows platform, you can also use the shortcut key CTRL+S to save your work.

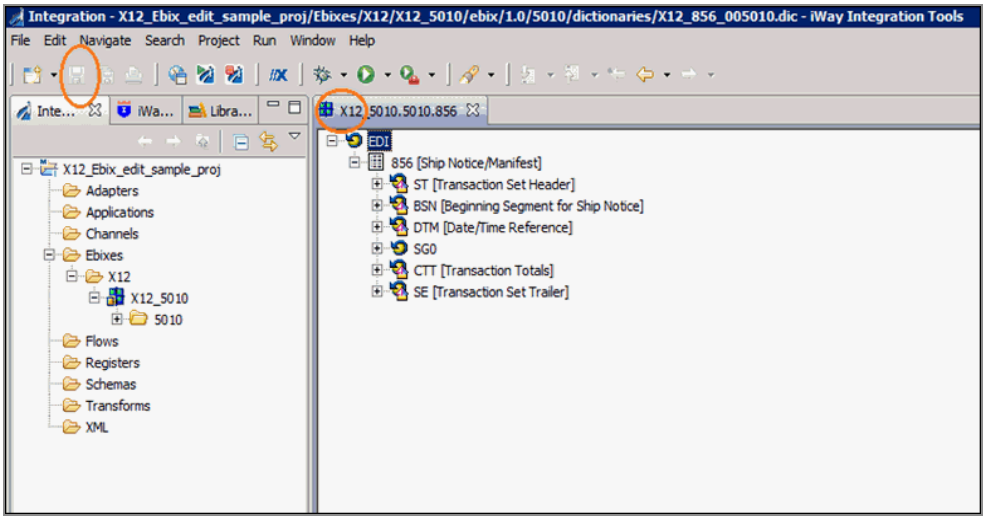
   An asterisk (*) character appears next to the file name until you have saved the edited changes, as shown in the following image.



5. Click on **Yes** to confirm your changes.

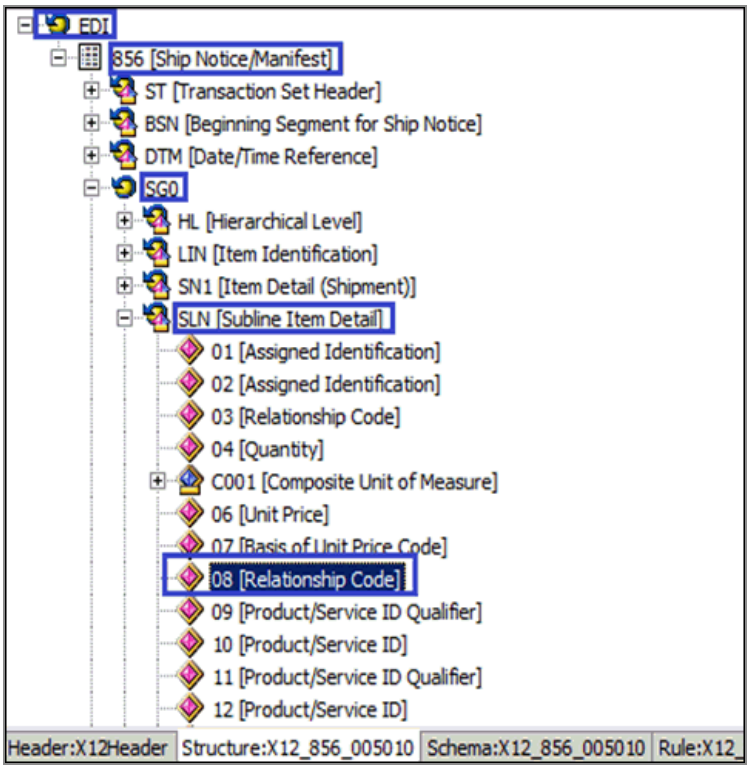


Your iIT interface should now resemble the following image:



Note: The asterisk (*) character will disappear once the edited Ebix has been saved successfully.

6. Click the **Structure:X12_856_005010** tab and navigate to the **08 [Relationship Code]** element by expanding **EDI**, **856 [Ship Notice/Manifest]**, **SG0**, and then **SLN [Subline Item Detail]**, as shown in the following image.



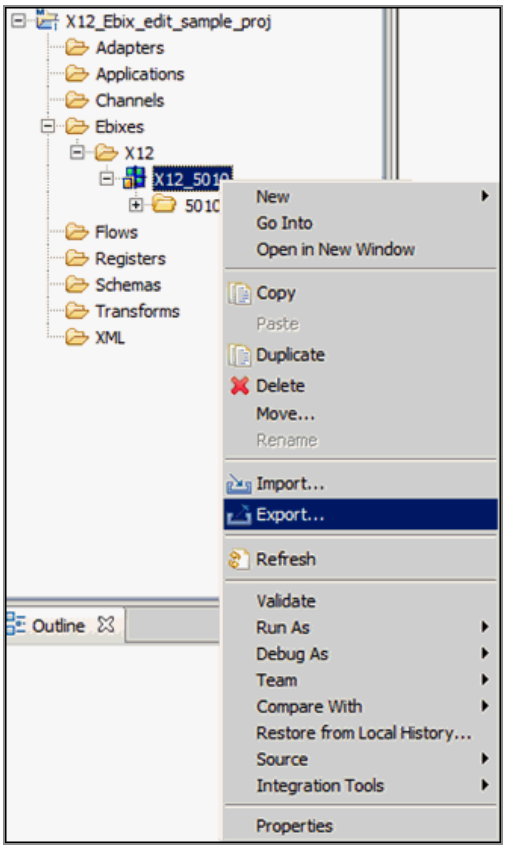
7. Repeat steps 2 - 4 in Edit an Ebix.

Export an Ebix

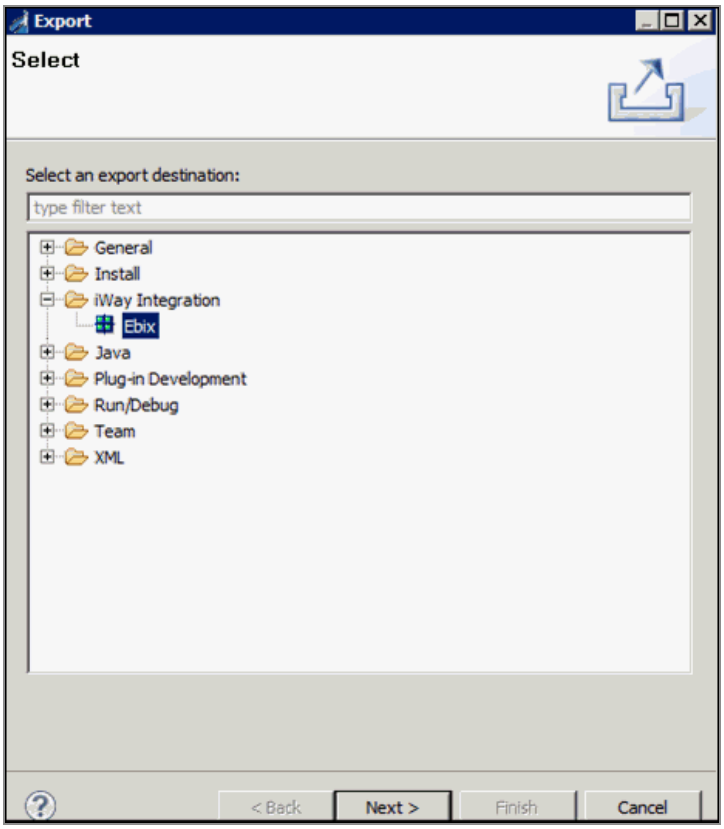
To export an Ebix:

Procedure

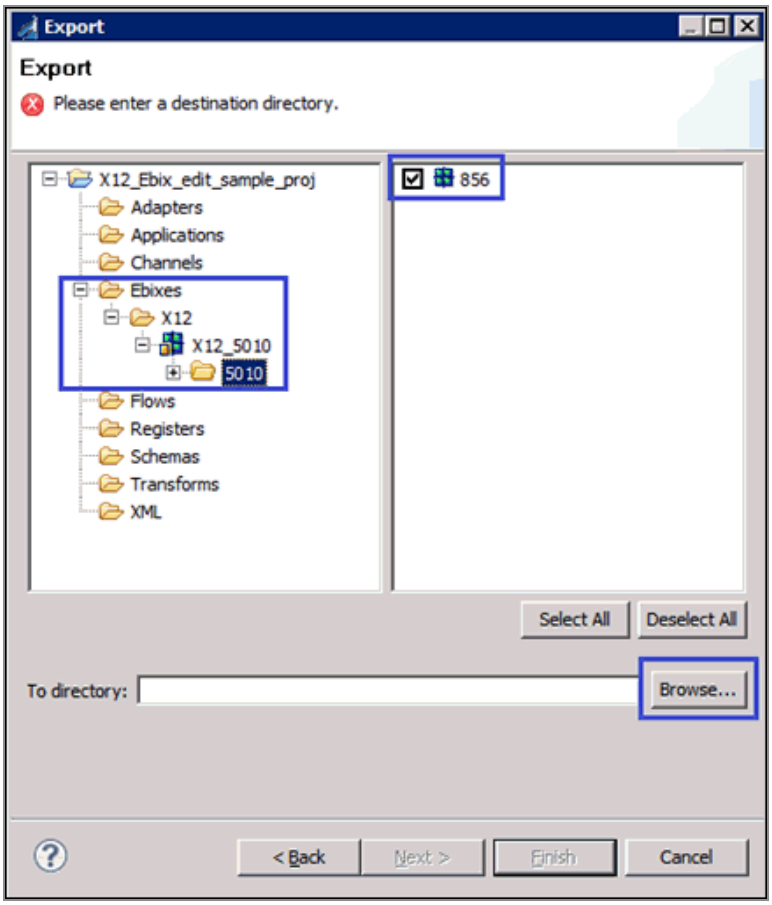
1. Right-click the **HIPAA_5010X299** Ebix from the Integration Explorer window and then select the **Export** option from the context menu, as shown in the following image.



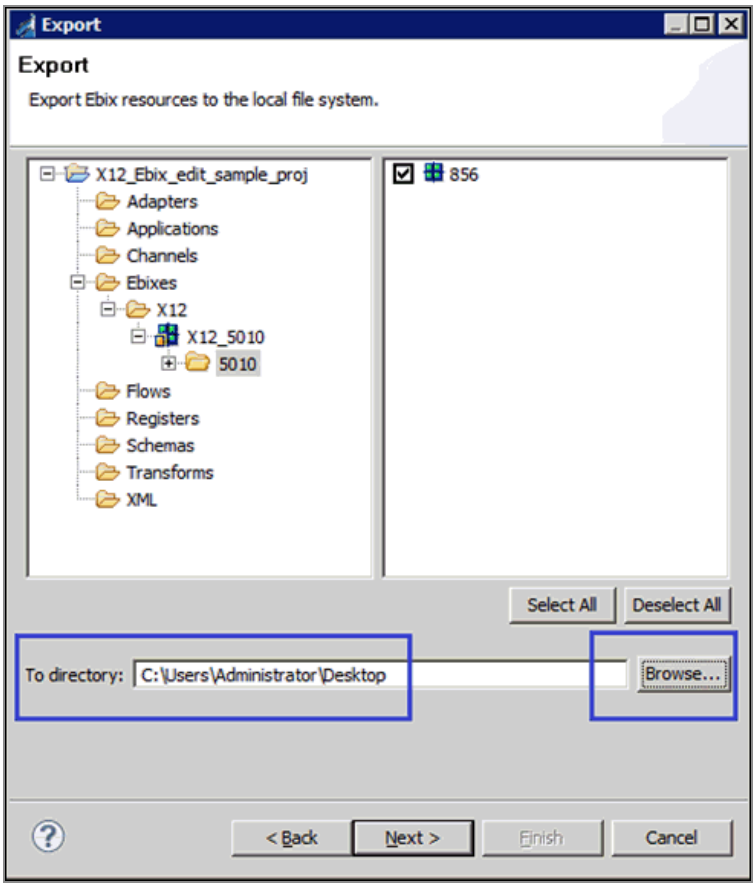
2. Expand the **iWay Integration** folder, select **Ebix**, and then click **Next**, as shown in the following image.



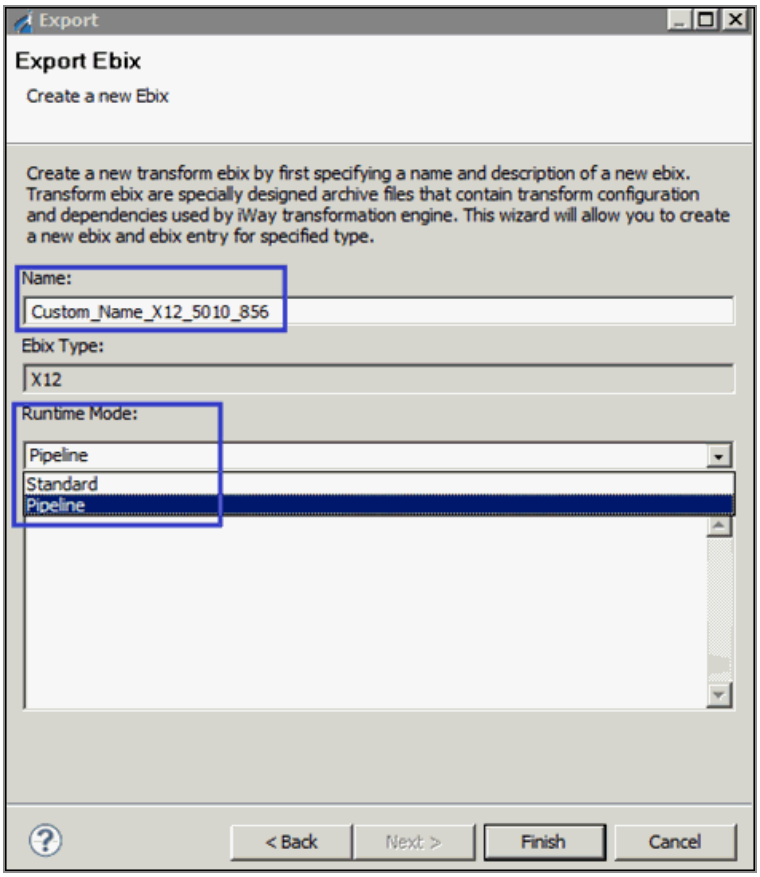
3. Expand **X12_Ebix_edit_sample_proj**, **Ebixes**, **X12**, **X12_5010**, select the **5010** folder in the left pane, and then select the **856** check box from the right pane, as shown in the following image.



4. Click **Browse** and choose a folder location to store the Ebix, and then click **Next**, as shown in the following image.



5. Provide a valid name for the Ebix in the Name field, select **Pipeline** from the Runtime Mode drop-down list, add a description (optional), and then click **Finish**, as shown in the following image.



Your exported Ebix is now available in the specified location.

Using EDI X12 Separators and Terminators

All EDI X12 documents are embedded with tokens that are separated by special characters called separators and terminators. Specifically, these special characters are used to identify:

- element separators
- sub-element separators
- segment terminators

This appendix provides a list of the separators and terminators that are allowed during the configuration of preparsers and preemitters in iWay Service Manager.

EDI X12 Separators and Terminators

Hex	Char	Hex	Char	Hex	Char
01	SOH	16	SYN	2F	/
02	STX	17	ETB	3A	:
03	ETX	18	CAN	3B	;
04	EOT	19	EM	3C	<
05	ENQ	1A	SUB	3D	=
06	ACK	1B	ESC	3E	>
07	BEL	1C	FS	3F	?
08	BS	1D	GS	40	@

Hex	Char	Hex	Char	Hex	Char
09	TAB	1E	RS	5B	[
0A	LF	1F	US	5C	\
0B	VT	21	!	5D	]
0C	FF	23	#	5E	^
0D	CR	24	$	5F	_
0E	SO	25	%	60	'
OF	SI	26	&	7B	{
10	DLE	27	'	7C	|
11	DC1	28	(	7D	}
12	DC2	29	)	7E	~
13	DC3	2A	*	7F	DEL
14	DC4	2B	+		
15	NAK	2D	-		

Using EDI X12 Special Register (SREG) Types

This section describes the Special Register (SREG) types that are created during EDI to XML transactions and 997 creation.

EDI X12 Special Register (SREG) Types

New Special Registers (SREGs) are available for EDI preparsers and EDI preemitters.

```
<variable type="USR" name="edi.transactionID" otype="0">823</variable>
<variable type="USR" name="edi.type" otype="0">X12</variable>
<variable type="USR" name="edi.version" otype="0">004010</variable>
```

These may be used to route your data by placing them in your process flow.

A new SREG (edi.ackstatus) is available for the acknowledgment agent. This SREG will contain the AK501 status from the 997 that corresponds to each XML output file. This value can be used to route error data (for example, a failed 997) from standard processing.

During EDI to XML transactions and 997 creation, the following types of SREGs are created:

- SYS (System) - These SREGs exist until you restart iWay Service Manager.
- USR/DOC - These SREGs exist throughout the life of the document.
- CFG - These SREGs are configuration related.

```
SEGMENT COUNT
```

```
1. <variable name="SEGCOUNT" type="USR">20</variable>
2. <variable name="basename" type="DOC">stephan_850_bad</variable>
3. <variable name="console-master-port" type="SYS">9999</variable>
```

```
CORRELATION ID
```

```
4. <variable name="correlid" type="USR">000001000</variable>
5. <variable name="doclocation" type="SYS">config</variable>
```

END OF STREAM FLAG

```
6. <variable name="eos" type="USR">1</variable>
7. <variable name="extension" type="DOC">x12</variable>
8. <variable name="filename" type="DOC">stephan_850_bad.x12</variable>
```

FROM PARTY

```
9. <variable name="fromparty" type="USR">NOTP           </variable>
```

GROUP CONTROL NUMBER – GE

```
10. <variable name="ge_groupctlnumber" type="USR">1000</variable>
```

NUMBER OF TRANSACTIONS – GE

```
11. <variable name="ge_numtransactions" type="USR">1</variable>
12. <variable name="ibse-port" type="CFG">9000</variable>
```

INTERCHANGE CONTROL NUMBER – IEA

```
13. <variable name="iea_interchangectlnum"
type="USR">000001000</variable>
```

VALIDATION REPORT/ACK

```
14. <variable name="iwaf.validationReport" type="USR">ISA*00*
*00*          *12*NOTP           *12*NOTP
*QAQAQA*QAQA*U*00401*000001000*0*P*&gt;
GS*FA*NOTP*NOTP*QAQAQAQA*QAQA*1000*X*004010
```

```
ST*997*0001
AK1*PO*1000
AK2*850*000000010
AK3*DTM*6**8
AK4*2**8*200100
AK5*R*5
AK9*E*1*1*1
SE*8*0001
GE*1*1000
IEA*1*000001000
</variable>
15. <variable name="iway.eos" type="DOC">true</variable>
16. <variable name="iwayconfig" type="SYS">base</variable>
17. <variable name="iwayhome" type="SYS">C:/Program
Files/iway7/</variable>
18. <variable name="iwayversion" type="SYS">8.0SM</variable>
19. <variable name="iwayworkdir" type="SYS">C:/Program
Files/iWay7/config/base</variable>
20. <variable name="locale" type="SYS">en_us</variable>
21. <variable name="name" type="SYS">EDI_XML</variable>
```

NUMBER OF FUNCTIONAL GROUPS

```
22. <variable name="numfunctionalgroups" type="USR">1</variable>
23. <variable name="parent"
type="DOC">c:\testing\edix12\input</variable>
24. <variable name="protocol" type="SYS">FILE</variable>
25. <variable name="source" type="DOC">C:\testing\edix12\input\stephan_
850_bad.x12</variable>
```

SPLIT COUNT

```
26. <variable name="splitcount" type="USR">1</variable>
27. <variable name="tid" type="DOC">EDI_XML-FILE-W.EDI_XML.1_
20080605152319600Z</variable>
```

TRANSACTION ID

```
28. <variable name=”edi.transactionID” type=”USR”>850</variable>
```

VERSION

```
29. <variable name=”edi.version” type=”USR”>004010</variable>
```

Sample EDI X12 Files

This appendix includes a sample Electronic Data Interchange (EDI) 4010 850 Purchase Order, 4010 810 Invoice, and 4010 856 Advanced Ship Notice. These are the key EDI documents in wholesale distribution.

For more information on obtaining EDI X12 sample files for testing purposes, see Downloading and Extracting EDI X12 User Samples.

Sample EDI 4010 850 Purchase Order

The following is a sample EDI 4010 850 Purchase Order.

```
ISA*00*          *00*          *12*NOTP           *12*NOTP
*080501*1700*U*00401*000001000*0*P*>
GS*PO*NOTP*NOTP*20080501*1700*1000*X*004010
ST*850*000000010
BEG*00*SA*08292243254**20010501*610385388
REF*DP*030
REF*PS*
ITD*14*3*2**45**46
DTM*001*20010510
PKG*F*68***PALLET, SHRINKWRAP 48W X 40D X 45H
PKG*F*66***REGULAR
TD5*A*92*P3**SEE ROUTING GUIDE FOR ROUTING
N1*ST*RETAIL STORE*9*0001234567890
N3*123 ANYWHERE AVENUE
N4*CITY*ST*12345
PO1*1*120*EA*9.25*TE*CB*(12) 0-083628-838*PR*RO*VN*ABA18783
PO1*2*220*EA*13.7 9*TE*CB*(69) 0-093 83 7-991*PR*RO*VN*RUP83112
PO1*3*126*EA*10.9 9*TE*CB*(71) 0-099172-837*PR*RO*VN*CPR19293
PO1*4*76*EA*4.35*TE*CB*(71) 0-012110-737*PR*RO*VN*PIW28173
PO1*5*72*EA*7.5*TE*CB*(71) 0-0848 88-9 75*PR*RO*VN*JBM1938 7
PO1*6*696*EA*9.55*TE*CB*(71) 0-003 922-121*PR*RO*VN*IUI19283
CTT*6
SE*20*000000010
GE*1*1000
IEA*1*000001000
```

Sample EDI 4010 810 Invoice

The following is a sample EDI 4010 810 Invoice.

```
ISA*00*          *01*          *ZZ*NOTP           *ZZ*NOTP
*050108*0954*U*00501*000000001*0*P*>
GS*IN*NOTP*NOTP*20050108*0954*1*X*004010
ST*810*0001
BIG*20021119*184*20021015*BMB
REF*IA*040682
N1*BT*WALGREEN*92*0000
ITD*02**1.000**30**31*****1% 30 NET 31
FOB*CC
PID*S**VI*FL
IT1*0001*267*CA*53.52**IN*859067
PID*F*08*VI**BARBIE SING W/ME DISC GRL CD PLYR
TDS*1421839*1428984
CAD*T***CFWY*CONSOLIDATED FREIGHTWAYS
SAC*A*D240***7145**********FREIGHT CHARGE
ISS*267*CA
CTT*1
SE*15*0001
GE*1*1
IEA*1*000000001
```

Sample EDI 4010 856 Advanced Ship Notice

The following is a sample EDI 4010 856 Advanced Ship Notice.

```
ISA*00*          *00*          *ZZ*NOTP           *ZZ*NOTP
*080105*1026*U*00501*100000001*0*P*:
GS*NOTP*NOTP*20080105*1026*1*X*004010
ST*856*0001
BSN*00*PC123456*20071205*1026*0004
DTM*067*20070717
HL*1**S
TD1******A3*5.750*EA*1*1N
TD5**S*DHL
REF*BM*PC123456
N1*SF*ACME PHARMA CO
N1*ST*DISTRIBUTION CENTER*92*0001
HL*2*1*O
```

```
PRF*PWS6***20080103
HL*3*2*P
MAN*GM*00007287900000256222
HL*4*3*I
LIN*10*UP*72879096026*LT*804813-5     50 Safety Pins*CH*CN
CTT*4
SE*17*0001
GE*1*1
IEA*1*100000001
```

Tutorial: Mapping an IDOC to an Invoice Document (810)

This topic provides a tutorial that demonstrates how to map an IDOC to an Invoice Document (810) using iWay Integration Tools (iIT).

Note: For your convenience, the 5010_810.zip file is attached to this PDF, which contains sample files that can be used with this tutorial. For PDF-compatibility purposes, the file extension of the 5010_810.zip file is temporarily renamed to .zap. After saving this file to your system, you must rename this extension back to .zip.

EDI X12 Invoice Document Mapping Tutorial Overview

The X12 transaction set contains the format and establishes the data contents of the Invoice document (810) for use within the context of an Electronic Data Interchange (EDI) environment. This transaction set can be used to provide for customary and established business and industry practice relative to the billing for goods and services provided.

iWay Integration Tools (iIT) provides a rule-based data transformation tool that converts an input document of one data format to an output document of another data format or structure. The easy-to use graphical user interface and function tool set facilitate the design of transform projects that are specific to your requirements.

This tutorial guides you through the following steps that are required to map a sample IDoc in XML format to an XML schema document.

- Creating a New Transform Project
- Mapping the Control Segments
- Mapping the Header Section
- Mapping the ITEM Detail
- Mapping the Invoice Summary Section

- Testing the Transform Project

The XML output data that is returned by this transformation can be used as an input document for outbound processing (XML to EDI) in iWay Service Manager.

Creating a New Transform Project

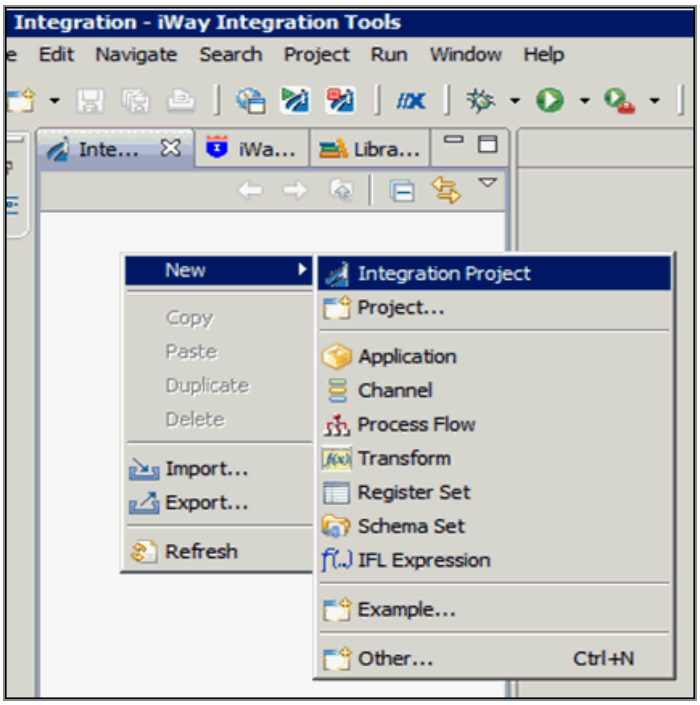
This section describes how to create a new Transform project.

Create a New Transform Project

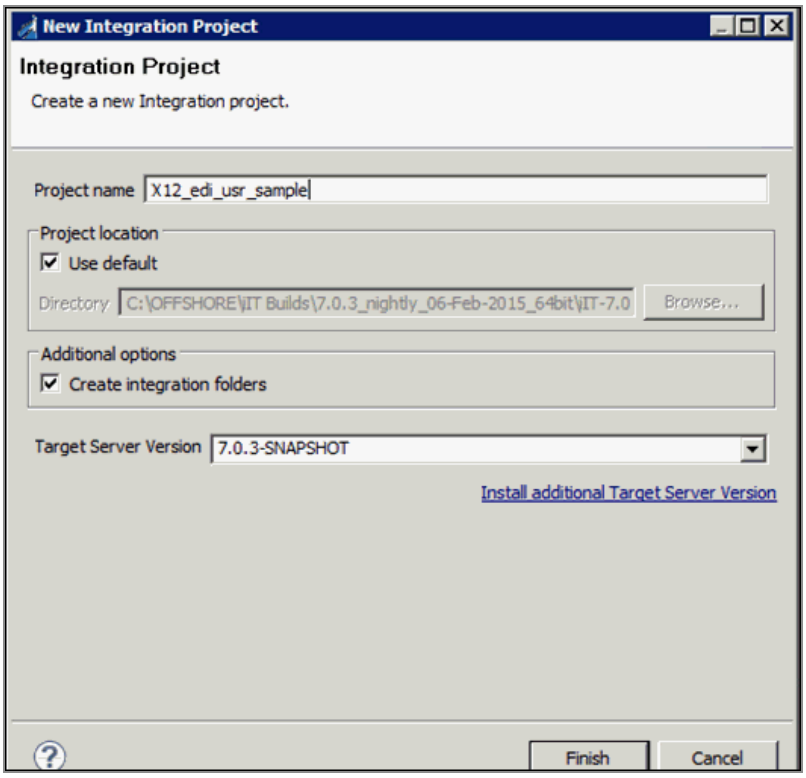
To create a new Transform project:

Procedure

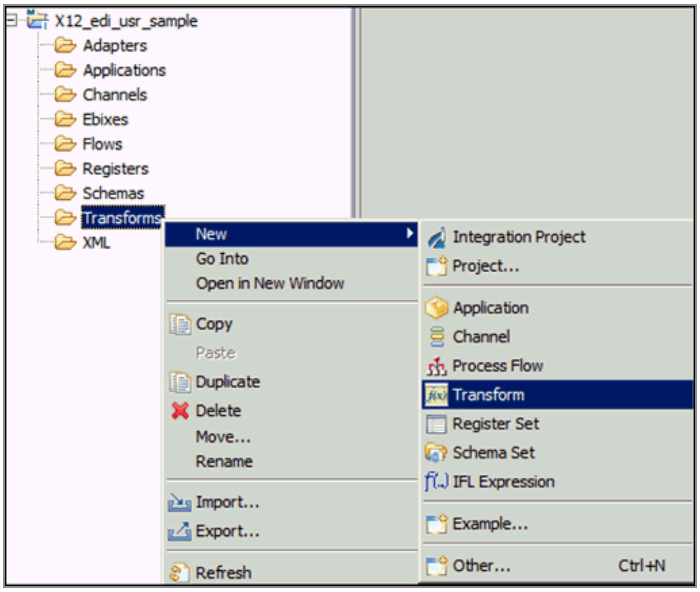
1. Open iWay Integration Tools (iIT).
2. Right-click on the Integration Explorer tab, select **New**, and then click **Integration Project**, as shown in the following image.



The New Integration Project dialog box opens as shown in the following image.

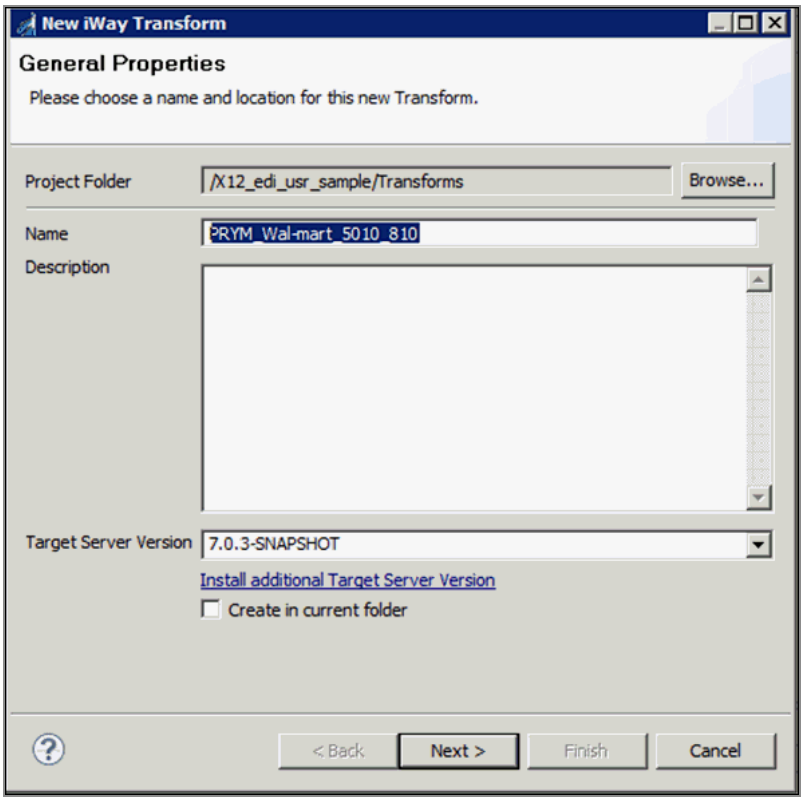


3. In the Project name field, type a name for your new project, and click **Finish**.
4. Right-click the **Transforms** folder, select **New**, and then click **Transform**.



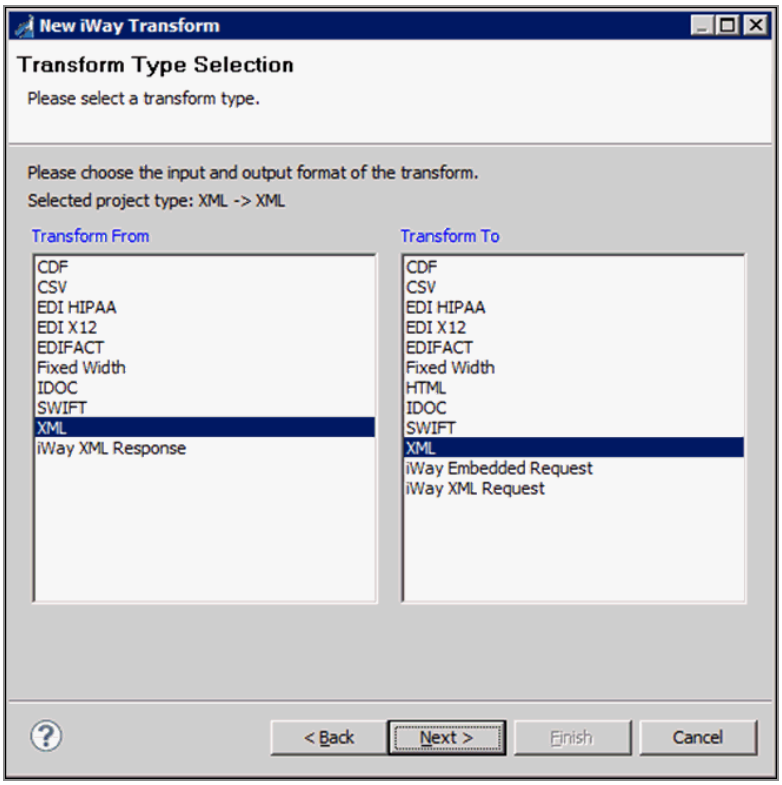
5. In the Name field, type a name for your new transform project, for example, *PRYM_*

Wal-mart_5010_810.



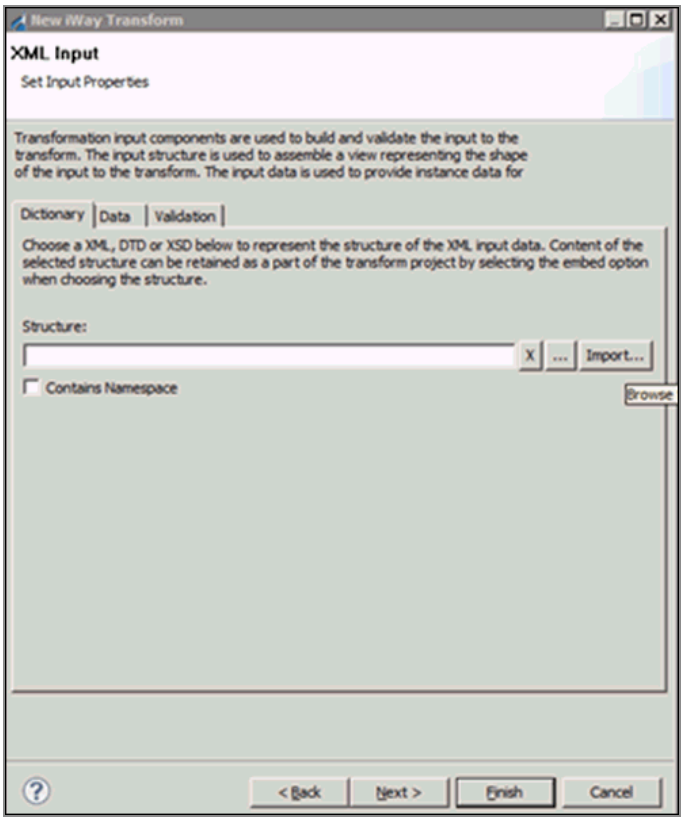
6. In the Description field, type a project description (optional).
7. Click **Next**.

   The Transform Type Selection dialog box opens as shown in the following image.

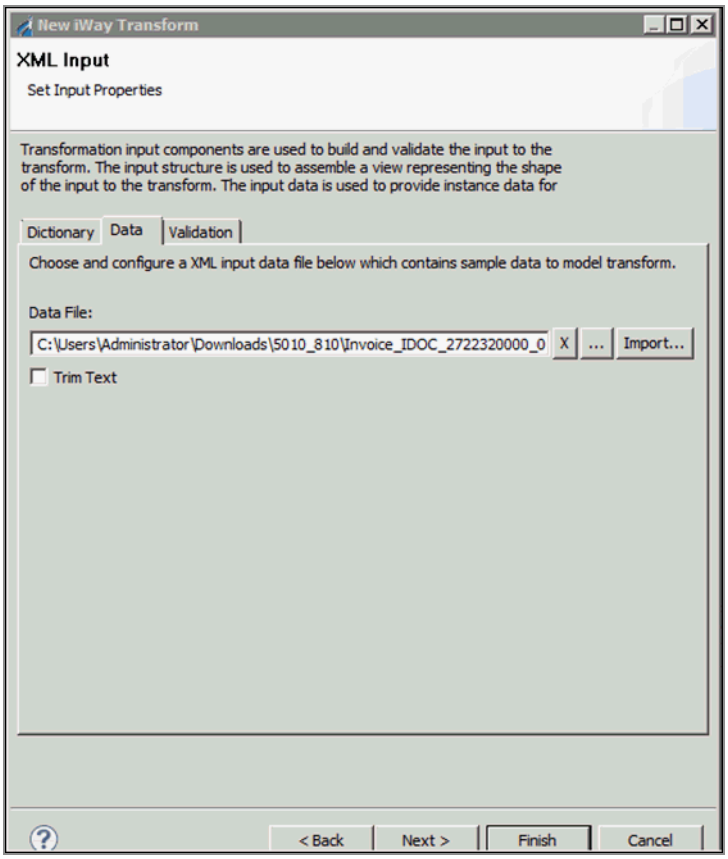


8. From the list in the Transform From pane, select the format of your input, for example, **XML**.
9. From the list in the Transform To pane, select the format of your output data, for example, **XML**.
10. Click **Next**.

    The Transform Project Wizard - XML Input properties dialog box opens with the Dictionary tab active as shown in the following image.

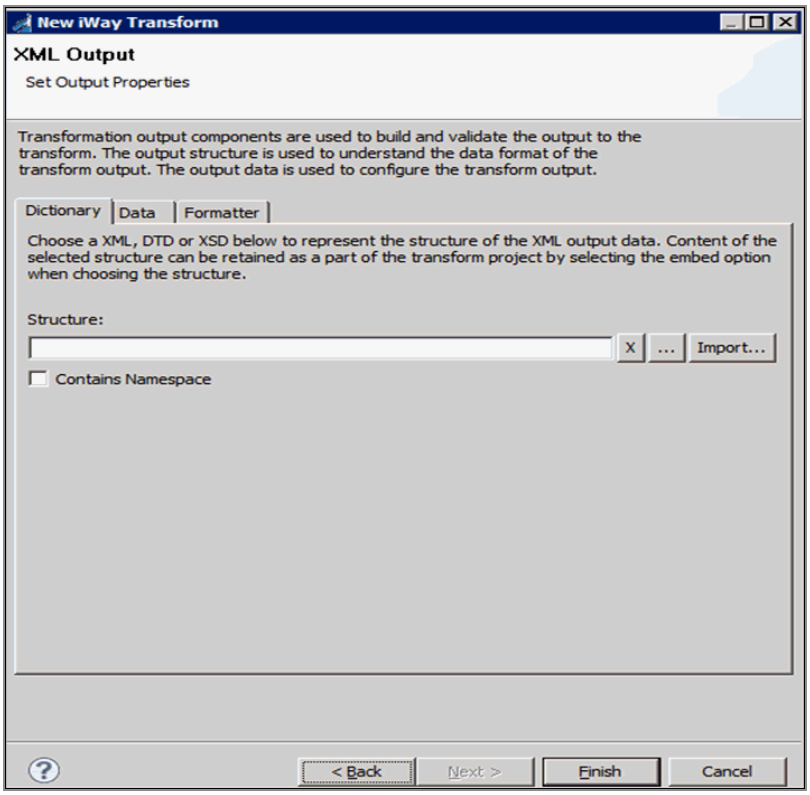


11. In the Structure field, enter the location of the sample IDoc file in XML format, which will be used as the dictionary, or click the **Import** button to bring a sample IDoc file to the transform project.
12. Click the **Data** tab.
13. Browse to the location of the sample IDoc file in XML format, which will be used as the input data file.

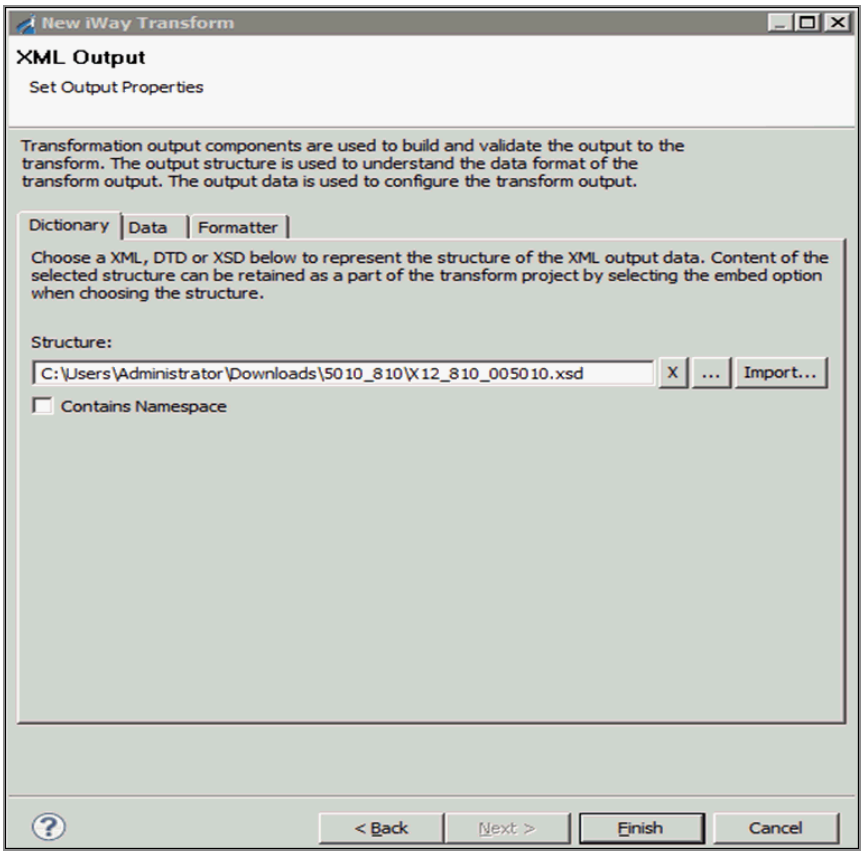


14. Click **Next**.

    The Transform Project Wizard - XML Output properties dialog box opens.

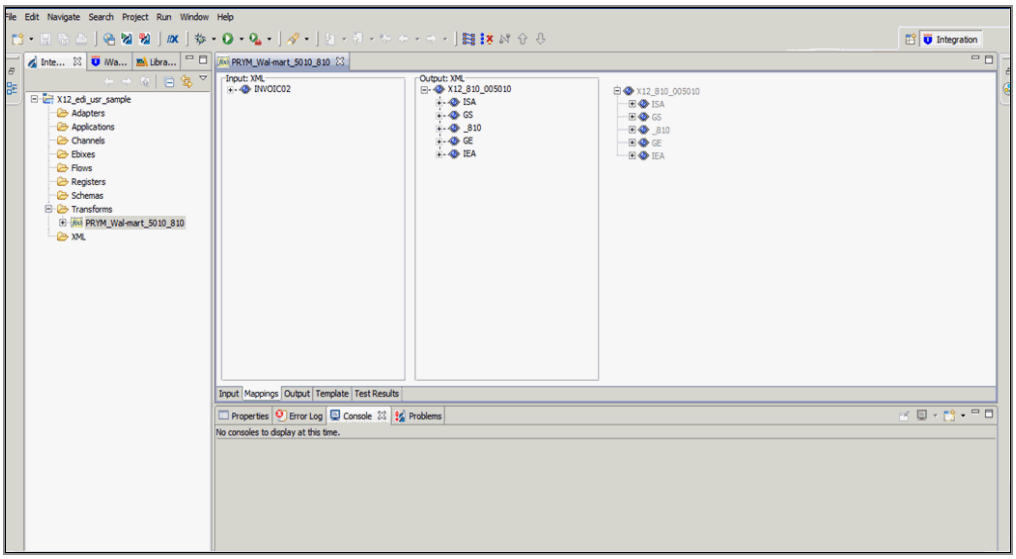


15. In the Structure field, enter the location of the XML schema document (.XSD file) that represents the EDI Invoice document (810).



16. Click **Finish**.

    The Transform Project Wizard closes. Your new Transform project is displayed in the iIT project workspace.



Understanding EDI Invoice Mapping

The EDI invoice is comprised of a header, detail lines and the trailer sections. The header contains general information about the invoice such as the invoice number, invoice date and purchase order number. The invoice also specifies the Currency used in the transaction. Other sections are parties involved in the transaction, customer information, such as the Bill-to address and the remit to information.

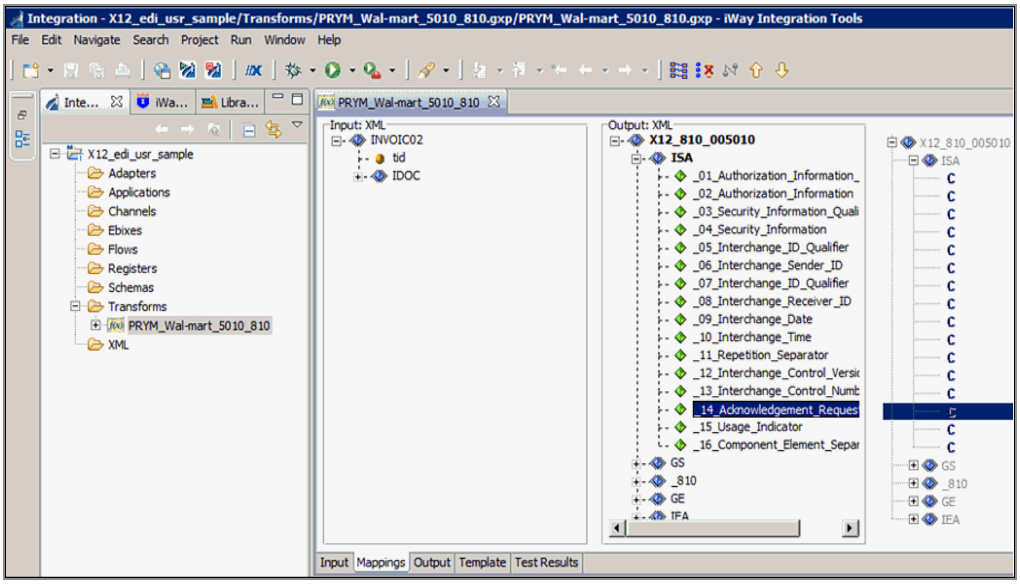
Item details include the item information (SKU and UPC/EAN) as well as the quantities billed and shipped.

The summary section of an invoice includes the total amount billed, the discounts allowed and the charges and allowances applied. As in all EDI transactions the invoice includes a Transaction Totals segment that contains control numbers to ensure there were no records lost in transformation from Internal (IDoc) format to EDI.

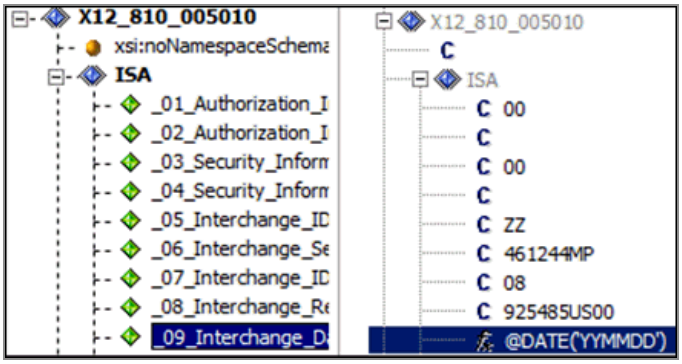
Mapping the Control Segments

The ISA is the first segment in an X12 EDI document. The ISA contains Sender and Receiver information, which supports the routing and transfer of the data. A primary function of the ISA is to contain the ISA control number which should match the IEA control number to verify the receiver has received a complete transmission.

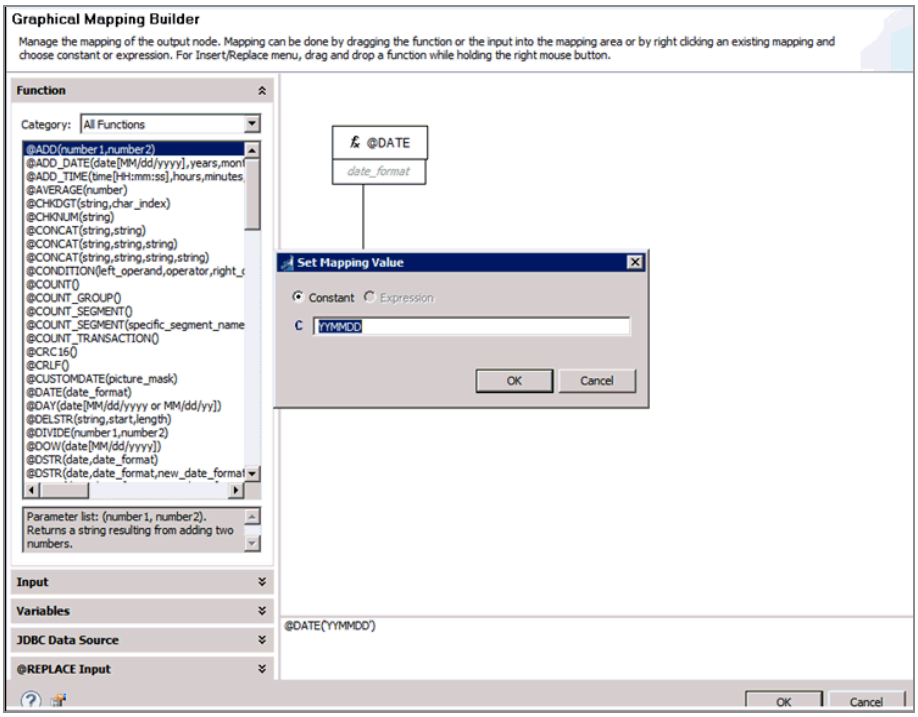
Most of the elements in the ISA are constants. One way to enter constants is to display the segment, double-click on the data entry area and enter the desired value.



ISA09

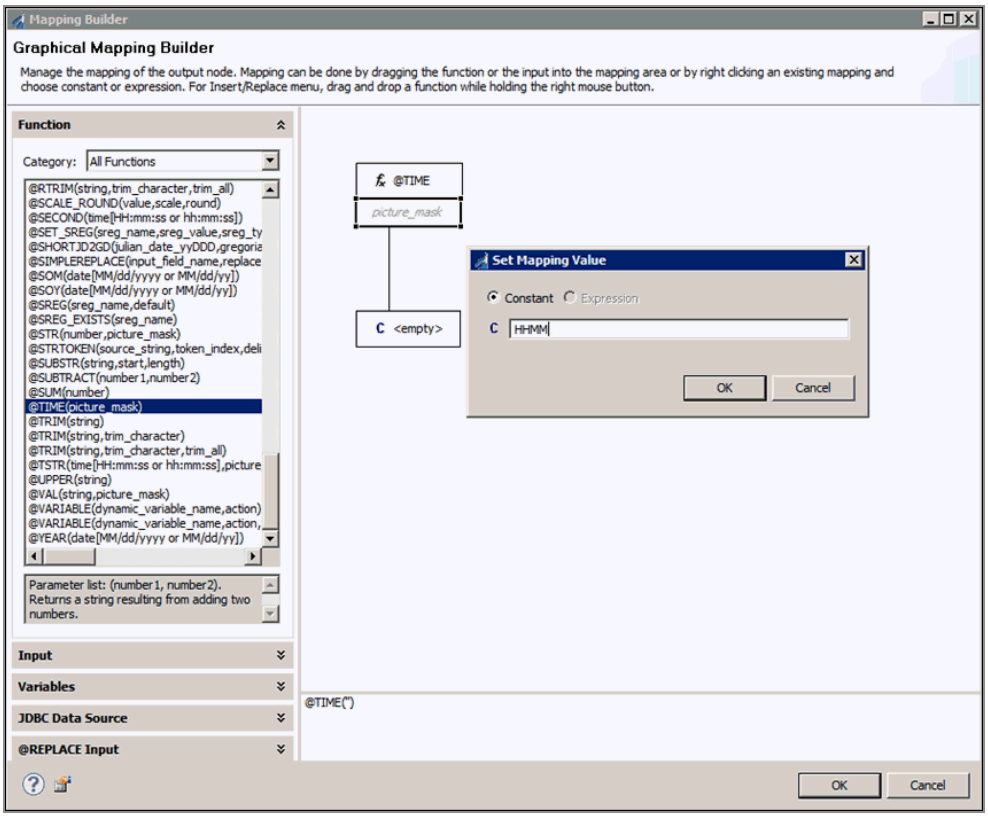


The ISA09 element is an exception to the statement about constants. This element is the date the transmission is created. In this case, select the @DATE function. The parameter for the date function is the Date_Format. The ISA09 is a 6 digit date using the YYMMDD format. Enter the Date_Format in the box under the @DATE function by double-clicking the parameter box. Another box opens, which will allow you to enter a Constant or Expression. Select Constant, enter the value YYMMDD, and click OK, then click OK again in the Graphical Mapping Builder.



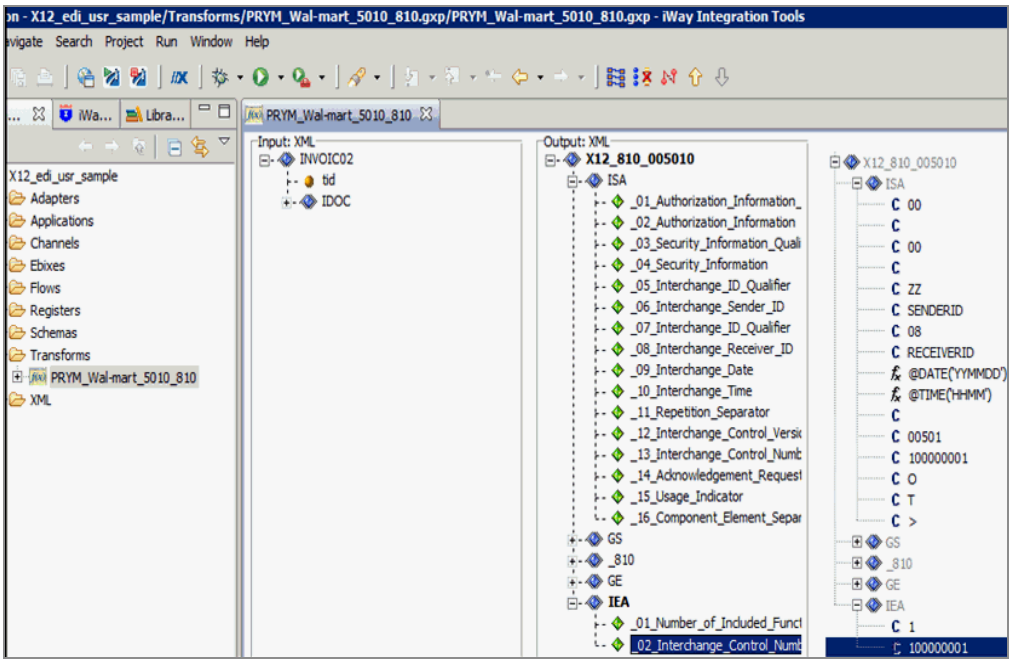
ISA10

This ISA10 element represents the time that the transmission is created. In this case select the @TIME function. The parameter for the @TIME function is the Picture_mask. The ISA10 is a 4 digit time using the HHMM format. Enter the Picture_mask in the box under the @TIME function by double-clicking the parameter box. Another box opens, which will allow you to enter a Constant or Expression. Select Constant, enter the value HHMM, click OK, then click the OK button again in the Graphical Mapping Builder.



Mapping ISA and IEA

You are now ready to map output fields. Since a Trading Partner is not used for this exercise, the envelope values need to be hardcoded. You can double-click on the line to the right of the split bar to enter constants, or click the button with the ellipsis.



Enter the values that are listed in the following table:

Field	Value
ISA01	00
ISA03	00
ISA05	ZZ
ISA06	SENDERID
ISA07	ZZ
ISA08	RECEIVERID
ISA09	@DATE('YYMMDD')
ISA10	@TIME('HHMM')
ISA11	:

Field	Value
ISA12	00501
ISA13	100000001
ISA14	0
ISA15	T
ISA16	>
IEA01	1
IEA02	100000001

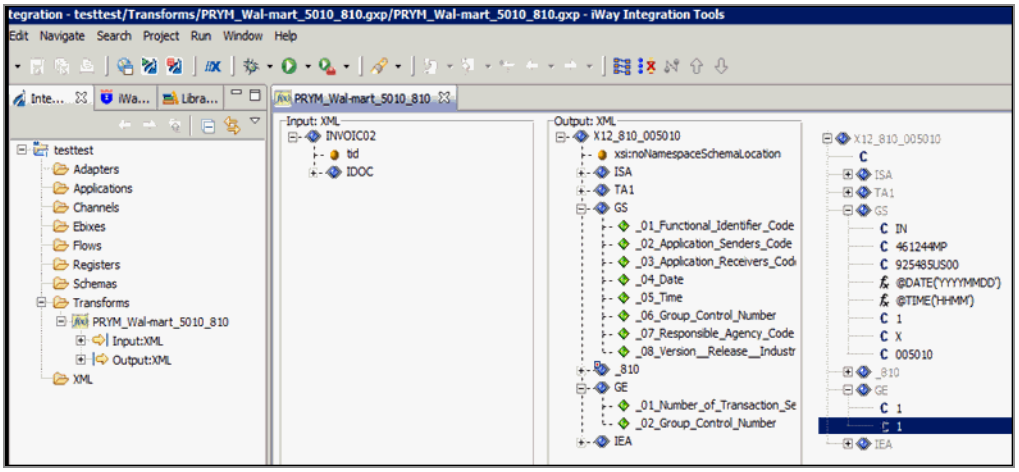
Mapping GS and GE

Enter the values that are listed in the following table:

Field	Value
GS01	IN
GS02	SENDERID
GS03	RECEIVERID
GS04	@DATE('YYYYMMDD')
GS05	@TIME('HHMM')
GS06	1
GS07	X

Field	Value
GS08	005010
GE01	1
GE02	1

Your iIT interface should resemble the following:

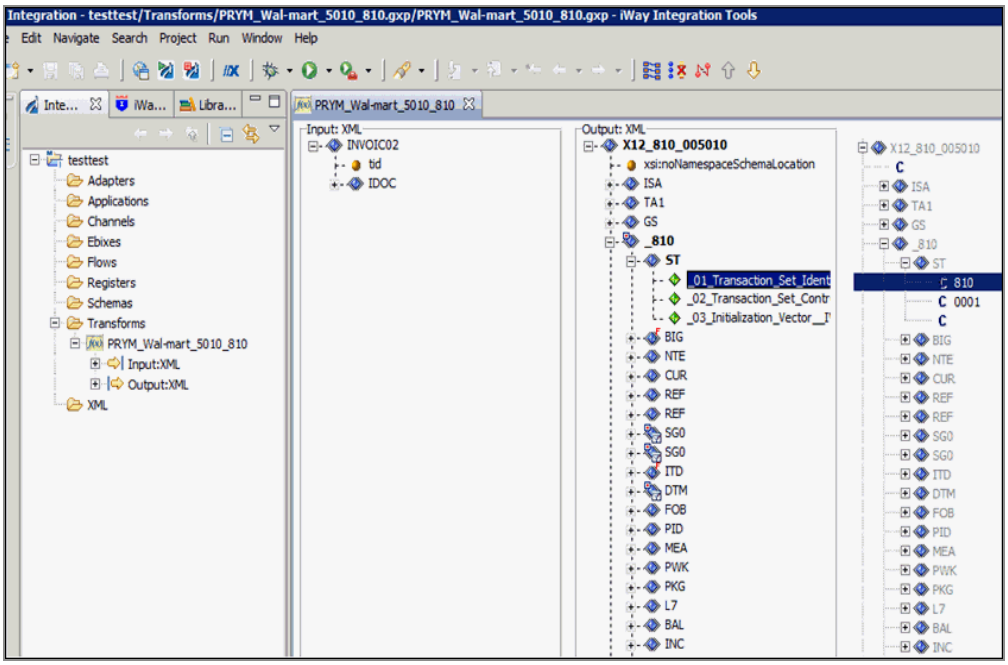


Mapping ST and SE

Enter the values that are listed in the following table:

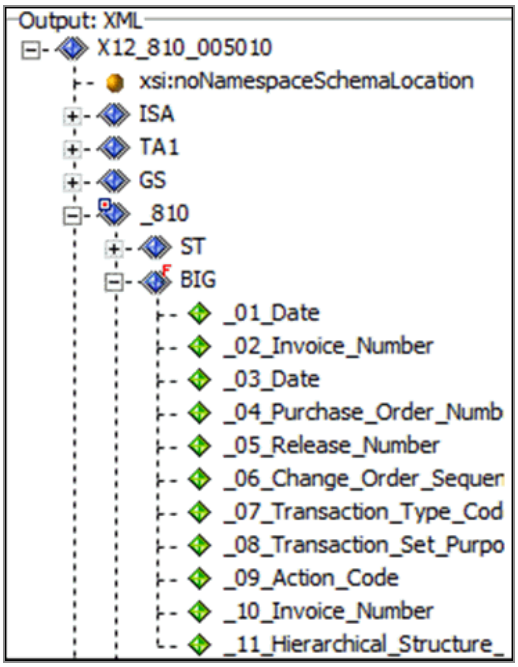
Field	Value
ST01	810
ST02	0001
SE01	1
SE02	0001

Your iIT interface should resemble the following:



Mapping the Header Section

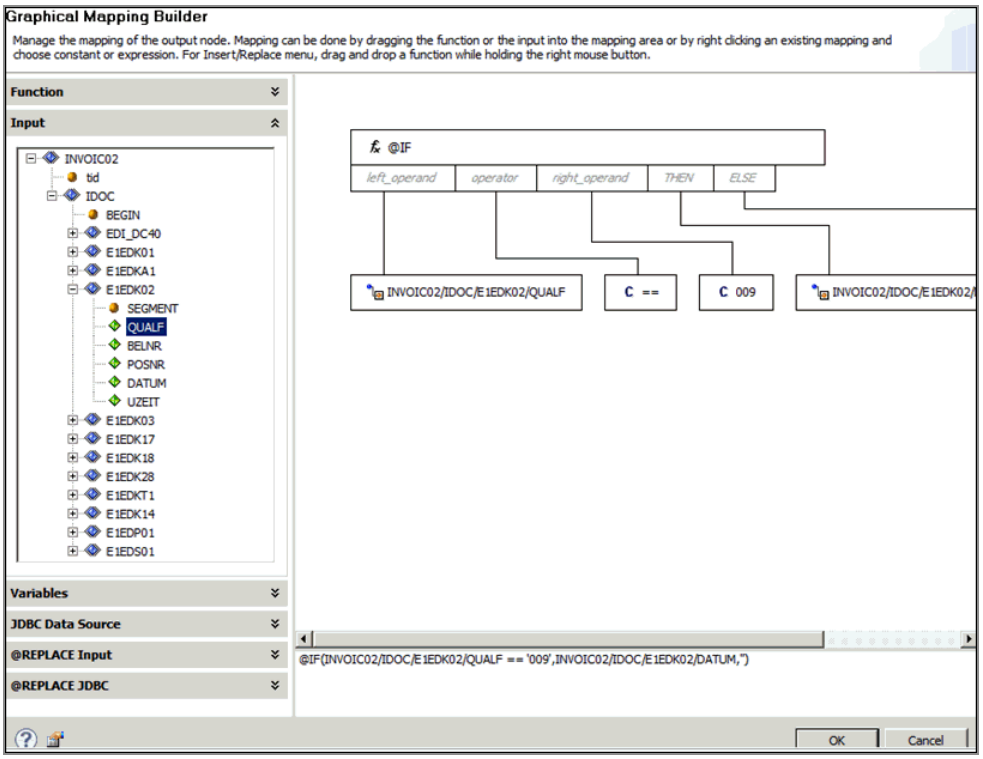
The BIG segment represents the beginning segment of the invoice.



Expand the 810 node and the BIG segment node. The BIG segment contains four elements, which will be mapped.

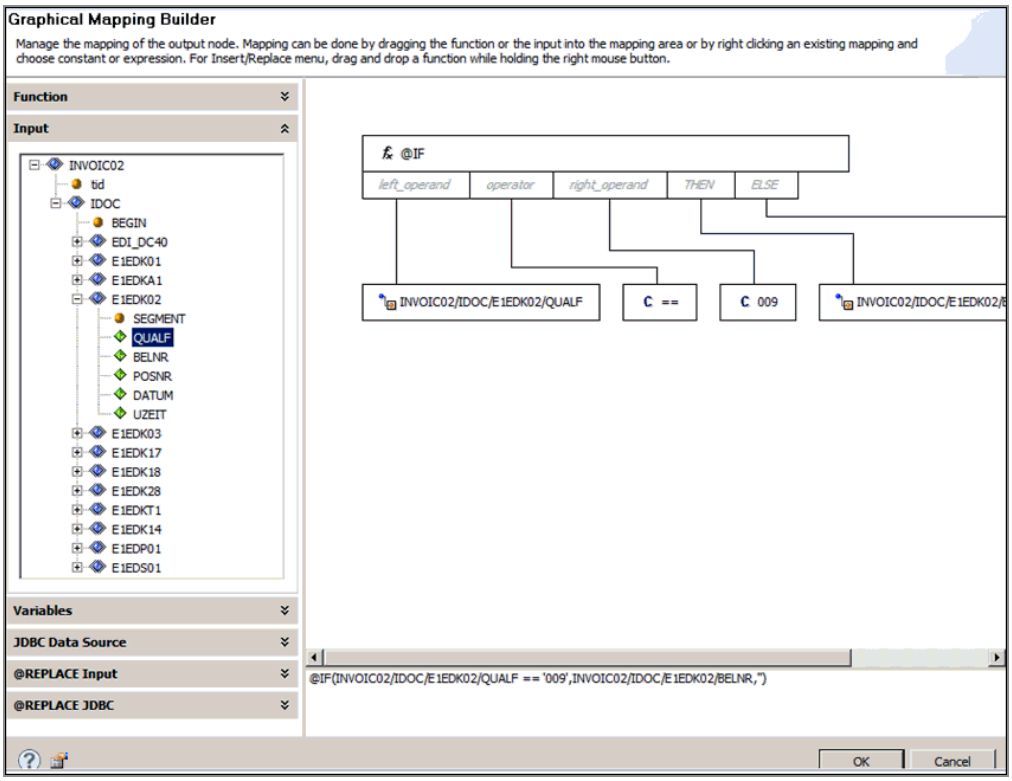
1. Map the following @IF function to the _01_Date element:

```
@IF(INVOIC02/IDOC/E1EDK02/QUALF ==
'009',INVOIC02/IDOC/E1EDK02/DATUM,'')
```



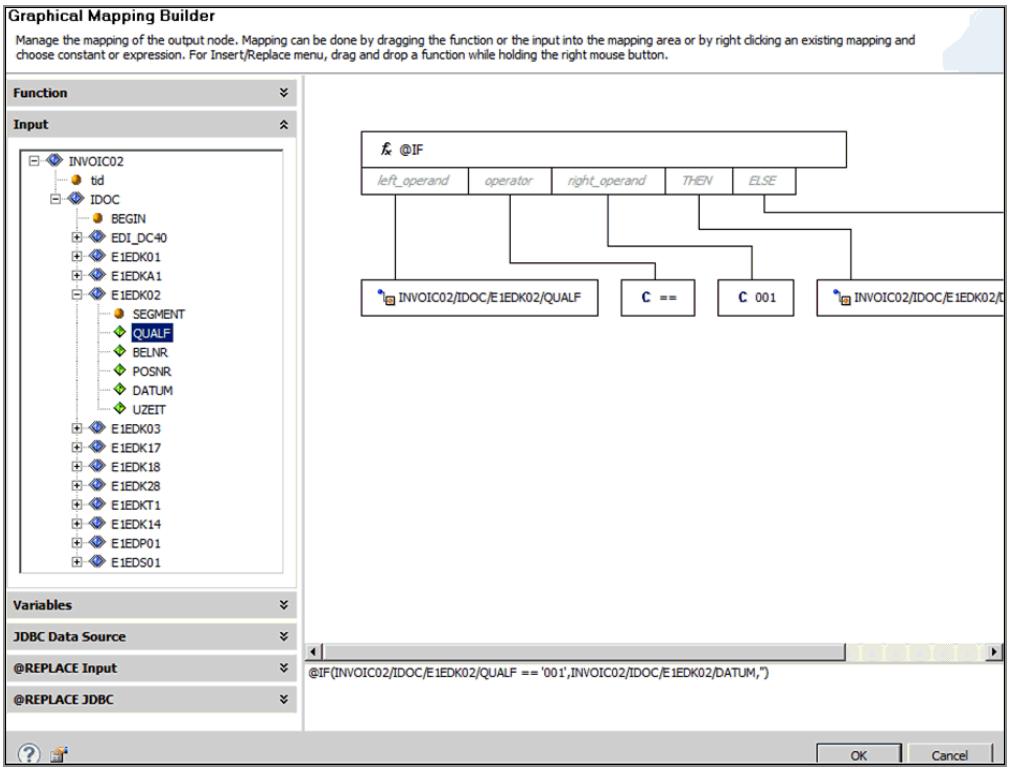
2. Map the following @IF function to the _02_Invoice_Number element:

```
@IF(INVOIC02/IDOC/E1EDK02/QUALF ==
'009',INVOIC02/IDOC/E1EDK02/BELNR,'')
```



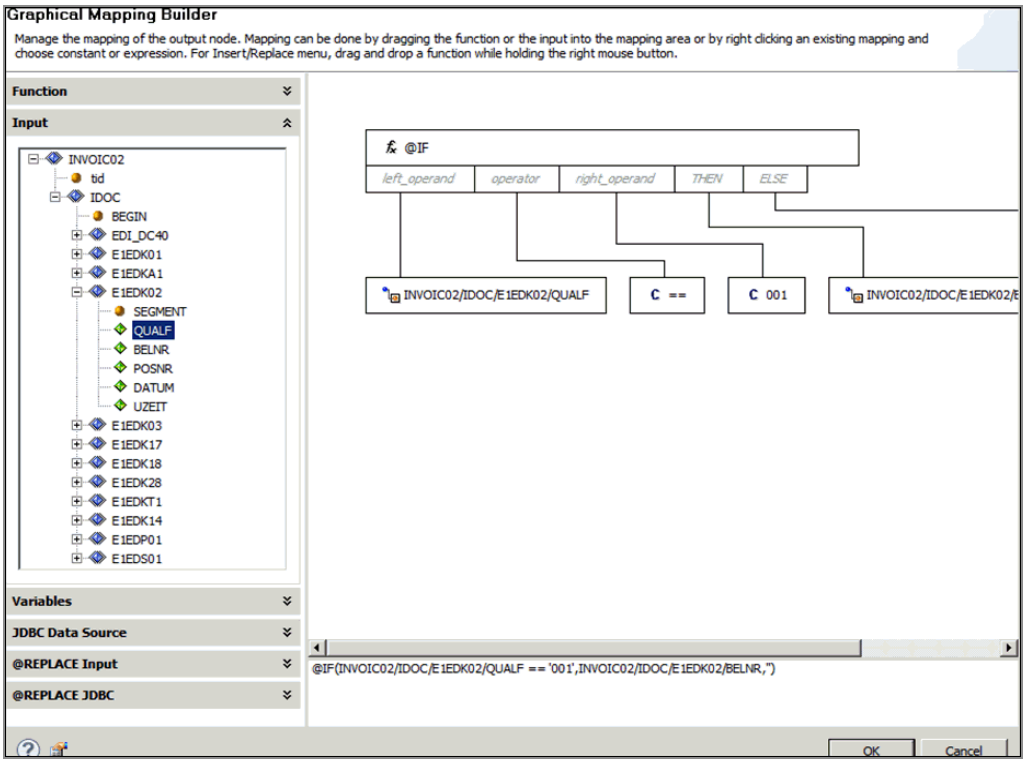
3. Map the following @IF function to the _03_Date element:

```
@IF(INVOIC02/IDOC/E1EDK02/QUALF ==
'001',INVOIC02/IDOC/E1EDK02/DATUM,'')
```

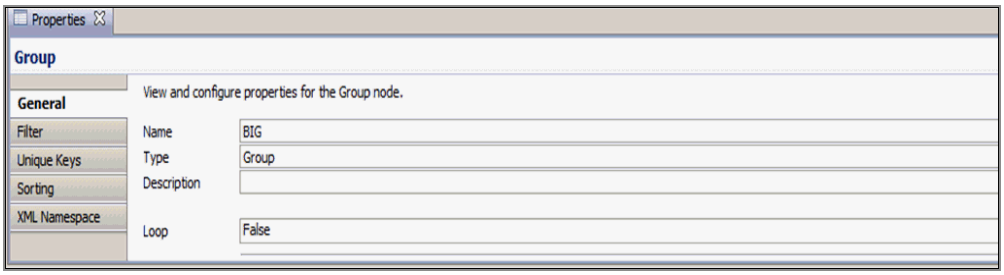


4. Map the following @IF function to the _04_Purchase_Order element:

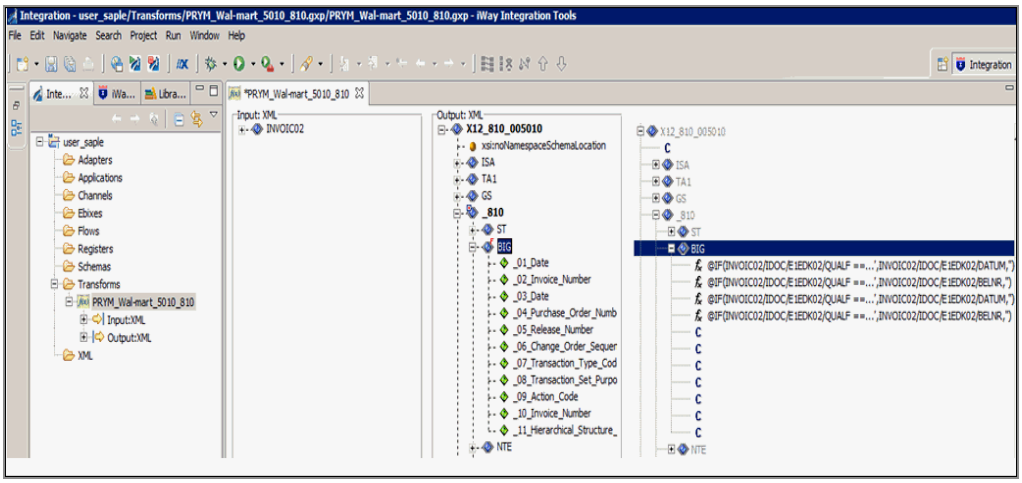
```
@IF(INVOIC02/IDOC/E1EDK02/QUALF ==
'001',INVOIC02/IDOC/E1EDK02/BELNR,'')
```



5. Set the looping property for the BIG segment to **False**.



Your iIT interface should resemble the following:



Currency Segment (CUR)

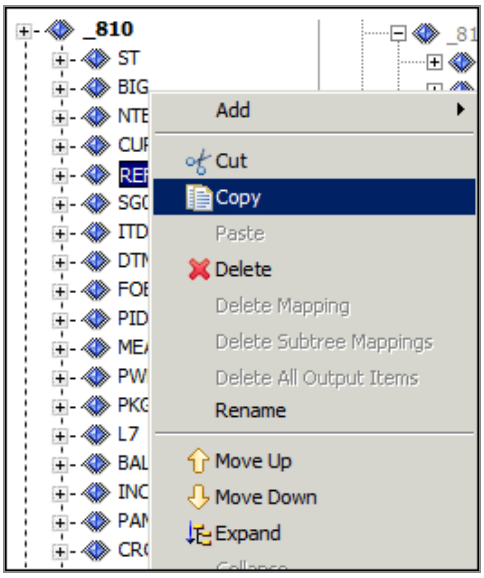
The CUR segment transmits the currency the invoice is billed in. There are two elements used that are constants.

1. Map the BY constant to CUR01.
2. Map the USD constant to CUR02.

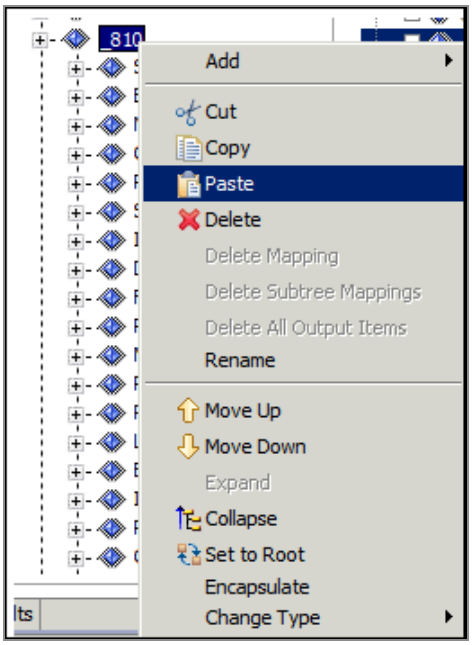
Reference Information Segment (REF)

There are two REF segments used in this Transform project. You will need to add the second by selecting the first one, copying it, and pasting it on the 810 node as a sub-tree.

1. Right-click the first instance of the REF segment and select **Copy** from the context menu.



2. Right-click the **_810** segment node and select **Paste**.



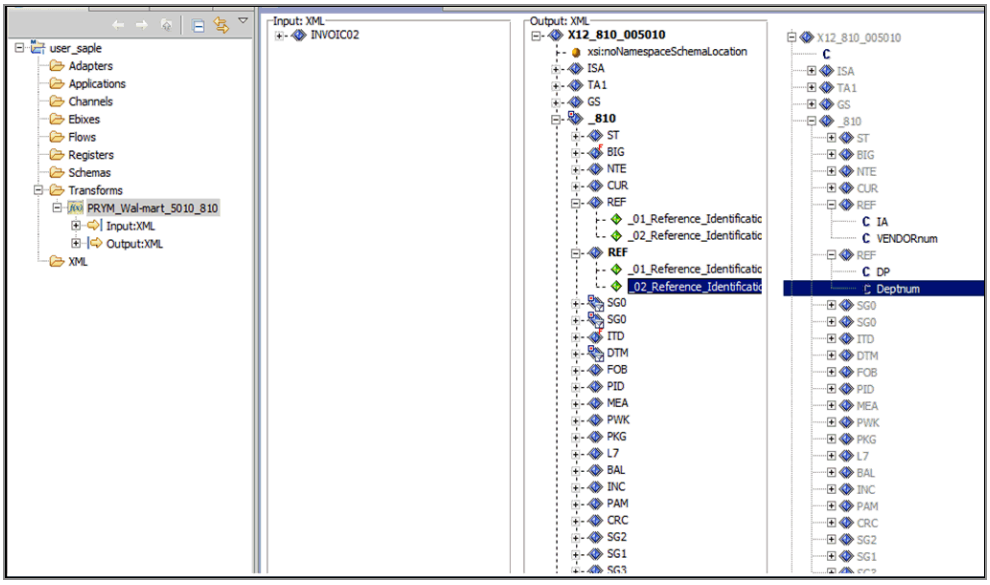
3. Use the Move Up option to position the new REF segment under the first REF segment.

   Next, map the values from the IDoc to both REF segments.

4. Map IA (Internal Vendor Number) to REF01.
5. Map the Vendor Number from the IDoc to REF02.

6. Map DP (Internal Vendor Number) to REF01.
7. Map the Department from the IDoc to REF02.

   Your iIT interface should resemble the following:



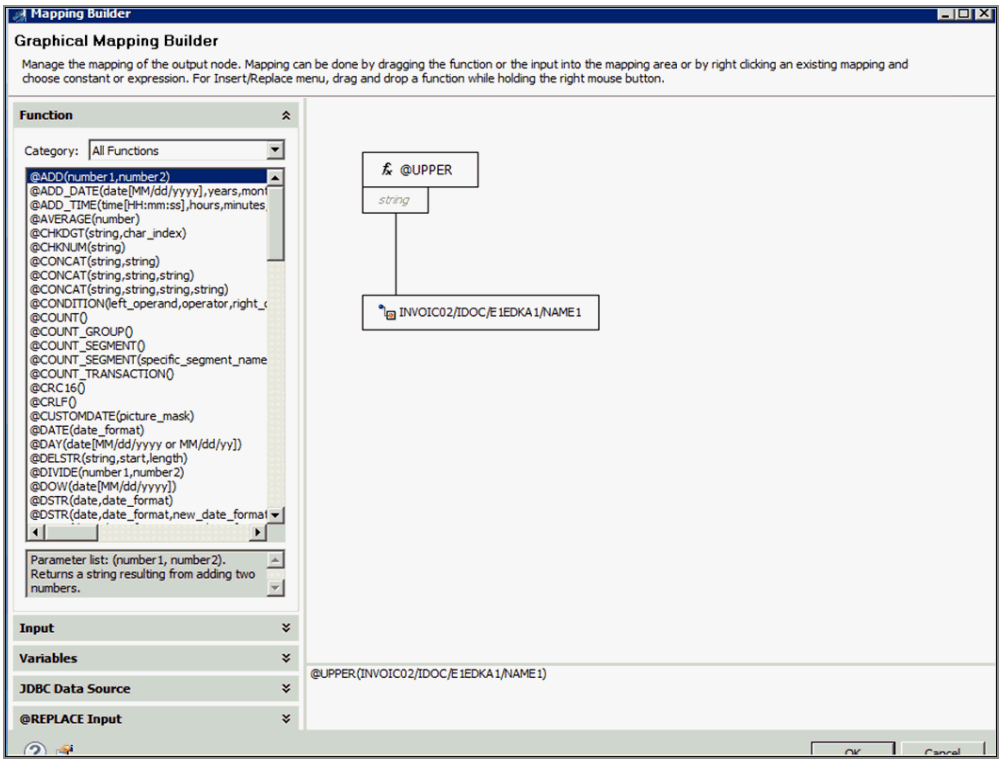
Name Loops

There are two Name loops used in this Transform project. One is used for the Supplier and the other is used for the Ship-To Address. To accomplish this, you will need to create a second SG0 in the same way you created the additional REF segment. Simply copy the SG0 node and paste it as a sub-tree over the existing _810 segment. You will then use the Move Up option to position the new SG0 node under the first SG0 node.

You are now ready to start mapping to the N1, N3, and N4 segments inside the SG0 nodes.

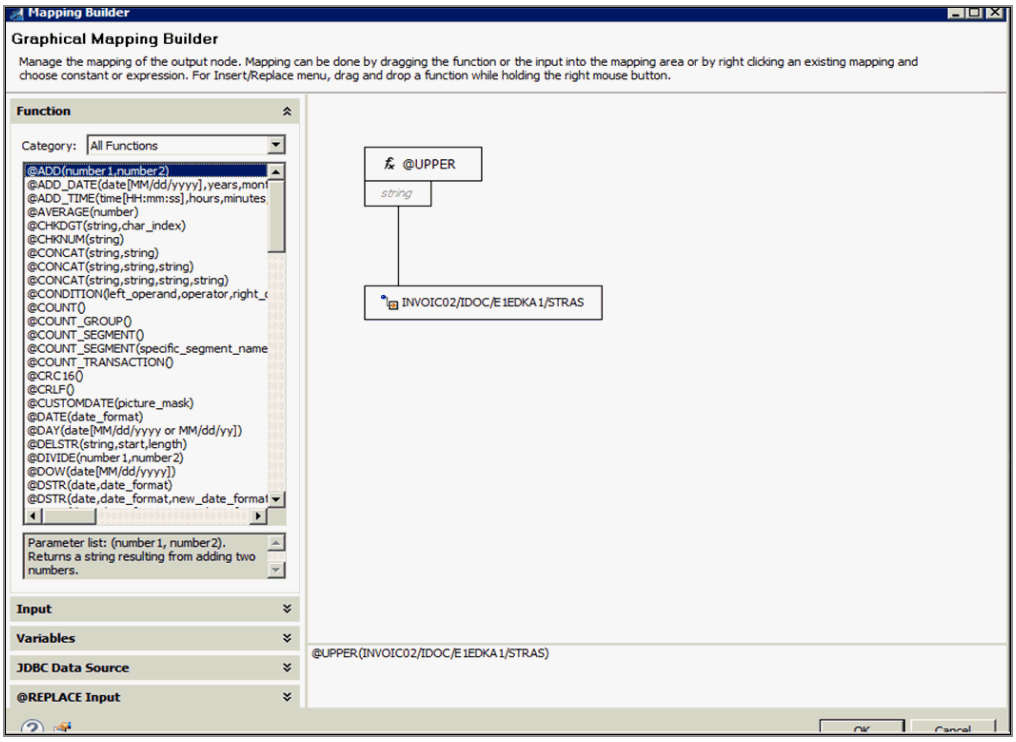
1. Map a constant of **SU** to the N101 element indicating the Supplier.
2. Map the UPPERCASE value of the Name to the N102 element.

```
@UPPER(INVOIC02/IDOC/E1EDKA1/NAME1)
```

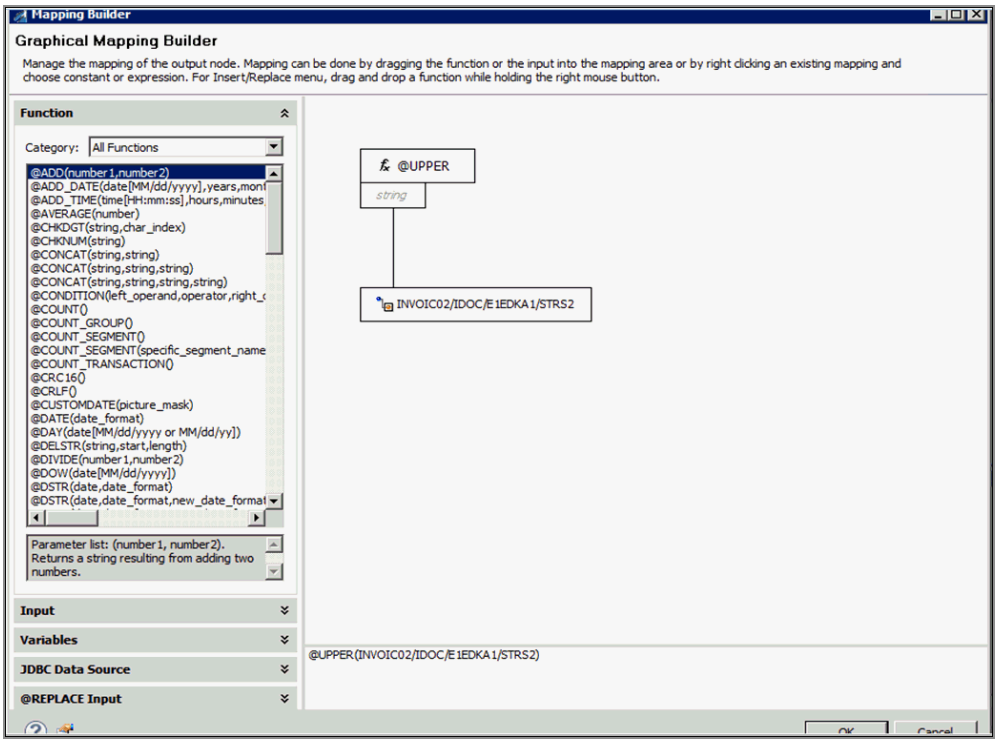


3. Map the Qualifier for the DUNS number **9** as a constant to the N103 segment. Then map the supplier DUNS number as a constant to the N104 element.
4. Map the UPPERCASE of the Address values to the N301 and N302 elements.

```
@UPPER(INVOIC02/IDOC/E1EDKA1/STRAS)
```

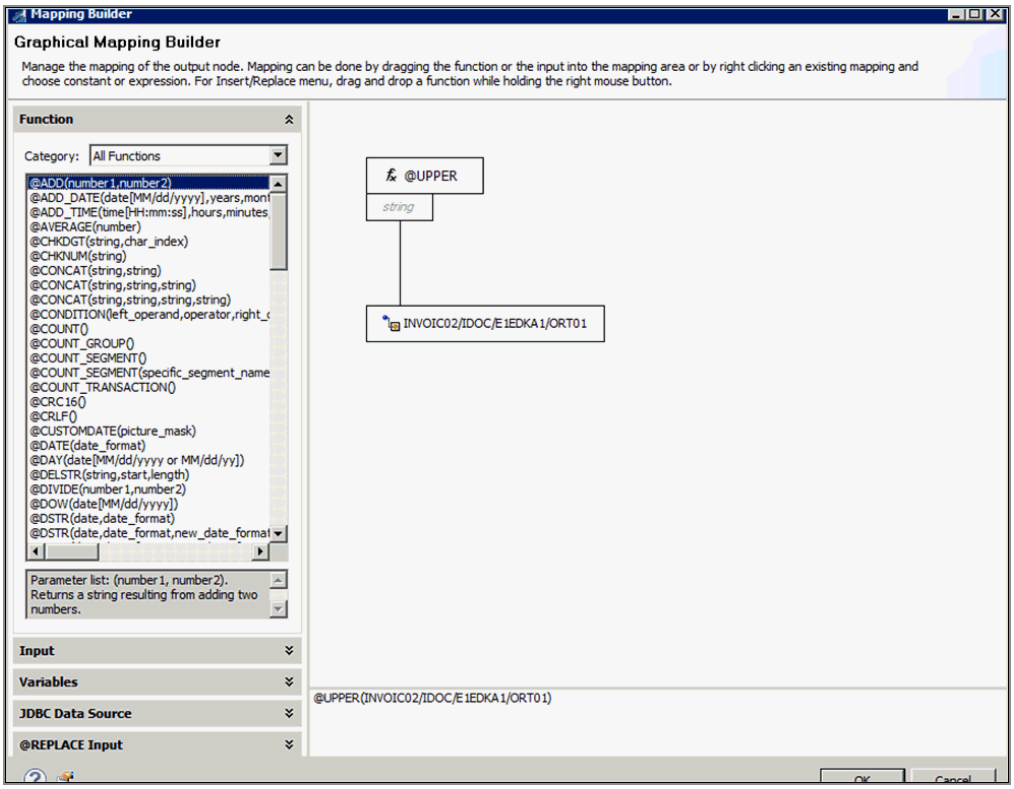



```
@UPPER(INVOIC02/IDOC/E1EDKA1/STRS2)
```

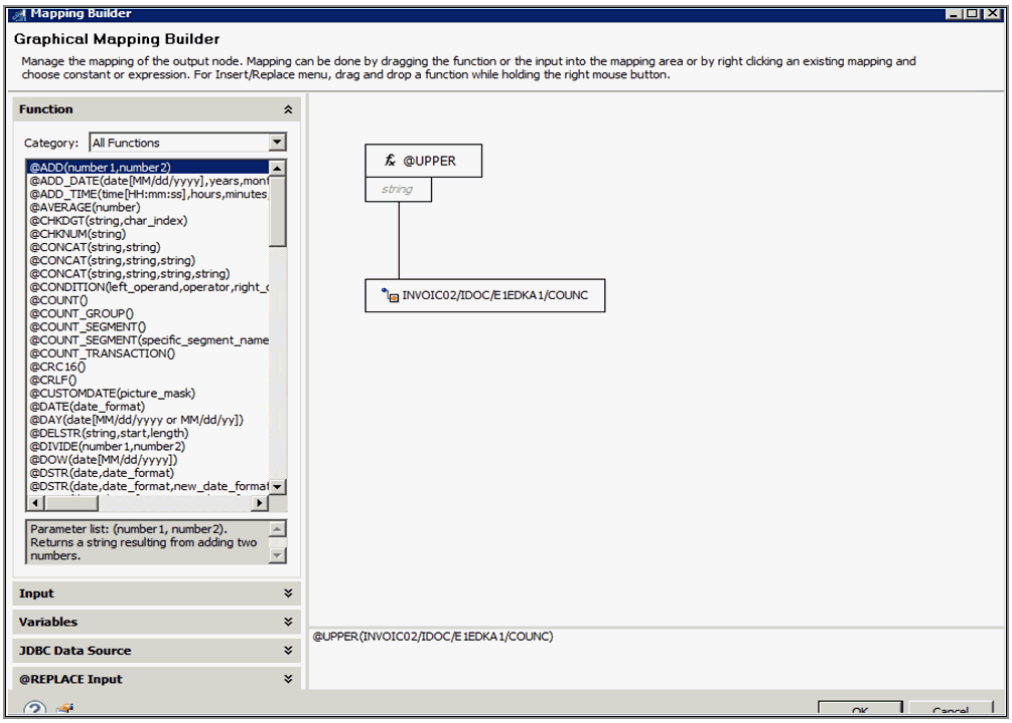


5. Map the UPPERCASE value of the City to the N401 element, then the State to the N402 element.

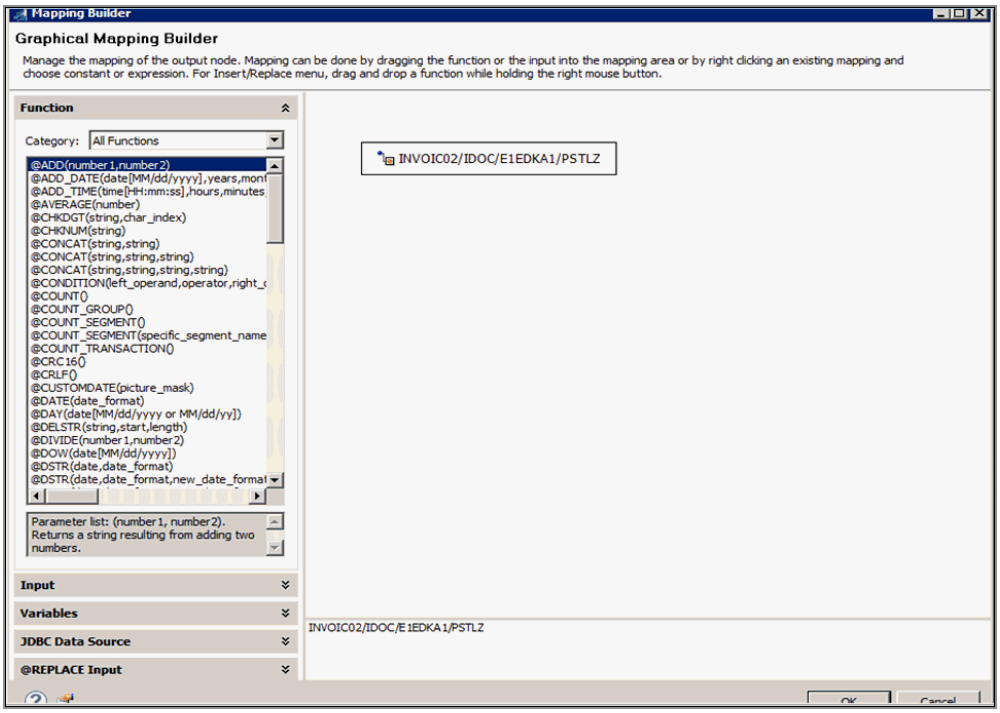
```
@UPPER(INVOIC02/IDOC/E1EDKA1/ORT01)
```



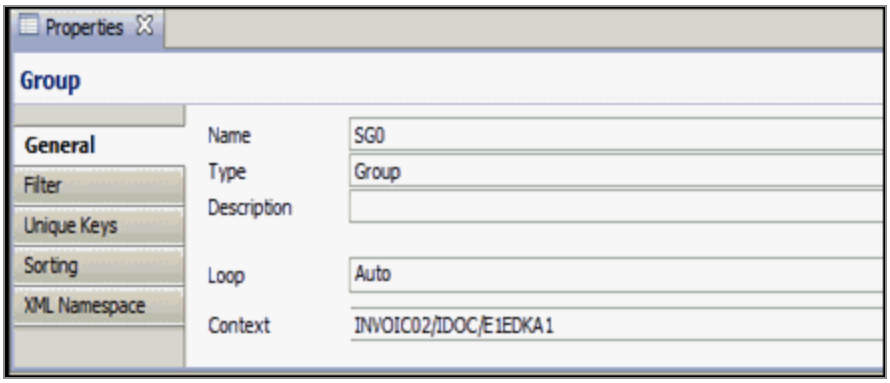

```
@UPPER(INVOIC02/IDOC/E1EDKA1/COUNC)
```



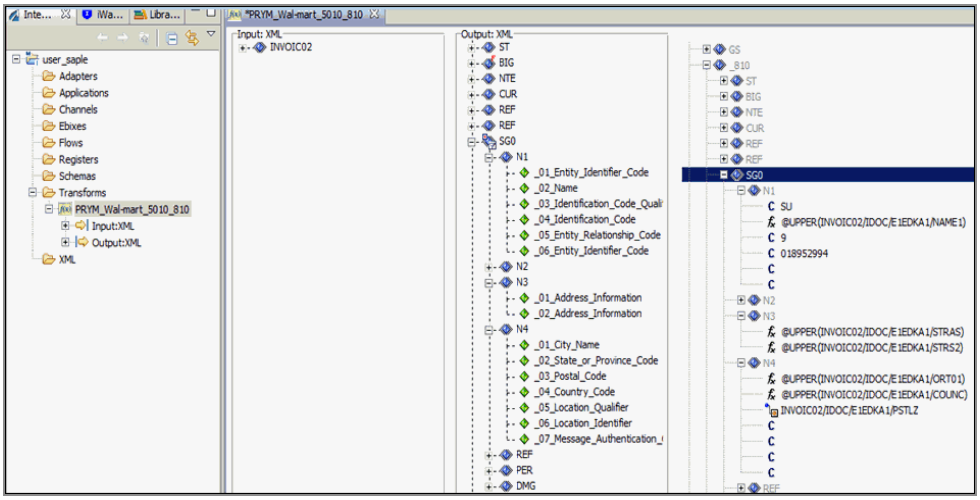
6. Map the ZIP to the N403 element.



7. Set the properties for the SG0 node.

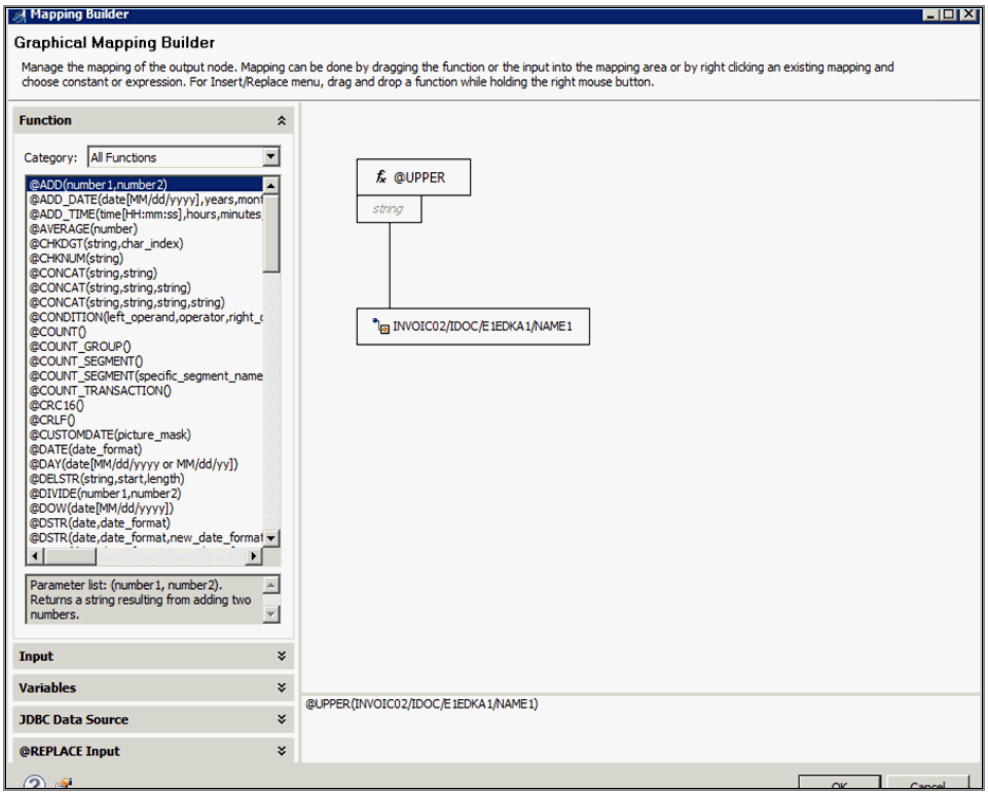


Your iIT interface should resemble the following:



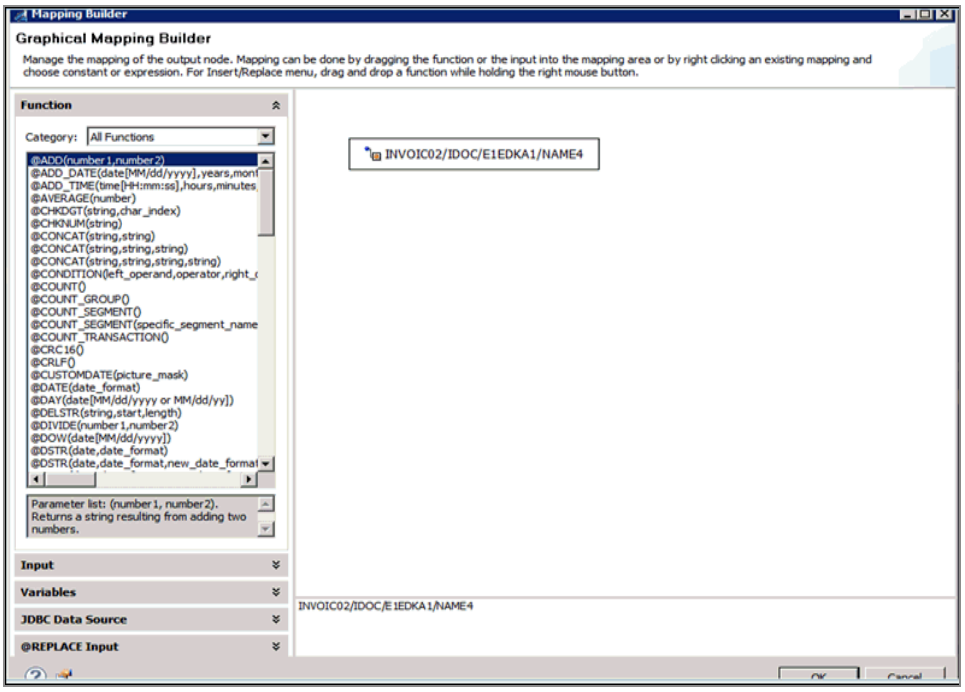
8. Map the Ship-To Address to the other occurrence of the SG0 node.
9. Map the constant ST to the N101 element.
10. Map the UPPERCASE value to the Name and map to the N102 element.

```
@UPPER(INVOIC02/IDOC/E1EDKA1/Name1)
```



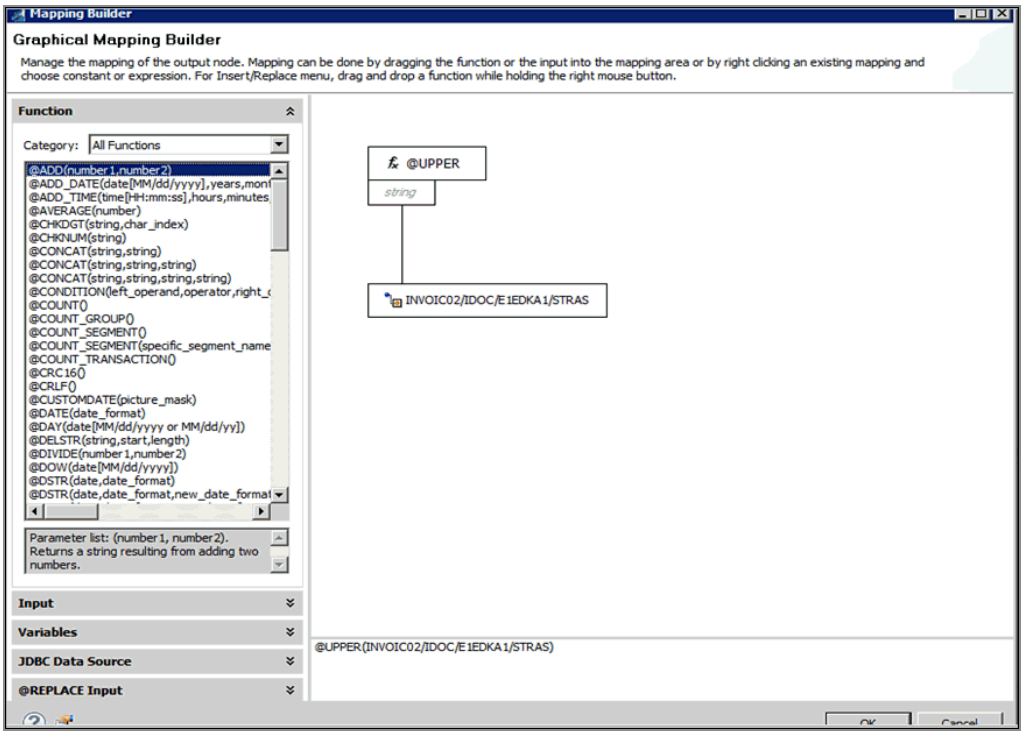
11. Map the constant UL to the N103 qualifier of the N104 element.
12. Map the Name4 to the N104 element.

```
INVOIC02/IDOC/E1EDKA1/Name4
```



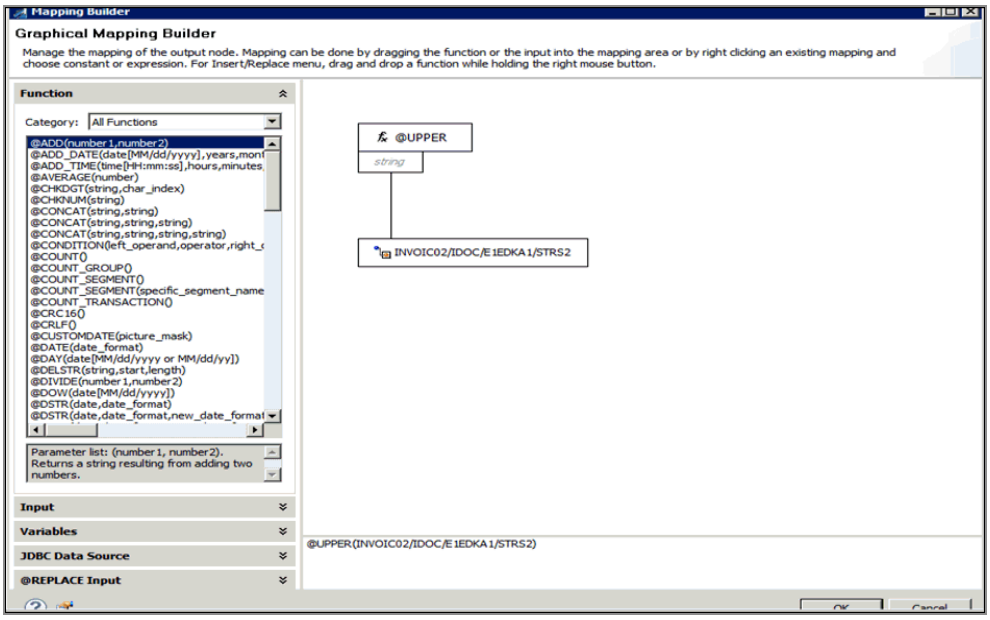
13. Map the IDoc City value to the N301 element.

```
@UPPER(INVOIC02/IDOC/E1EDKA1/STRAS)
```



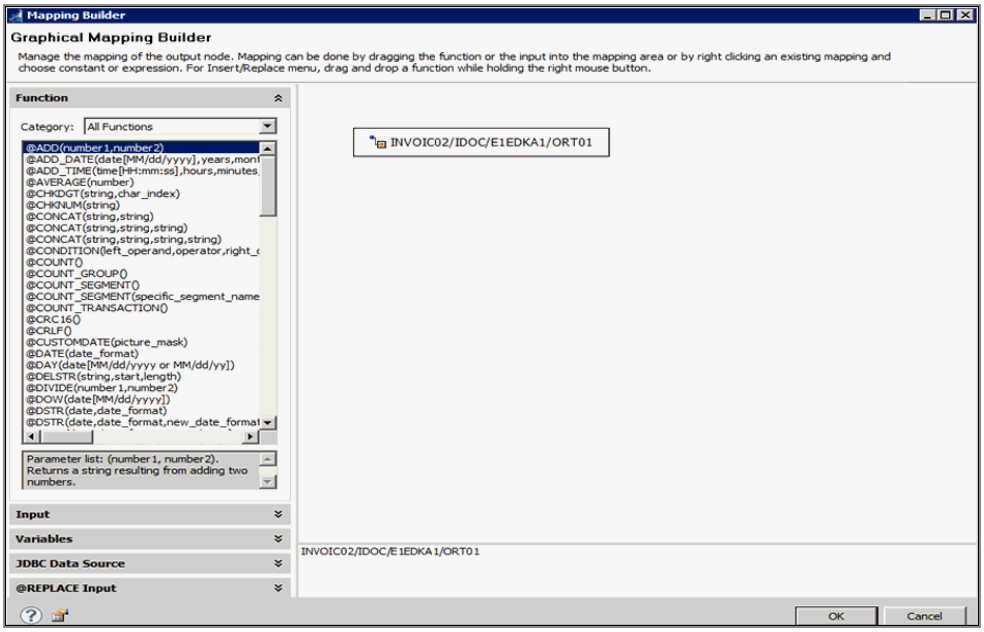
14. Map the IDoc State value to the N302 element.

```
@UPPER(INVOIC02/IDOC/E1EDKA1/STRS2)
```



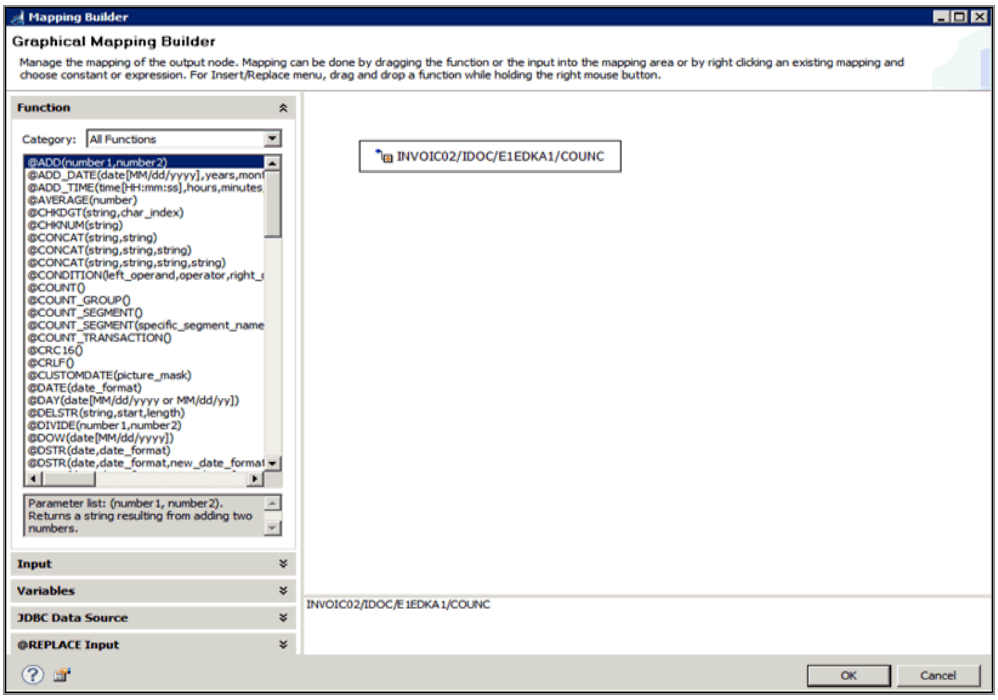
15. Map the City to the N401 element.

```
INVOIC02/IDOC/E1EDKA1/ORT01
```



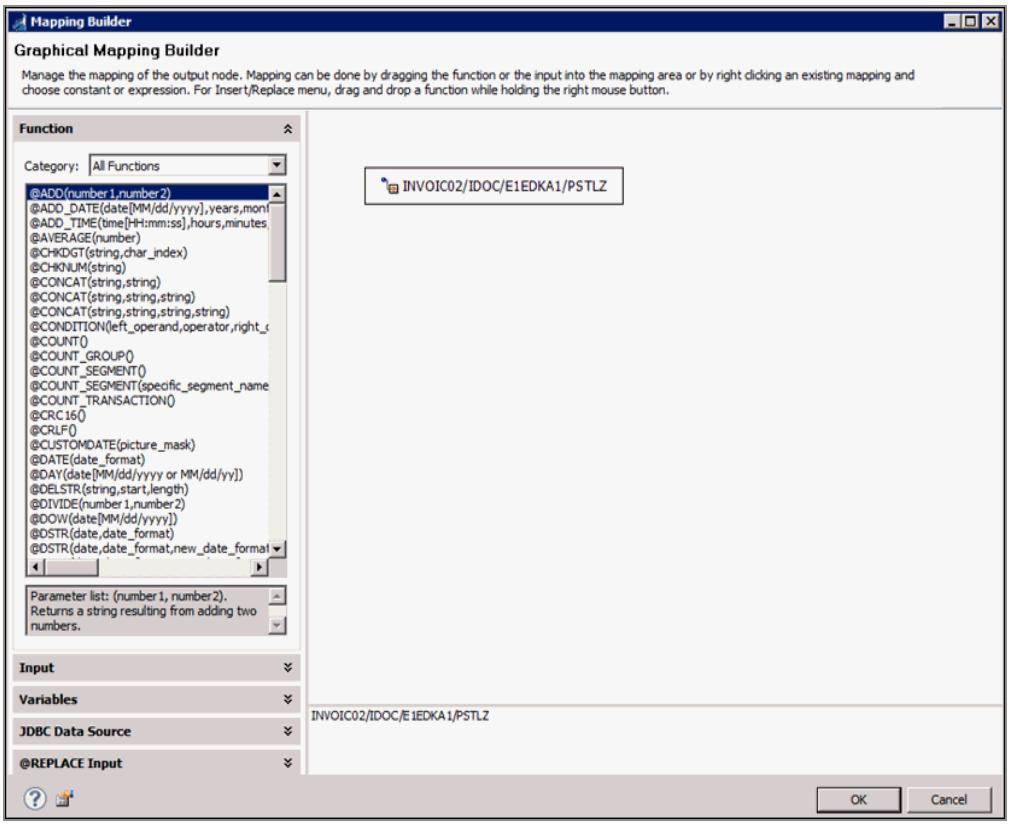
16. Map the State to the N402 element.

```
INVOIC02/IDOC/E1EDKA1/COUNC
```

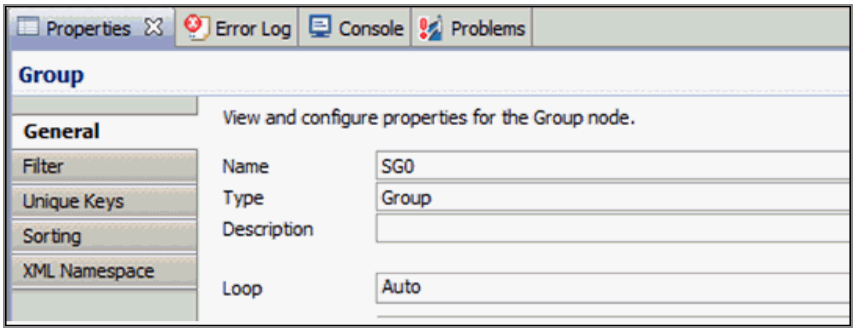


17. Map the ZIP to the N403 element.

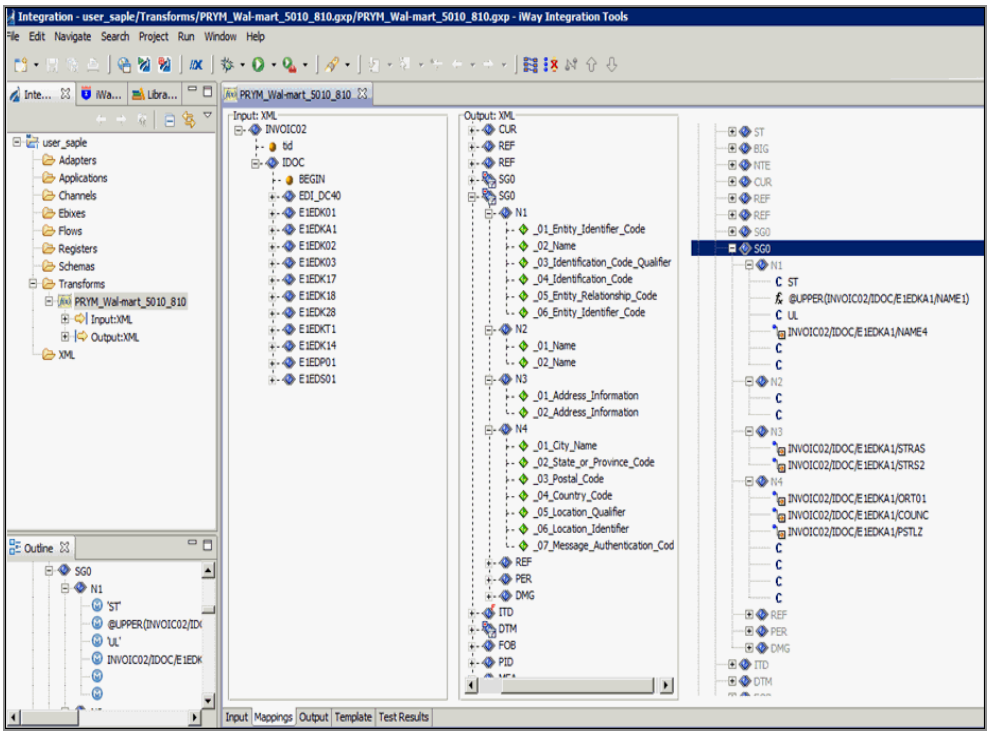
```
INVOIC02/IDOC/E1EDKA1/PSTLZ
```



18. Set the properties for this SG0 node.



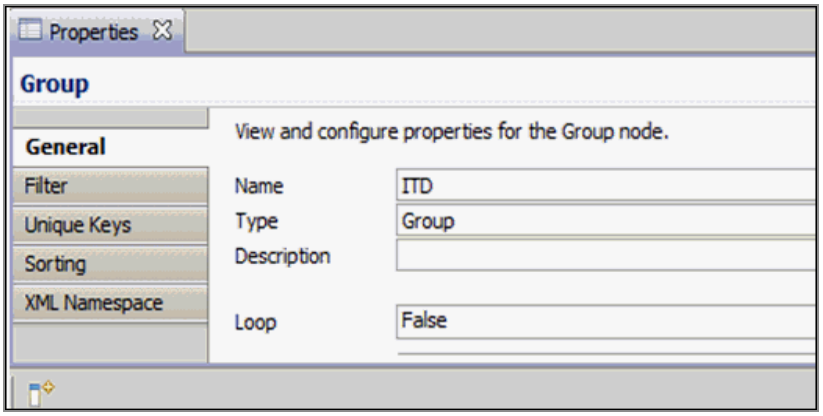
Your iIT interface should resemble the following:



Terms of Sale Segment (ITD)

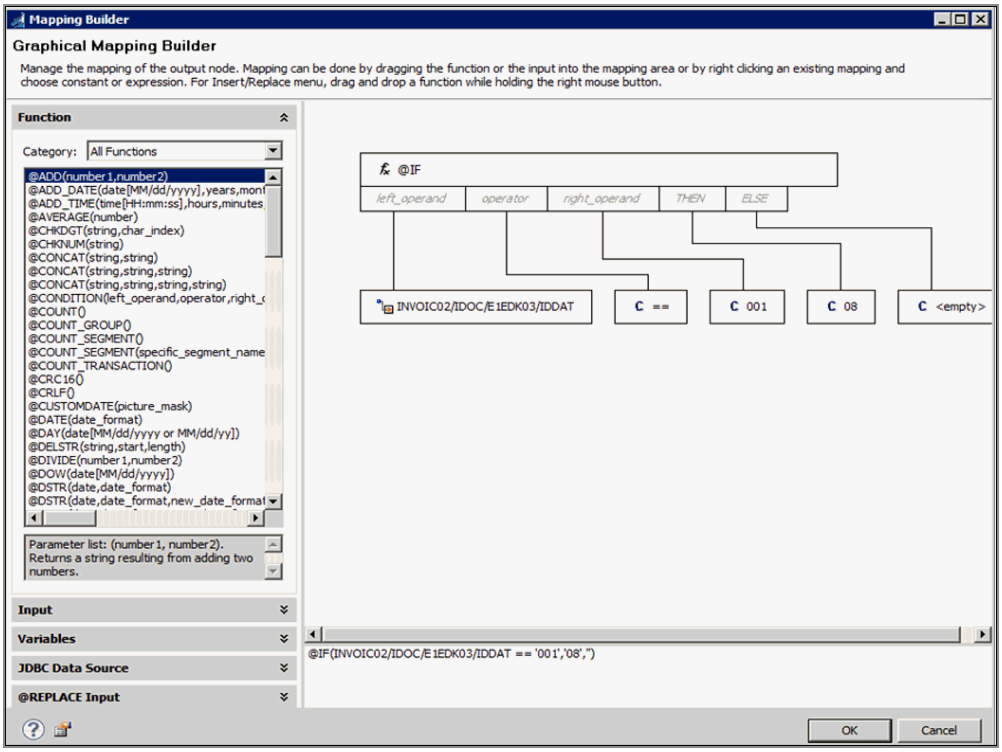
There is one ITD segment that is used in this Transform project and must be configured.

1. Set the looping property for the ITD segment to **False**.



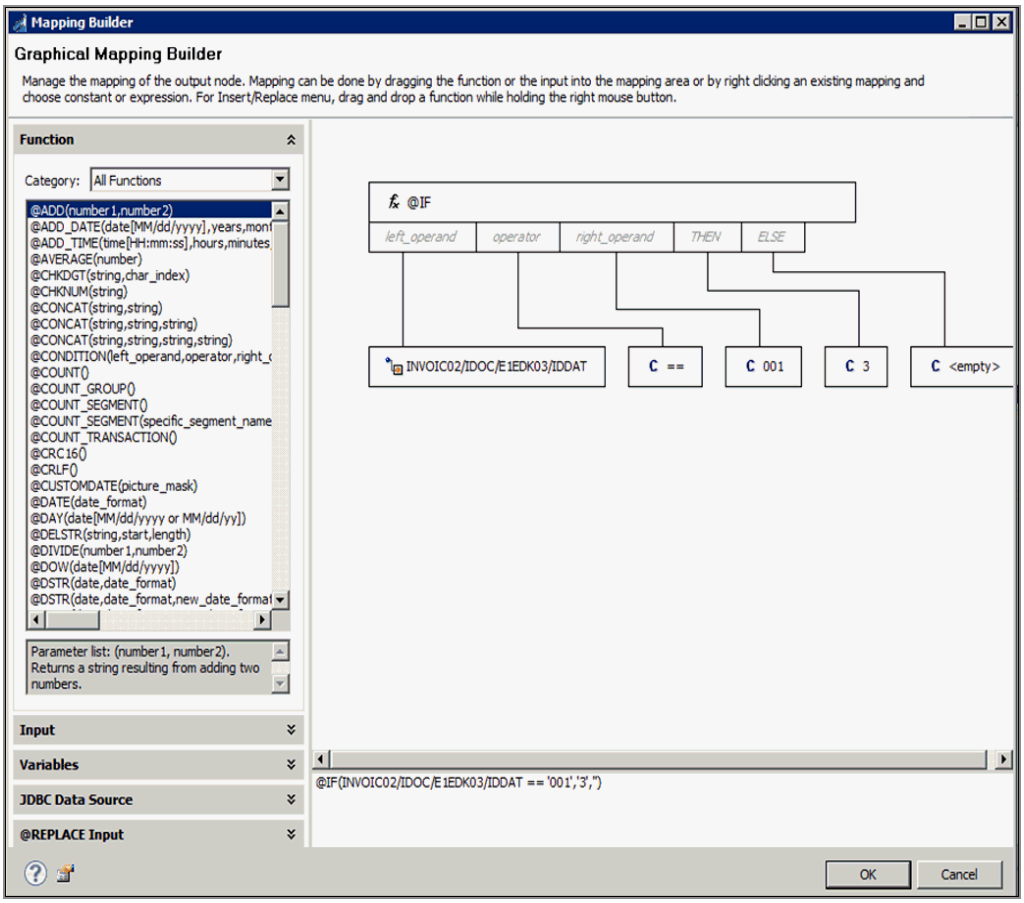
2. Map the terms and the terms dates.
3. Based on the qualifier, map the ITD01 segment.

```
@IF(INVOIC02/IDOC/E1EDKA1/IDDAT == '001','08','')
```



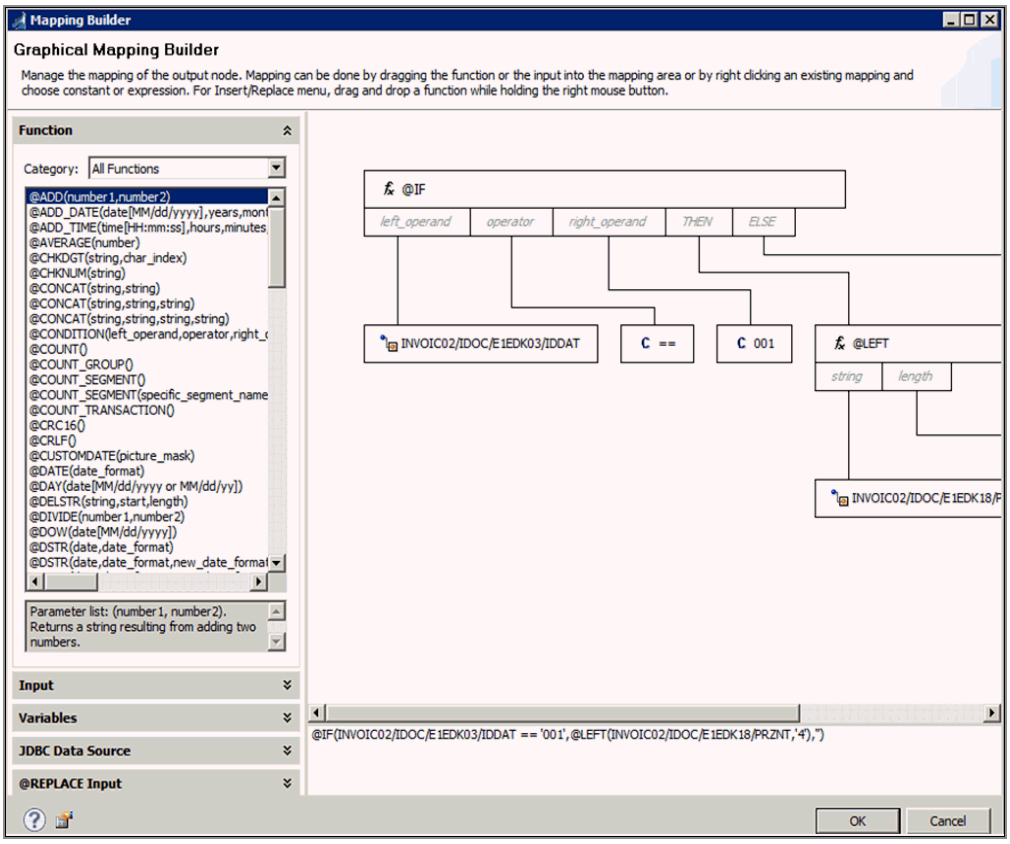
4. Map the ITD02 segment.

```
@IF(INVOIC02/IDOC/E1EDK03/IDDAT == '001','3','')
```



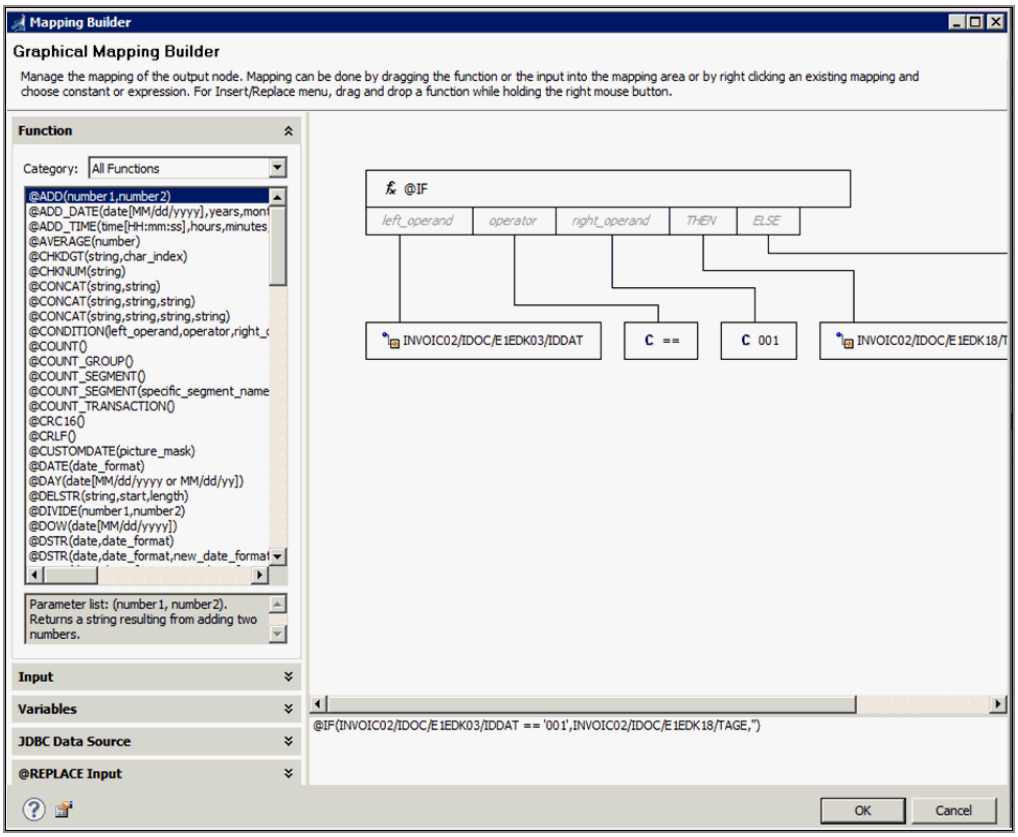
5. Map to the ITD03 segment.

```
@IF(INVOIC02/IDOC/E1EDK03/IDDAT == '001',@LEFT
(INVOIC02/IDOC/E1EDK18/PRZNT),'4'),'')
```



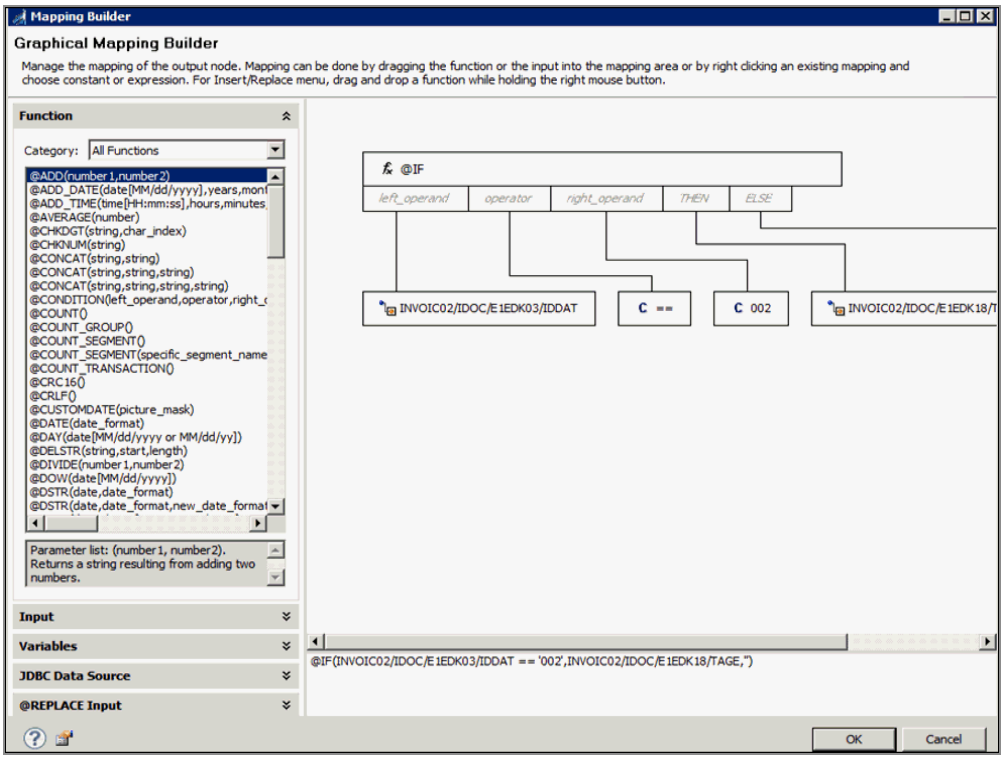
6. Map the ITD05 segment.

```
@IF(INVOIC02/IDOC/E1EDK03/IDDAT == '001','
INVOIC02/IDOC/E1EDK18/TAGE','')
```

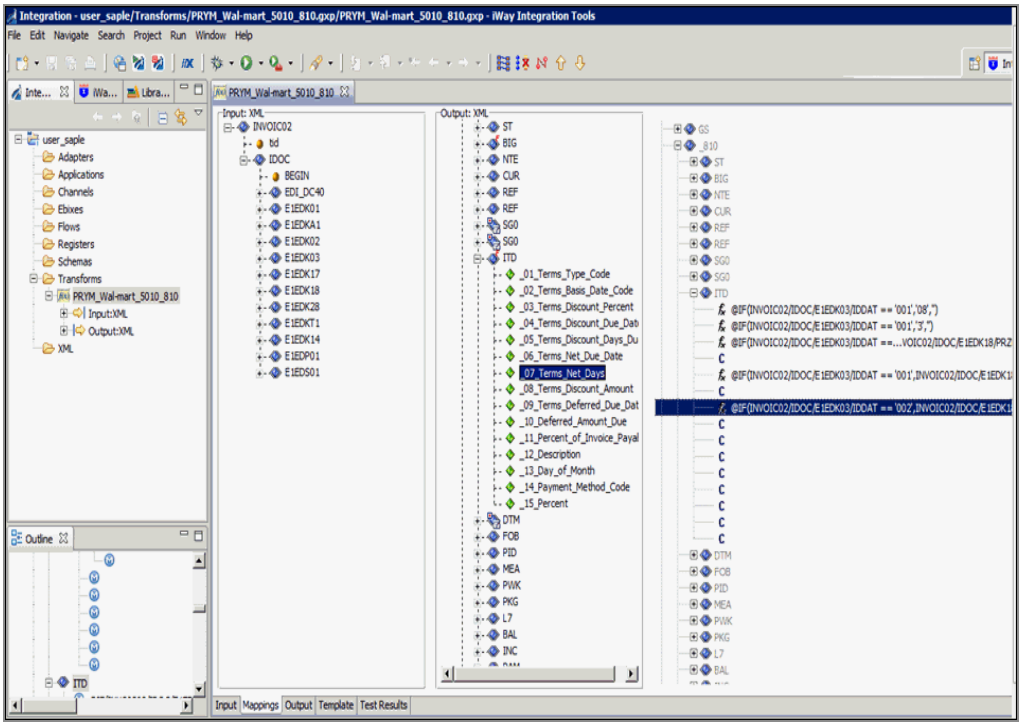


7. Map the ITD07 segment.

```
@IF(INVOIC02/IDOC/E1EDK03/IDDAT == '002','
INVOIC02/IDOC/E1EDK18/TAGE','')
```



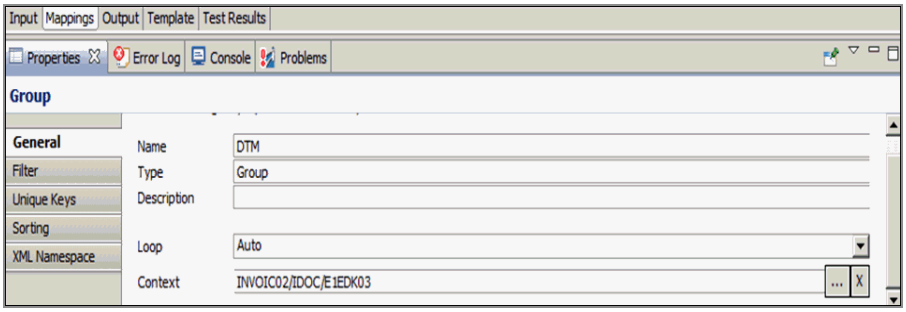
Your iIT interface should resemble the following:



Date/Time Segment (DTM)

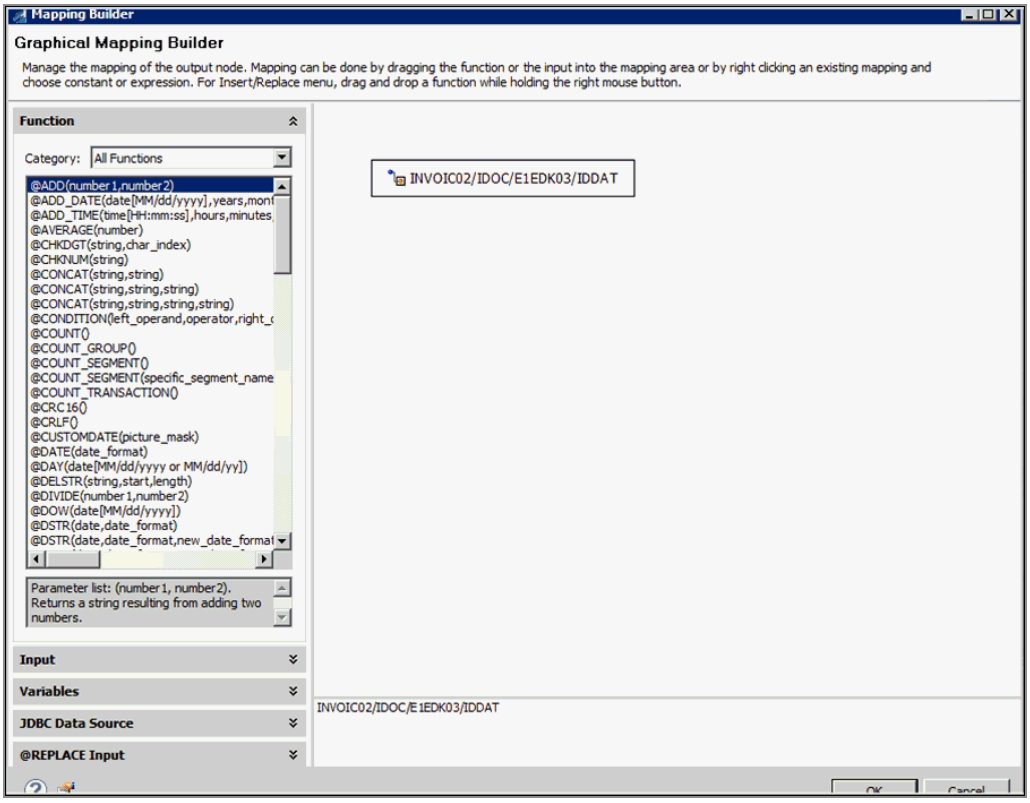
There is one DTM segment that is used in this Transform project and must be configured.

1. Set the looping property for the DTM segment to **Auto**.



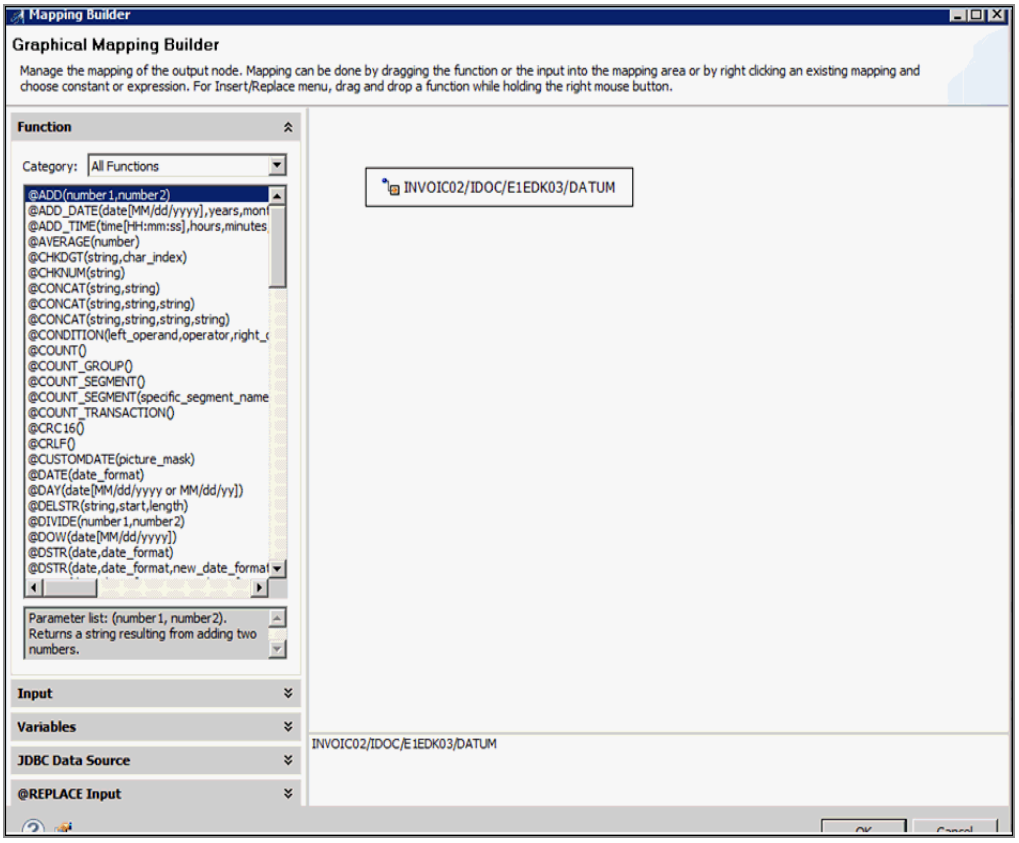
2. Map the Qualifier for the terms date to the DTM01 segment if the terms date exists.

```
@IF(INVOIC02/IDOC/E1EDK03/DATUM > '0',' INVOIC02/IDOC/E1EDK03/IDDAT
','')
```

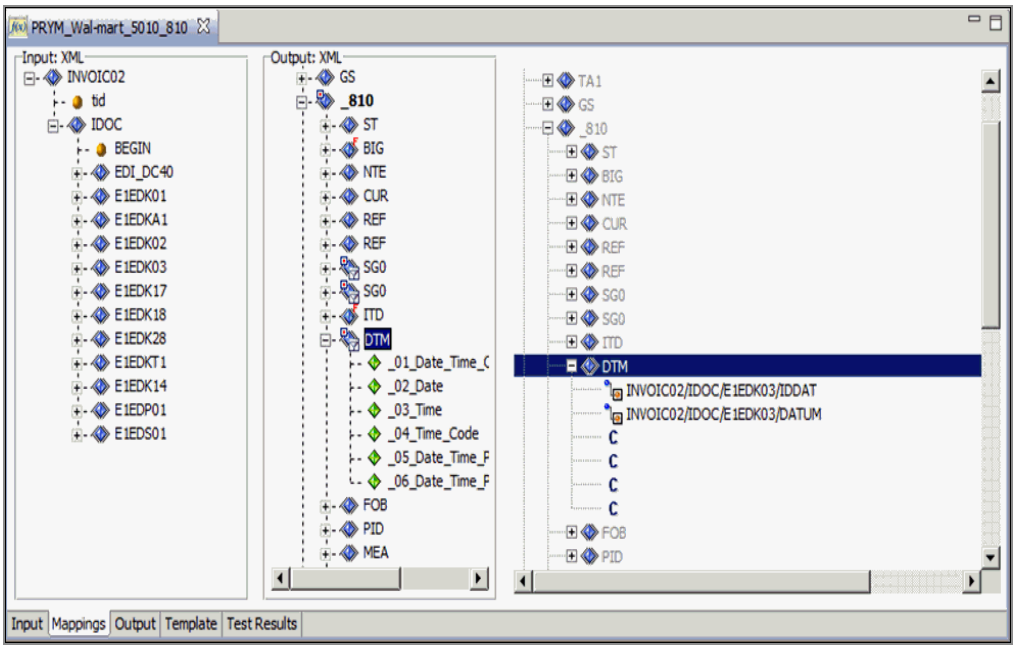


3. Map the date into the DTM02 segment.

```
INVOIC02/IDOC/E1EDK03/DATUM
```



Your iIT interface should resemble the following:

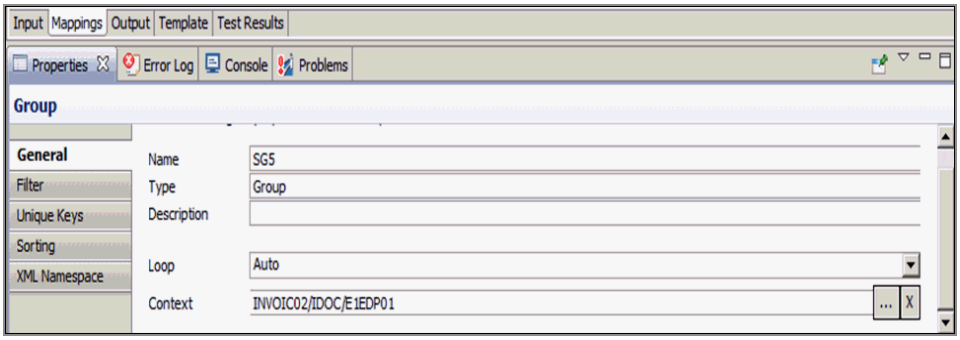


Mapping the ITEM Detail

This section describes how to map the ITEM detail.

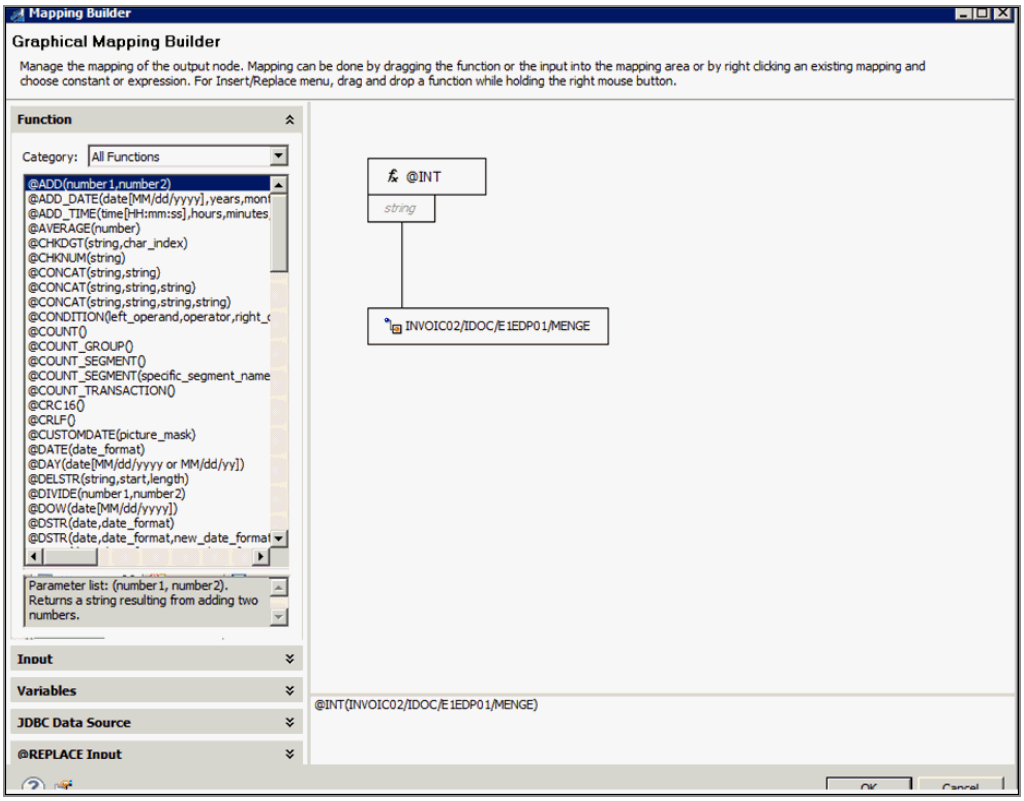
Baseline Invoice Item Data (IT1)

1. In the Output Tag Properties dialog box for the SG5 segment, set the context and looping.



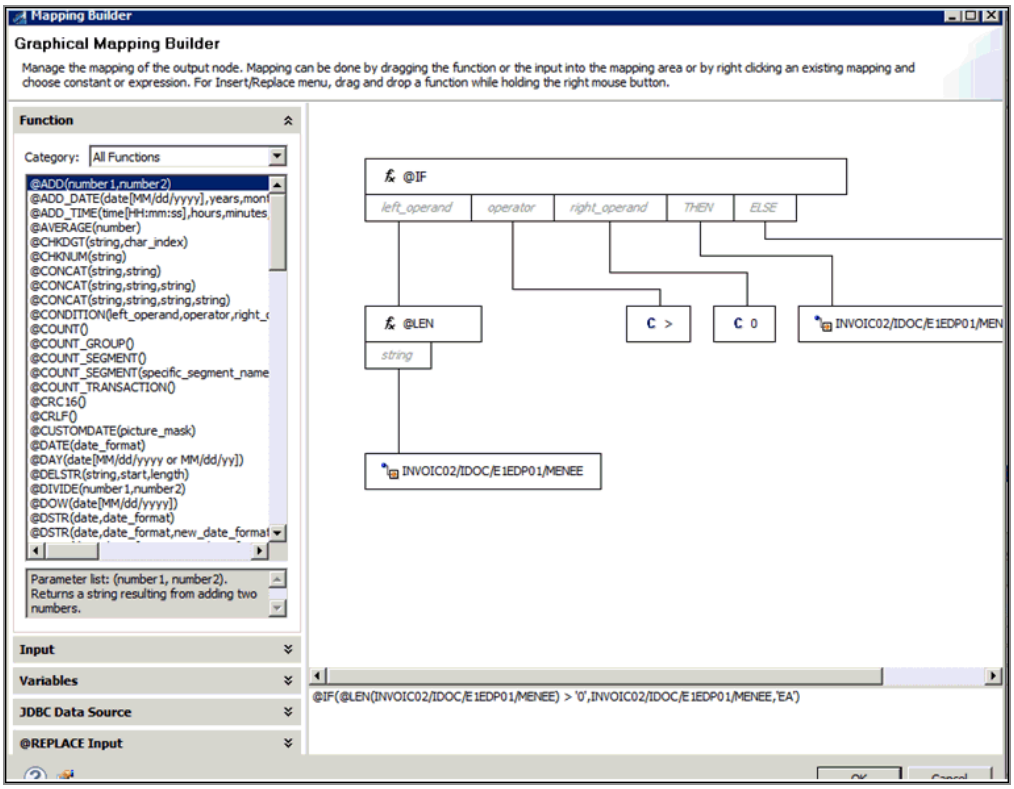
2. Map the Integer of the IDoc line number value to the IT102 element.

```
@INT(INVOIC02/IDOC/E1EDP01/MENGE)
```



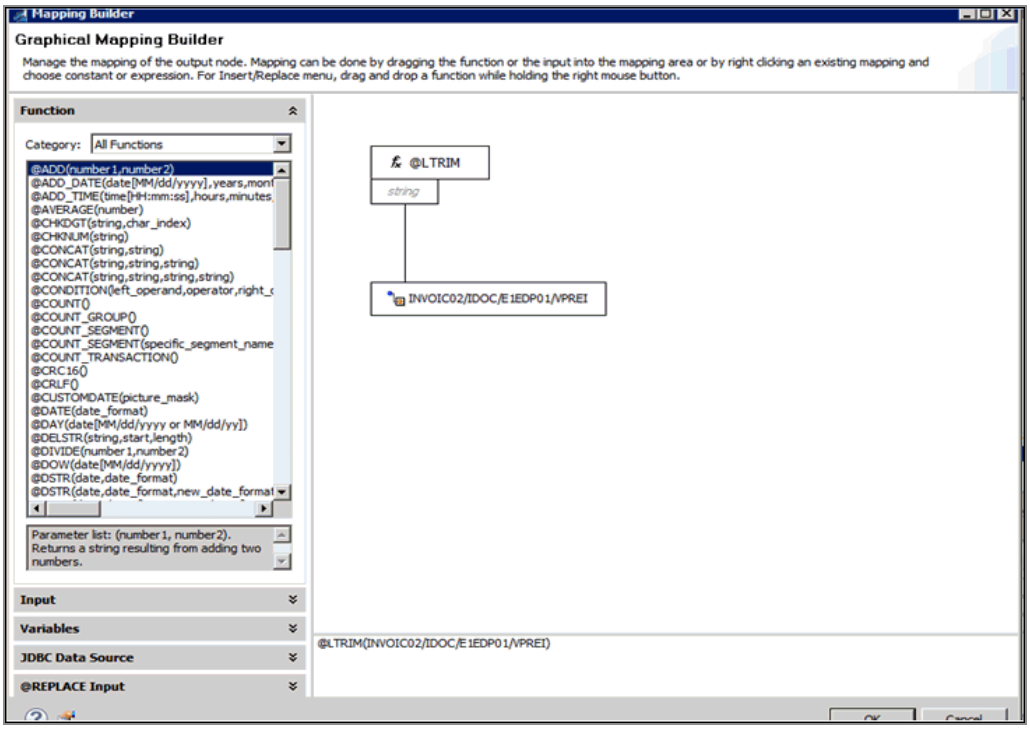
3. If the length of the Unit of Measure is greater than 0, map to the IT103 element, otherwise, map the constant EA.

```
@IF(@LEN(INVOIC02/IDOC/E1EDP01/MENEE) > '0','
INVOIC02/IDOC/E1EDP01/MENEE','EA')
```

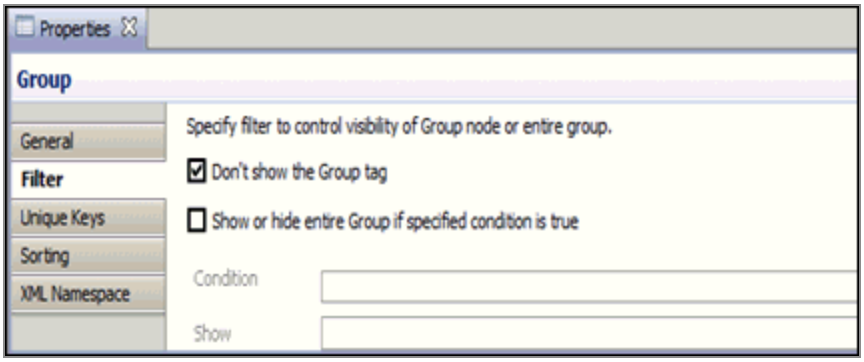


4. Trim leading spaces from the unit price and map it to the IT104 element.

```
@LTRIM(INVOIC02/IDOC/E1EDP01/VPREI)
```

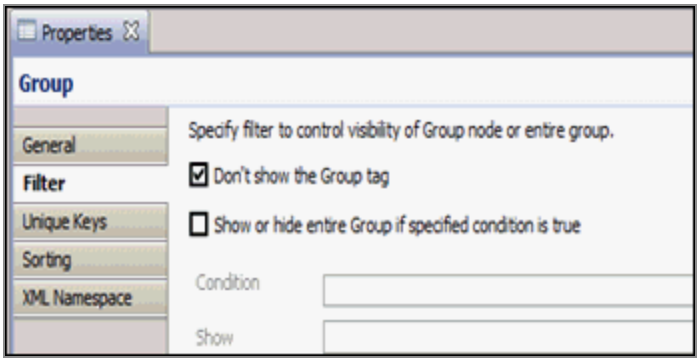


5. Add an output group node to the IT1 segment and rename it to OUTPUT_GROUP_NODE.
6. Use the Move Up option to position the new output group node under the IT104 element.
7. In the Output Tag Properties dialog box for the OUTPUT_GROUP_NODE, set the following properties:

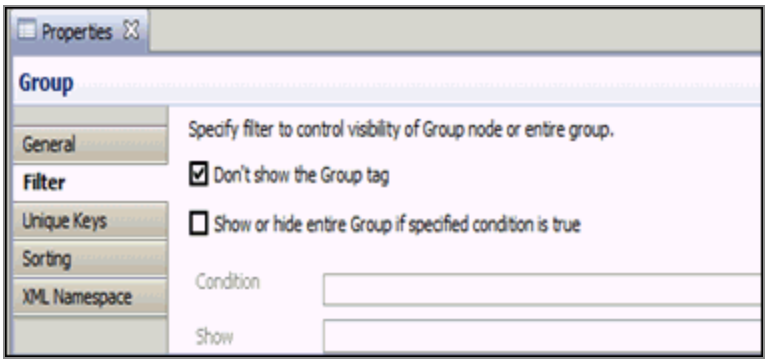


8. Add an output group node to the IT1 segment and rename it to OUTPUT_PARENT_TAG.
9. Use the Move Up option to position the new output group node under the IT105 element.

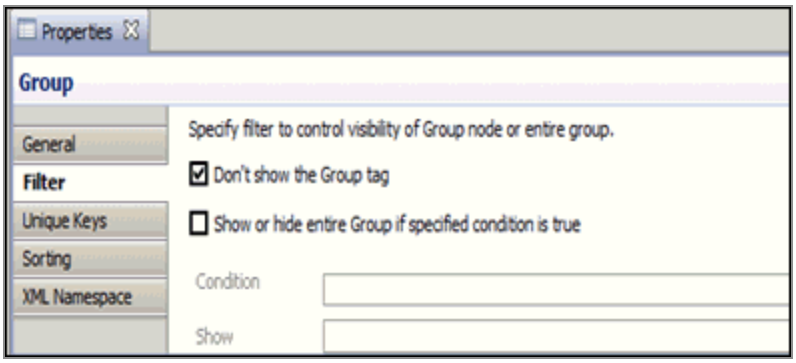
10. In the Output Tag Properties dialog box for the OUTPUT_PARENT_TAG, set the following properties:



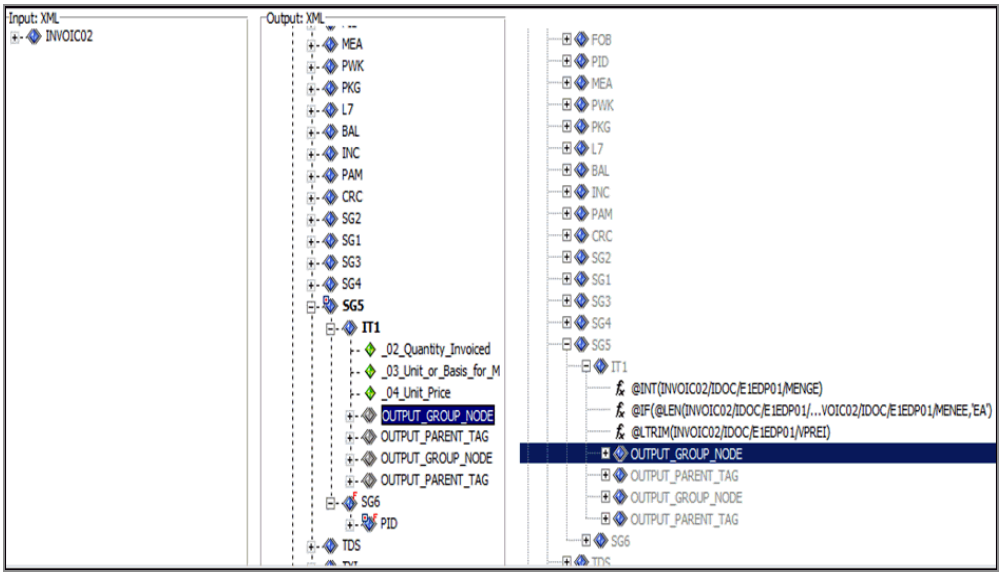
11. Add an output group node to the IT1 segment and rename it to OUTPUT_GROUP_NODE.
12. Use the Move Up option to position the new output group node under the IT106 element.
13. In the Output Tag Properties dialog box for the OUTPUT_GROUP_NODE, set the following properties:



14. Add an output group node to the IT1 segment and rename it to OUTPUT_PARENT_TAG.
15. Use the Move Up option to position the new output group node under the IT107 element.
16. In the Output Tag Properties dialog box for the OUTPUT_PARENT_TAG, set the following properties:

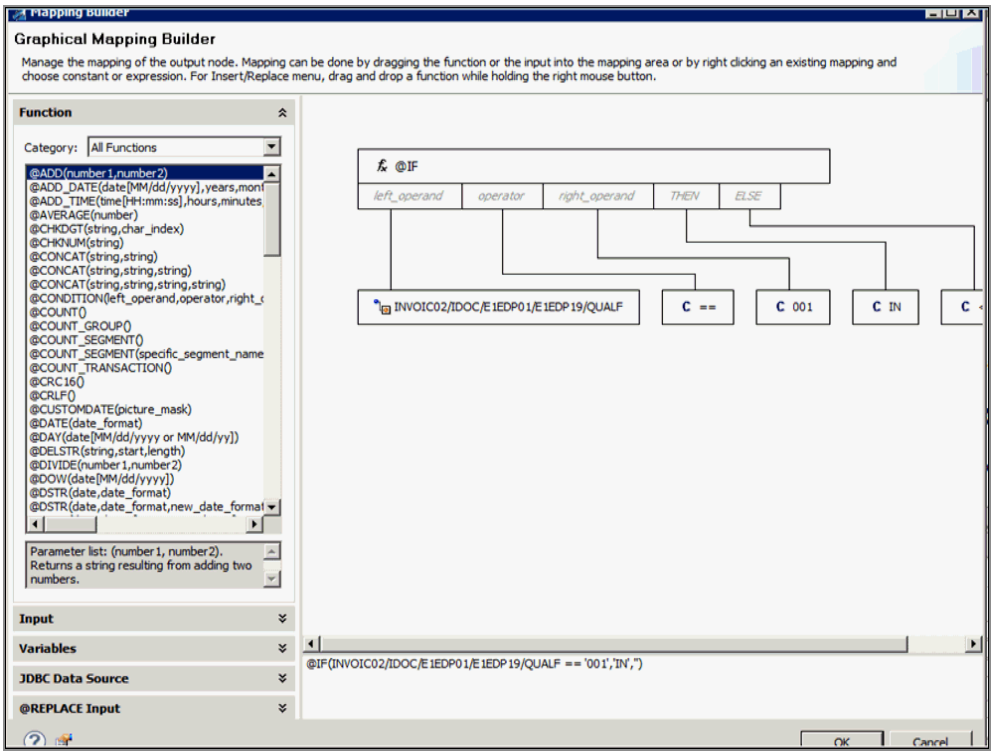


Your iWay Integration Tool interface should resemble the following image:



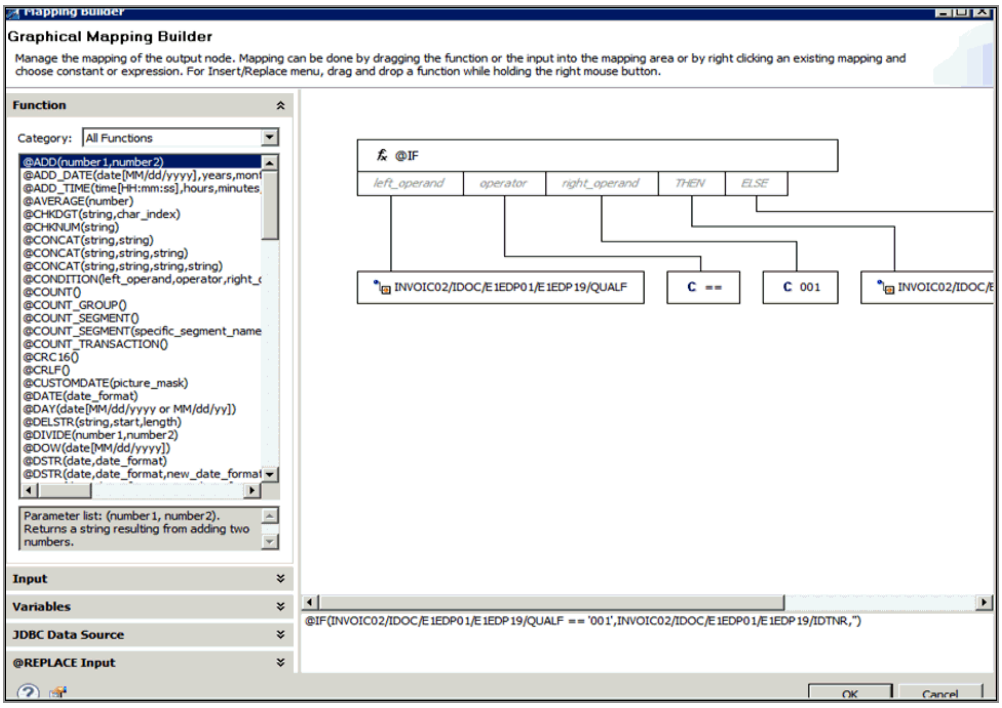
17. Map the Qualifier to the IT106 element.

```
@IF(INVOIC02/IDOC/E1EDP01/E1EDP19/QUALF == '001',' IN','')
```



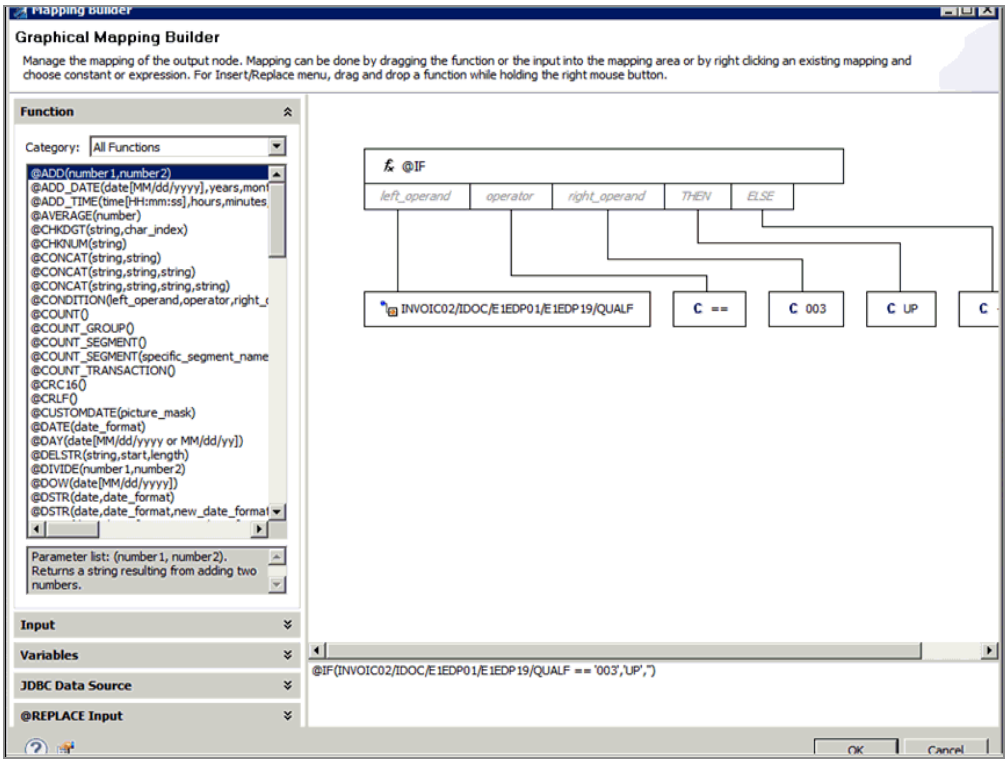
18. Map the value to the IT107 element.

```
@IF(INVOIC02/IDOC/E1EDP01/E1EDP19/QUALF == '001','
INVOIC02/IDOC/E1EDP01/E1EDP19/IDTNR ','')
```



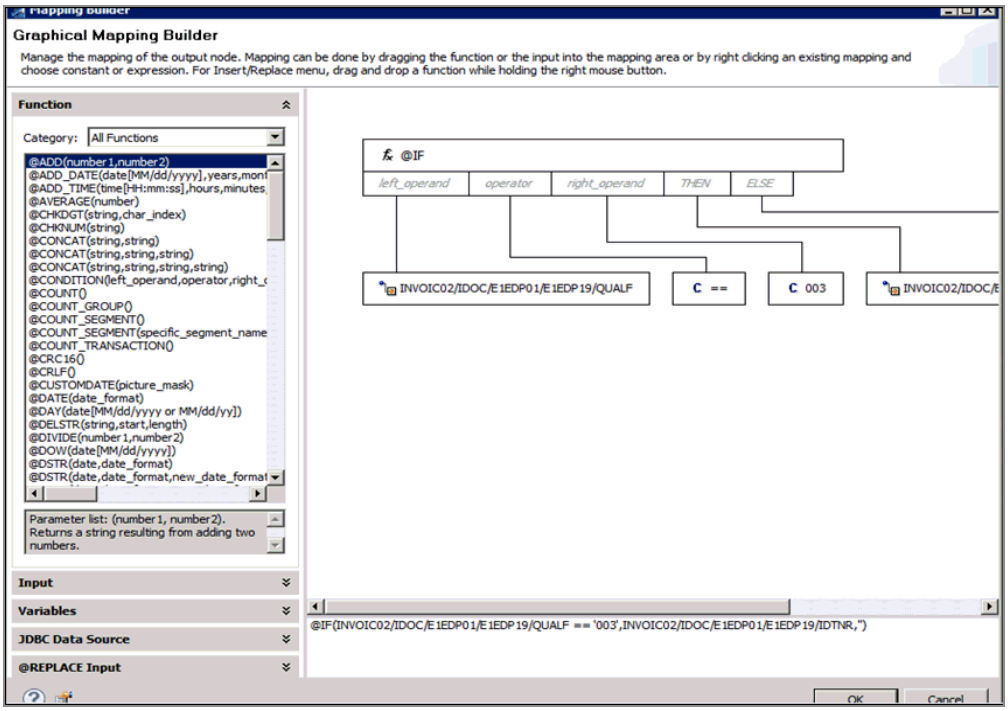
19. Map the qualifier constant to the IT108 element.

```
@IF(INVOIC02/IDOC/E1EDP01/E1EDP19/QUALF == '003',' IN','')
```

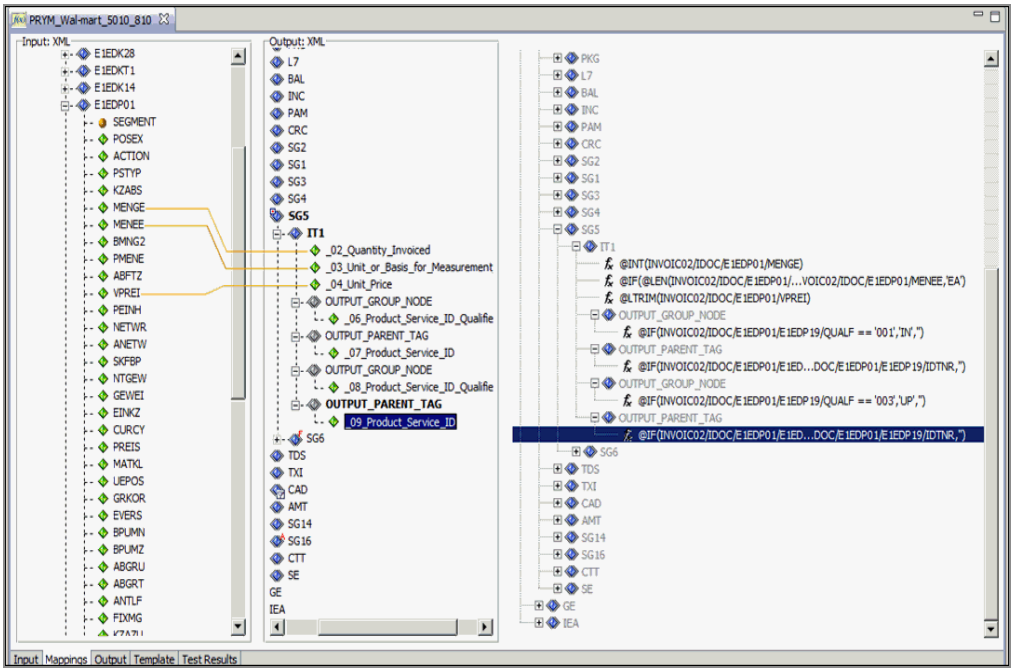


20. Map the IDoc value to the IT109 element.

```
@IF(INVOIC02/IDOC/E1EDP01/E1EDP19/QUALF == '003','
INVOIC02/IDOC/E1EDP01/E1EDP19/IDTNR ','')
```

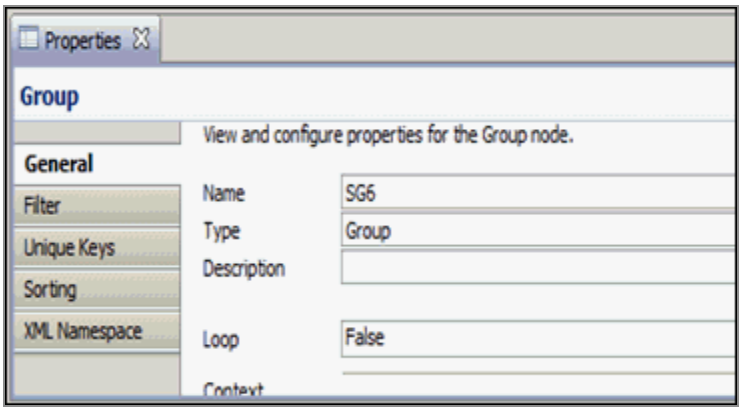


Your iIT interface should resemble the following image:

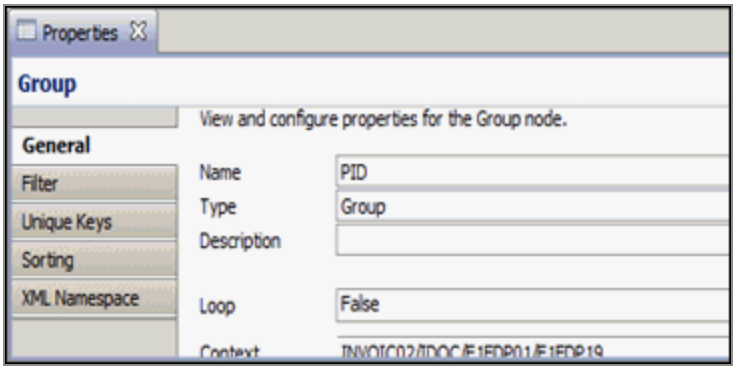


Product Item Description Segment (PID)

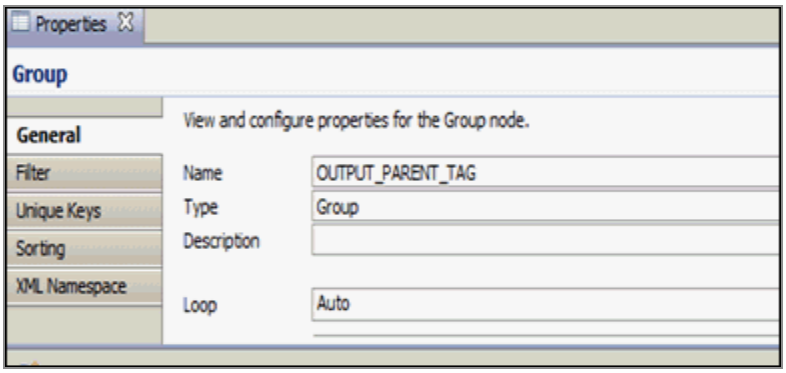
1. In the Output Tag Properties dialog box for the SG6 segment, set the looping.



2. In the Output Tag Properties dialog box for the PID segment, set the following properties:



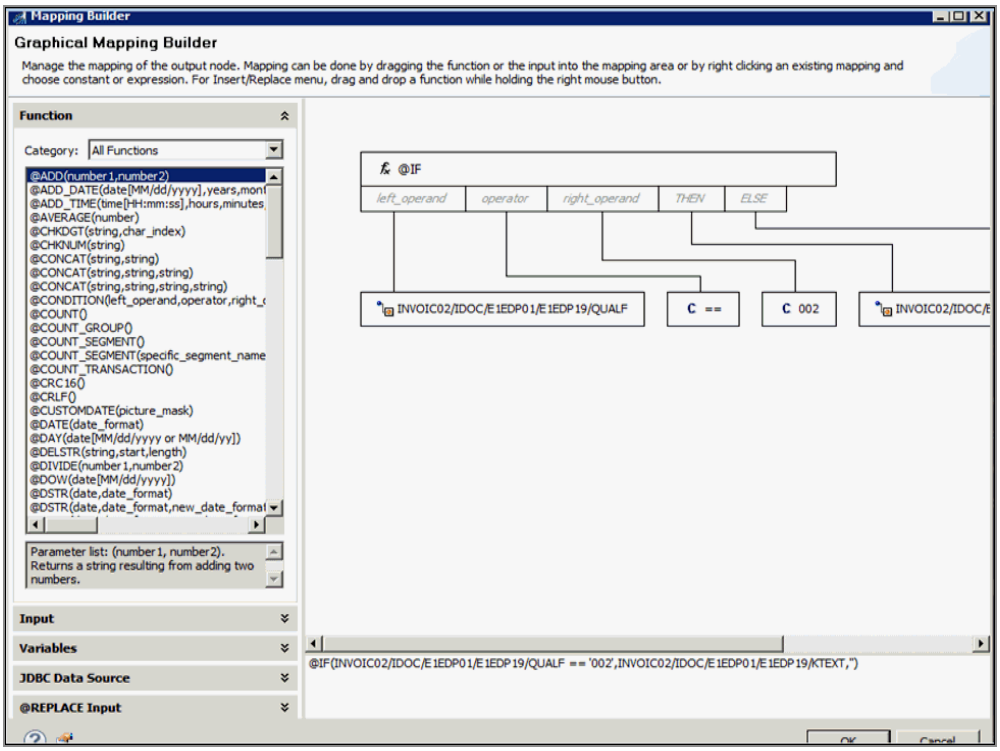
3. Add an output group node to the PID segment and rename it to OUTPUT_PARENT_TAG.
4. In the Output Tag Properties dialog box for the output group node, set the following properties:



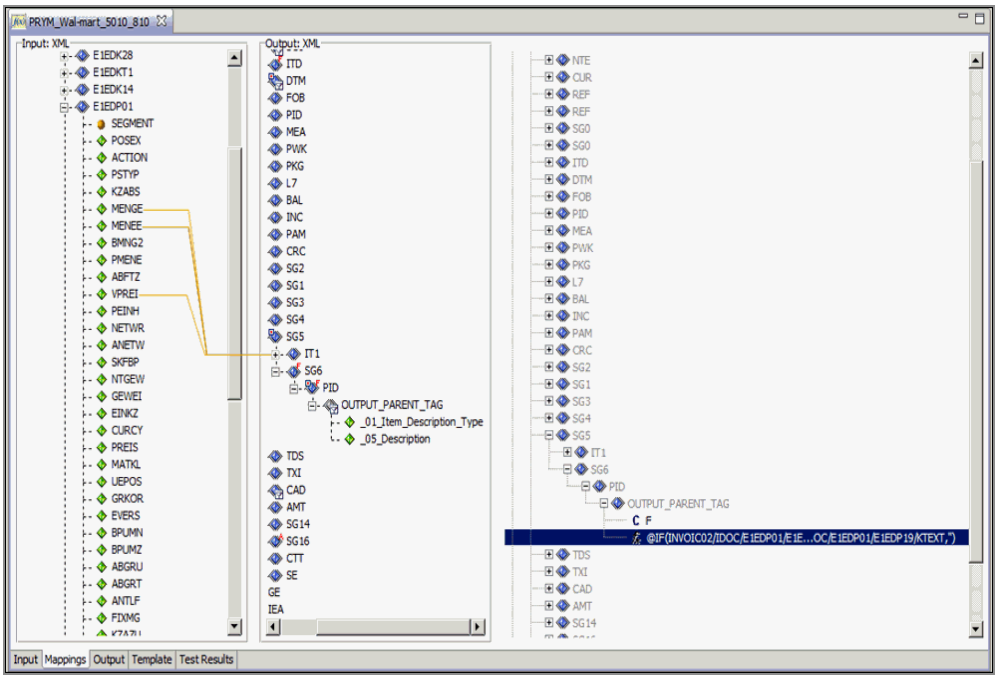
5. Map a constant F to the PID01 element.

6. Map the following to the PID05 segment:

```
@IF(INVOIC02/IDOC/E1EDP01/E1EDP19/QUALF ==
'002',INVOIC02/IDOC/E1EDP01/E1EDP19/KTEXT,'')
```



Your iIT interface should resemble the following:



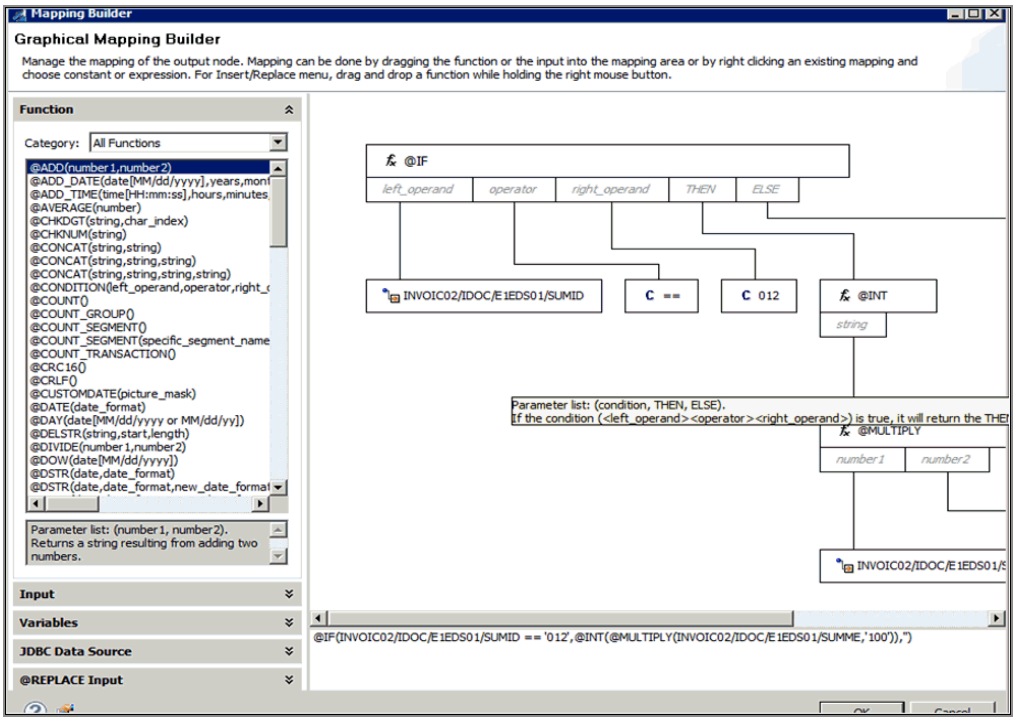
Mapping the Invoice Summary Section

This topic describes how to map the Invoice Summary section.

Total Monetary Value Segment (TDS)

1. Map the following to the TDS01 element:

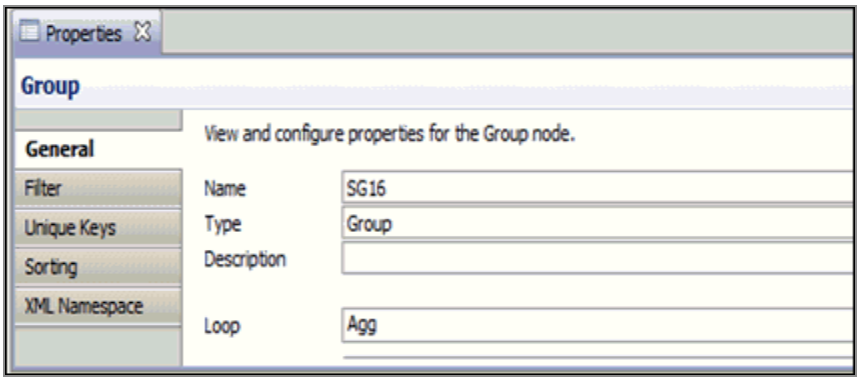
```
@IF(INVOIC02/IDOC/E1EDS01/SUMID == '012',@INT(@MULTIPLY
(INVOIC02/IDOC/E1EDS01/SUMME,'100')),'')
```



The integer value is taken after multiplying the Invoice total by 100, since there is no decimal masking in XML and to avoid rounding errors.

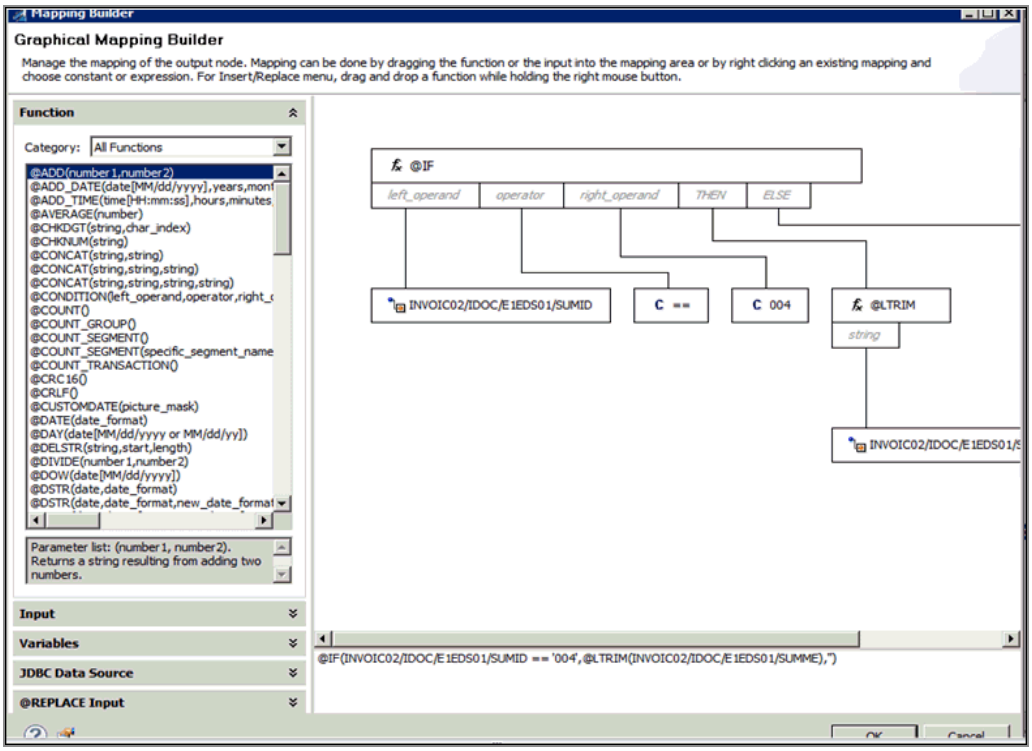
Invoice Shipment Summary Segment (ISS)

1. Expand the SG16 segment to show the ISS segment.
2. In the Output Tag Properties dialog box for the SG16 segment, set the looping:

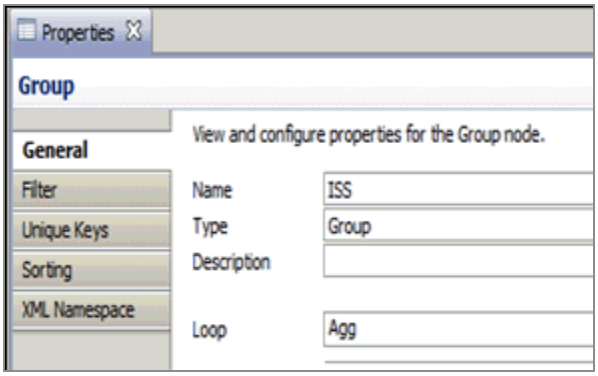


3. Map the following to the ISS01 segment:

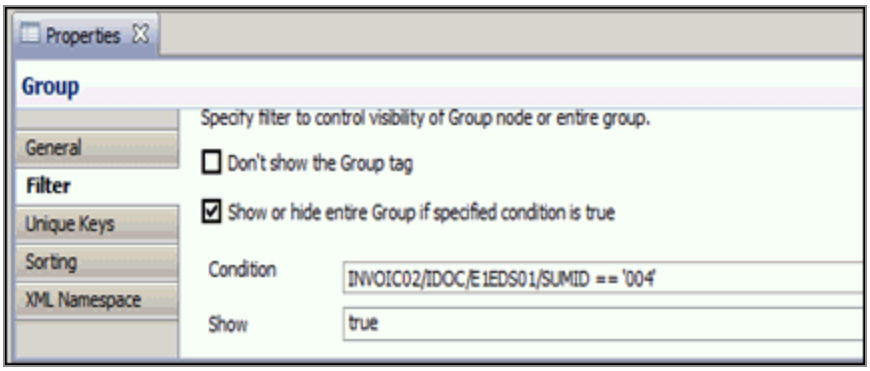
```
@IF(INVOIC02/IDOC/E1EDS01/SUMID == '004',@LTRIM
(INVOIC02/IDOC/E1EDS01/SUMME),'')
```



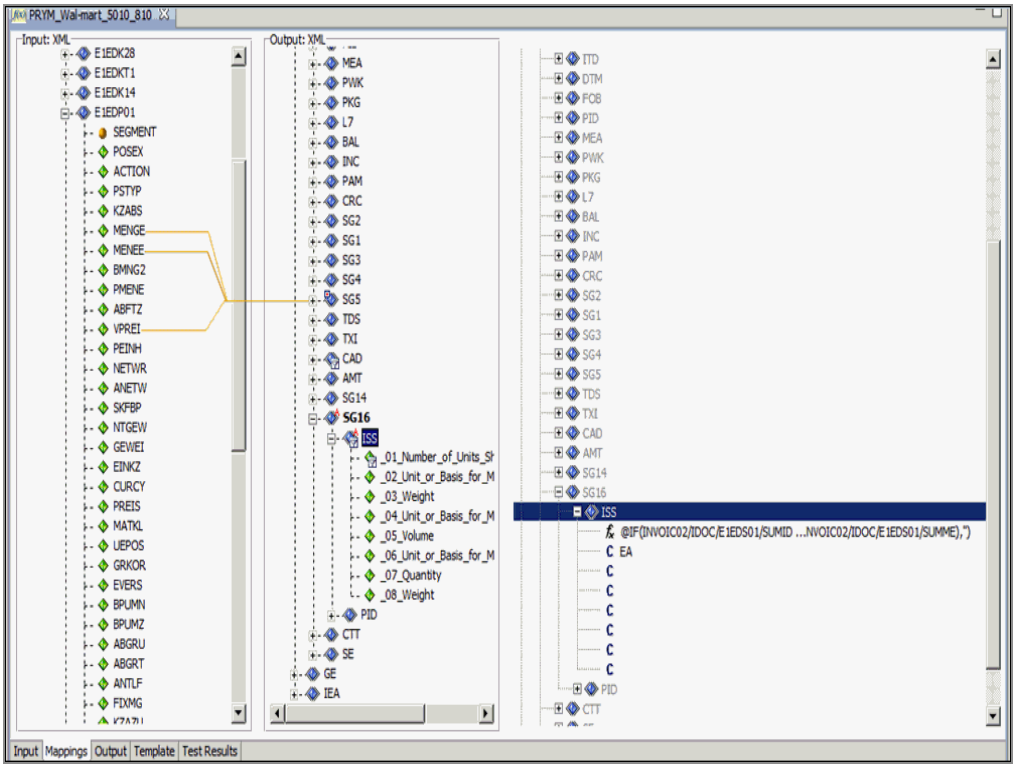
4. Map the constant EA to the ISS02 element.
5. In the Output Tag Properties dialog box for the ISS segment, set the following properties:



6. Click the **Filter** tab and set the following properties:



Your iIT interface should resemble the following:



7. Save the Transform project.

Testing the Transform Project

This section describes how to test the Transform project that was created.

1. Click the **Test Transform** tab.

The transform output results appear in the Test Transform tab.

2. Click the **Save** icon to save the XML output file.

   This XML output file can be used as an input document for outbound processing (XML to EDI) in iWay Service Manager. For more information on how to build an outbound channel, see Outbound Processing: XML to EDI X12.

   The following is a sample EDI output file:

```
ST*810*0001
BIG*20081002*0090689331*20081001*6100051905
REF*IA*VENDORnum
REF*DP*Deptnum
N1*SU*YOUR CUSTOMER NAME*9*012345678
N3*901 SOUTH ST
N4*CITY*NY*01234
N1*ST*WAL-MART DC 6011D DSDC DEPT 19*UL*0078742028682
N3*2200 MANUFACTURERS BOULEVARD
N4*BROOKHAVEN**39601
ITD*08*3*1.50**35
DTM*011*20081002
FOB*CC
IT1**10*EA*2.00**IN*005201460*UP*036346317427
PID*F****5201460 Schmetz Needles Embroidery
IT1**40*EA*1.20**IN*005201544*UP*036346317113
PID*F****5201544 Schmetz Neelde Asst
.
.
.
IT1**18*EA*0.90**IN*005202623*UP*072879104325
TDS*478594
CAD*T***9999*PUT SCAC CODE HERE**BM*GRN0571922196
ISS*7848*EA
SE*1*0001
```

Tutorial: Mapping an IDOC to an Advanced Ship Notice (ASN)

This topic provides a tutorial that demonstrates how to map an IDOC to an Advanced Ship Notice (ASN) using iWay Integration Tools (iIT).

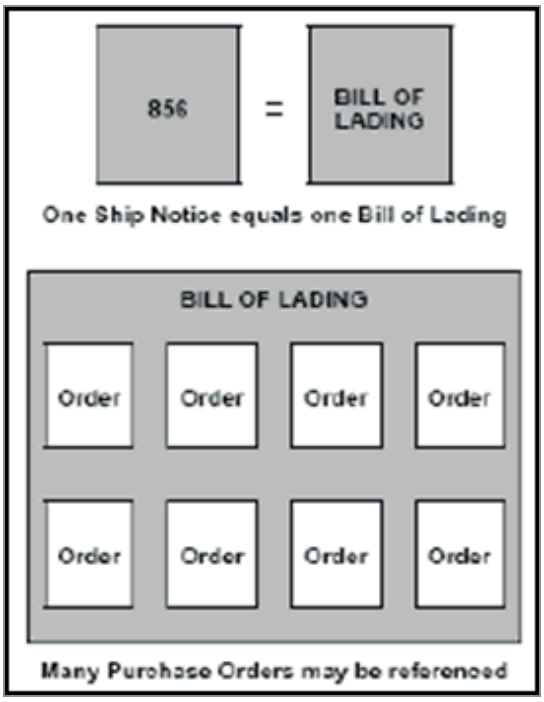
Note: For your convenience, the ASN_With_Variables.zip file is attached to this PDF, which contains sample files that can be used with this tutorial. For PDF-compatibility purposes, the file extension of the ASN_With_Variables.zip file is temporarily renamed to .zap. After saving this file to your system, you must rename this extension back to .zip.

EDI X12 ASN Mapping Tutorial Overview

From the ANSI X12 Standards Board:

"This Draft Standard for Trial Use contains the format and establishes the data contents of the Ship Notice/Manifest Transaction Set (856) for use within the context of an Electronic Data Interchange (EDI) environment. The transaction set can be used to list the contents of a shipment of goods as well as additional information relating to the shipment, such as order information, product description, physical characteristics, type of packaging, marking, carrier information, and configuration of goods within the transportation equipment. The transaction set enables the sender to describe the contents and configuration of a shipment in various levels of detail and provides an ordered flexibility to convey information. The sender of this transaction is the organization responsible for detailing and communicating the contents of a shipment, or shipments, to one or more receivers of the transaction set. The receiver of this transaction set can be any organization having an interest in the contents of a shipment or information about the contents of a shipment."

When the merchandise is packaged and put on the truck, the ASN is the EDI document that is sent to the recipient. The ASN tells the recipient what is contained in the shipment, down to the level of what merchandise is in each carton.

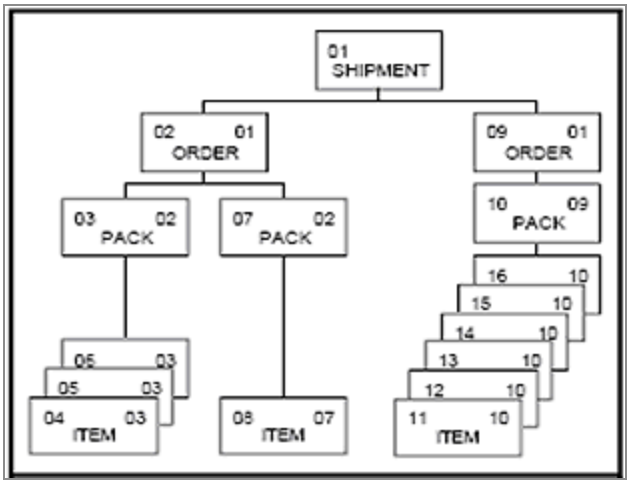


Each ASN is a single shipment. There can be one or multiple shipments on a truck, and each one would have an associated ASN. Each shipment represents a single batch of merchandise bound for a single Ship-To location. Each shipment will contain one or multiple orders. An order can comprise of one or many cartons. Each carton (usually) has a label on it with a bar code. This label physically ties the carton to the EDI document. The following is an example of a label.

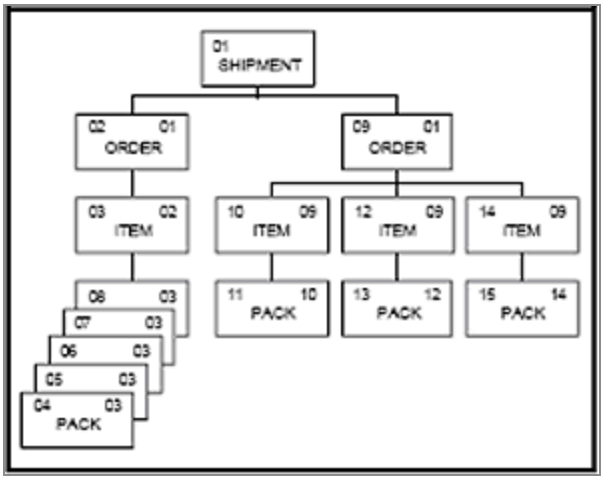
Zone A Zone title: FROM Height: 0.8" Width: 1.25" Data Content: Ship from Name and Address Text size: 8 to 10 Pts	**Zone D** Zone title: Carrier: Height: 1.0" Width: 1.5" Data Content: Carrier name SCAC Text size: 10 to 16 Pts	**Zone H** Zone title: FOR Height: 1.2" Width: 1.5" Data Content: Store number and address Text size: 12 Pt
Zone B Zone title: TO: Height: 0.8" Width: 2.75" Data Content: Ship to Name and Address Text size: 12 to 14 Pts	**Zone E** Zone title: (not applicable) Height: 1.2" Width: 4" Data Content: 9 digit PO number (24 pt) Style, color, size* (10 pt) Quantity in Carton (10 pt)	**Zone I** Zone title: (not applicable) Height: 2" Width: 4" Data Content: Serial Shipping Container Code Bar Code height: 1.25" minimum X Dimension 0.020" minimum
Zone C Zone title: Ship to Postal code Height: 1.0" Width: 2.5" Data Content: Ship to Postal bar code Bar Code height: 0.5" minimum Text size: 8 to 10 Pts	**Zone G** Zone title: Store number Height: 1.2" Width: 2.5" Data Content: 3 digit Store number Text size: 36 Pt Bar Code height: 0.5" minimum X Dimension 0.020" minimum	**NOTES** * Style, Color and Size (up to 12 bytes each) * Multi SKU (style, color, size)cartons must include a content label

ASNs are typically packed using one of two methods. **Pick-Pack** is a multiple quantity of one or more sku in a carton. **Standard Pack** is all items in the carton are the same.

The following diagram is a Pick-Pack example.



The following is a Standard Pack example.



You may find pick-pack in the mass-market industries, such as garment. A single box contains three dozen shirts in an assortment of sizes and colors. An example of standard pack is the grocery business, where a single carton of cereal contains 12 boxes of the same item.

As you can see from the two diagrams, the ASN contains several levels of nested information. The ASN levels are called Shipment, Order, Tare, Pack, and Item.

Shipment contains data that reflects all of the goods being transported.

- Where are they being shipped?

- Who shipped them?
- Ship date
- Total number of cartons
- Total shipment weight
- Shipment number
- Bill of lading number of the trucker

There is one Shipment level per ASN. Next is the Order level. There can be multiple Order levels on a single ASN. Each Order level corresponds to a purchase order from the trading partner. This level includes:

- Purchase order number
- Order date
- Ordering store or location
- Department number
- Total number of cartons

Beneath the Order level, depending on the type of ASN, will be one or several looping structures that denote the packaging and what is inside the packaging. A tare is usually a pallet full of boxes that has been sealed with plastic (shrink) wrap. The pallet has a single bar code, called a license plate. The boxes on the pallet may or may not be individually labeled, but the pallet is usually not broken down until it reaches its final destination. A Tare level usually contains a bar code number.

The Pack level contains information about the carton. In most cases, it contains nothing more than a bar code number.

For common carrier, this would be the UCC128 (GS1) 20 digit bar code. For small package services such as Federal Express or UPS this would be the package tracking number.

The Item level contains information about the merchandise.

- Item number
- UPC
- Style
- Color
- Size

- Item weight
- Inner pack/outer pack
- On an 857, unit price

A pick-pack style ASN is ordered SOPI (Ship, Order, Pack, Item). A standard pack ASN would be ordered SOIP (Ship, Order, Item, Pack). Pick-pack has one or many items per carton, and standard pack has one or many cartons per item.

At the beginning of each segment is the HL segment.

- HL01 contains a counter. The shipment level is always 1. Each HL segment afterwards increments by 1. The CTT segment at the bottom of the document, after the last HL, contains the same counter as the last HL.
- HL03 contains a constant that tells you the level (S,O,T,P, or I).
- HL02 contains a pointer. The pointer is the location of the parent level to the current level. The shipment level has no parents, and contains nothing in this element. The order segments all point to the shipment segment, so they all contain "1" in this element. The tare, pack and item levels all point to the higher level. In the diagrams on pages 8 and 9, each box (level) contains two numbers. The left number is the sequence number (HL01) and the right number is the pointer to the higher level (HL02).

Not all levels are required. It is perfectly fine to have SOI (no pack labels), although you would probably show the items in summary rather than box-by-box details. You could also have SOT (tare with no item details). The contents and details are subject to agreement by the partners.

If you cannot figure out the type of looping, BSN05 usually contains a code that will tell you.

- 0001 – Ship, Order, Pack, Item (Pick-Pack)
- 0002 – Ship, Order, Item, Pack (Standard Pack)
- 0003 – Ship, Pack, Order, Item
- 0004 – Ship, Order, Item

Unfortunately BSN05 is not a mandatory element.

This tutorial guides you through the following steps.

1. Creating a Transform project for processing an XML IDoc into an Advanced Ship

Notice (ASN).

2. Creating constant and direct mappings for your Transform project.
3. Using filters for mapping segments.
4. Setting context for mapping segments.
5. Using variables to count HL segments.
6. Using variables to count the total number of cartons.
7. Testing your Transform project.

Creating a New Transform Project

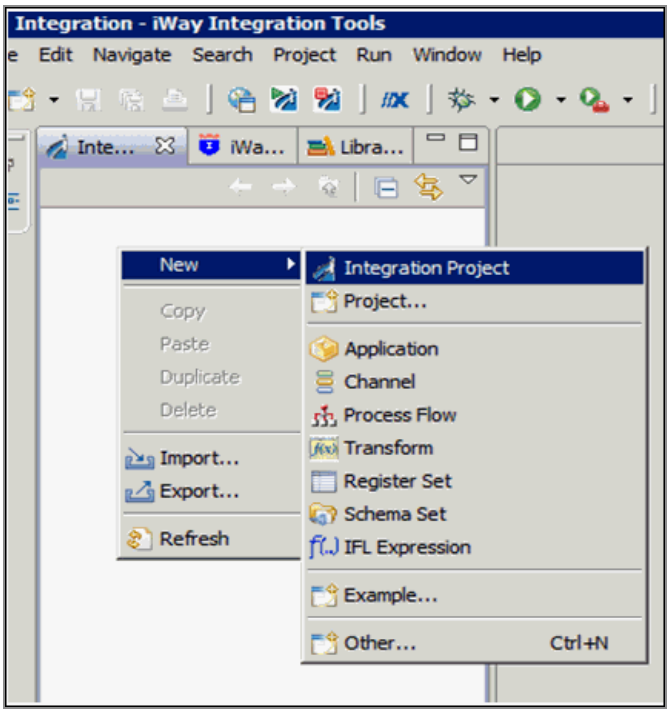
This section describes how to create a new Transform project.

Create a New Transform Project

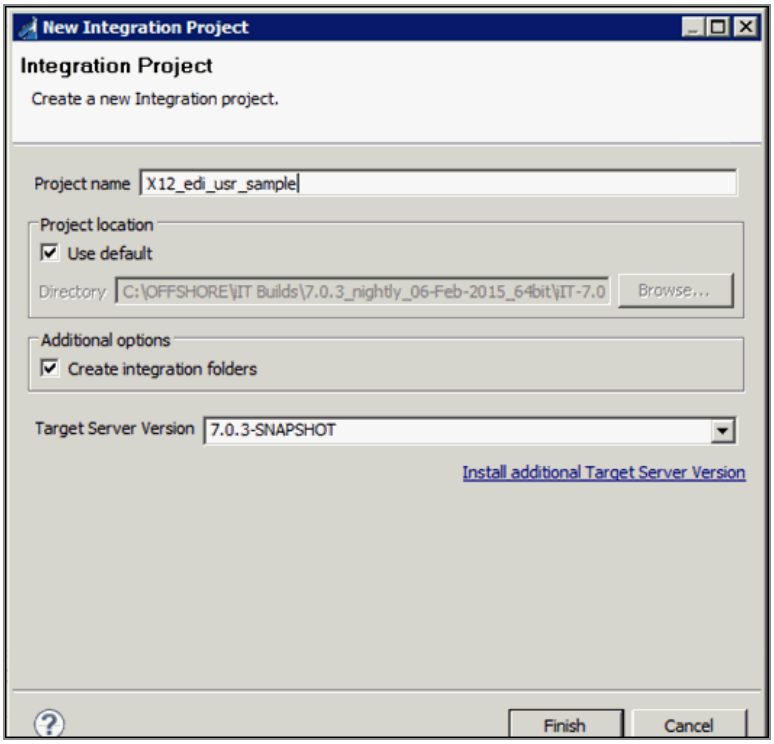
To create a new Transform project:

Procedure

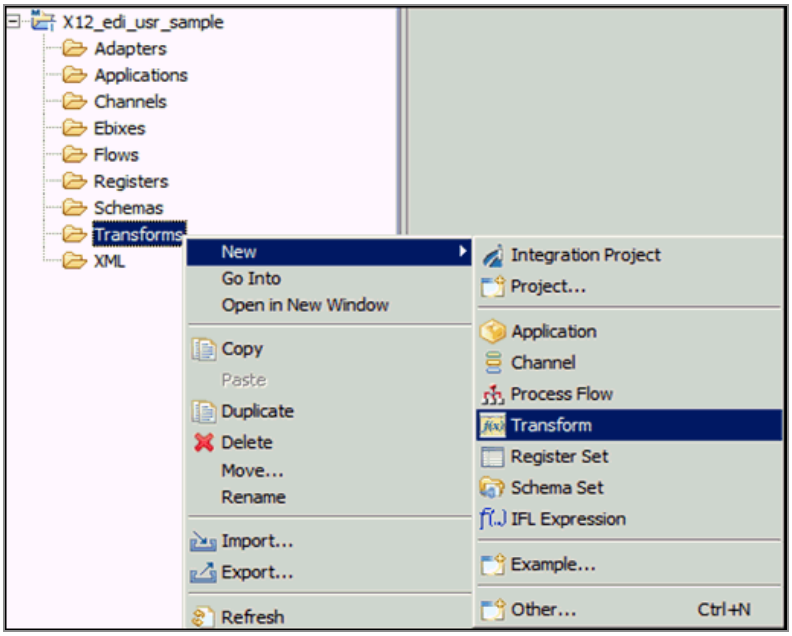
1. Open iWay Integration Tools (iIT).
2. Right-click the Integration explorer window, select **New**, then select **Integration Project**.



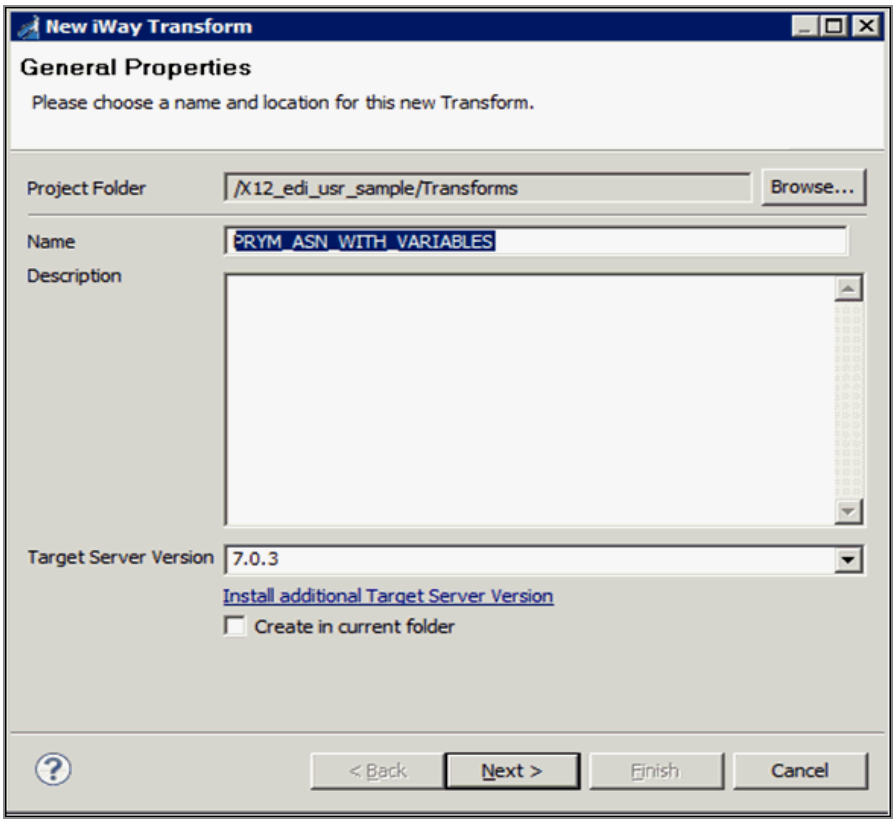
3. In the New Integration Project window, provide a Project name and click **Finish**.



4. Right-click the **Transforms** folder/Integration project, select **New**, and click **Transform**.

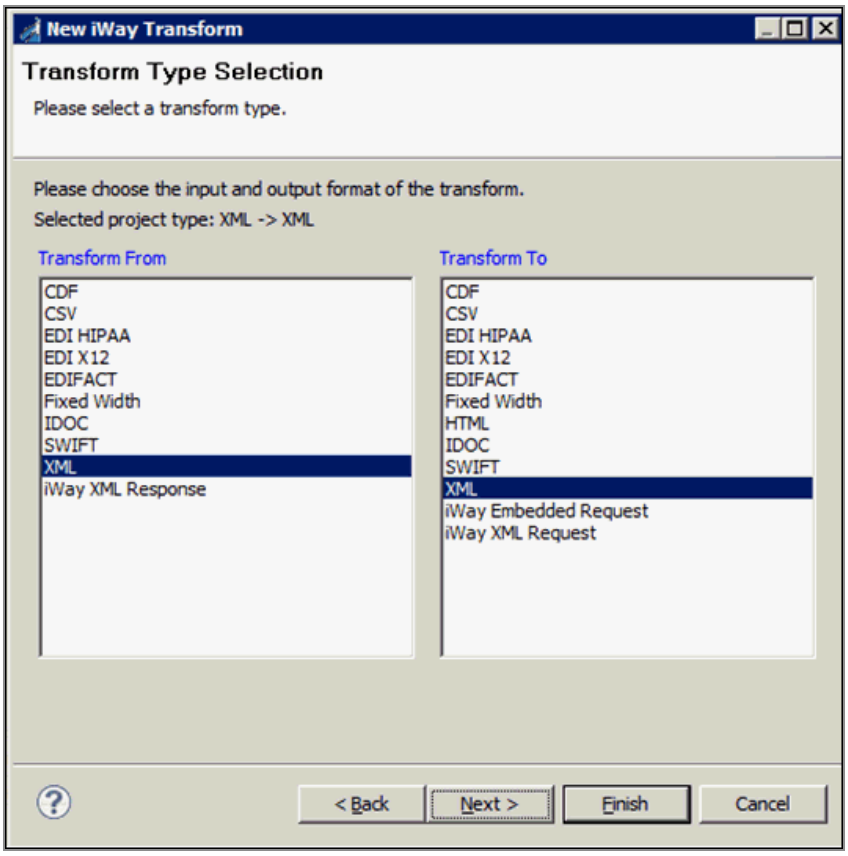


5. In the Name field, type a name for your new project, for example, PRYM_ASN-WITH_VARIABLES.



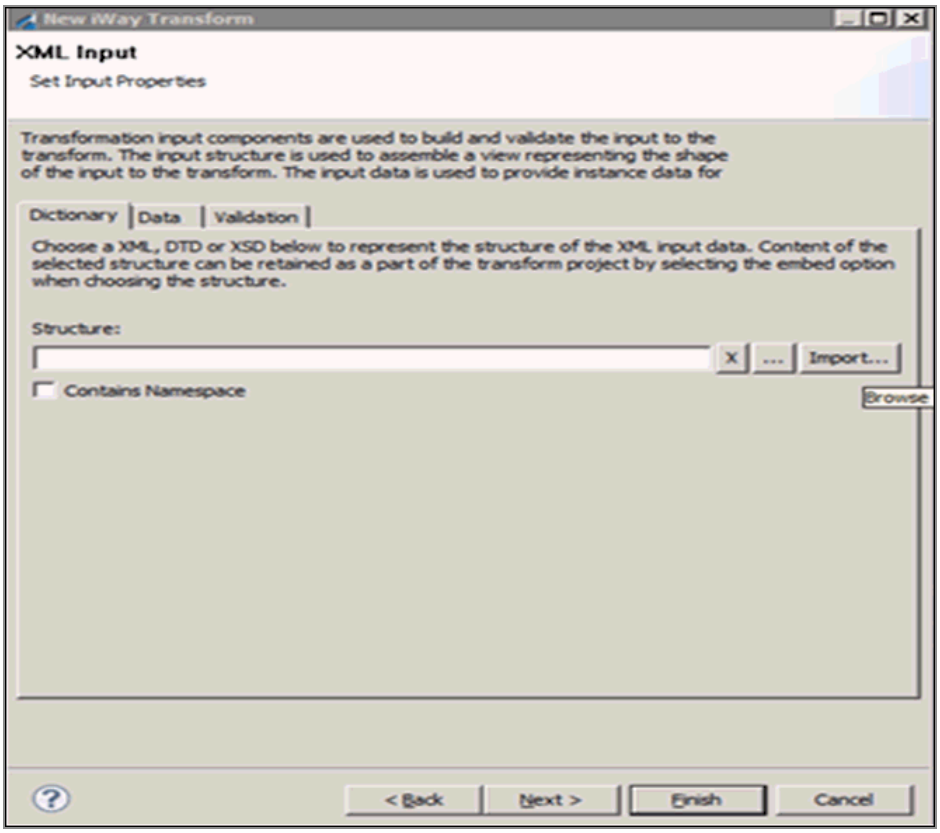
6. In the Description field, type a project description (optional).
7. Click **Next**.

   The Transform Type Selection dialog box opens as shown in the following image.

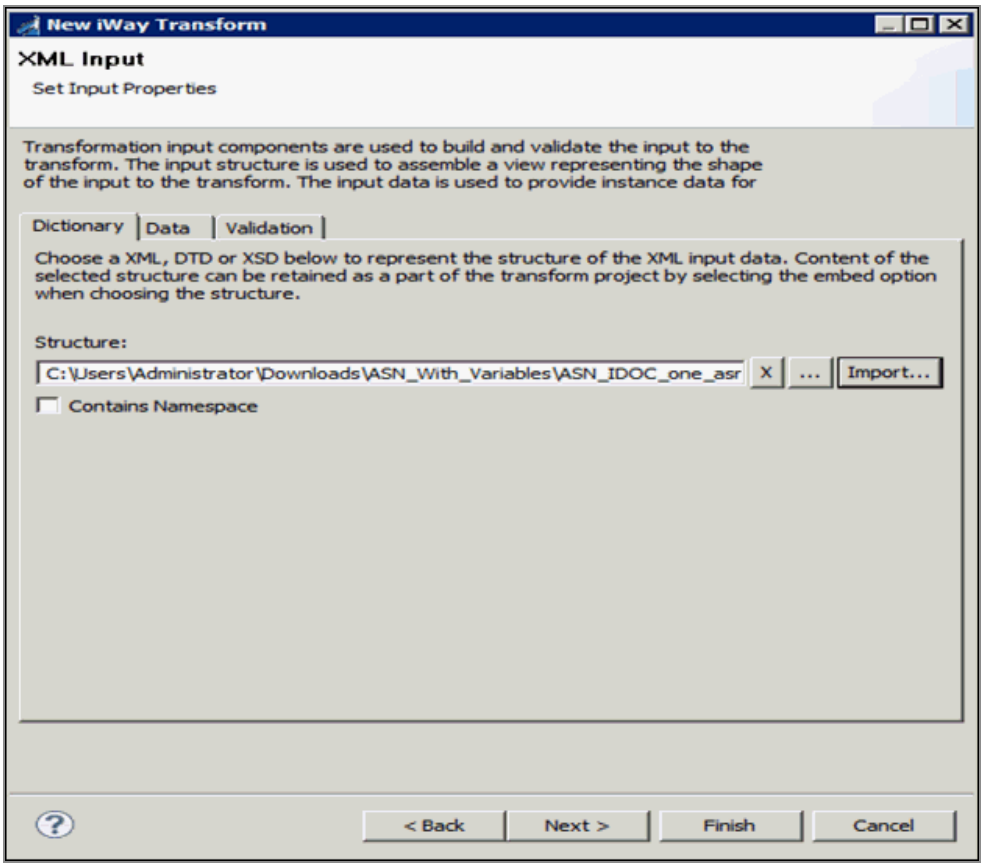


8. From the list in the Transform From pane, select the format of your input, for example, XML.
9. From the list in the Transform To pane, select the format of your output data, for example, XML.
10. Click **Next**.

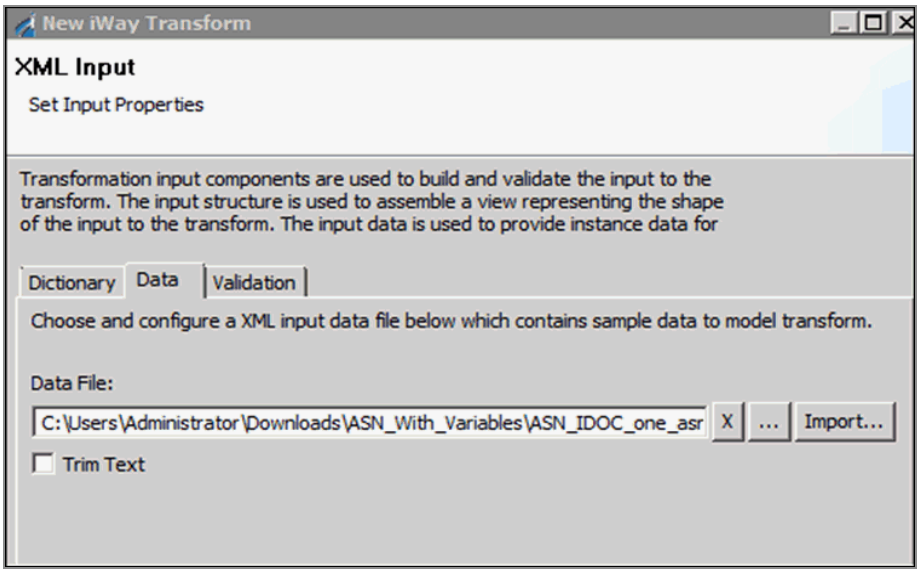
    The XML Input properties dialog box opens with the Dictionary tab active, as shown in the following image.



11. In the Structure field, enter the location of the sample IDoc file in XML format, which is used as the dictionary, or click the **Import** button to import the sample IDoc file to the transform project.
12. Select the sample IDoc file in XML format, for example, ASN_IDOC_one_asn.xml.

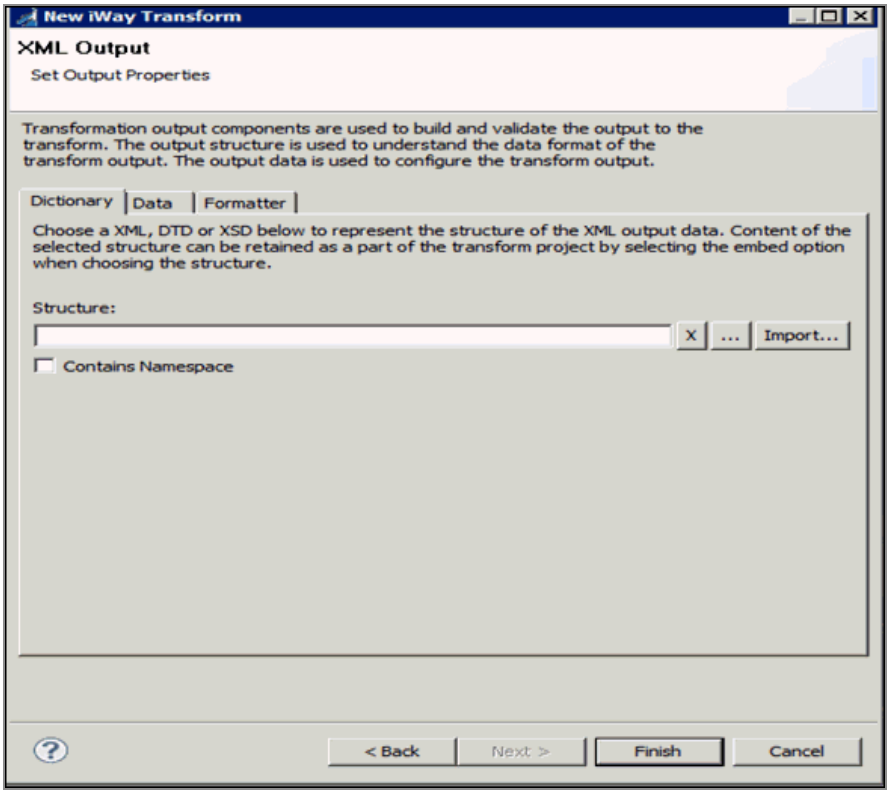


13. Click the Data tab and browse to the location of the sample IDoc file in XML format, which is used as the input data file.



14. Click **Next**.

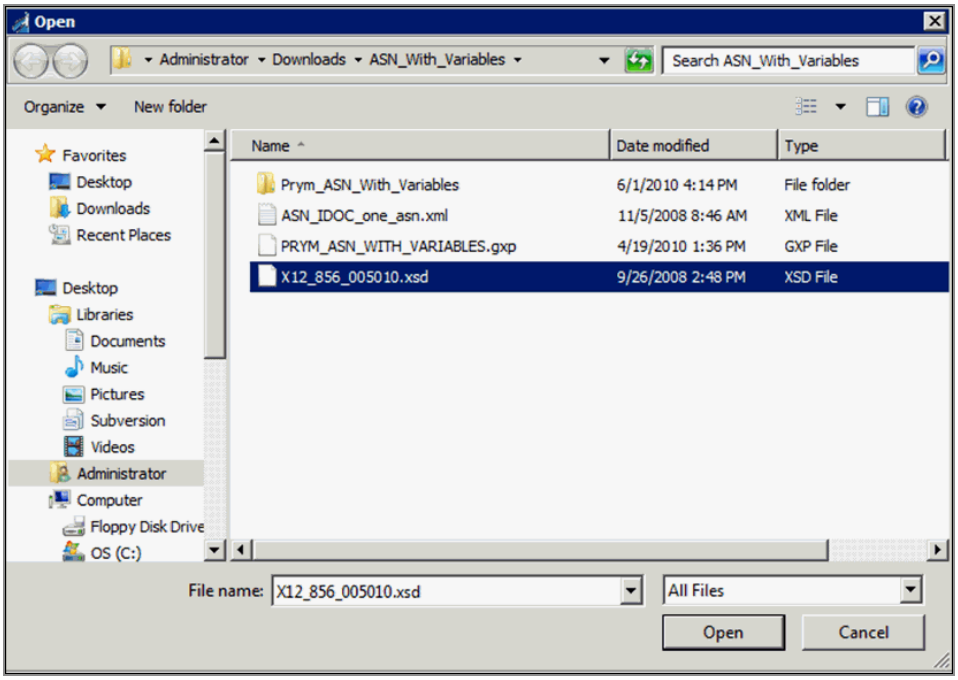
    The XML Output properties dialog box opens.



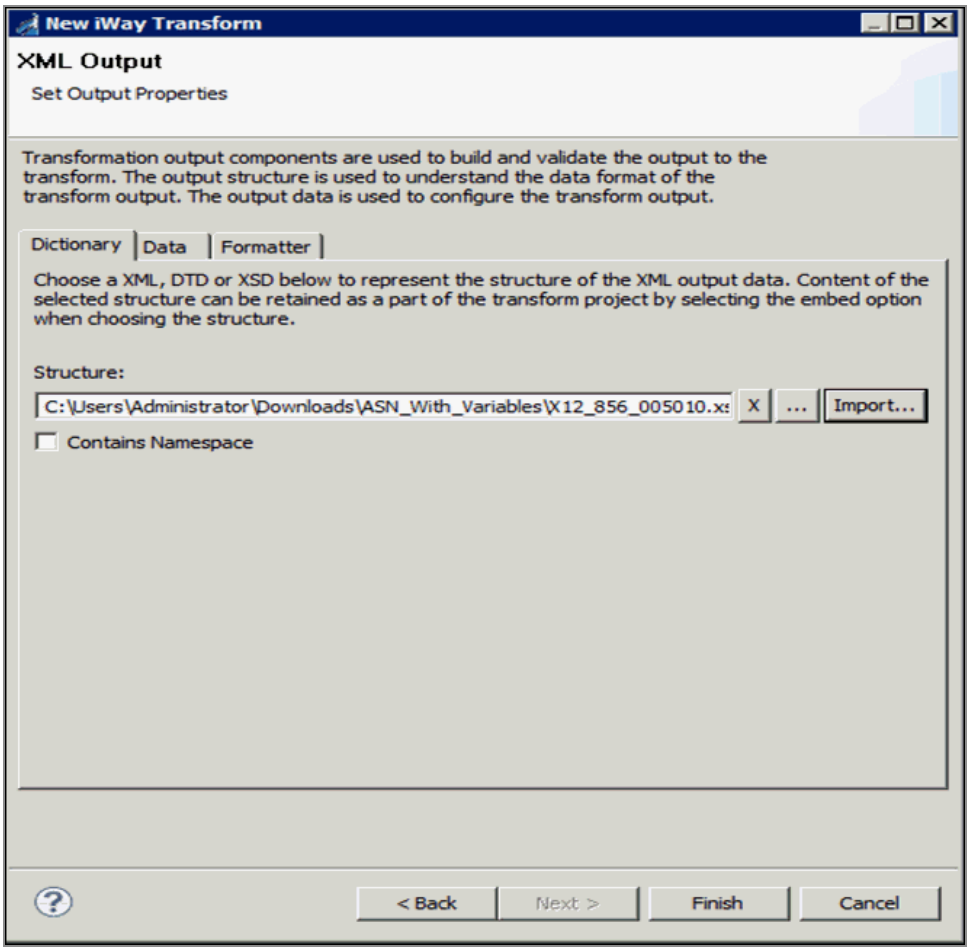
15. In the Structure field, enter the location of the sample EDI XML schema document

(.xsd file) which is used as the dictionary, or click the **Import** button to import a sample EDI XML schema file to the transform project.

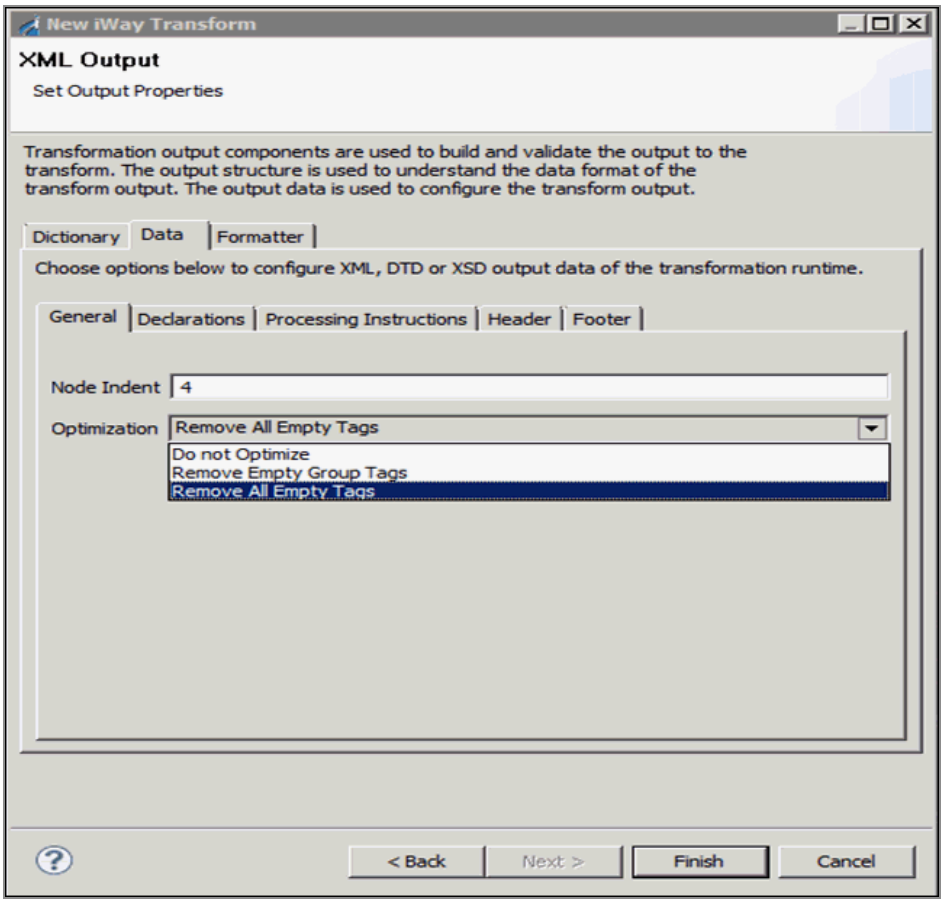
The Open dialog box is displayed.



16. Select the sample EDI XML schema document, for example, X12_856_005010.xsd.
17. Click **Open**.

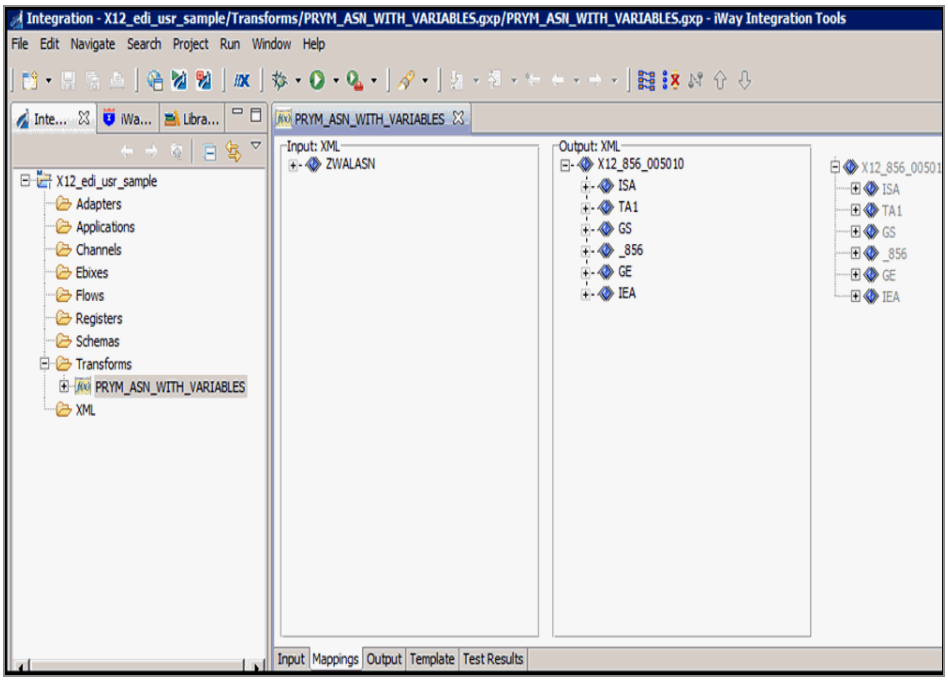


18. Click the **Data** tab.



19. Select **Remove All Empty Tags** from the Optimization drop-down list.
20. Click **Finish**.

    The new Transform project is created in iWay Integration Tools (iIT), as shown in the following image.



21. Click the **Save** icon on the toolbar to Save your project.

    Tip: As a best practice, it is recommended to Save your Transform project often as you proceed through this tutorial.

Maintaining the HL Counters in a Transform Project

You can maintain the ASN level numbers and the parent level numbers using iWay Integration Tools (iIT). The example that is used is a four-level SOPI ASN. You can easily change the number or type of levels if required. For more information, see Flattening the Output Structure.

This approach uses standard functions that are available in iIT. It also uses variables in iIT, which are evaluated at design time and run time.

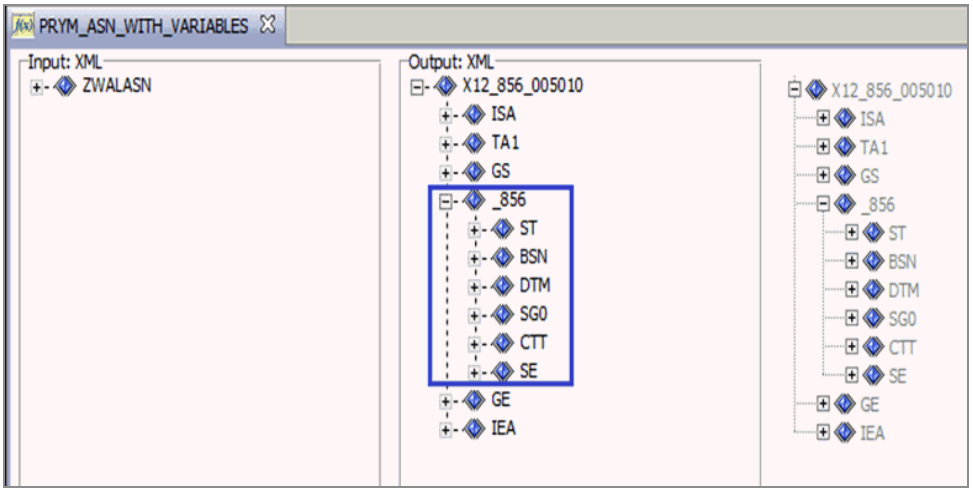
An example of the sample output document can be viewed in The X12 ADN Mapping Final Results.

The HL structures need to be nested in the XML output in order to produce the loops properly. This nesting allows you to know which level you are working in without tracking the level value itself.

Maintain the HL Counters in a Transform Project

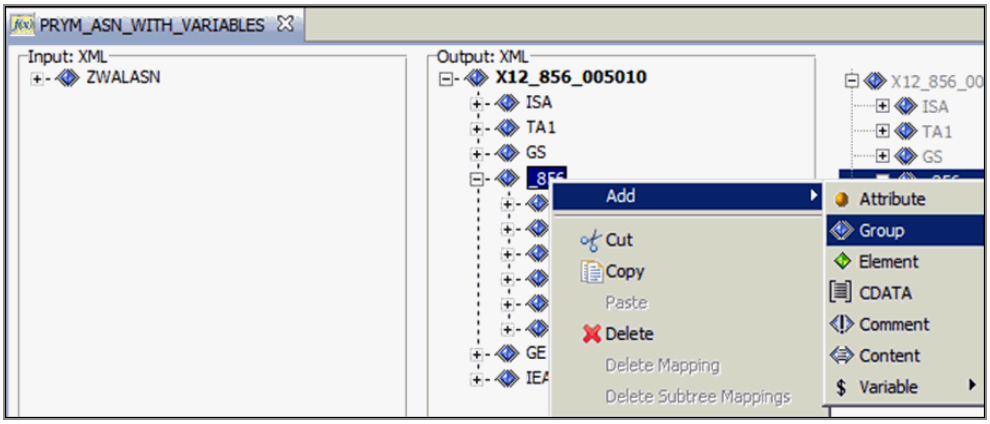
Procedure

1. Expand the **_856** tag in the Output pane.



   You can enrich the output structure with hidden parent group tags to make the looping easier to follow. You can also add a group to be used to count the number of total cartons.

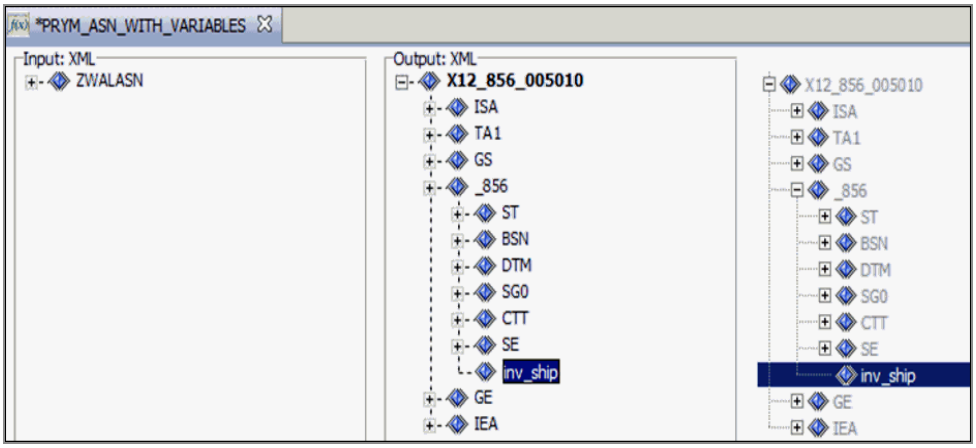
2. Right-click the **_856** tag, select **Add**, and then select **Group**.



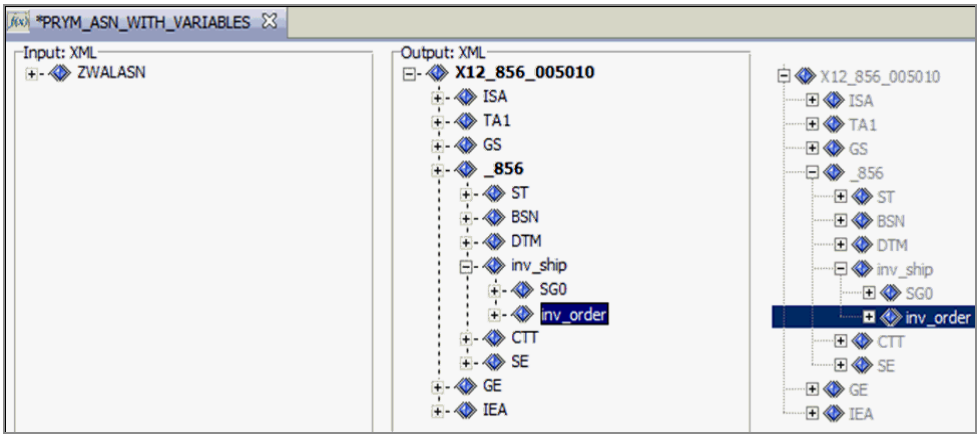
3. Double-click the default name of the added group (OUTPUT_GROUP_NODE) and change it to **inv_ship**.

4. Use the position icons (**Move Up** and **Move Down**) on the toolbar (or right-click and

select from the control menu) to position the new **inv_ship** group below the DTM segment.

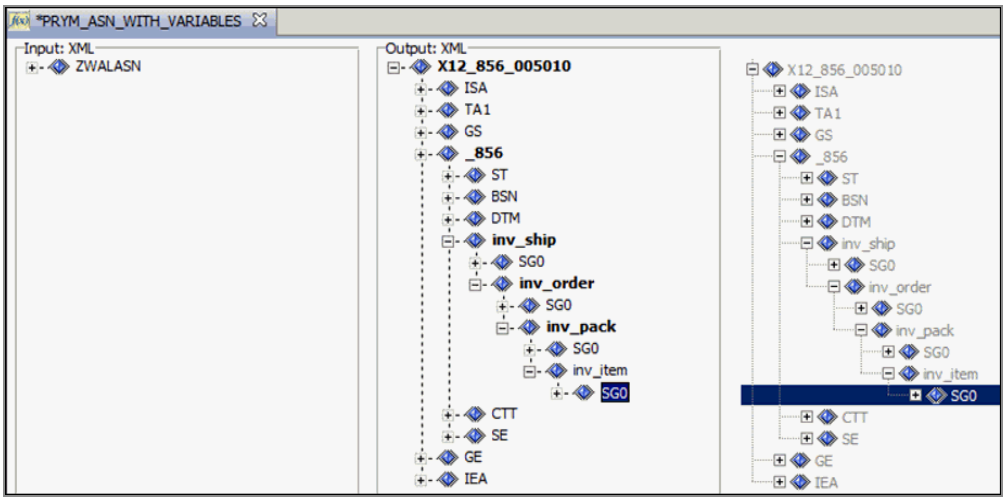


5. Click and drag the **SG0** loop into the **inv_ship** group.
6. Right-click **inv_ship** and add a new group called **inv_order**.



7. Right-click and copy the **SG0** loop, and then right-click and paste the sub-tree under **inv_order**.
8. Right-click **inv_order** and add a new group called **inv_pack**.
9. Right-click and copy the **SG0** loop, then right-click and paste the sub-tree under **inv_pack**.
10. Right-click **inv_pack** and add a new group called **inv_item**.
11. Right-click and copy the **SG0** loop, and then right-click and paste the sub-tree under **inv_item**.

When you are finished, the Output pane now has the following structure:



Showing and Hiding Tags

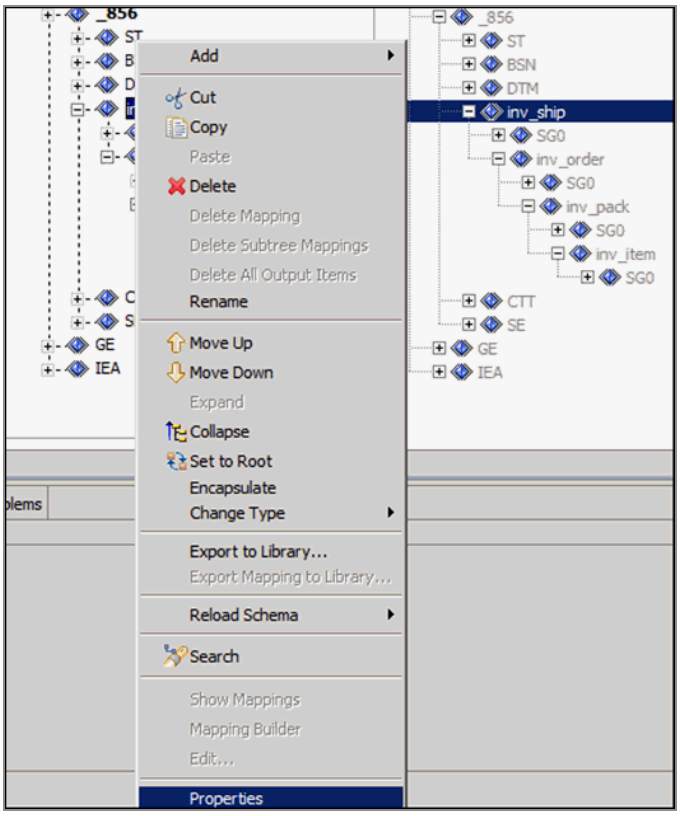
When you add new tags to a structure, you must ensure that relative paths are maintained. The SG0 tags and their looping properties will be affected when you add new parent tags. Invisible group tags exist in the mapping structure, but do not receive output in the XML that is created by iWay Integration Tools (iIT). The blue group (diamond) tag is a visible tag and the gray group (diamond) tag is invisible. These tags are used to help organize your mapping and maintain correct looping properties in iIT.

Show and Hide Tags

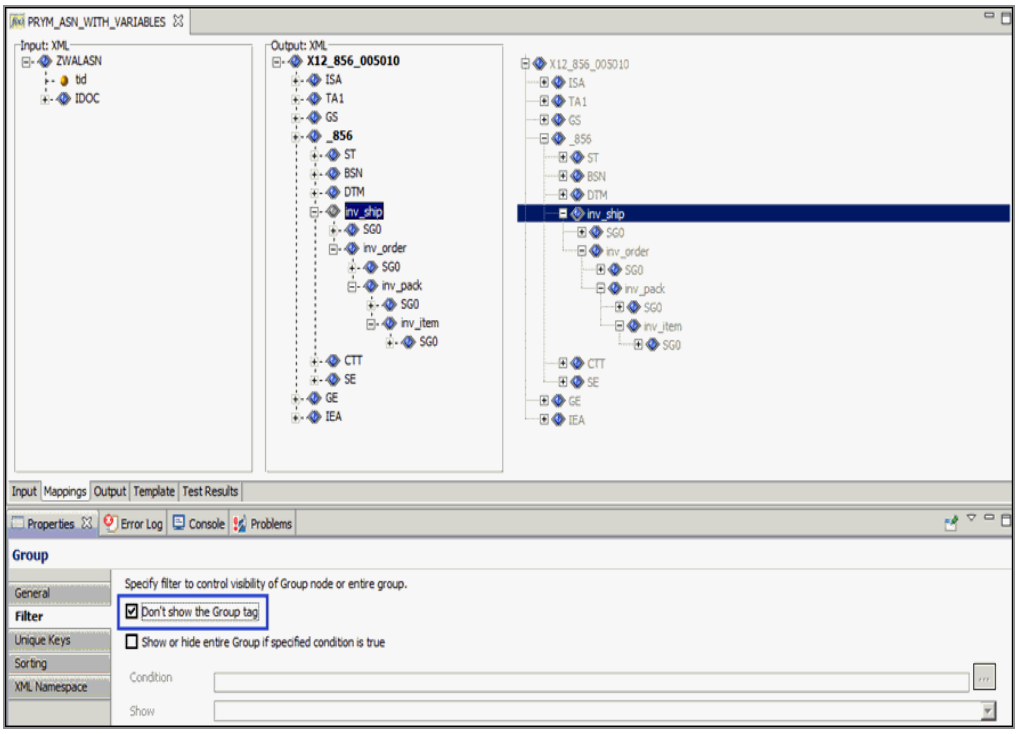
To show and hide tags:

Procedure

1. Right-click the **inv_ship** group in the Output pane and select **Properties** from the context menu.

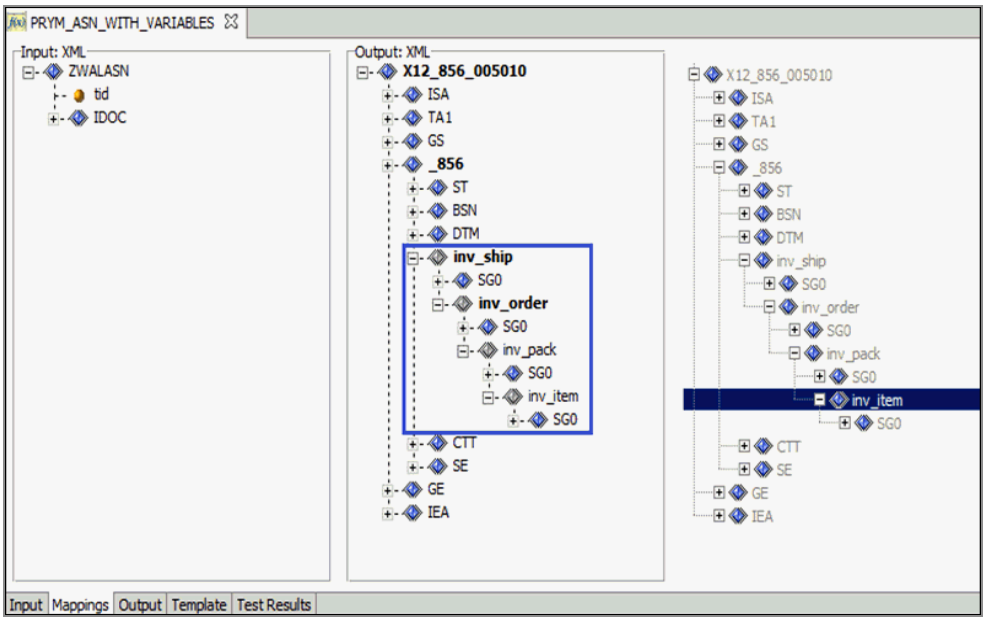


The Output Node Properties - inv_ship dialog box opens.



2. Click the **Filter** tab.
3. Select the **Don't show the Group node** option, and then click **OK**.
4. Repeat this procedure for the remaining groups (inv_order, inv_pack, and inv_item).

   Your screen should now resemble the following image:



Mapping the Control Segments

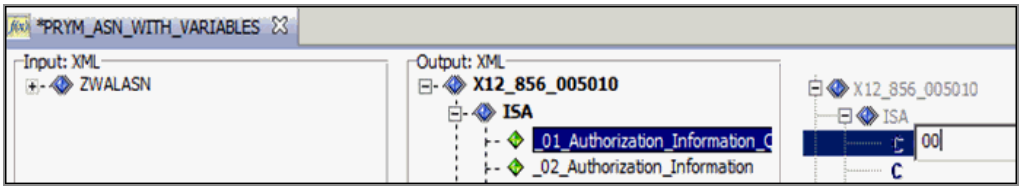
The following sections describe how to map the control segments.

Map ISA and IEA (Interchange) Segments

Since you are not using a trading partner for this exercise, you will hard code the envelope status.

Procedure

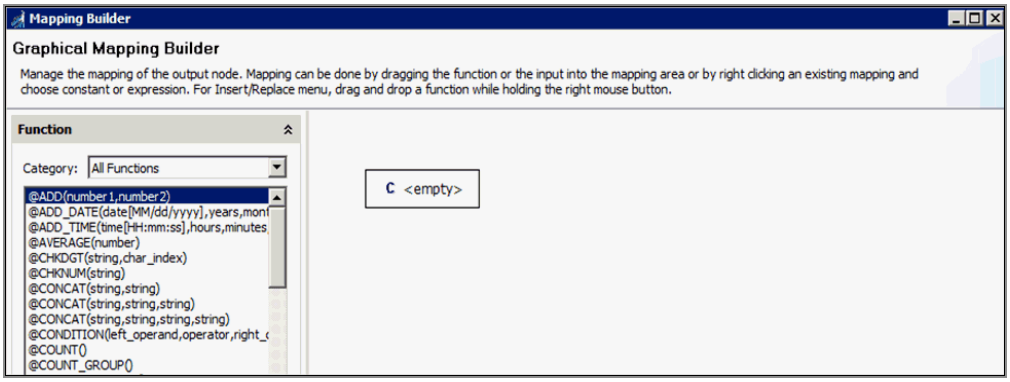
1. Click the empty field in the Mapping Values pane to specify a value for the output node.



You can also double-click the empty field in the mapping values or right-click an

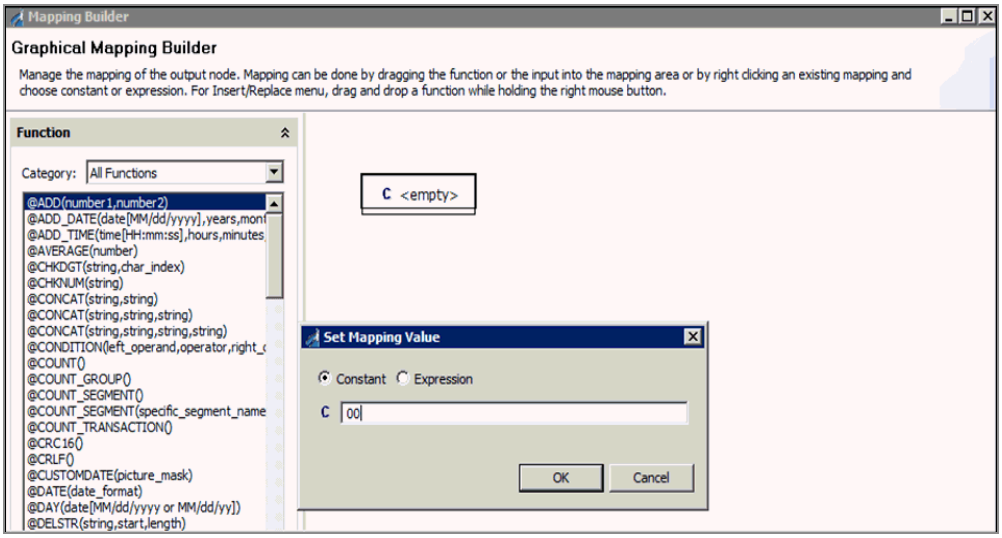
empty field mapping values and select **Mapping Builder** and provide a value for the empty field.

For example:



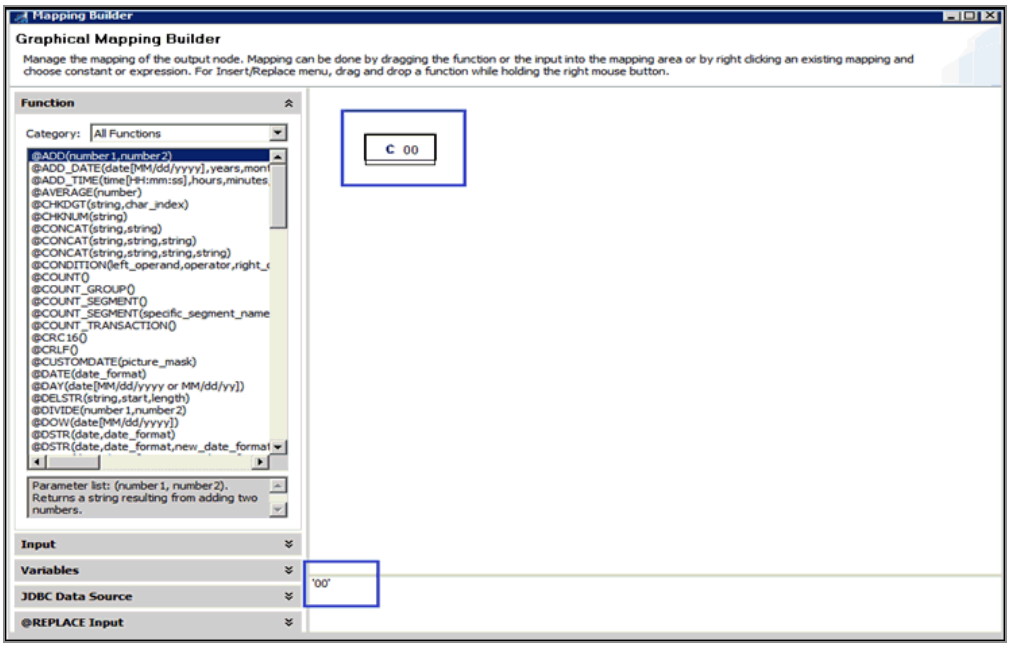
Double-click the empty icon in the Graphical Mapping Builder area and select **Set Constant** from the context menu.

The Set Mapping Value dialog box opens.



Enter a value in the field (for example, 00, which is a constant), and then click **OK**.

You are returned to the Output Node Mapping Builder.



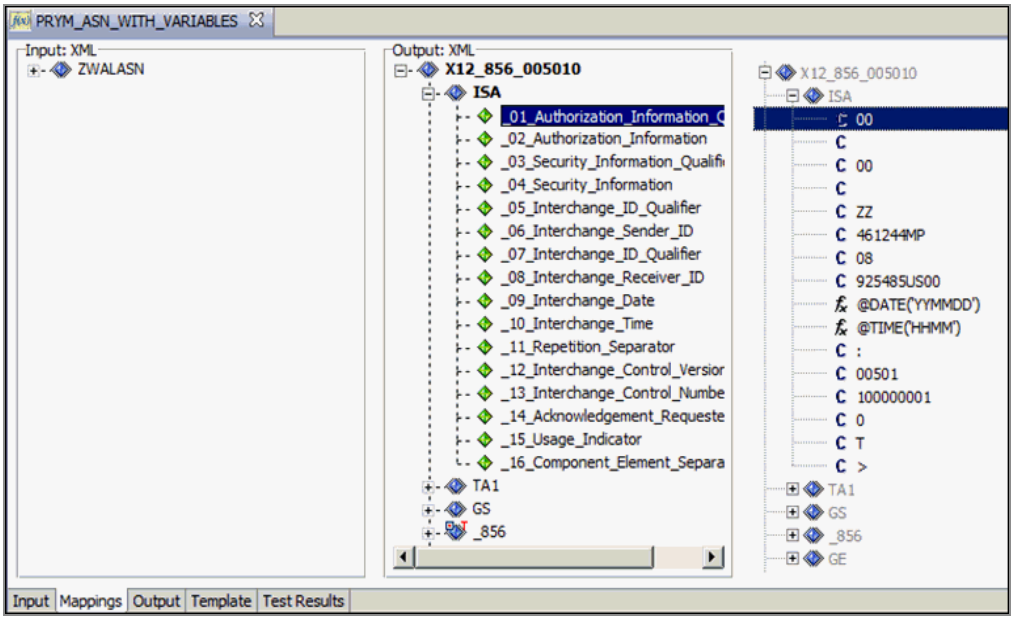
Click **OK** to return to the Mapping Values pane.

2. Enter the values for the ISA and IEA segments that are listed in the following table:

Segment	Value
ISA01	00
ISA03	00
ISA05	ZZ
ISA06	SENDERID
ISA07	ZZ
ISA08	RECEIVERID
ISA09	@DATE('YYMMDD')
ISA10	@TIME('HHMM')

Segment	Value
ISA11	:
ISA12	00501
ISA13	100000001
ISA14	0
ISA15	T
ISA16	>
IEA01	1
IEA02	100000001

Your screen should now resemble the following image:



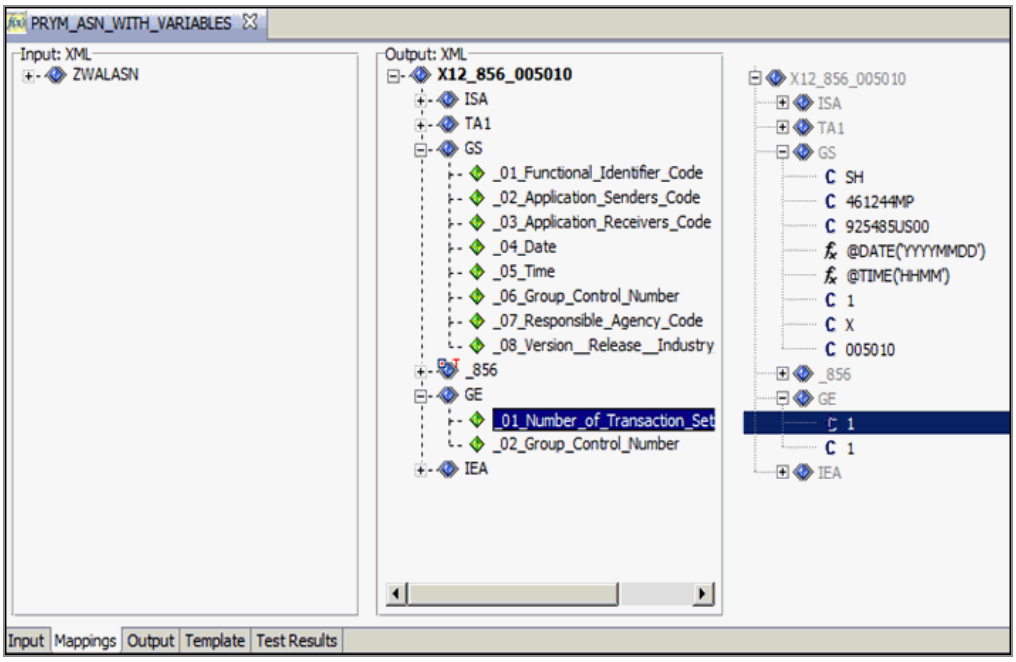
Map GS and GE (Group) Segments

Procedure

1. Enter the values for the GS and GE segments that are listed in the following table:

Segment	Value
GS01	SH
GS02	SENDERID
GS03	RECEIVERID
GS04	@DATE('YYYYMMDD')
GS05	@TIME('HHMM')
GS06	1
GS07	X
GS08	005010
GE01	1
GE02	1

Your screen should now resemble the following image:



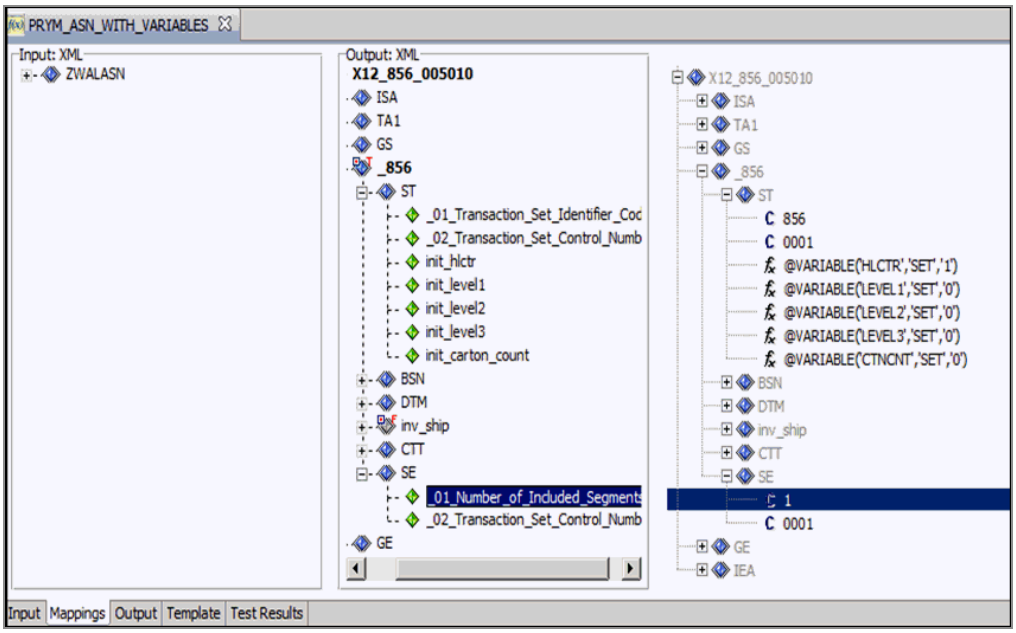
Map ST and SE (Document)

Procedure

1. To complete the envelopes, enter the values for the ST and SE segments that are listed in the following table:

Segment	Value
ST01	856
ST02	0001
SE01	1
SE02	0001

Your screen should now resemble the following image:



Note: You have specified default values for the SE01, GE01, and IEA01 segments. The preemitter transform contains functions that will correctly calculate these values. This example has only one document per envelope. As a result, the ST02/SE02 is 0001. If you have more than one document per envelope, you can use some of the techniques that are described in this tutorial to insert a sequential counter in these fields.

Initializing Constant Values

You can start to configure the variables that you are going to use to calculate the HL counts and carton count. A best practice is to initialize these variables at the start of the process.

Initialize Constant Values

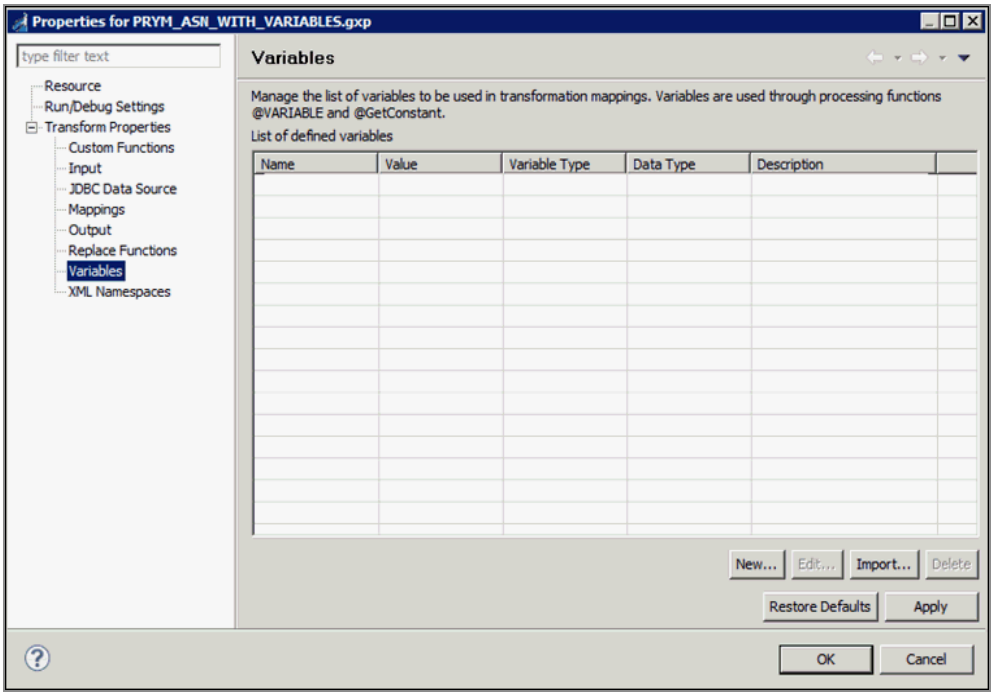
To initialize constant values:

Procedure

1. Right-click the Transform project PRYM_ASN_WITH_VARIABLES and select **Properties**

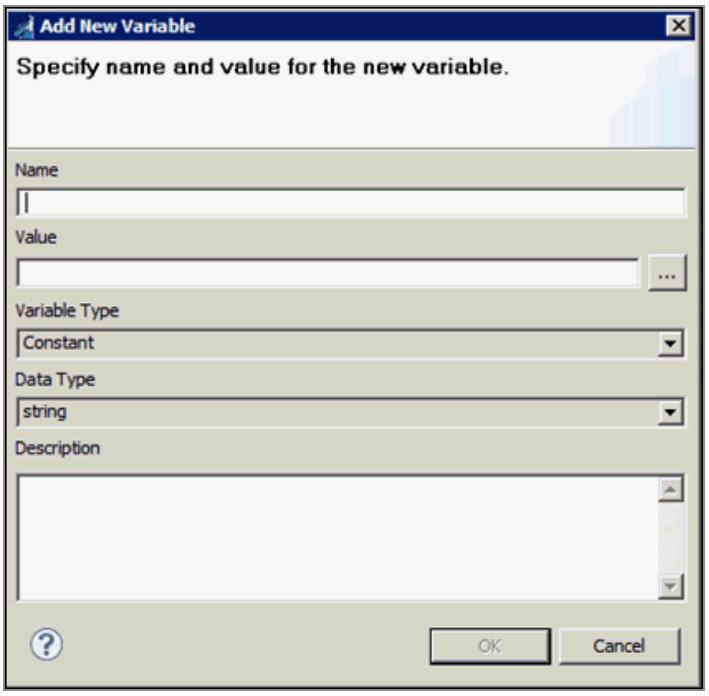
to open the Transform Project Properties dialog box.

2. Expand the **Transform Properties** tree and select **Variables** in the left pane, as shown in the following image.

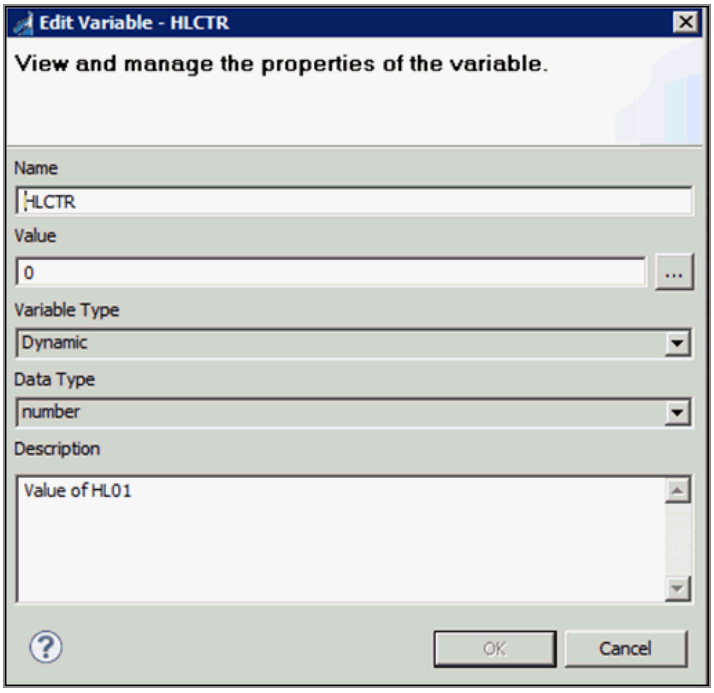


3. Click **New**.

   The Add New Variable dialog box opens.



4. Enter **HLCTR** for the name of the constant, **0** for the value, select **Dynamic** from the Variable Type drop-down list, and **number** from the Data Type drop-down list.

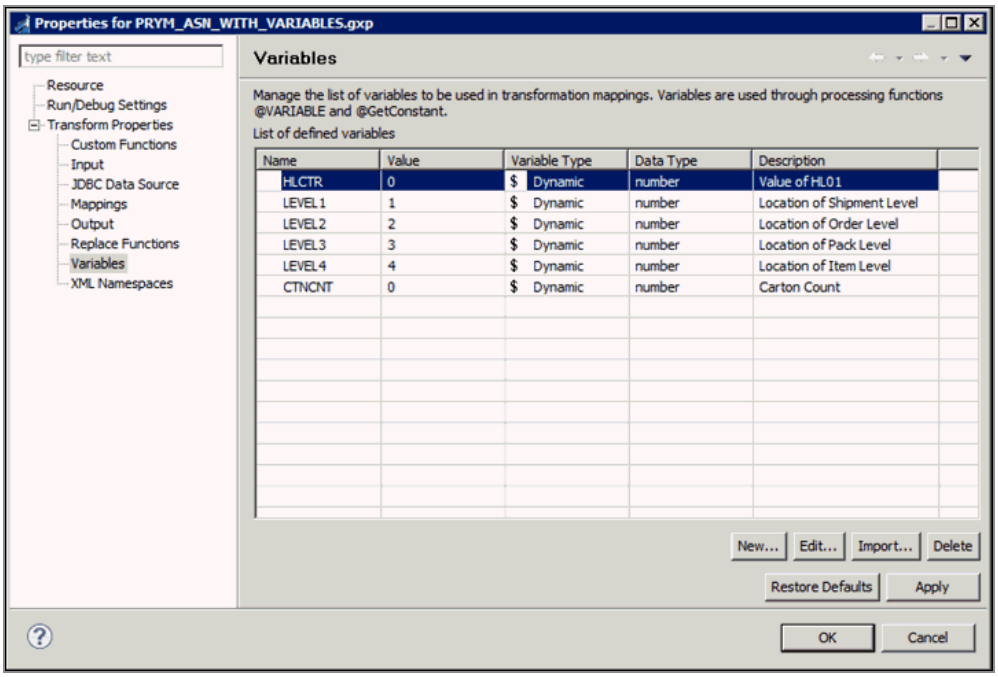


5. Click **OK** to save and continue.

6. Enter the remaining values, as specified in the following table:

Variable Name	Value
LEVEL1	1
LEVEL2	2
LEVEL3	3
LEVEL4	4
CTNCT	0

The Project Properties dialog box should now resemble the following image:



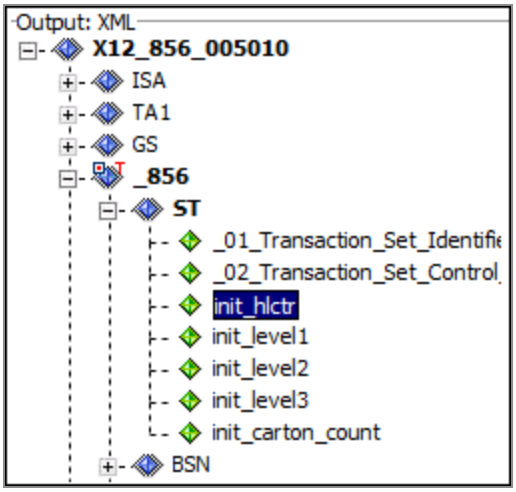
7. Click **OK**.

You also need to reset these variables in the Transform as it runs. The proper place to do this is the ST segment. At each new ST segment you can reinitialize the counters. To do this, you will add new element tags at the end of the ST segment.

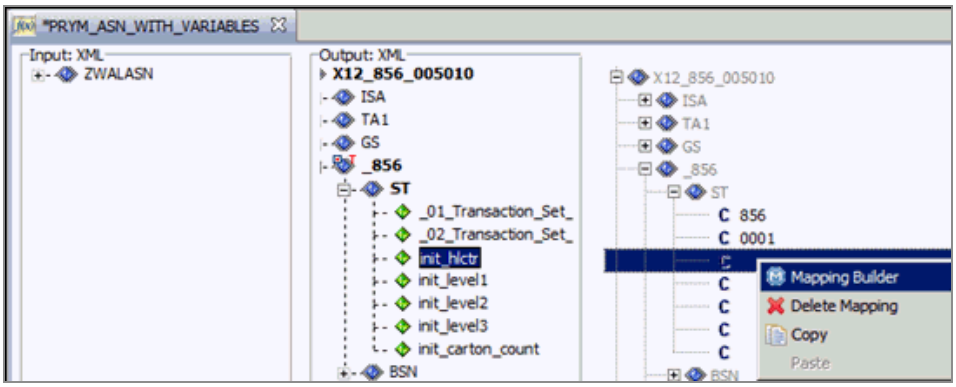
You do not have to hide these elements because there are no corresponding elements in the XML to X12 transform in the Ebix.

8. Right-click **ST**, select **Add**, and then click **Element** from the context menu.
9. Rename the new element to **init_hlctr**.
10. Add the following additional elements:
    - init_level1
    - init_level2
    - init_level3
    - init_carton_count

    Your screen should now resemble the following image:

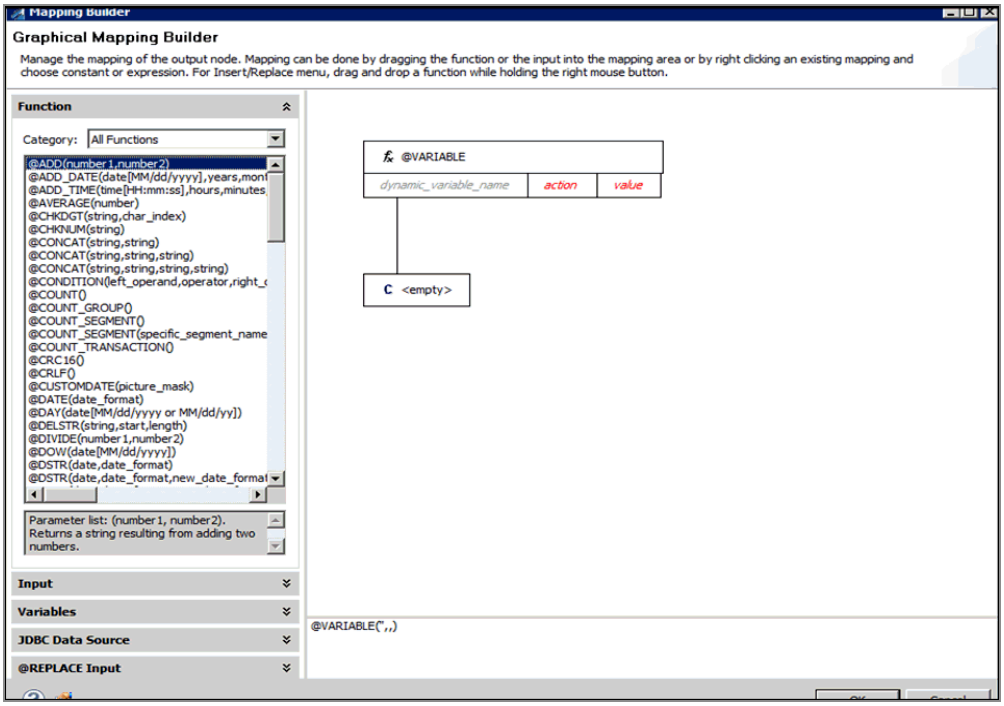


11. Right-click the Mapping Values pane for an element (for example, init_hlctr), and then select **Mapping Builder** from the context menu.



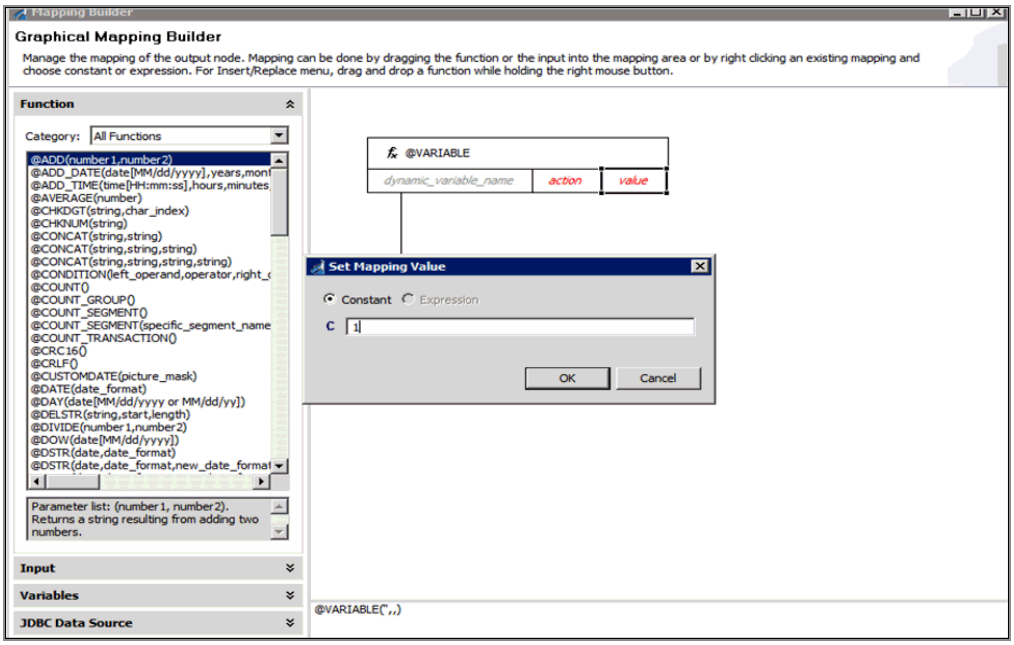
The Output Node Mapping Builder opens.

12. From the function list, select **@VARIABLE(dynamic_variable_name, action, value)** and drag it to the workspace area.



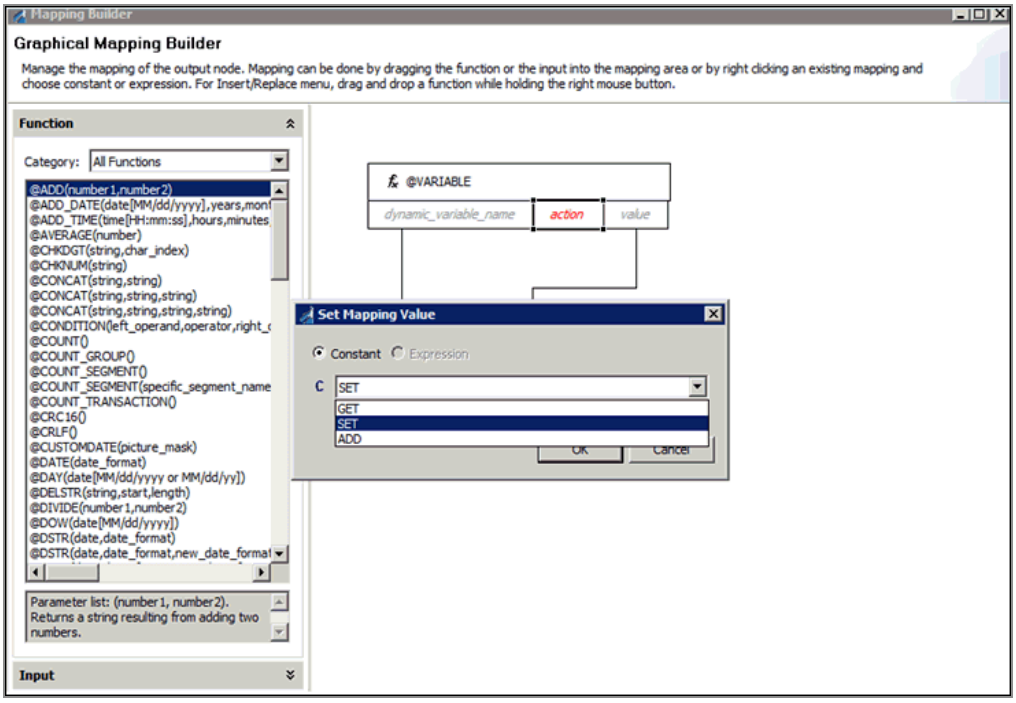
13. Double-click the **value** parameter of the function.

    The Set Mapping Value dialog box opens.



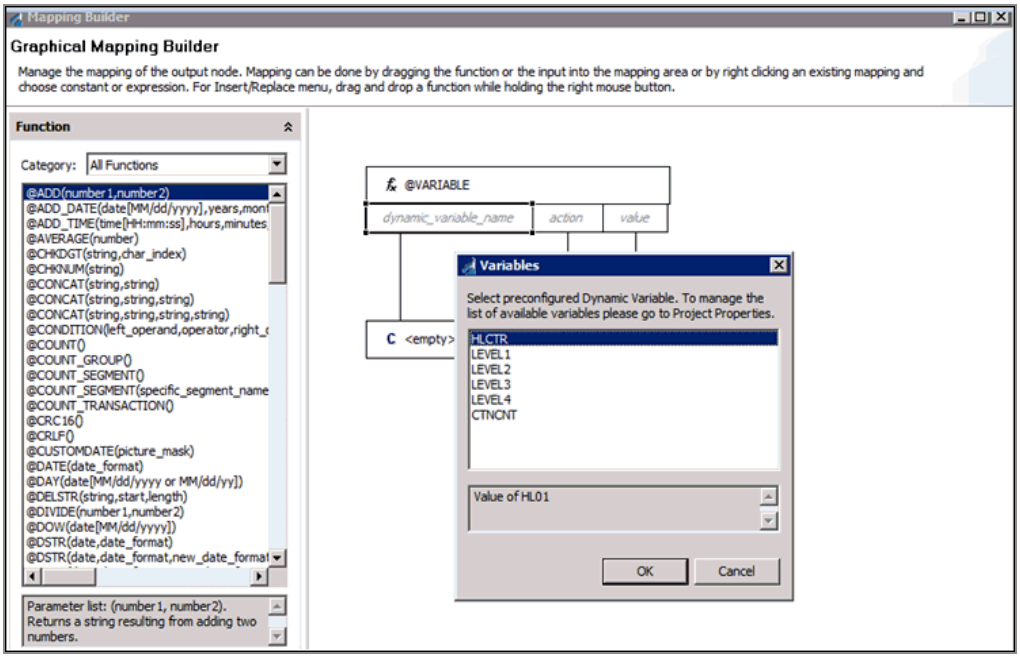
14. Type a constant value (for example, 1) and click **OK**.
15. Double-click the **action** parameter of the function.

    The Set Mapping Value dialog box opens.



16. Select **SET** from the drop-down list and click **OK**.
17. Double-click the **dynamic_variable_name** parameter of the function.

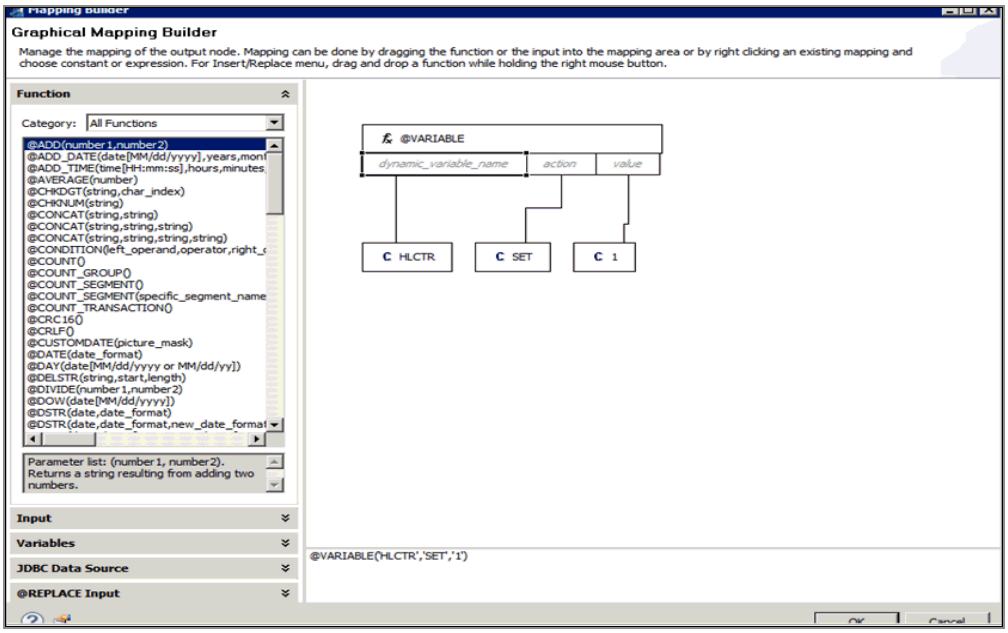
    The Variables dialog box opens.



    The Variables dialog box lists the variables that you configured earlier.

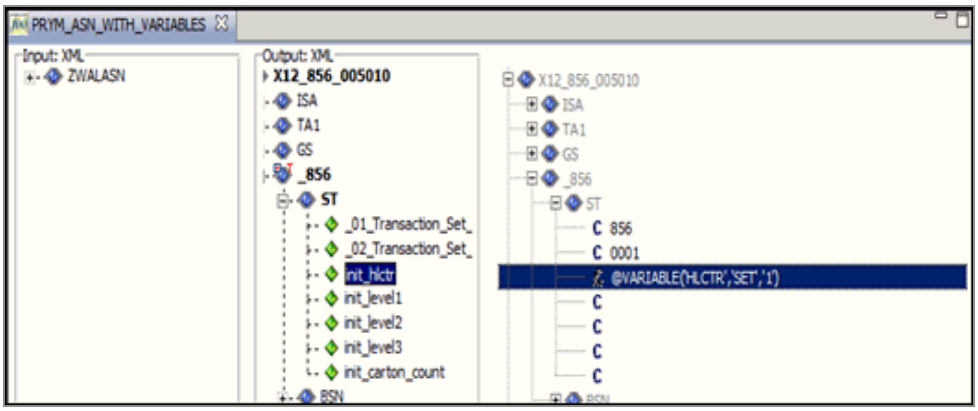
18. Select the **HLCTR** variable and click **OK**.

    The configured function for the *init_hlctr* element is shown in the following image.



19. Click **OK**.

    The new function is appended to the *init_hlctr* element in the Mapping Values pane, as shown in the following image.

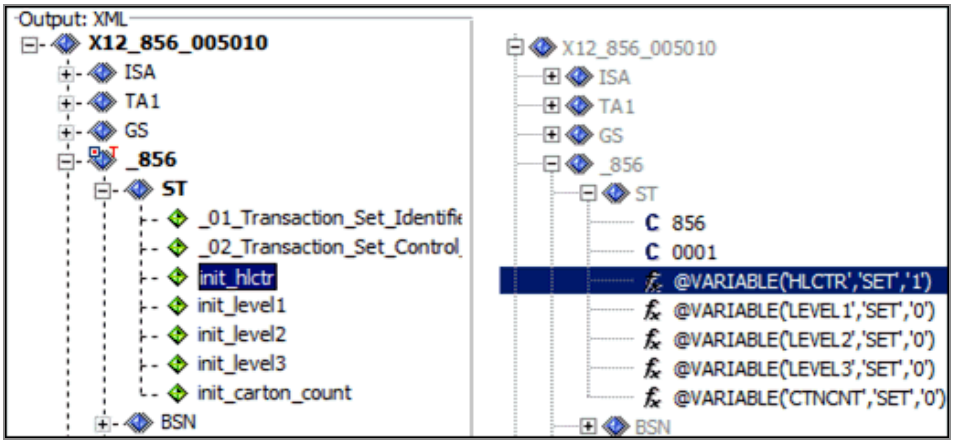


20. Repeat steps 11 through 19 for the remaining elements (init_level1, init_level2, init_level3, and init_carton_count). Use the values that are listed in the following table:

Element	Value
init_level1	@VARIABLE('LEVEL1','SET','0')

Element	Value
init_level2	@VARIABLE('LEVEL2','SET','0')
init_level3	@VARIABLE('LEVEL3','SET','0')
init_carton_count	@VARIABLE('CTNCNT','SET','0')

Your screen should now resemble the following image.

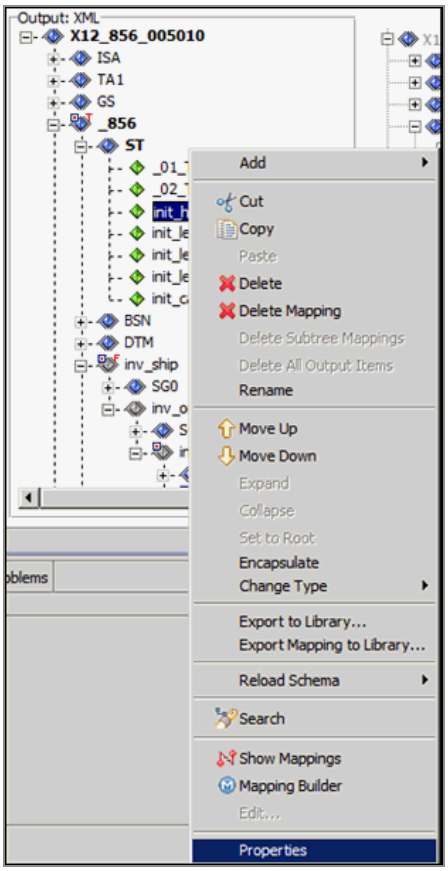


Hide Elements in a Transform

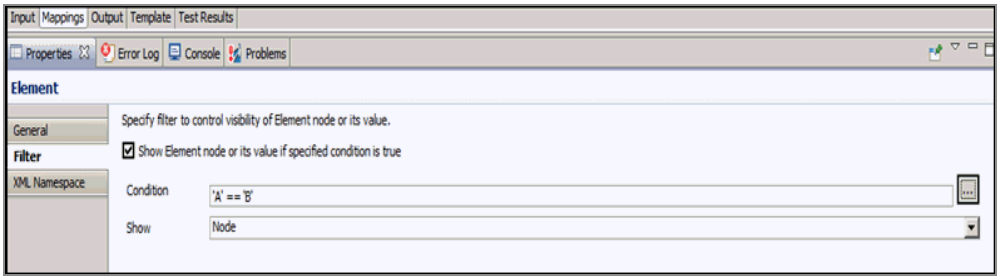
To maintain the integrity of the EDI structure, you must hide the elements that have been added. These elements are initial values, which if displayed in the XML, will cause the XML to EDI transform to fail. Perform the following steps to ensure that these elements do not appear in the transformed output.

Procedure

1. Right-click the **init_hlctr** tag in the Output pane and select **Properties** from the context menu.



The Output Node Properties - init_hlctr dialog box opens.



2. Click the **Filter** tab.
3. Select the **Show Element node or its value if specified condition is true** check box.
4. Select **Node** from the Show drop-down list.
5. Enter the following in the Condition field:

```
'A' == B'  (OR) '1' == '2'
```

Note: Since ‘A’ will never equal ‘B’ and the check box indicates if the condition is true to show the node, this condition ensures that the node will not be shown in the output XML.

6. Click **OK**.
7. Repeat steps 1 through 6 for the remaining *init_* elements that were added:
    - init_level1
    - init_level2
    - init_level3
    - init_carton_count

Mapping the Begin Ship Notice (BSN) Segments

From top to bottom, you will now map the output segments.

Map the Begin Ship Notice (BSN)

Procedure

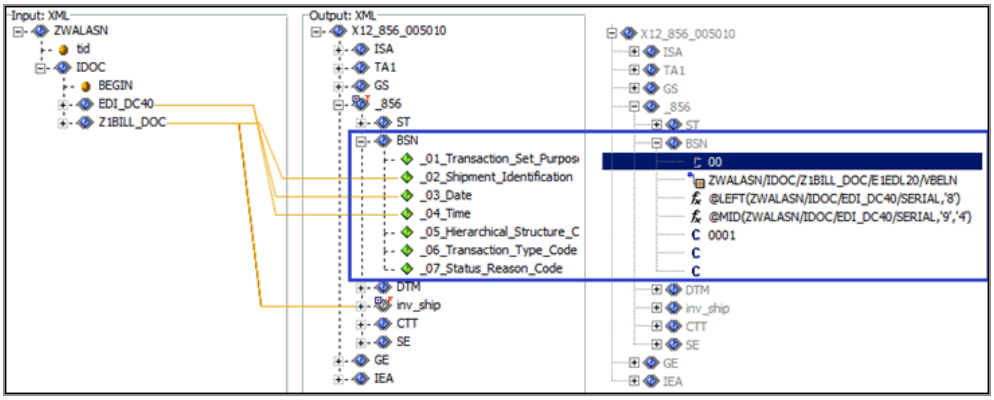
1. Configure the mapping values for the BSN elements as indicated by the following table:

Element	Value
BSN01	00
BSN02	ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/VBELN
BSN03	@LEFT(ZWALASN/IDOC/EDI_DC40/SERIAL,'8')

Element	Value
BSN04	@MID(ZWALASN/IDOC/EDI_DC40/SERIAL,'9','4')
BSN05	0001

The first eight characters of SERIAL are the ship date and the next four represent the ship time. You can use the LEFT and MID functions to correct the contents of these fields.

Your screen should now resemble the following image:



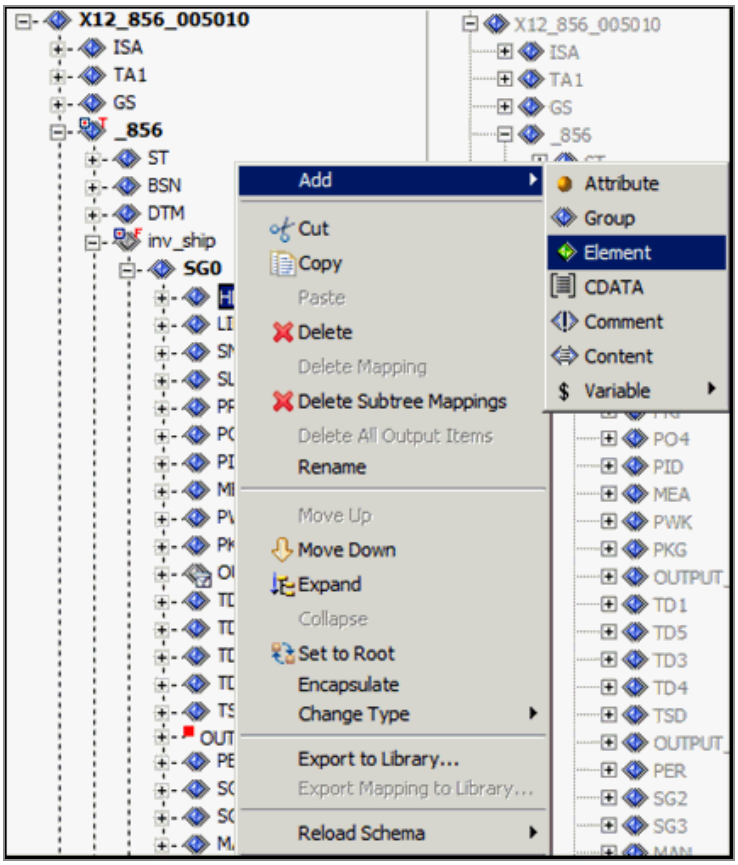
Configuring Shipment Level Segments

The following sections describe how to map the shipment level segments.

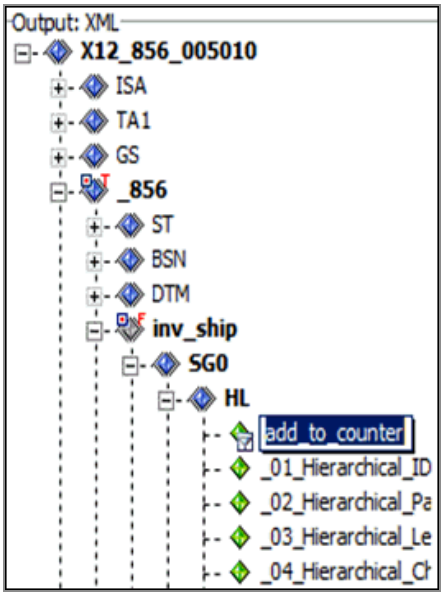
Configure the HL (Hierarchy) Elements

Procedure

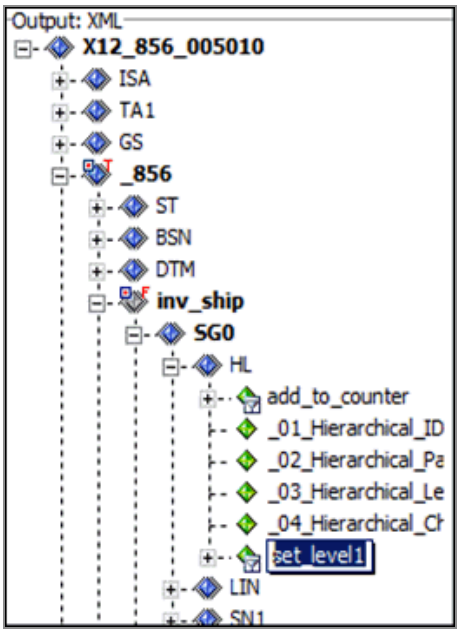
1. Under the inv_ship tag, expand **SG0**, right-click the **HL** tag, select **Add**, and then click **Element**.



2. Name the newly added element as **add_to_counter**.



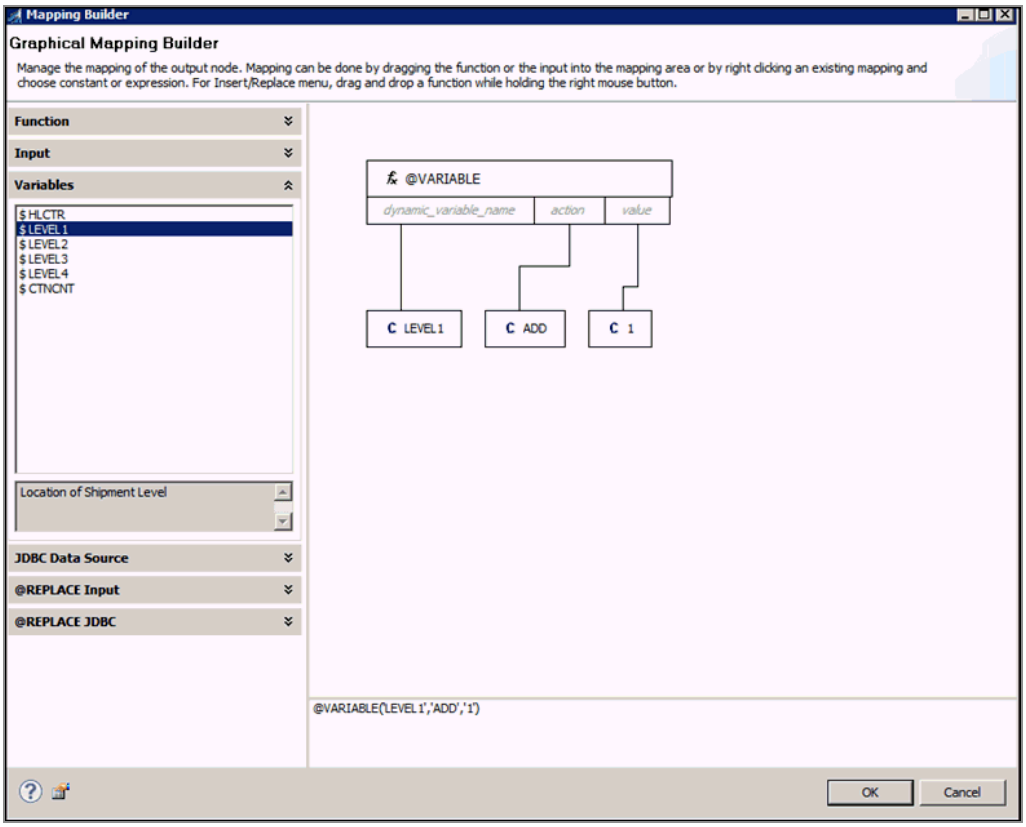
3. Repeat the same process to create the **set_level1** element.



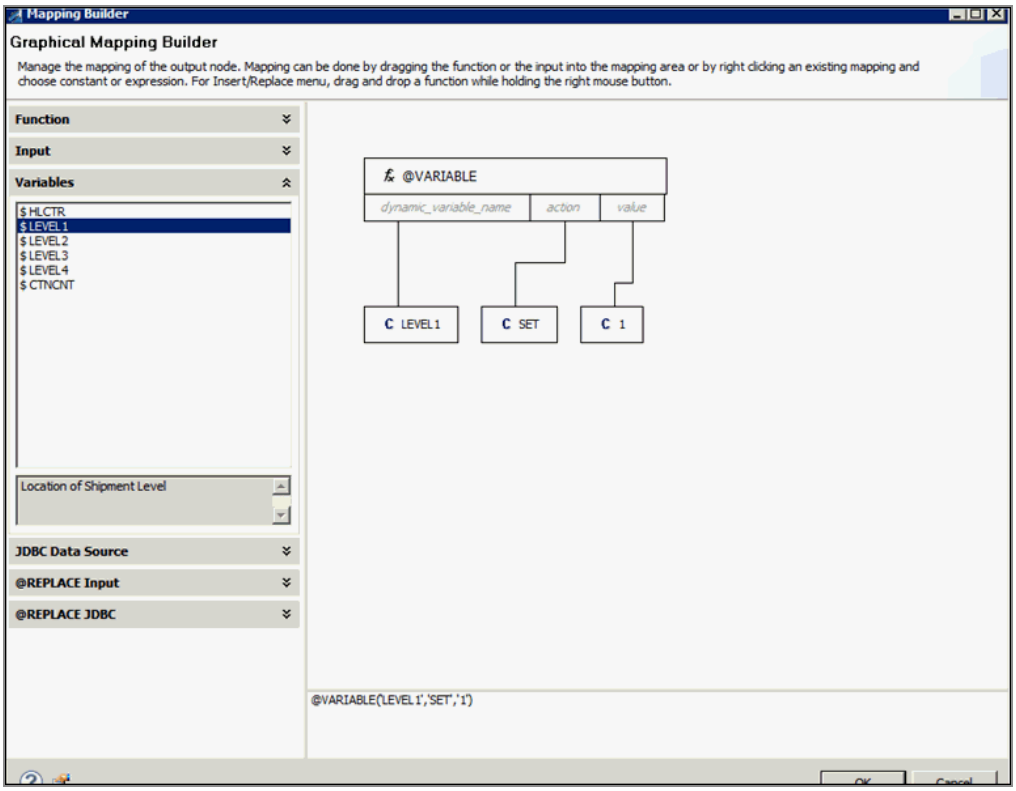
4. Configure the mapping values for the HL elements as indicated by the following table:

Element	Value
HL01	@VARIABLE('HLCTR','GET')
HL03	S
add_to_counter	@VARIABLE('LEVEL1','ADD','1')
set_level1	@VARIABLE('LEVEL1','SET','1')

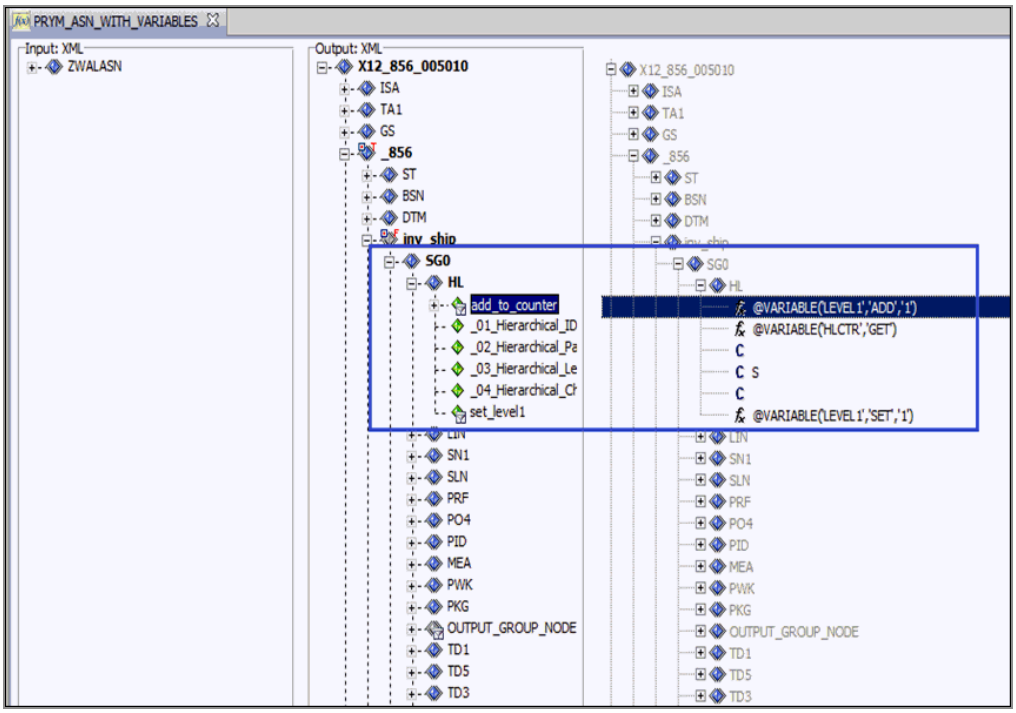
5. Map add_to_counter to the $LEVEL1 variable and the following:
   - action: ADD
   - value: 1



6. Map sel_level1 to the $LEVEL1 variable and the following:
    - action: SET
    - value: 1



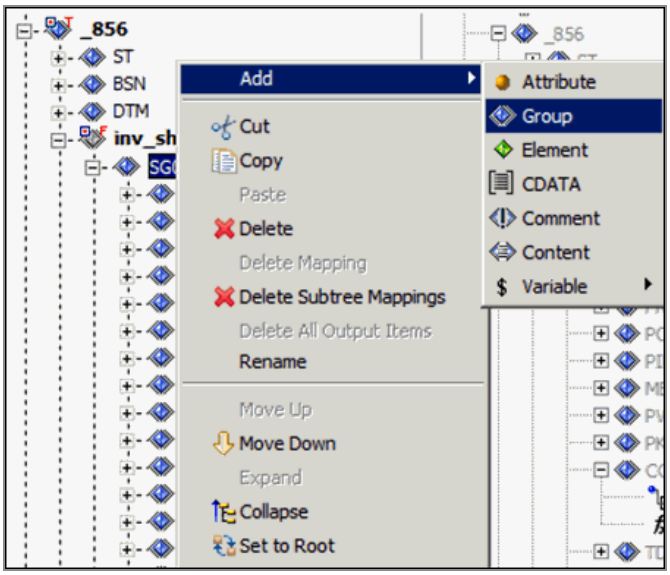
Your screen should now resemble the following image:



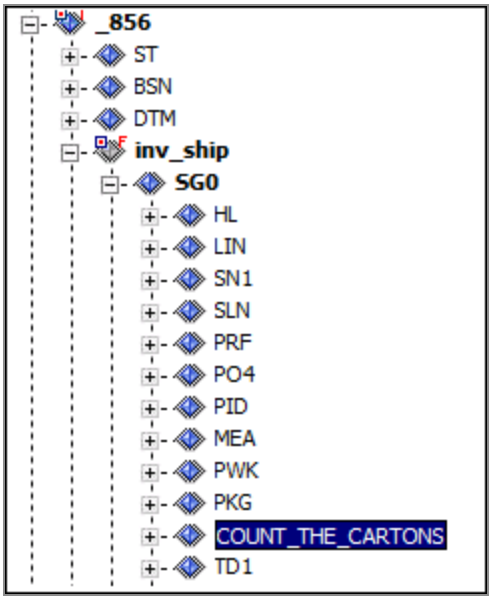
Configure the Carton Count

Procedure

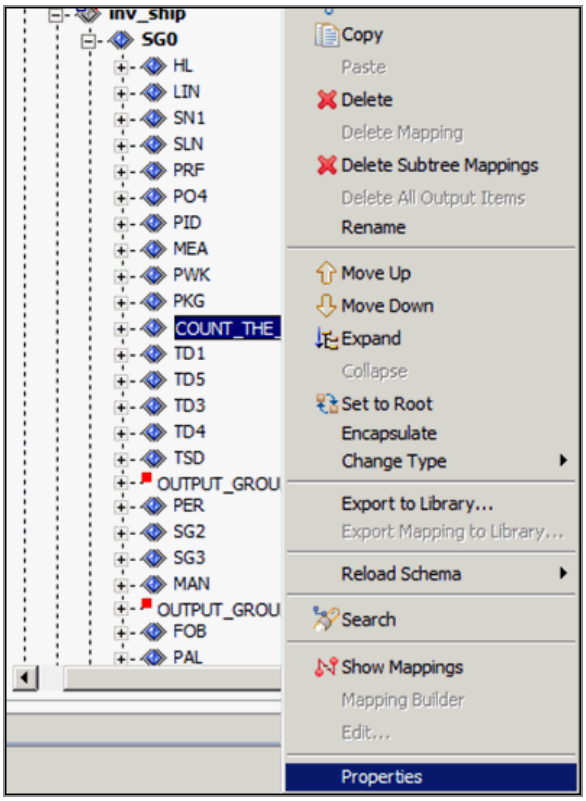
1. Under inv_ship, right-click the **SGO** tag, select **Add**, and then select **Group**.



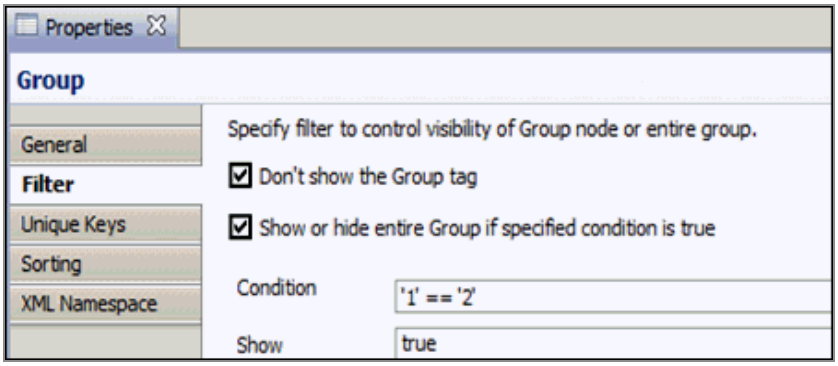
2. Double-click the default name of the added group (OUTPUT_GROUP_NODE) and change it to **COUNT_THE_CARTONS**.
3. Use the position icons (**Move Up** and **Move Down**) on the toolbar (or right-click and select from the control menu) to position the new *COUNT_THE_CARTONS* group below the PKG group.



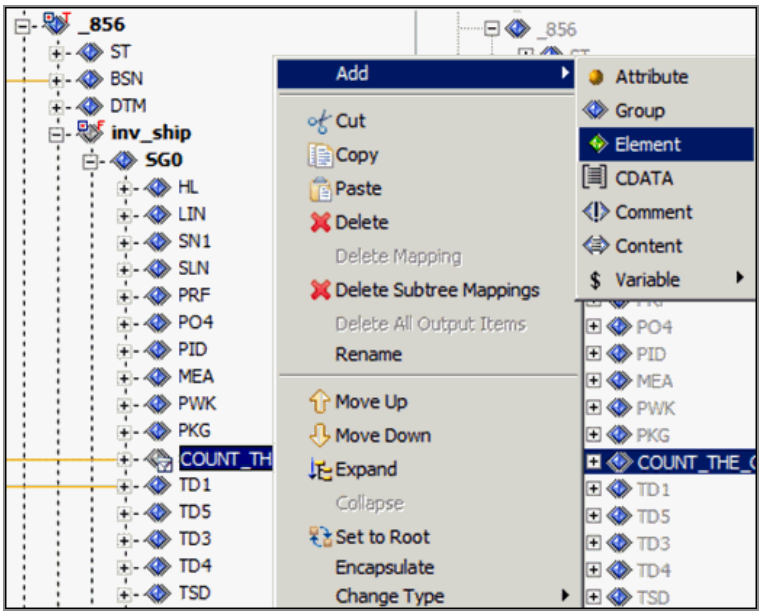
4. Right-click the **COUNT_THE_CARTONS** group in the Output pane and select **Properties** from the context menu.



The Output Node Properties - COUNT_THE_CARTONS dialog box opens.

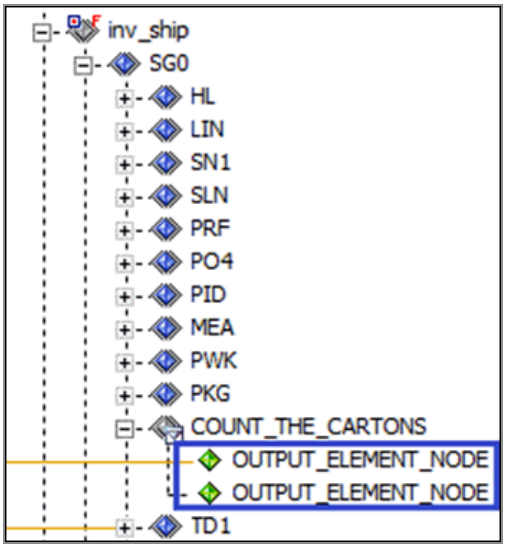


5. Click the **Filter** tab.
6. Select both the **Don't show the Group node** and **Show or hide entire Group if specified condition is true** options, set the Condition field to **‘1’==’2’**, and select the show value to **true**, and then click **OK**.
7. Right-click the **COUNT_THE_CARTONS** tag, select **Add**, and click **Element**.

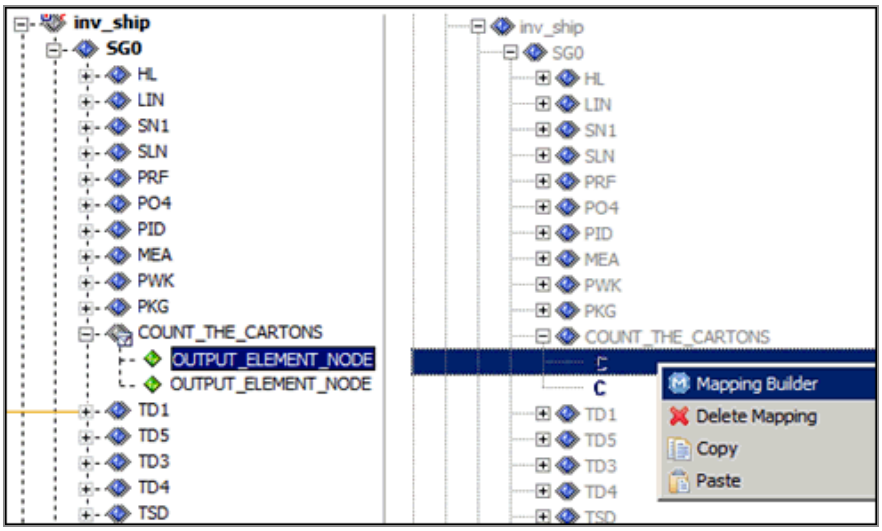


8. Repeat step 7 again to create another new element, and then rename both elements as **OUTPUT_ELEMEMT_NODE**.

   Your screen should now resemble the following image.

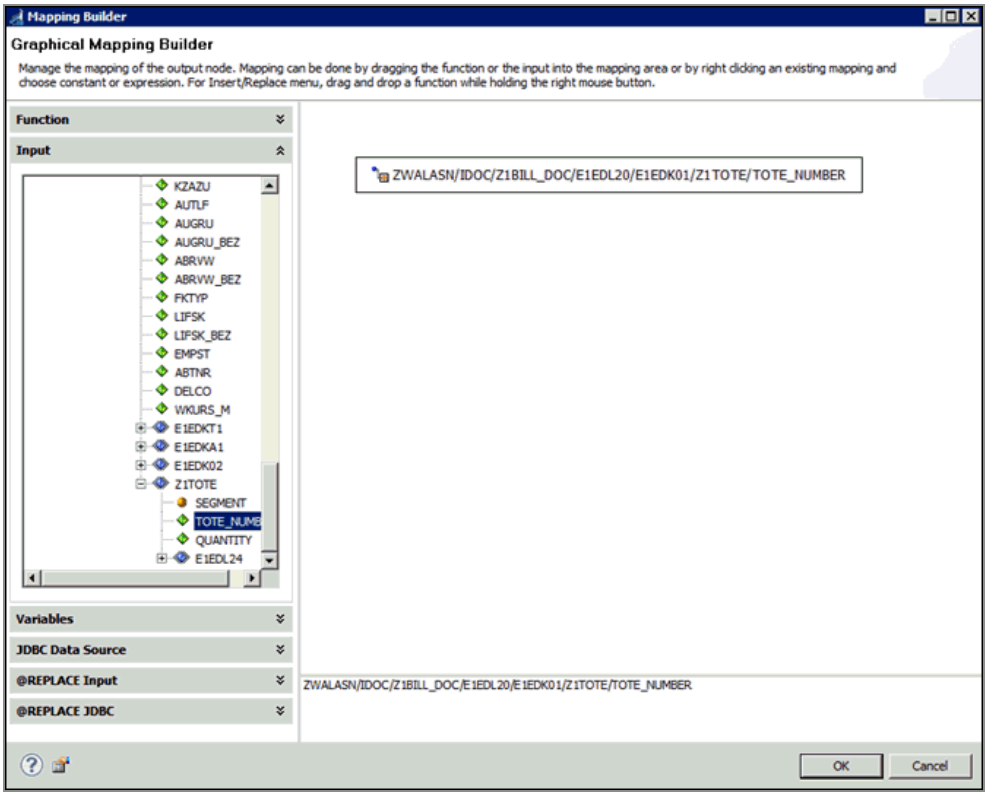


9. Right-click the first **OUTPUT_ELEMENT_NODE** mapping value, and select **Mapping Builder** from the context menu.

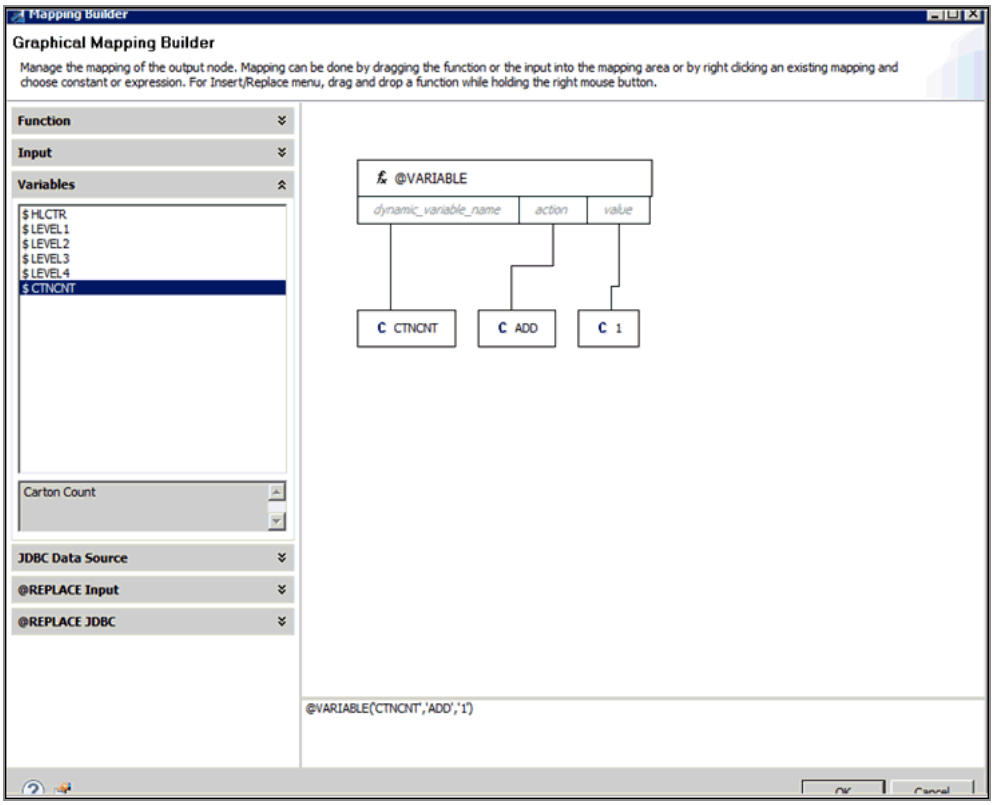


10. Map the following field to the first **OUTPUT_ELEMENT_NODE** element from the **COUNT_THE_CARTONS** group tag:

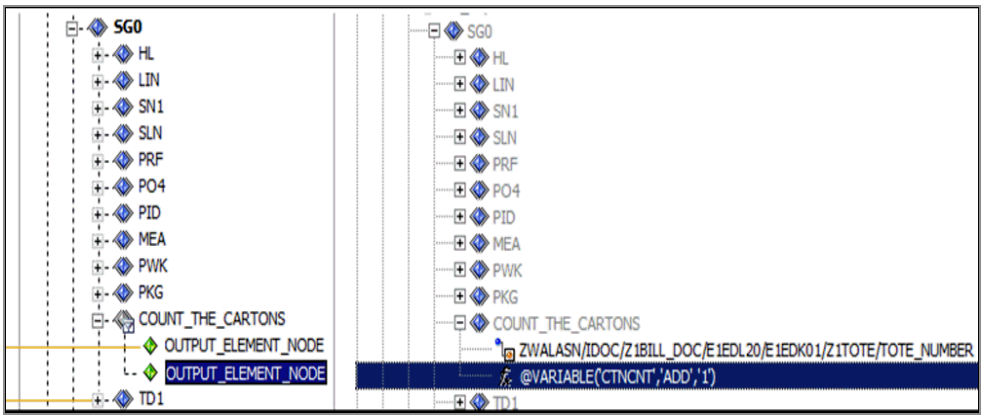
```
ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/Z1TOTE/TOTE_NUMBER
```



11. Click **OK**.
12. Map the **$CTNCNT** variable to the second **OUTPUT_ELEMENT_NODE** under **COUNT_THE_CARTONS**, as well as the following variables:
    - action: ADD
    - value: 1



Your screen should now resemble the following image.



Configure the TD1 (Total Details) Elements

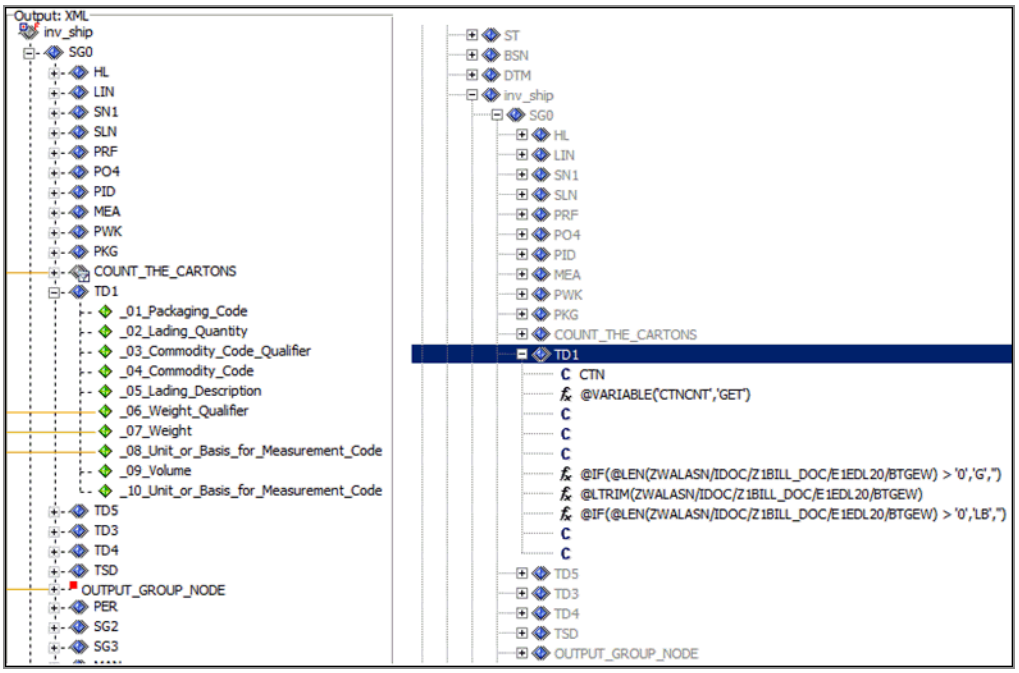
Procedure

1. Configure the mapping values for the TD1 elements as indicated by the following table:

Element	Value
TD101	CTN
TD102	@VARIABLE('CTNCNT','GET')
TD106	@IF(@LEN(ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/BTGEW) > '0','G','')
TD107	@LTRIM(ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/BTGEW)
TD108	@IF(@LEN(ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/BTGEW) > '0','LB','')

This is where the carton count gets inserted into the output document. As a best practice, only insert qualifiers if the data is present.

Your screen should now resemble the following image:



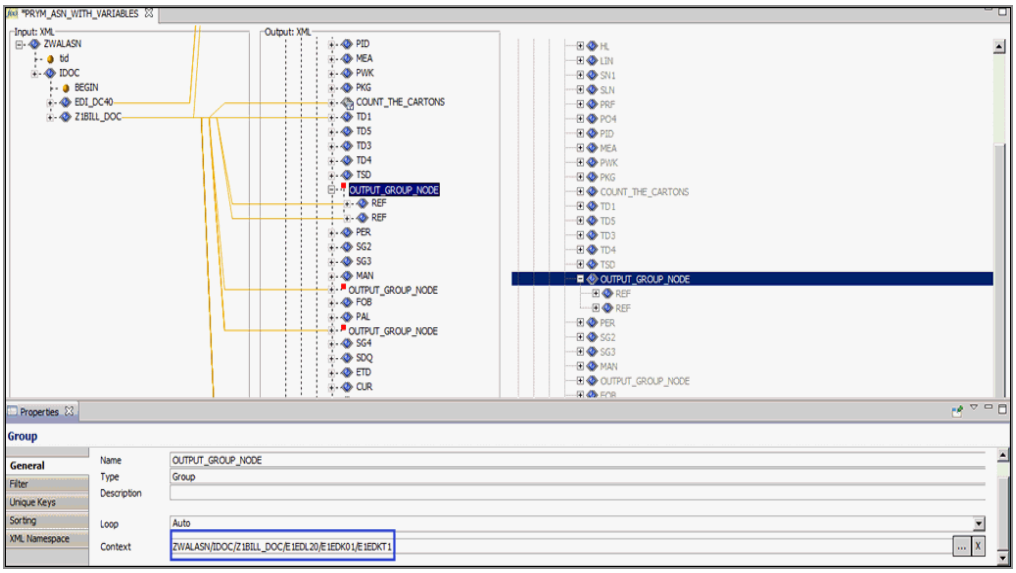
Configure the REF BM/CN (Bill of Lading Number) Groups

You need to output two REF groups from the same source element.

Procedure

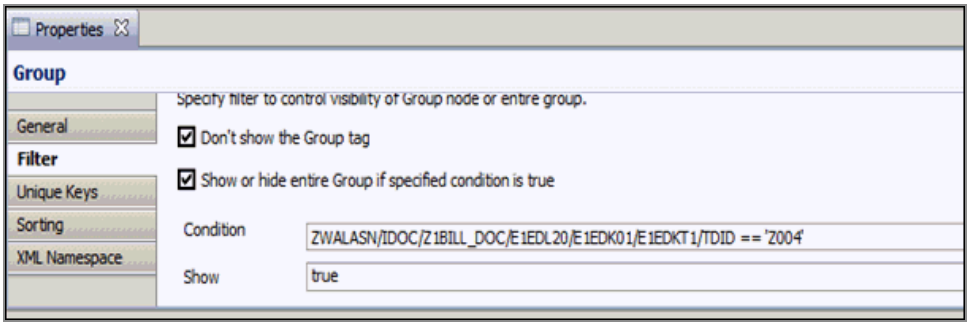
1. Add a new invisible parent group node (OUTPUT_GROUP_NODE) to contain the REF group.
2. Copy the REF group so there are two instances of this group.
3. Right-click and open the properties of OUTPUT_GROUP_NODE and set the context of the invisible parent group node to the following:

```
ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/E1EDKT1
```



4. Click the **Filter** tab.
5. Enter the following in the Condition field:

```
ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/E1EDKT1/TDID == 'Z004'
```

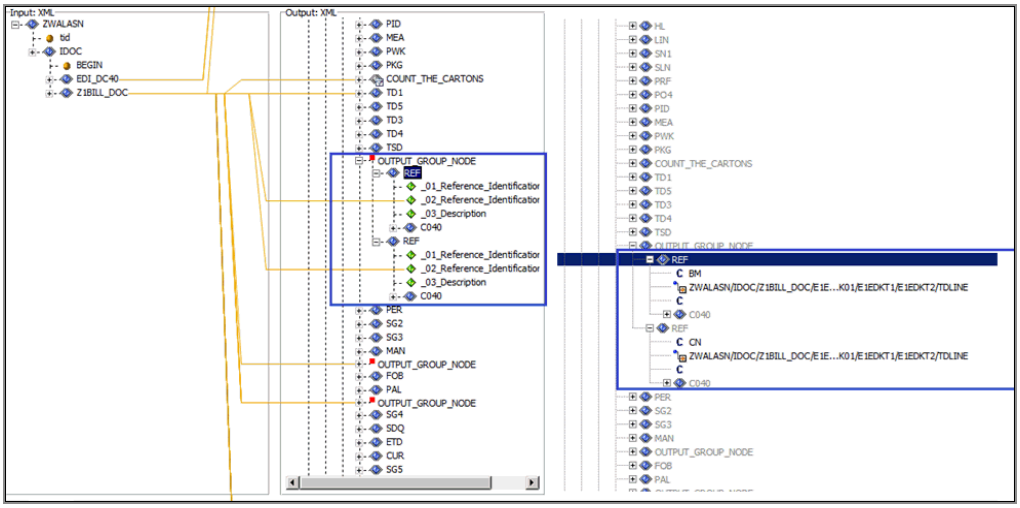


6. Click **OK**.
7. Configure the mapping values for the REF groups as indicated by the following table:

Group	Value
REF02	ZWALASN/IDOC/Z1BILL_ DOC/E1EDL20/E1EDK01/E1EDKT1/E1EDKT2/TDLINE

Group	Value
REF02	ZWALASN/IDOC/Z1BILL_ DOC/E1EDL20/E1EDK01/E1EDKT1/E1EDKT2/TDLINE

Your screen should now resemble the following image:

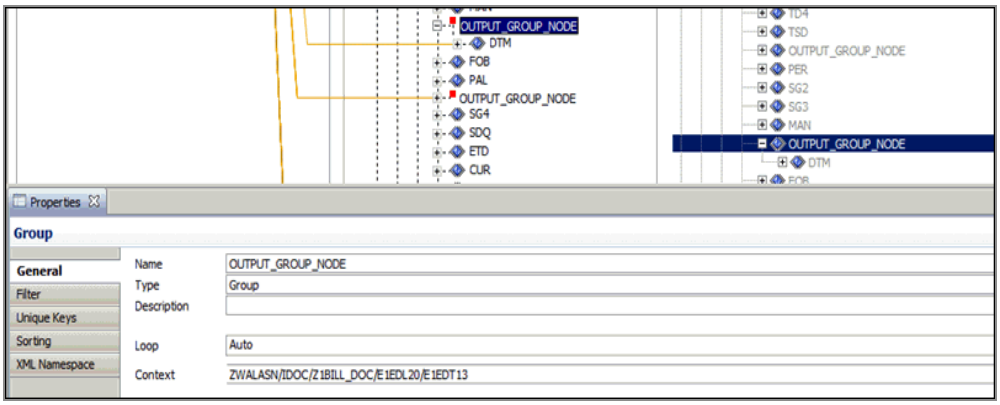


Configure the DTM 011 (Shipment Date) Elements

Procedure

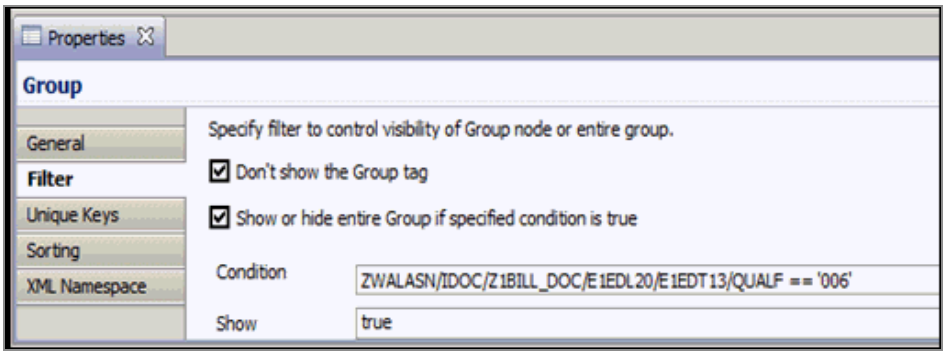
1. Add a new group element (OUTPUT_GROUP_NODE) and align it below the DTM element in the HL loop.
2. Drag and drop the DTM element into this new group.
3. Right-click and open the OUTPUT_GROUP_NODE properties and then set the context to the following:

```
ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDT13
```



4. Click the **Filter** tab.
5. Enter the following in the Condition field:

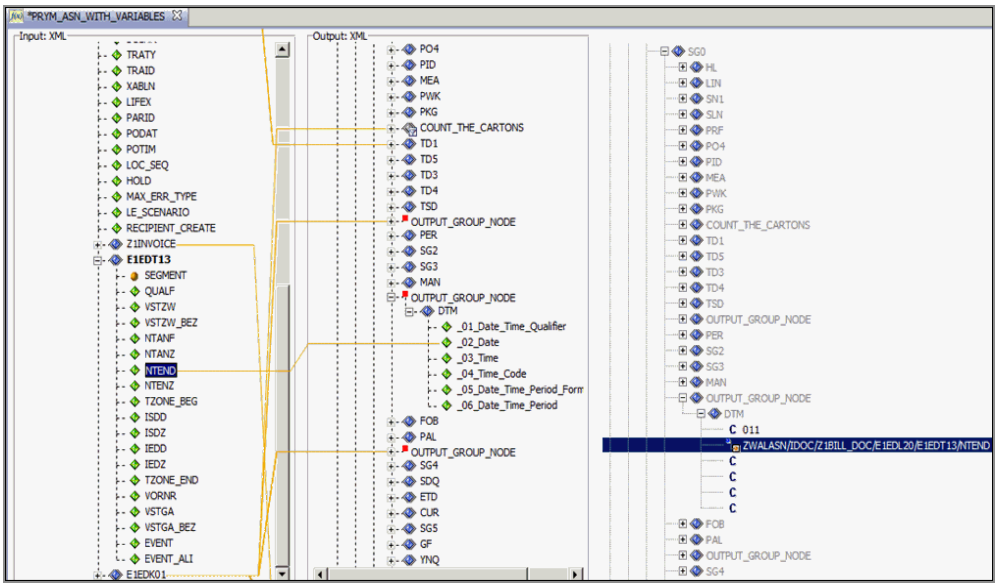
```
ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDT13/QUALF == '006'
```



6. Select **true** from the Show drop-down list.
7. Click **OK**.
8. Configure the mapping values for the DTM elements as indicated by the following table:

Element	Value
DTM01	011
DTM02	ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDT13/NTEND

Your screen should now resemble the following image:



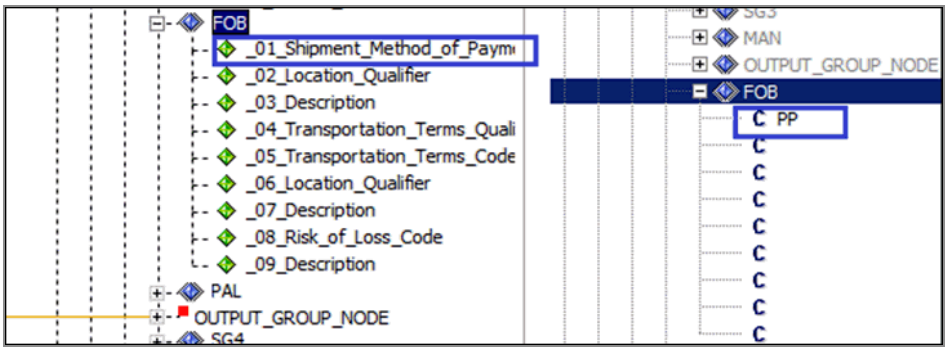
Configure the FOB Element

Procedure

1. Configure the mapping value for the FOB element as indicated by the following table:

Element	Value
FOB01	PP

Your screen should now resemble the following image:

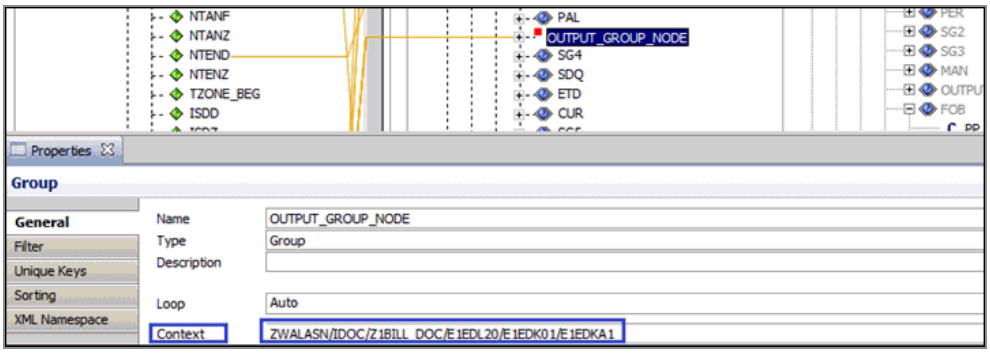


Configure the N1 ST (Ship-To) Segment

Procedure

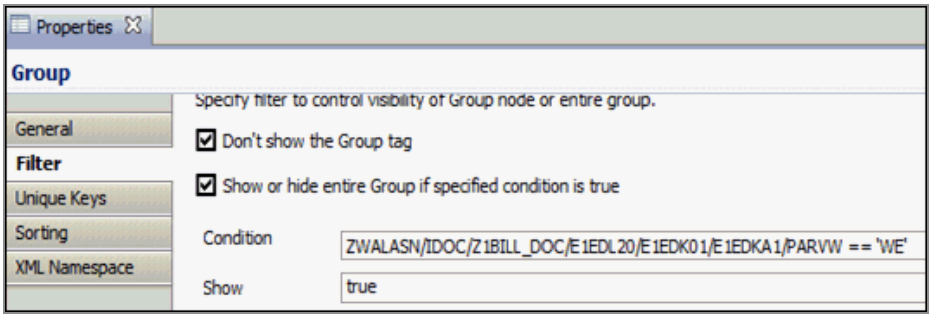
1. Add a new hidden group tag.
2. Move the new tag up so it is above the SG4 group.
3. Copy the SG4 group and paste the copy under the new group.
4. Right-click the newly created **OUTPUT_GROUP_NODE** and select **Properties** from the context menu, then set the context field to the following:

```
ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/E1EDKA1
```

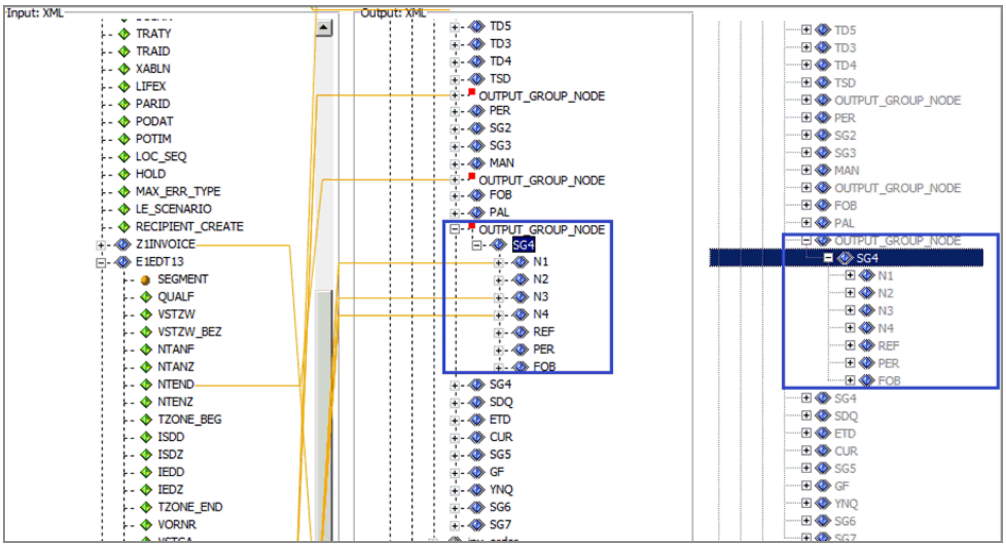


5. Click the **Filter** tab.
6. Enter the following in the Condition field and then select **true** in the Show field.

```
ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/E1EDKA1/PARVW == 'WE'
```



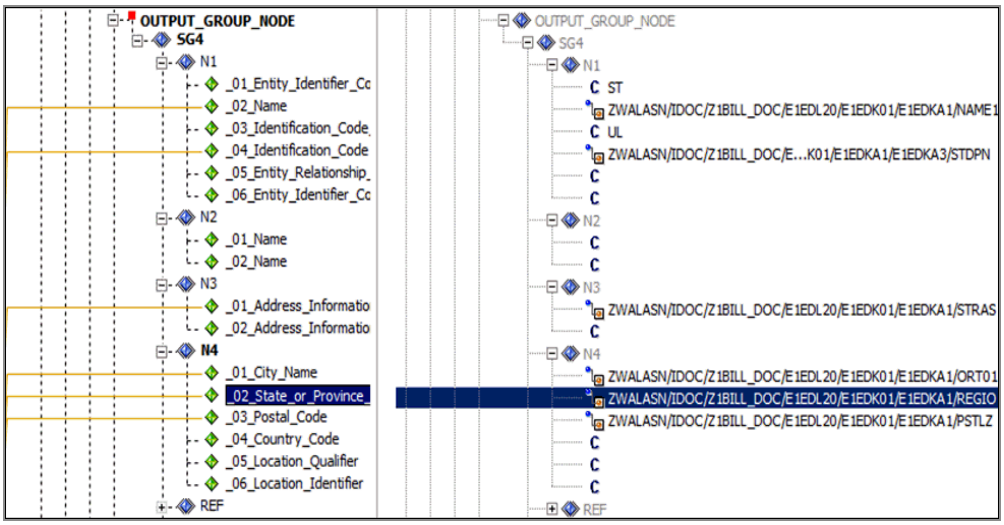
Your screen should now resemble the following image:



7. Configure the mapping values for the N1 ST elements as indicated by the following table:

Element	Value
N101	ST
N102	ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/E1EDKA1/NAME1
N103	UL
N104	ZWALASN/IDOC/Z1BILL_ DOC/E1EDL20/E1EDK01/E1EDKA1/E1EDKA3/STDPN
N301	ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/E1EDKA1/STRAS
N401	ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/E1EDKA1/ORT01
N402	ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/E1EDKA1/REGIO
N403	ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/E1EDKA1/PSTLZ

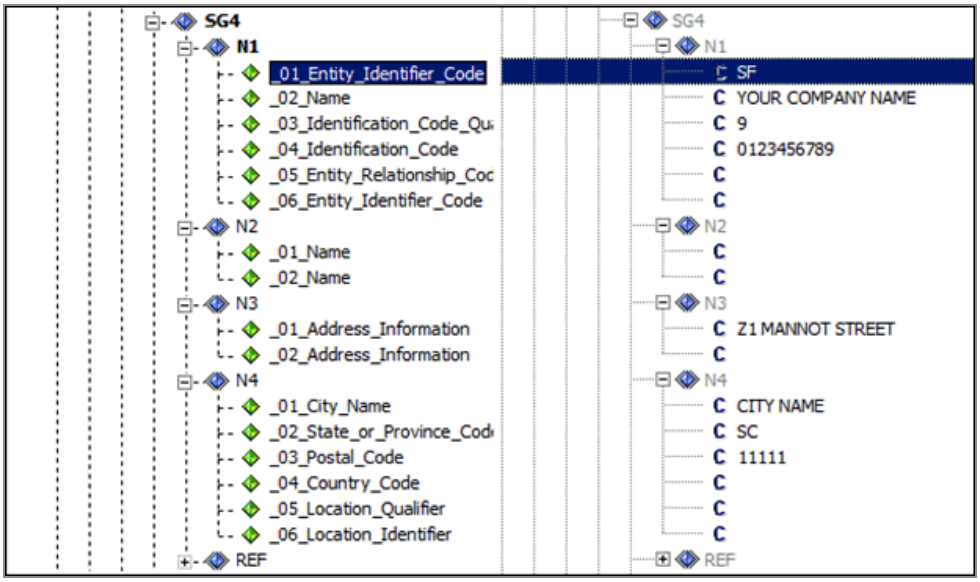
Your screen should now resemble the following image:



Configure the N1 SF (Ship From) Element

Procedure

1. The N1 SF element is hard-coded, as shown in the following image:



This information was extracted from the IDoc, but can be accepted from other sources as well.

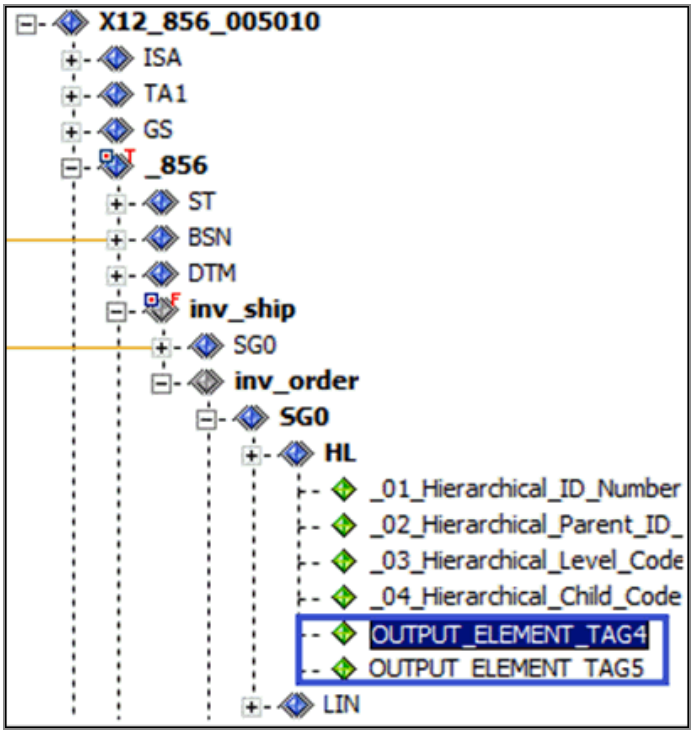
Configuring Order Level Segments

The following sections describe how to map the order level segments.

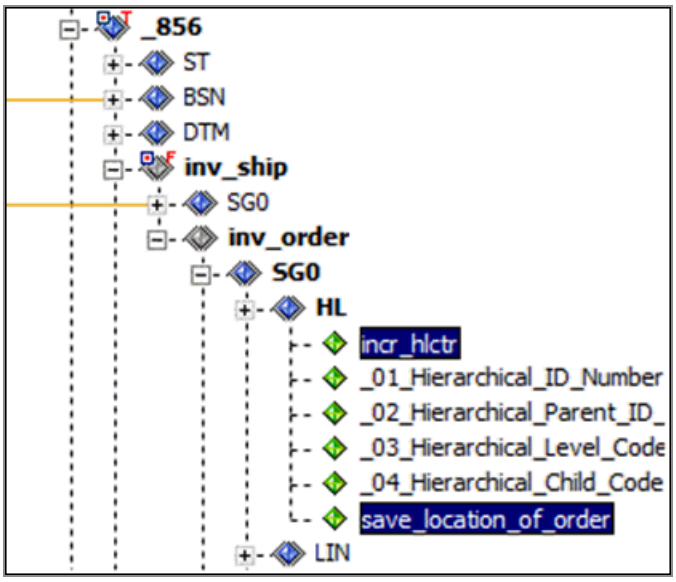
Configure the HL (Hierarchy) Elements

Procedure

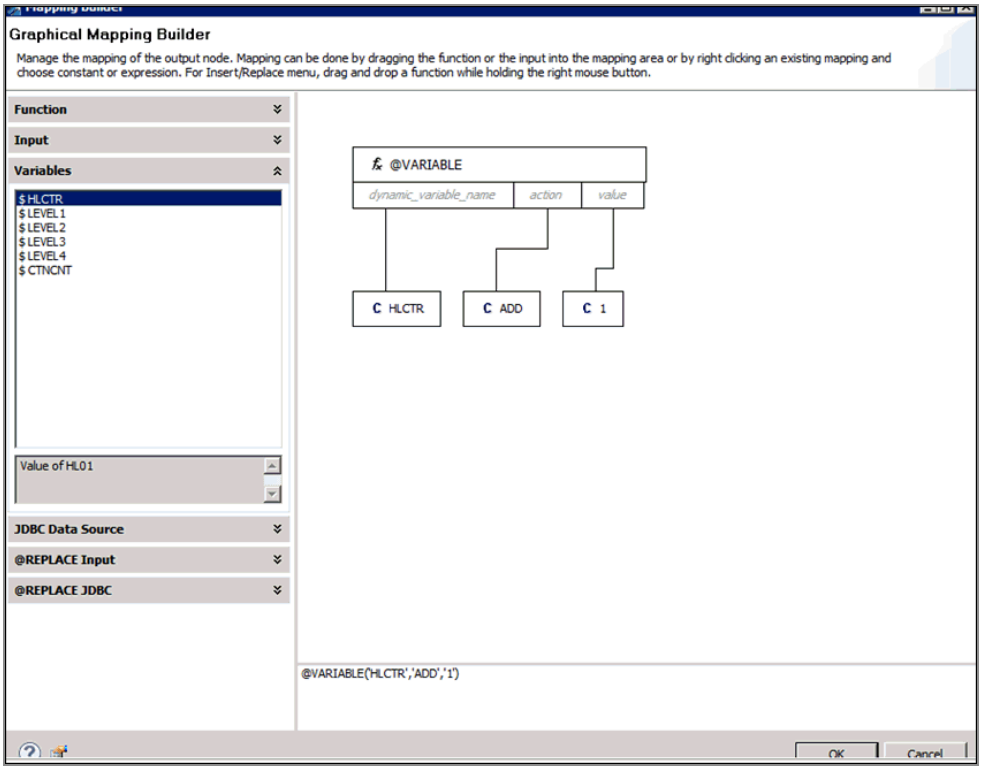
1. Under the **SG0** tag, which is located under *inv_order*, right-click the **HL** tag, select **Add**, and then click **Element**.



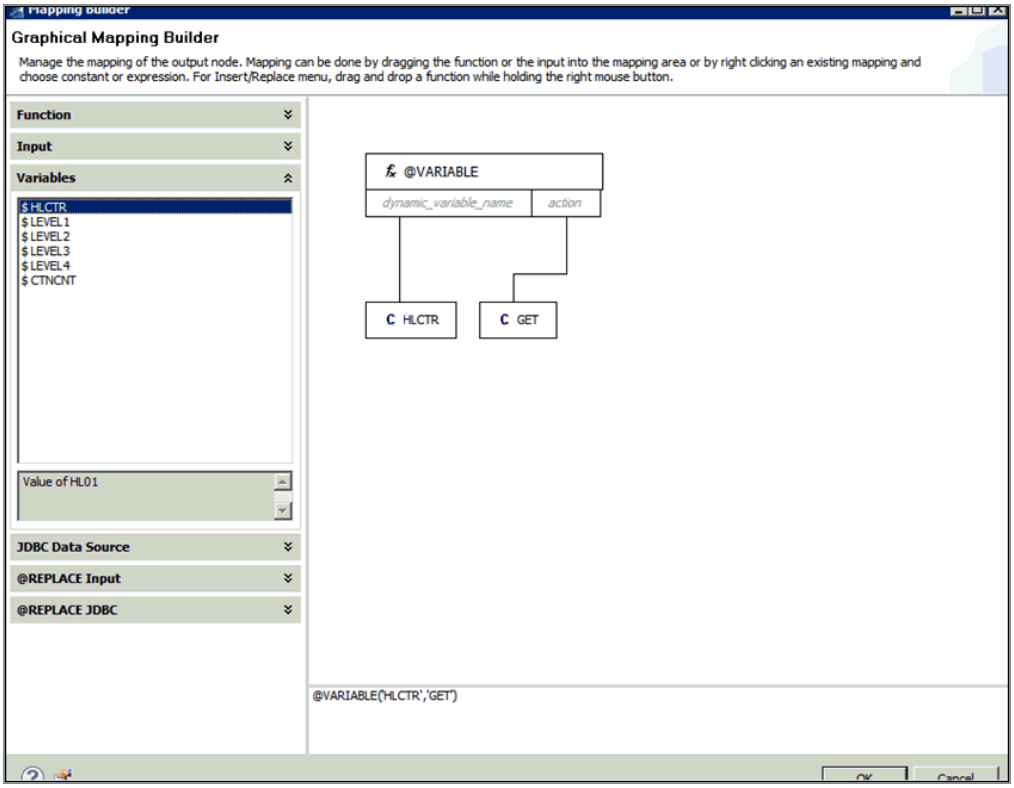
2. Repeat step 1 to create another element.
3. Use the position icons (Move Up and Move Down) on the toolbar (or right-click and select from the control menu) to position one of the newly created elements at the top of all HL elements, and the other at the bottom of all HL elements.
4. Rename the top HL element as **incr_hlctr** and the other element at the bottom, **save_location_of_order**, as shown in the following image.



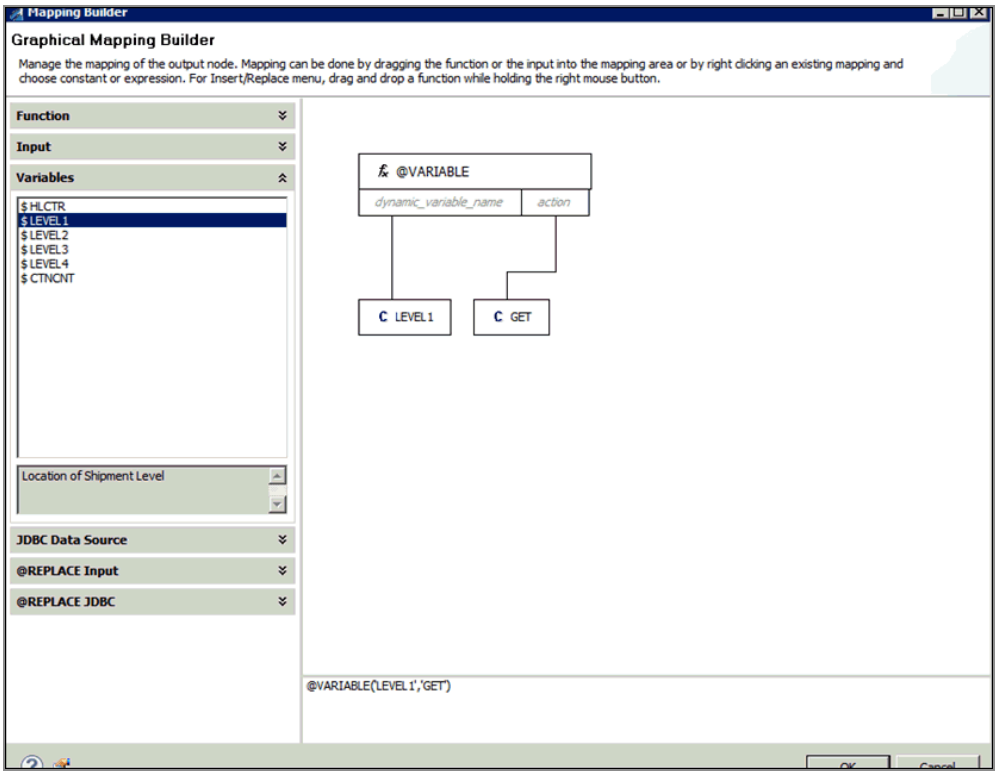
5. Right-click the **incr_hlctr** element, select **Properties** in the context menu, click the ellipses (...) button under the General tab, and then map the HLCTR variable.
6. Select **ADD** in the action, then set the constant value to **1**.



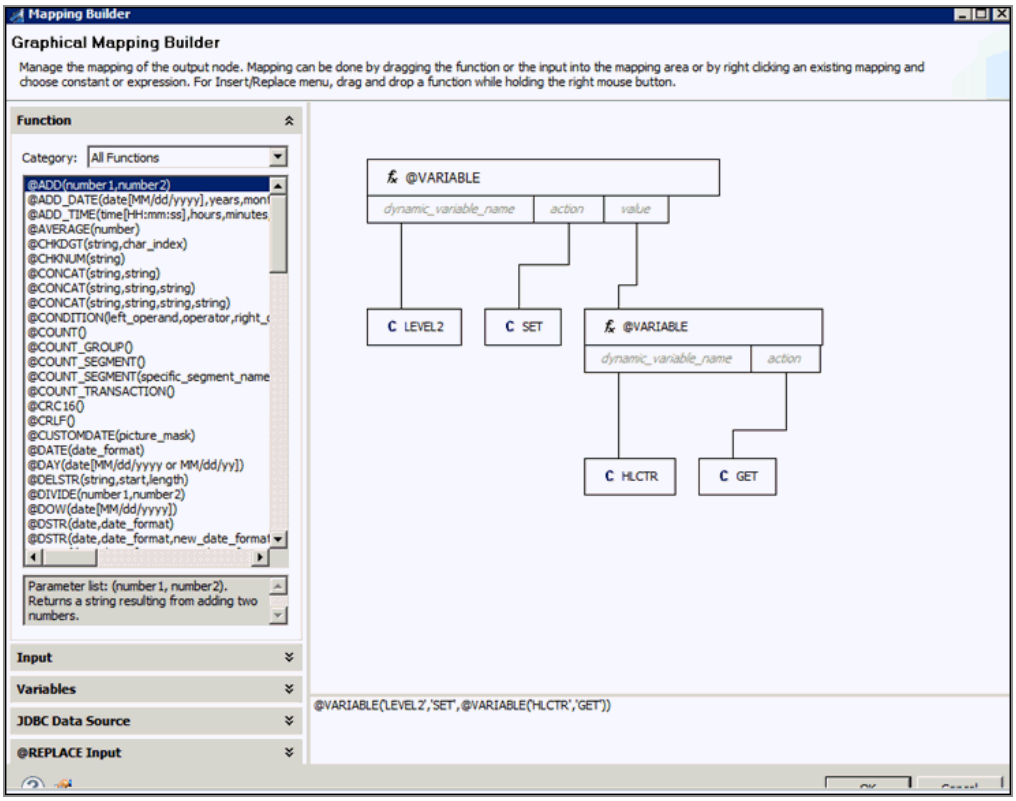
7. Map the HLCTR variable to the HL01 element, located in HL, under SG0 after expanding inv_order, and select GET as the action.



8. Map the **LEVEL1** variable to the **HL02** element and set the action to **GET**.



9. Expand **inv_order**, **SG0**, and then **HL**, and map the constant **O** to **HL03**.
10. Map the **LEVEL2** variable to the **save_location_of_order** element and set the GET current value to **HLCTR** (@VARIABLE(HLCTR,GET)).

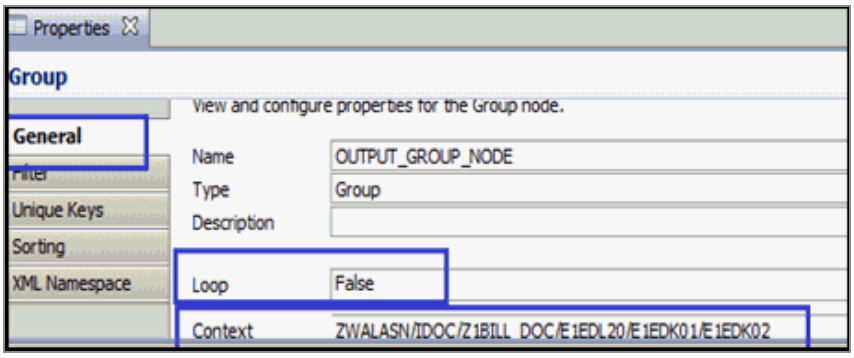


Configure the PRF (PO Number) Elements

Procedure

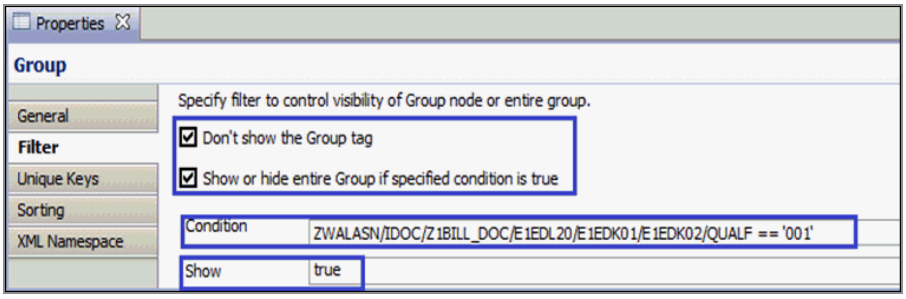
1. Add a hidden group node (OUTPUT_GROUP_NODE) above the PRF element and move the PRF element into the new group.
2. Set the looping to **false**.
3. Set the context to the following:

```
ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/E1EDK02
```



4. Click the **Filter** tab.
5. Enter the following in the Condition field:

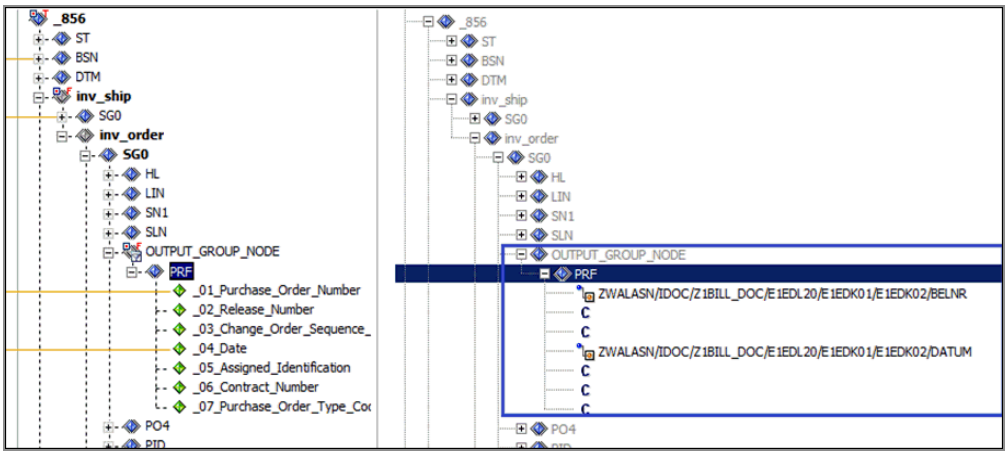
```
ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/E1EDK02/QUALF == '001'
```



6. Configure the mapping values for the PRF elements as indicated by the following table:

Element	Value
PRF01	ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/E1EDK02/BELNR
PRF04	ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/E1EDK02/DATUM

Your screen should now resemble the following image:

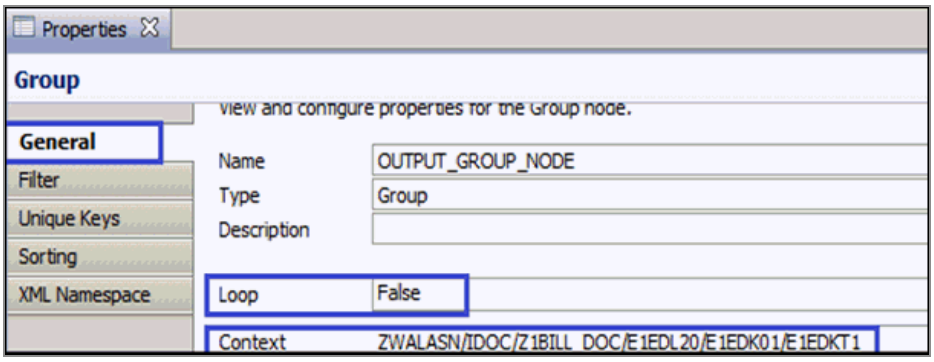


Configure the REF DP (Department) Elements

Procedure

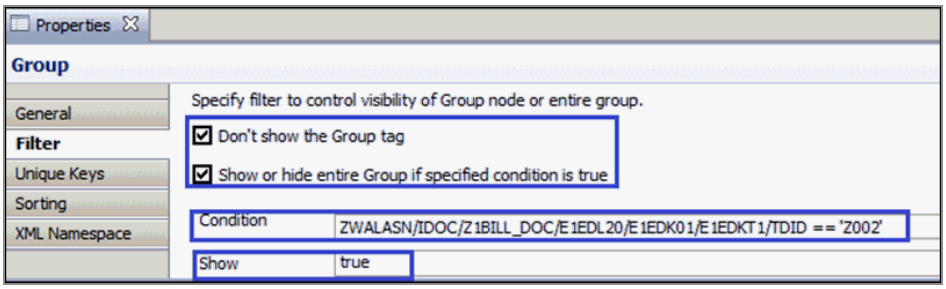
1. Add a hidden group node and move the PRF segment into it.
2. Set the looping to *false*.
3. Set the context to the following:

```
ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/E1EDKT1
```



4. Click the **Filter** tab.
5. Enter the following in the Condition field:

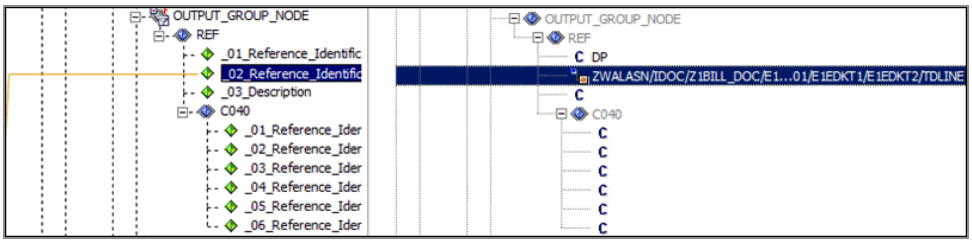
```
ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/E1EDKT1/TDID == 'Z002'
```



6. Configure the mapping values for the REF elements as indicated by the following table:

Element	Value
REF01	DP
REF02	ZWALASN/IDOC/Z1BILL_ DOC/E1EDL20/E1EDK01/E1EDKT1/E1EDKT2/TDLINE

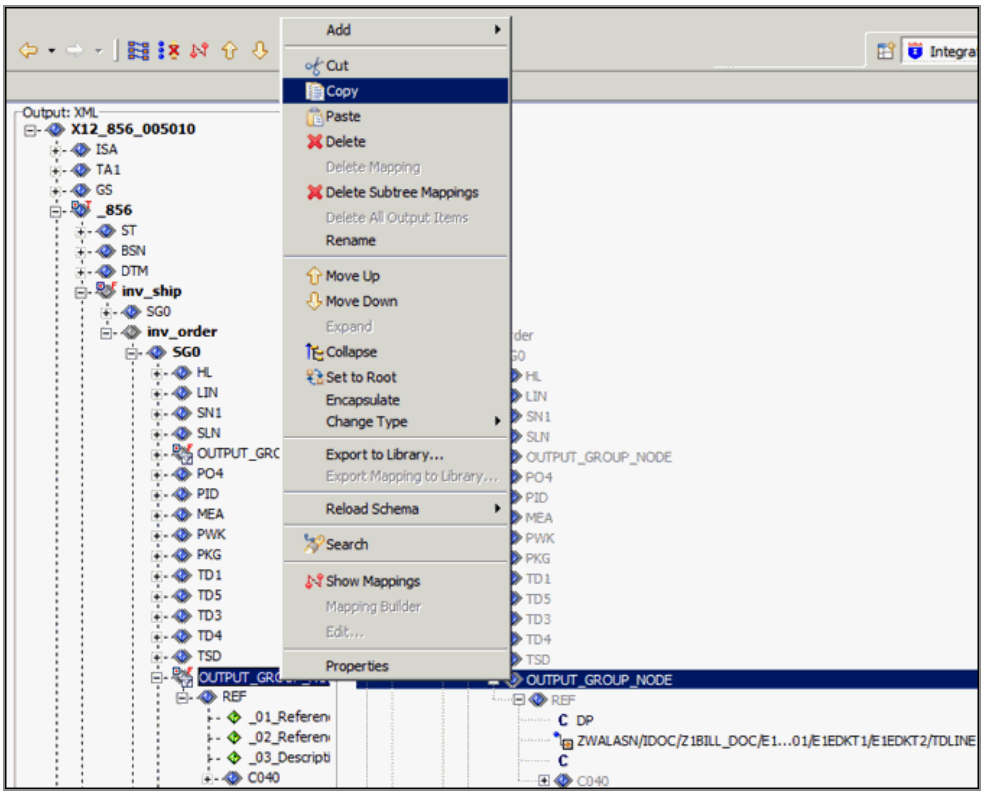
Your screen should now resemble the following image:



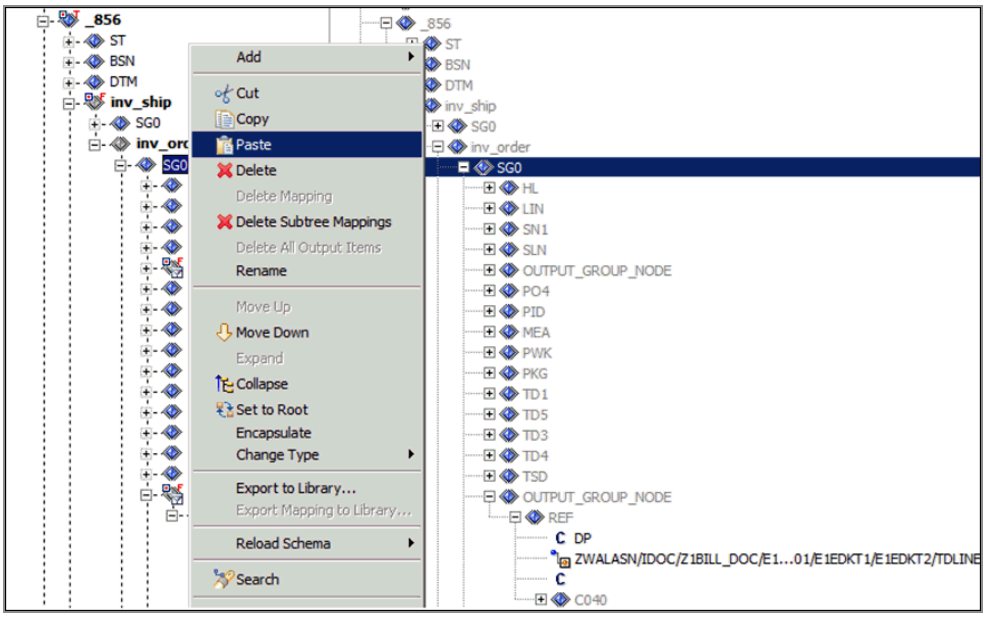
Configure the REF MR (Merchandise Type) Element

Procedure

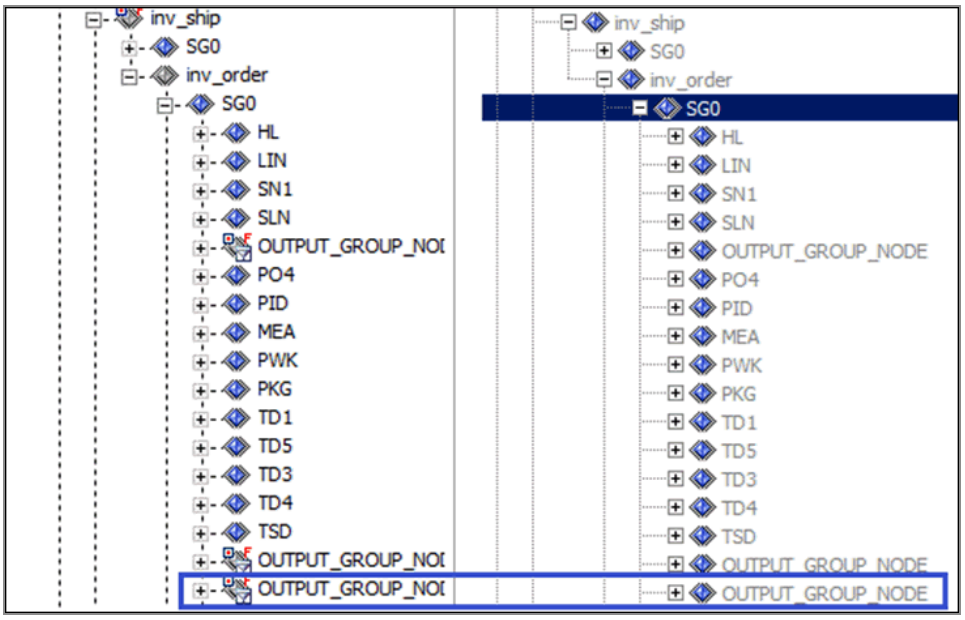
1. Copy the group you just mapped.



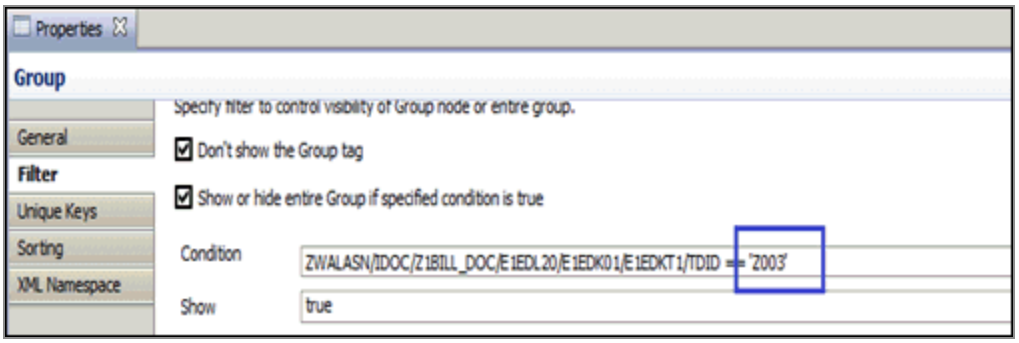
2. Highlight SG0 at the top of the order group.
3. Paste the REF DP element.



4. Move the group to below the REF DP element.

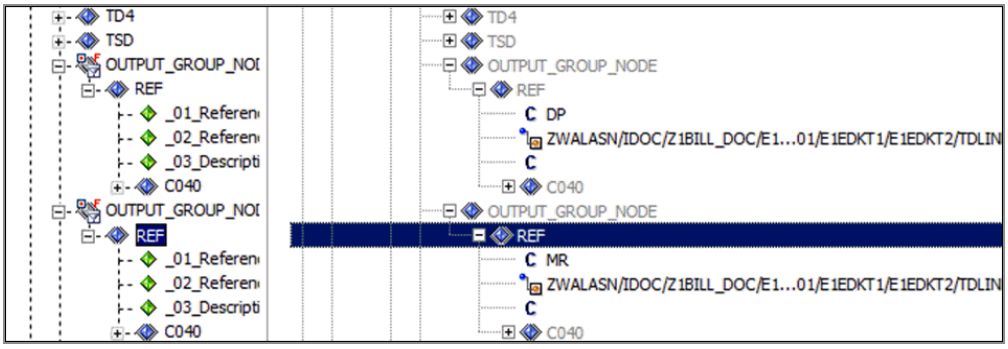


5. Change the specified condition in the Filter tab from Z002 to Z003.



6. Change REF01 from DP to MR.

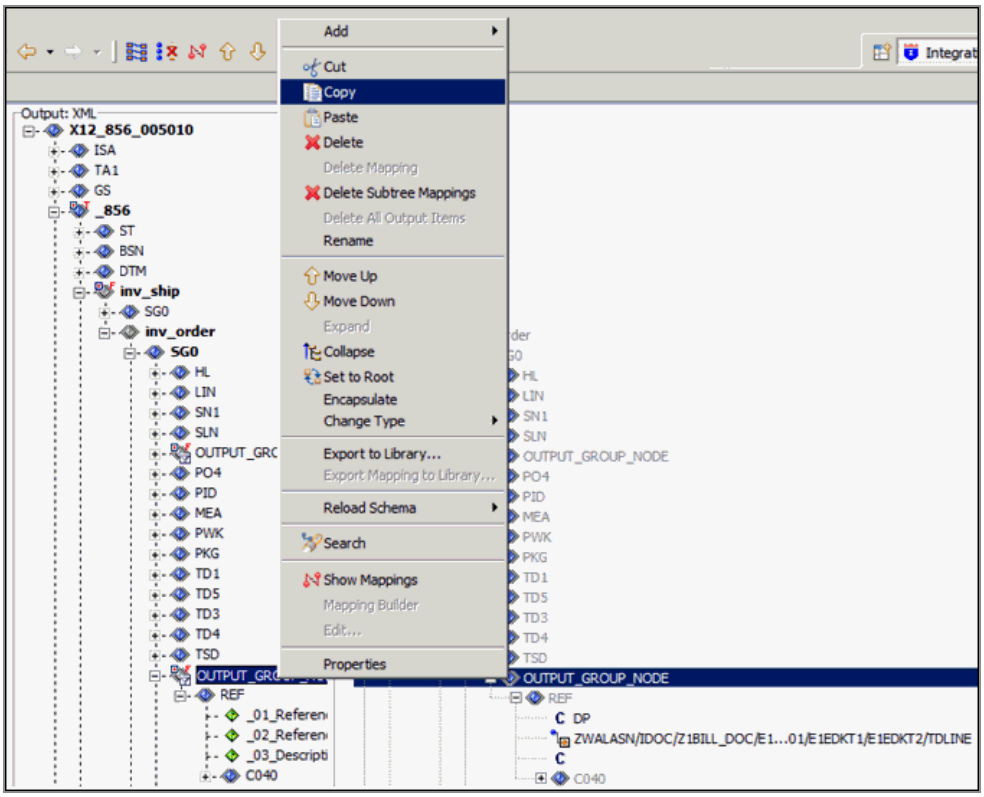
   Your screen should now resemble the following image:



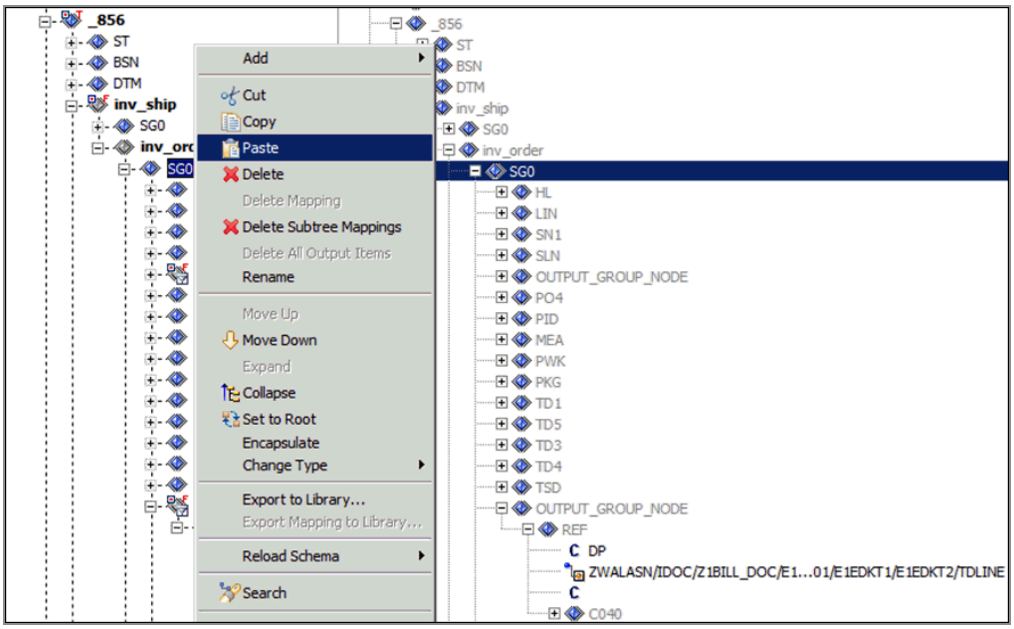
Configure the REF IA (Internal Vendor Number) Element

Procedure

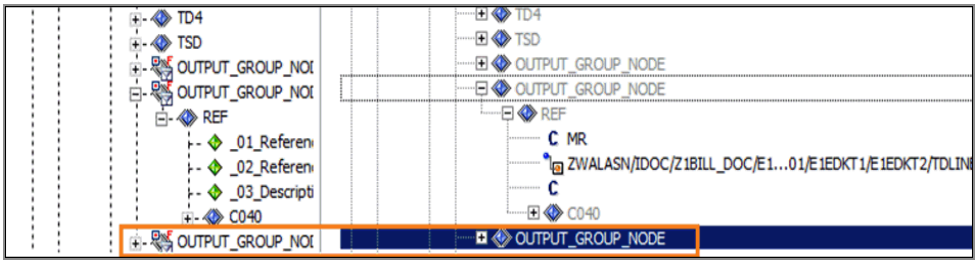
1. Copy the group you just mapped.



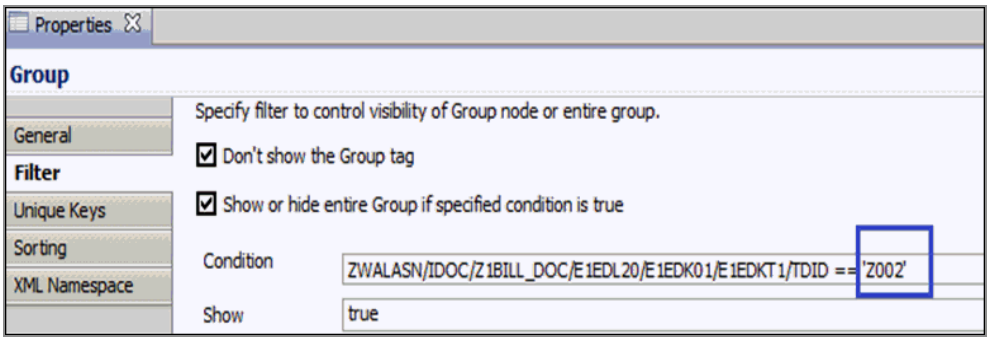
2. Highlight SG0 at the top of the order group.
3. Paste the REF MR element.



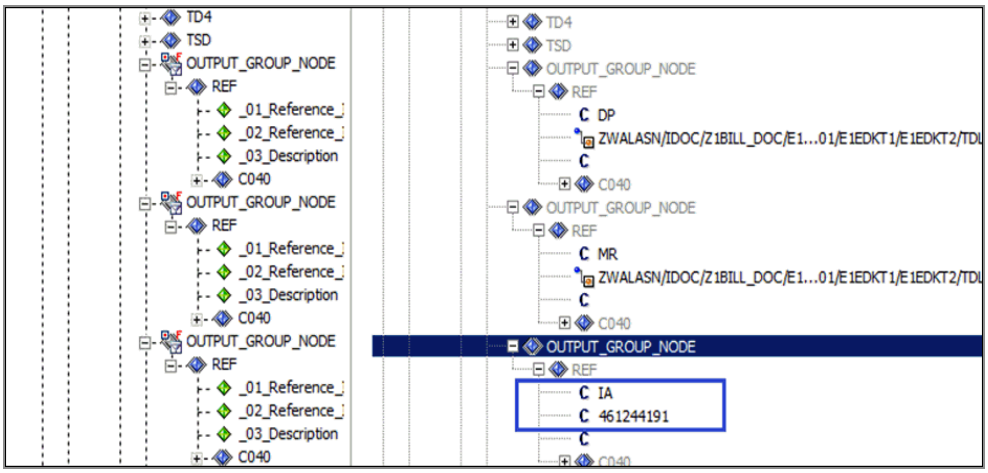
4. Move the group up until it is located below the REF MR element.



5. Leave the filter as Z002 which is copied from REF DP.



6. Set value IA to REF01 and value VENDORID to REF02.



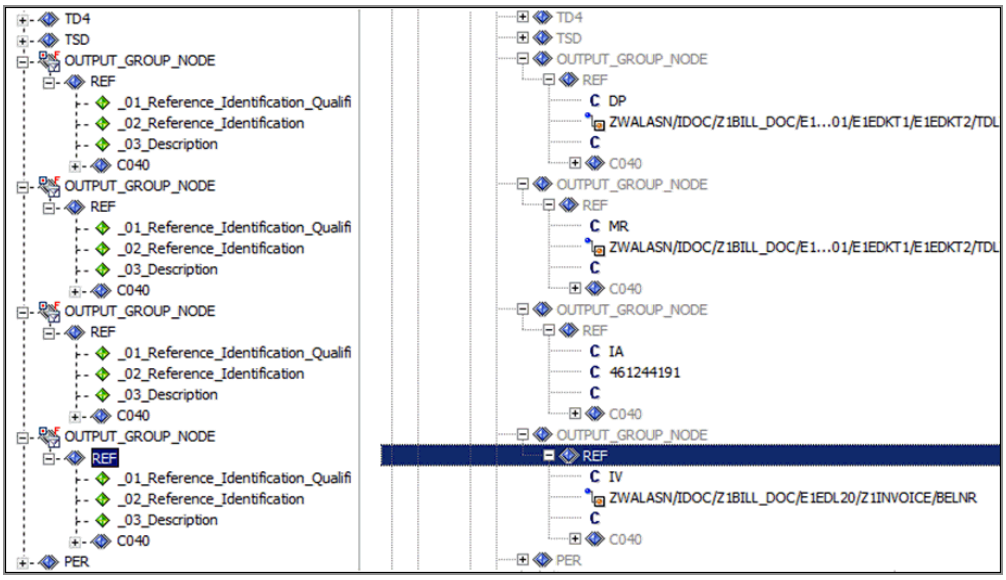
You have now mapped three of the four required REF segments.

Configure the REF IV (Invoice Number) Element

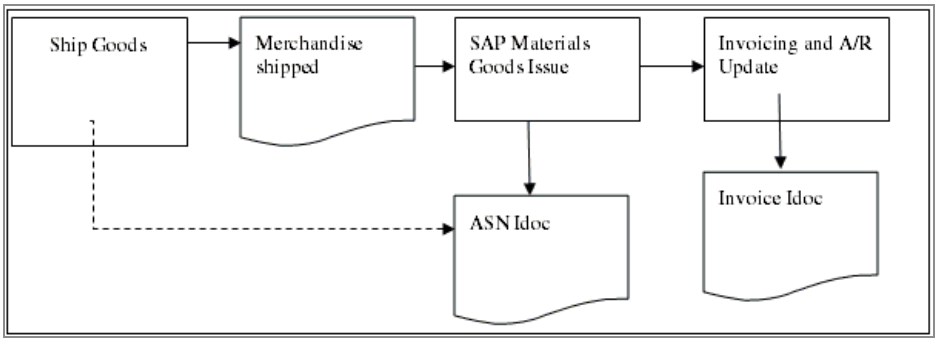
Procedure

1. Paste one more REF node and leave it as the last one in the group. Leave the filter set to Z002 and the context set the same as the previous REF elements.
2. Set value REF01 to IV and map ZWALASN/IDOC/Z1BILL_ DOC/E1EDL20/Z1INVOICE/BELNR to REF02.

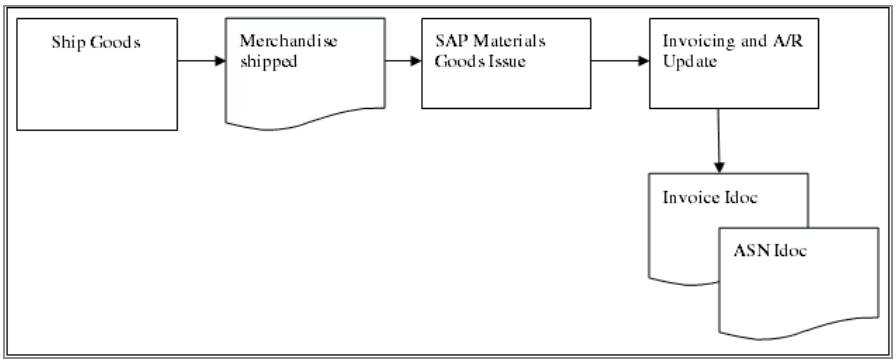
   Your screen should now resemble the following image:



Note that the invoice number is located in ZWALASN/IDOC/Z1BILL_ DOC/E1EDL20/Z1INVOICE/BELNR. This field is an extension field to the IDoc. The stock SAP IDoc for the ASN does not contain the invoice number. This is due to the order of operations in the flow to ship merchandise, as shown in the following diagram.



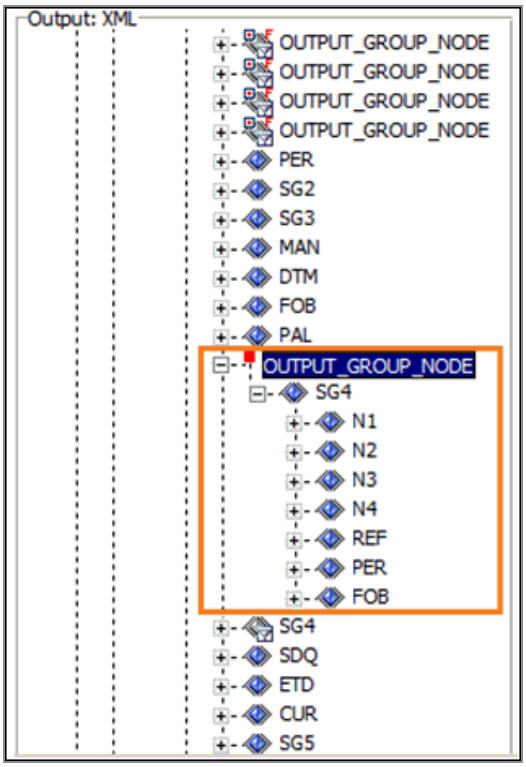
The ASN IDoc is usually created as a by-product of the Material Goods Issue Update process. Depending on the shipping system, it might be possible to create the ASN IDoc from the "truck has left" notification from the shipping system to SAP. Invoicing is usually a batch process that runs overnight. In this example, because the trading partner requires the invoice number on the ASN, the Invoicing process is run periodically during the day, either for all customers or just those EDI customers that require the invoice number on the ASN. The invoice number appears in an extension field on the IDoc. The following image is how the flow should now appear.



Configure the N1 BY (Buyer) Element

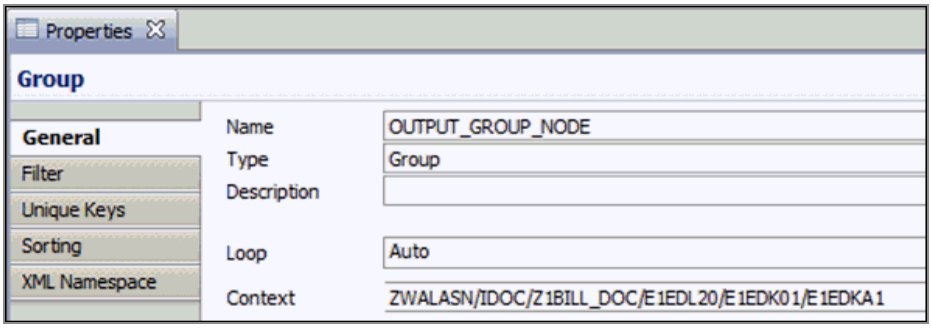
Procedure

1. Add a new hidden group tag, and move it up so the new tag is above the SG4 group and below the SG3 group under the SG0 node in inv_order.
2. Copy the SG4 group to the new group.



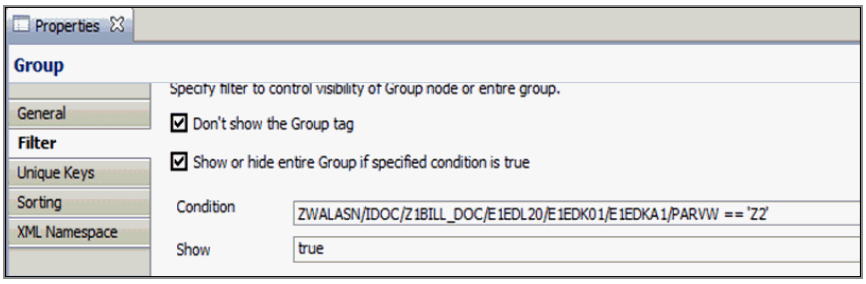
3. Set the context to the following:

```
ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/E1EDKA1
```



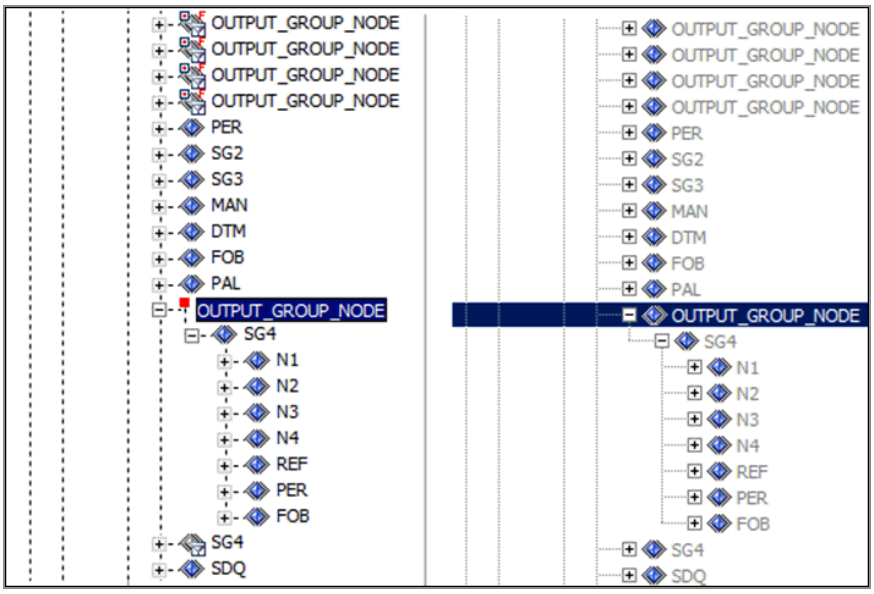
4. Click the **Filter** tab.
5. Enter the following in the Condition field:

```
ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/E1EDKA1/PARVW == Z2
```



6. Click **OK**.

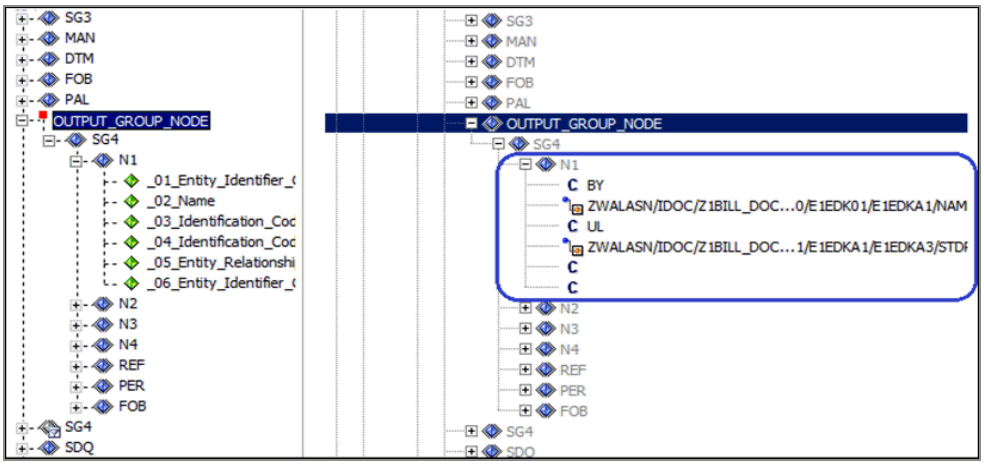
   Your screen should now resemble the following image:



7. Configure the mapping values for the N1 elements as indicated by the following table.

Element	Value
N101	BY
N102	ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/E1EDKA1/NAME1
N103	UL
N104	ZWALASN/IDOC/Z1BILL_ DOC/E1EDL20/E1EDK01/E1EDKA1/E1EDKA3/STDPN

Your screen should now resemble the following image:



Configuring Pack Level Segments

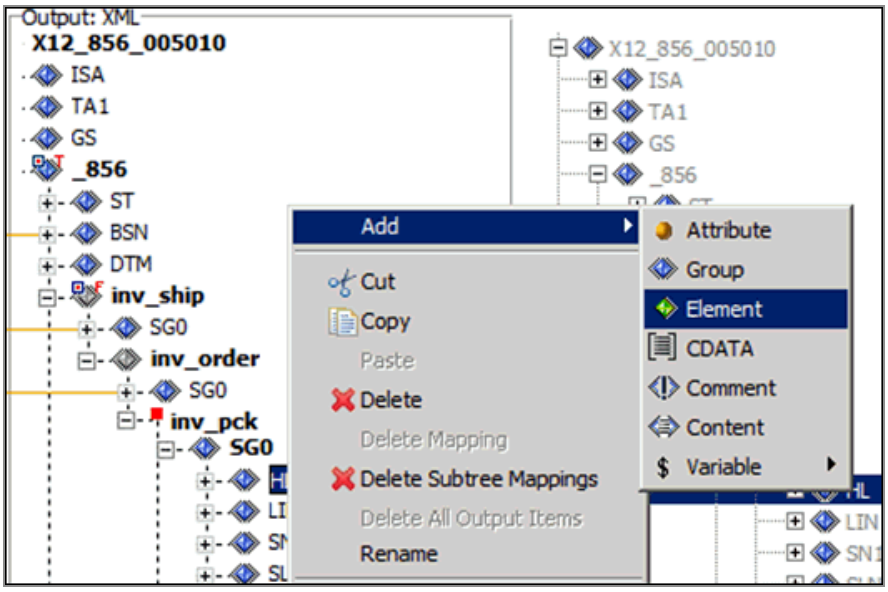
The following sections describe how to map the pack level segments.

Configure the HL (Hierarchy) Segment

Each pack (box, for example) has a bar code label with a serial number. This segment represents that pack and contains that serial number. For common carriers, the shipper and recipient mutually agree on the format and content of the labels. If the package is being transported by a "small package" carrier like Federal Express or UPS, the labels are formatted to the specifications of the freight company. UPS and Federal Express allow shippers to create labels over the web, and often provide hardware and software to allow the shipper to create the labels and accompanying manifests at the location of the shipper. We previously discussed the looping format of the advanced ship notice for common carrier shipments. A small package ASN may have only one pack (and possibly no items) per ship level, and may contain multiple ship levels in a document. The common carrier ASN usually contains the 20 character readable bar code from the label with a GM qualifier. The small package ASN contains the package tracking number with a SM qualifier.

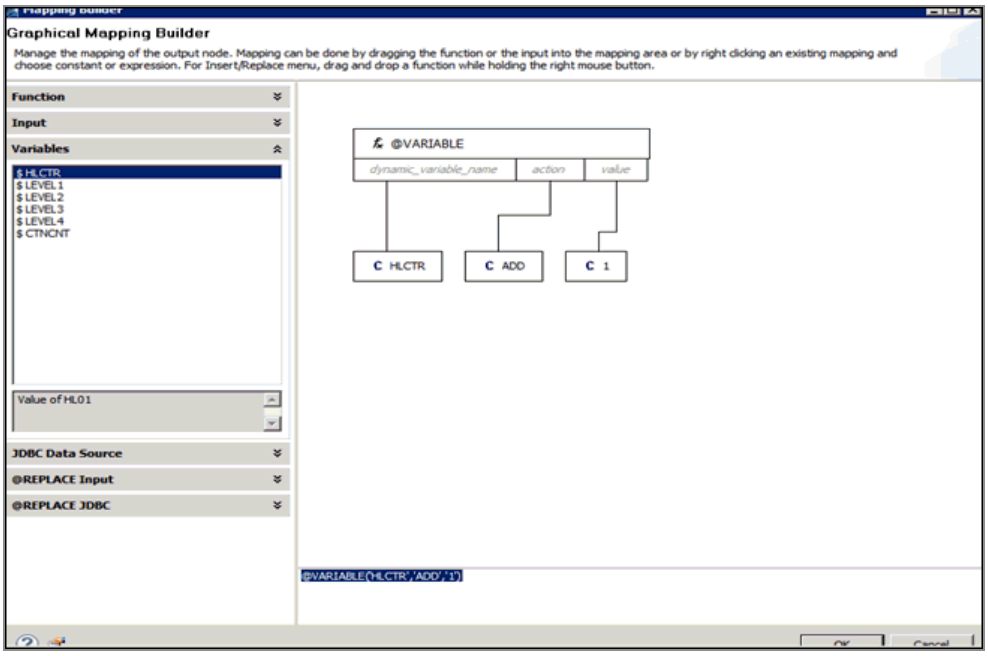
Procedure

1. After expanding **inv_pck** and then **SG0**, right-click the **HL** tag, select **Add**, and click **Element**.

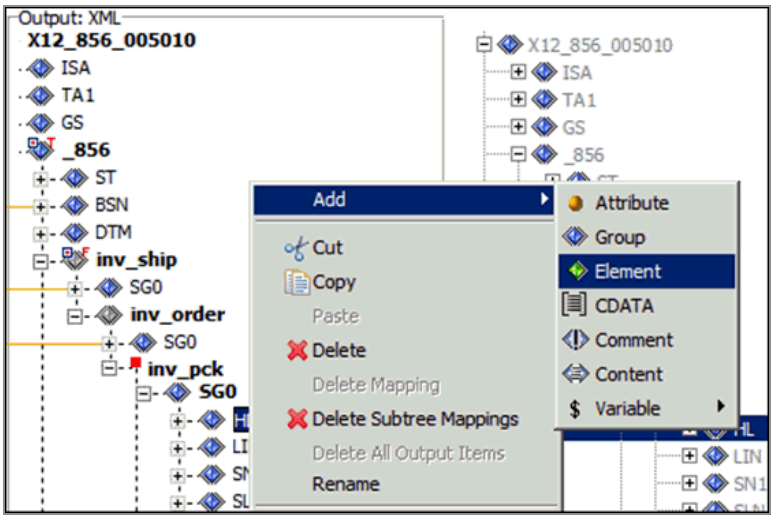


2. Rename the element to *incr_hlctr*.
3. Use the position icons (Move Up and Move Down) on the toolbar (or right-click and select from the control menu) to position the *incr_hlctr* element below the HL tag.
4. Configure the *incr_hlctr* element as indicated in by the following syntax:

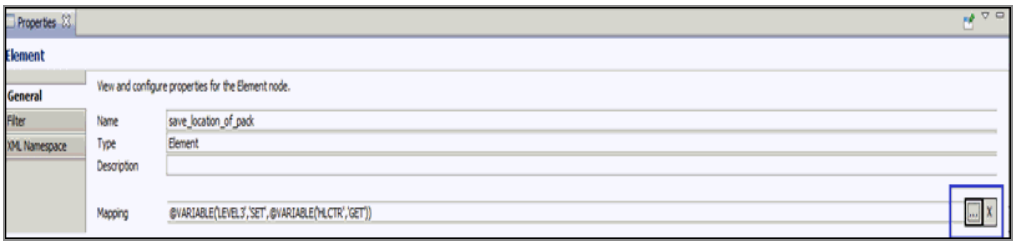
```
@VARIABLE('HLCTR','ADD','1')
```



5. Right-click the **HL** tag under *SG0*, which is located under *inv_pck*, select **Add**, and then select **Element**.



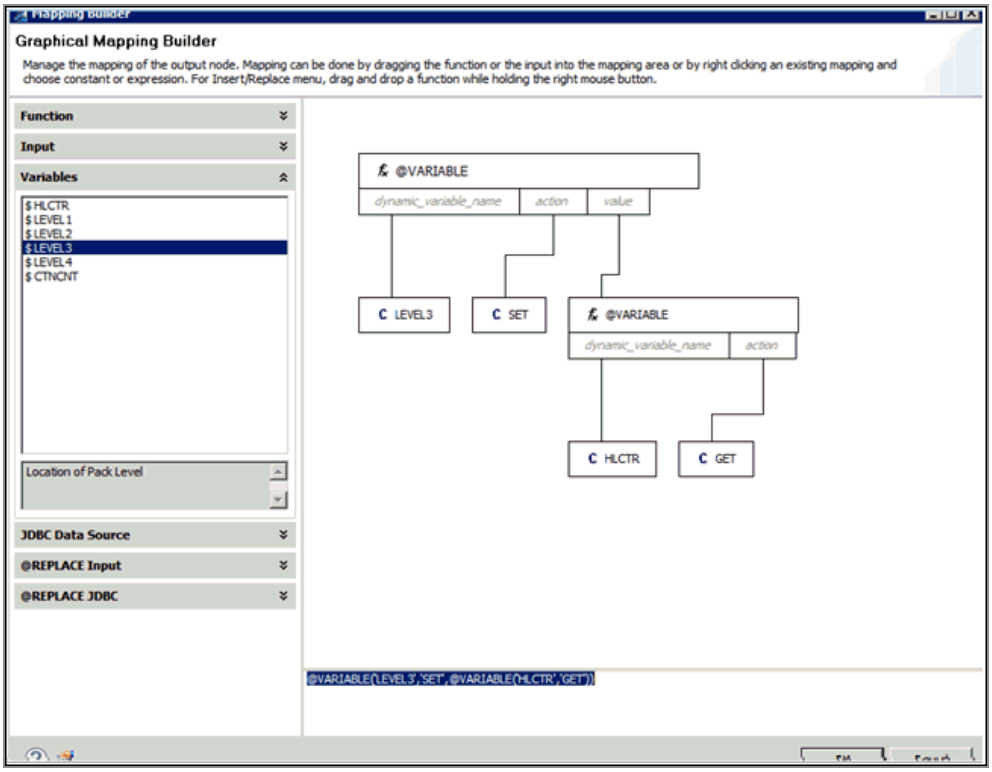
6. Rename this element to *save_location_of_pack*.
7. Right-click the **save_location_of_pack** element tag in the Output pane and select **Properties** from the context menu.
8. Click on the ellipses (...) icon for mapping the **save_location_of_pack** element.



9. Configure the *save_location_of_pack* element as indicated by the following table:

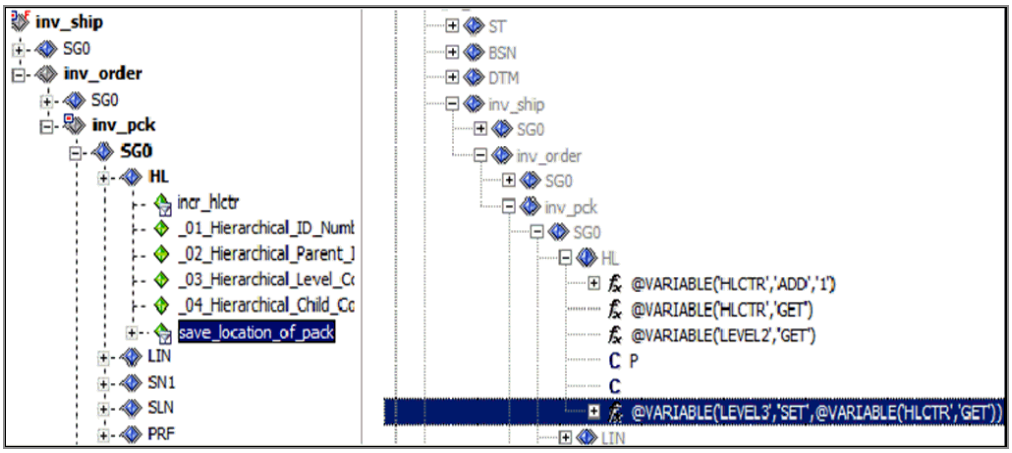
Element	Value
Save_location_of_pack	@VARIABLE('LEVEL3','SET',@VARIABLE ('HLCTR','GET'))

Your screen should now resemble the following image:



10. Use the position icons (Move Up and Move Down) on the toolbar (or right-click and select from the control menu) to position the *save_location_of_pack* element at the bottom of the **HL** tag, if it is not already positioned at the bottom.

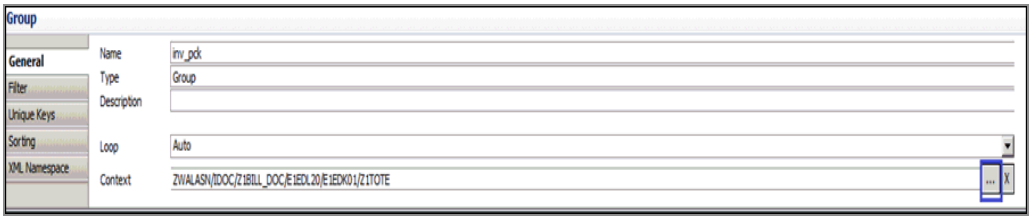
Your screen should now resemble the following image:



11. Configure the mapping values for the HL elements as indicated by the following table:

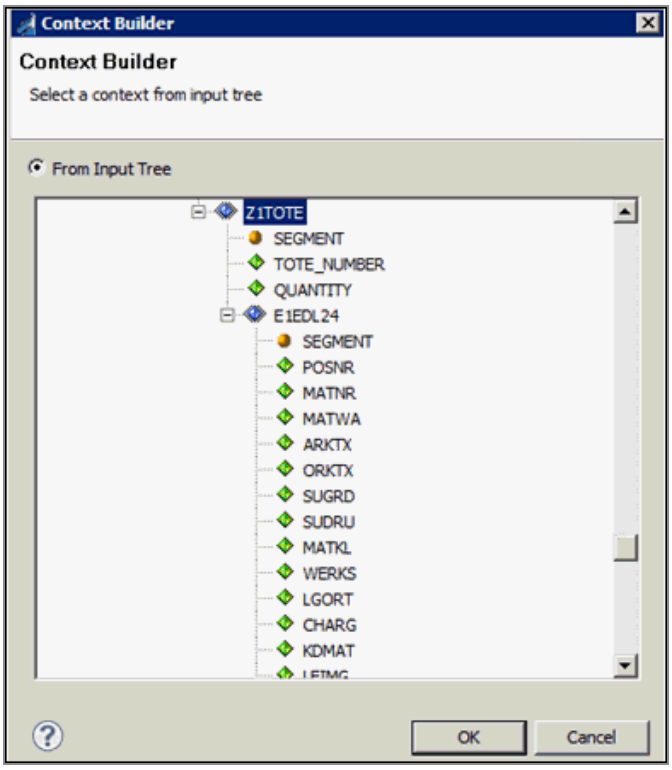
Element	Value
HL01	@VARIABLE('HLCTR','GET'
HL02	@VARIABLE('LEVEL1','GET')
HL03	P

12. Right-click the **inv_pck** node, select **Properties**, and then click the ellipses (...) button in the Properties window/General tab.



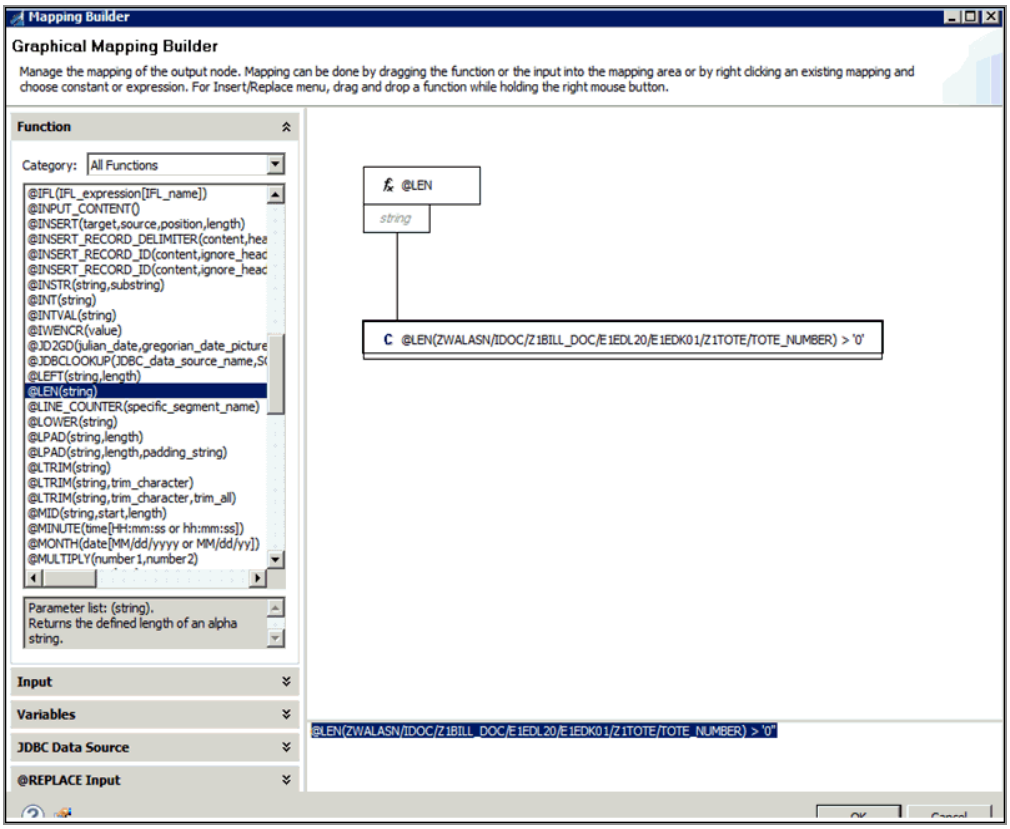
13. Set the context to the following:

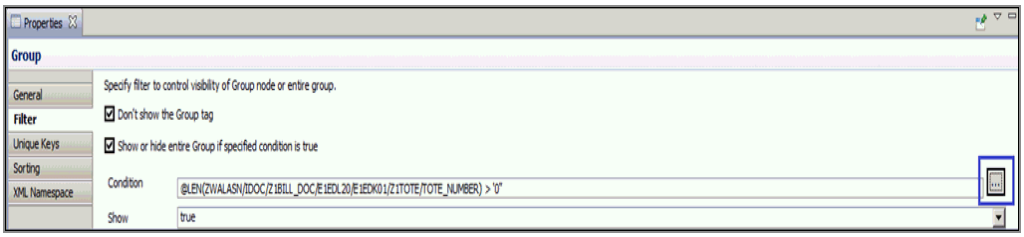
```
ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/Z1TOTE
```



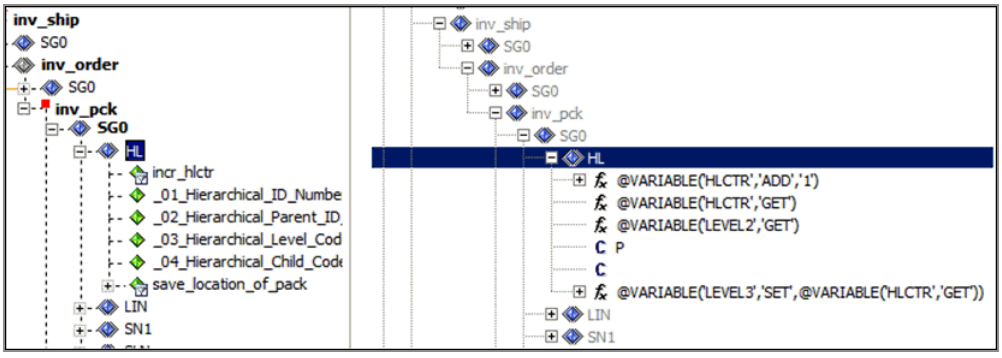
14. Set the filter condition to the following:

```
@LEN(ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/Z1TOTE/TOTE_NUMBER) >
'0'
```





Your screen should now resemble the following image:

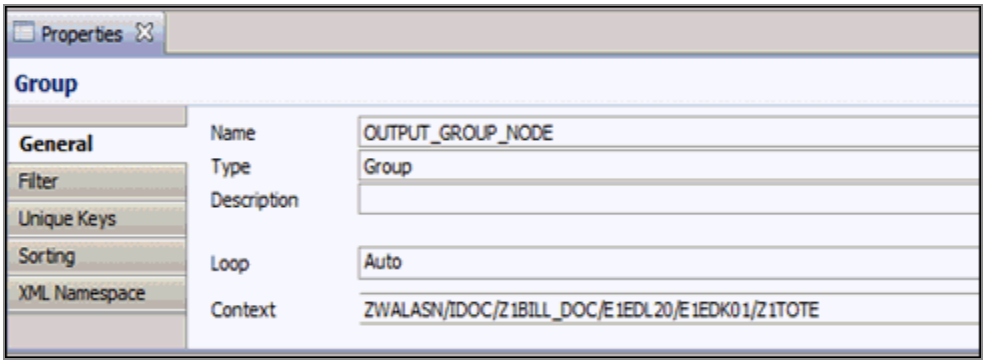


Configure the Man (Manifest) Element

Procedure

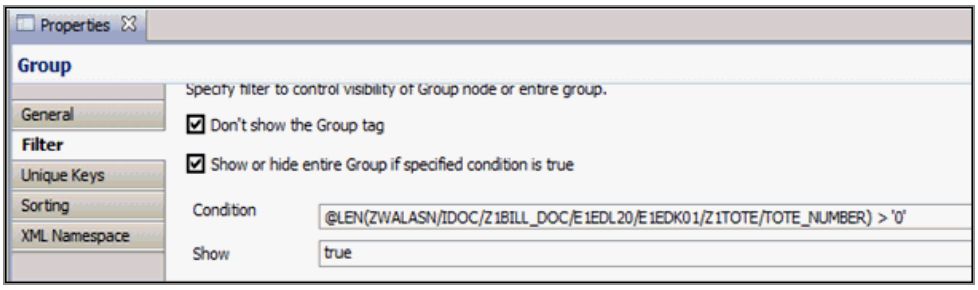
1. Add a new hidden group tag (OUTPUT_GROUP_NODE). Move it up so the new tag is above the MAN segment.
2. Move the MAN segment to the new group.
3. Set the context to the following:

```
ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/Z1TOTE
```



4. Click the **Filter** tab.
5. Set the filter condition to the following:

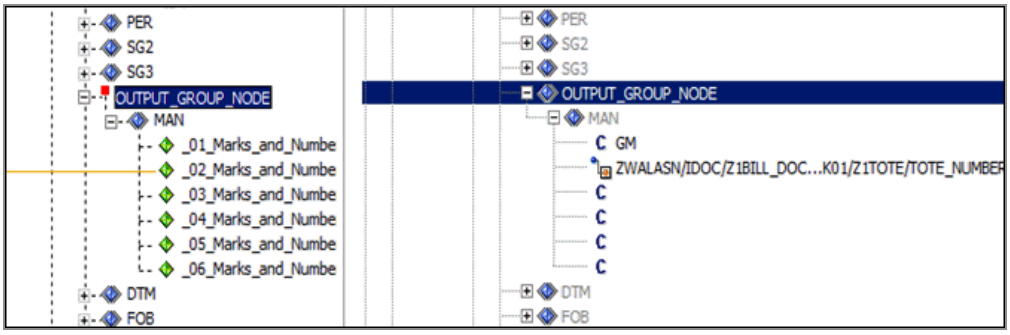
```
@LEN(ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/Z1TOTE/TOTE_NUMBER) >
'0'
```



6. Click **OK**.
7. Configure the mapping values for the Man elements as indicated by the following table:

Element	Value
MAN01	GM
MAN02	ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/Z1TOTE/TOTE_NUMBER

Your screen should now resemble the following image:



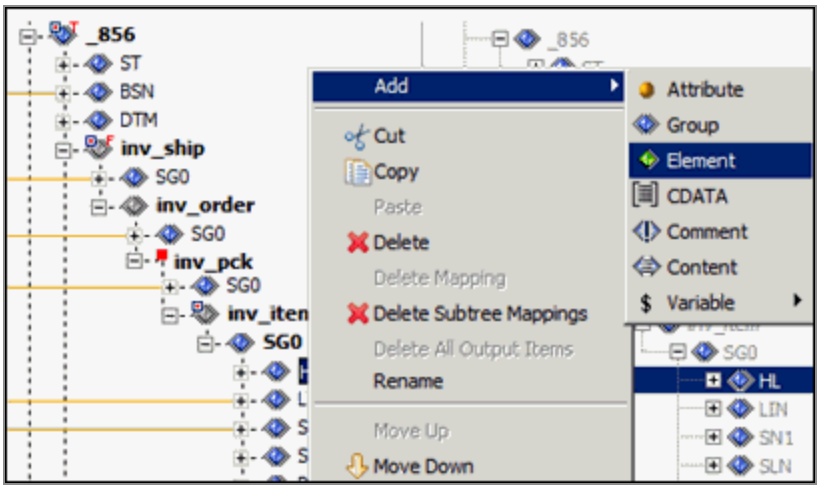
Configuring Item Level Segments

The following sections describe how to map the item level segments.

Configure the HL (Hierarchy) Element

Procedure

1. Expand **inv_item** and then expand **SG0**, right-click the **HL** tag, select **Add**, and then click **Element**.

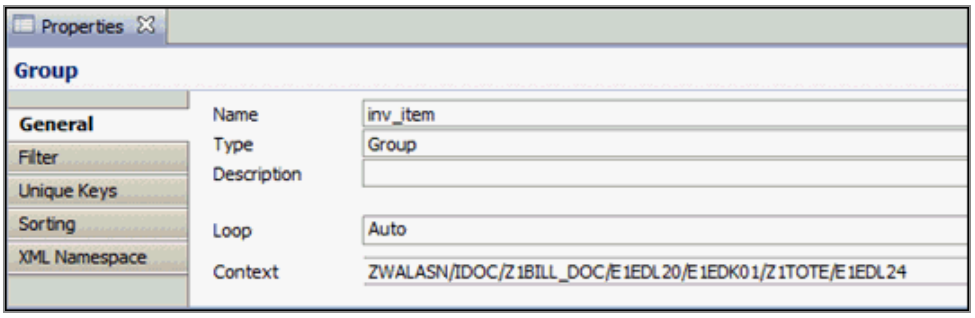


2. Name the element *incr_hlctr*.
3. Use the position icons (Move Up and Move Down) on the toolbar (or right-click and select from the control menu) to position the *incr_hlctr* element below the **HL** tag.
4. Configure the mapping values for the HL elements as indicated by the following table:

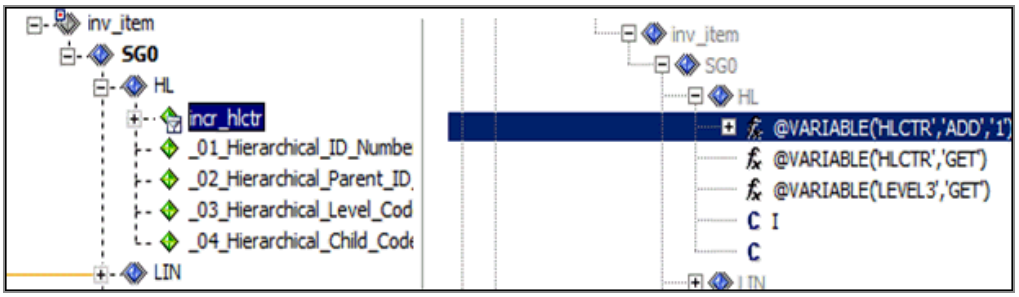
Element	Value
HL01	@VARIABLE('HLCTR','GET')
HL02	@VARIABLE('LEVEL3','GET')
HL03	I

5. Set the context (inv_item) to the following:

```
ZWALASN/IDOC/Z1BILL_DOC/E1EDL20/E1EDK01/Z1TOTE/E1EDL24
```



Your screen should now resemble the following image:



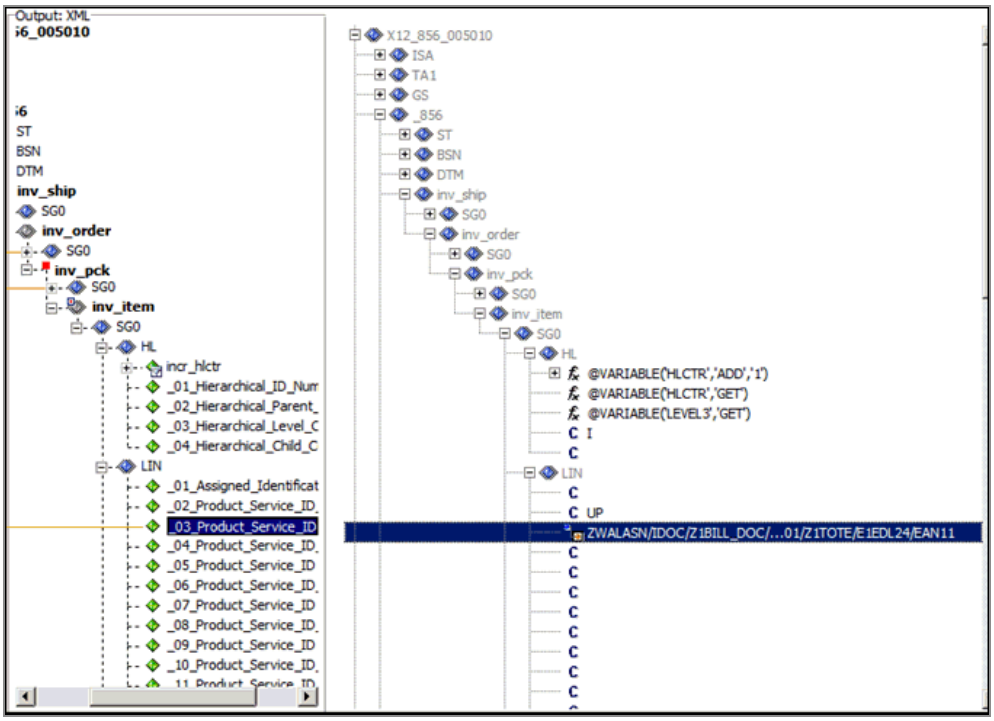
Configure the LIN (Item Identification) Element

Procedure

1. Configure the mapping values for the LIN elements as indicated by the following table:

Value	Data
LIN01	UP
LIN02	ZWALASN/IDOC/Z1BILL_ DOC/E1EDL20/E1EDK01/Z1TOTE/E1EDL24/EAN11

Your screen should now resemble the following image:



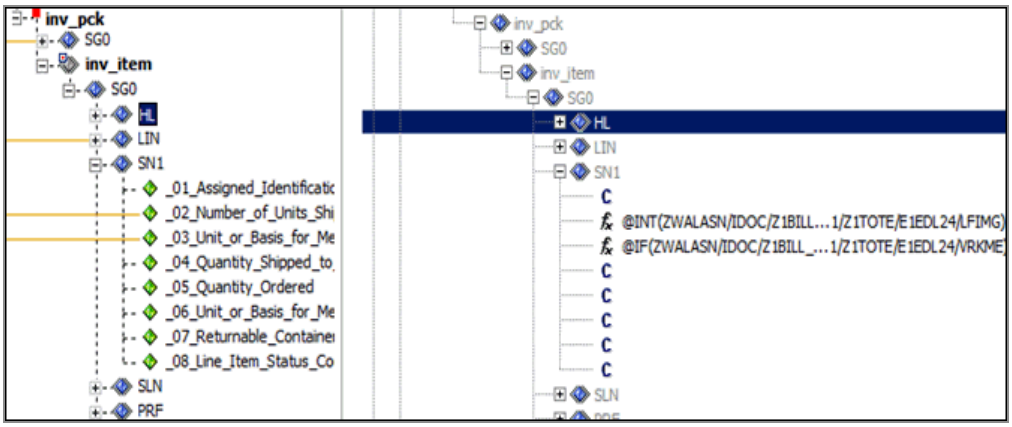
Configure the SN1 (Shipped Item Details) Element

Procedure

1. Configure the mapping values for the SN1 elements as indicated by the following table:

Value	Data
SN102	@INT(ZWALASN/IDOC/Z1BILL_ DOC/E1EDL20/E1EDK01/Z1TOTE/E1EDL24/LFIMG)
SN103	@IF(ZWALASN/IDOC/Z1BILL_ DOC/E1EDL20/E1EDK01/Z1TOTE/E1EDL24/VRKME == 'DZN','DZ',ZWALASN/IDOC/Z1BILL_ DOC/E1EDL20/E1EDK01/Z1TOTE/E1EDL24/VRKME)

Your screen should now resemble the following image:



Configuring the Summary Level Segment

The following section describes how to map the summary level segment.

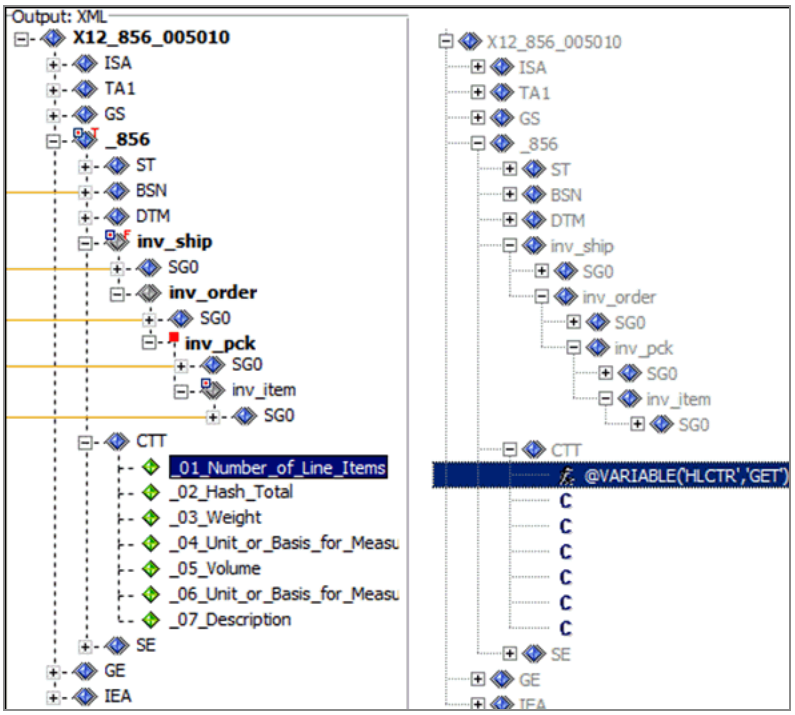
Configure the CTT (Transaction Total) Group

Procedure

1. Configure the mapping value for the CTT element as indicated by the following table:

Value	Data
CTT01	@VARIABLE('HLCTR','GET')

Your screen should now resemble the following image:



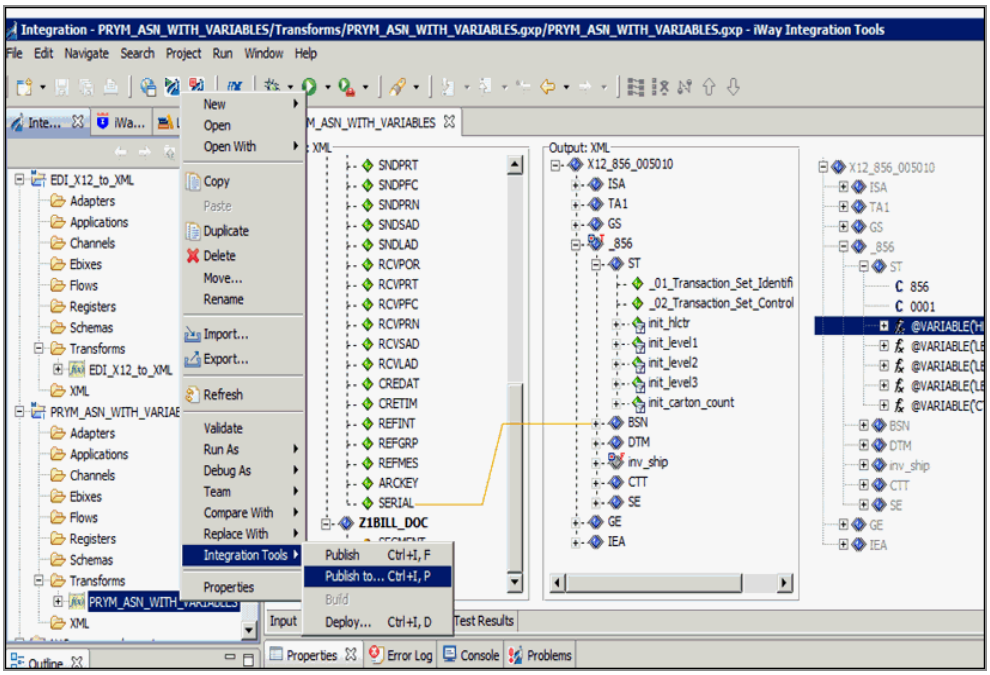
Publishing the Transform Project

In order to build process flows to test and implement the documents, you must first publish the Transform project. For more information on publishing transformations, see the *iWay Integration Tools Transformer User's Guide*.

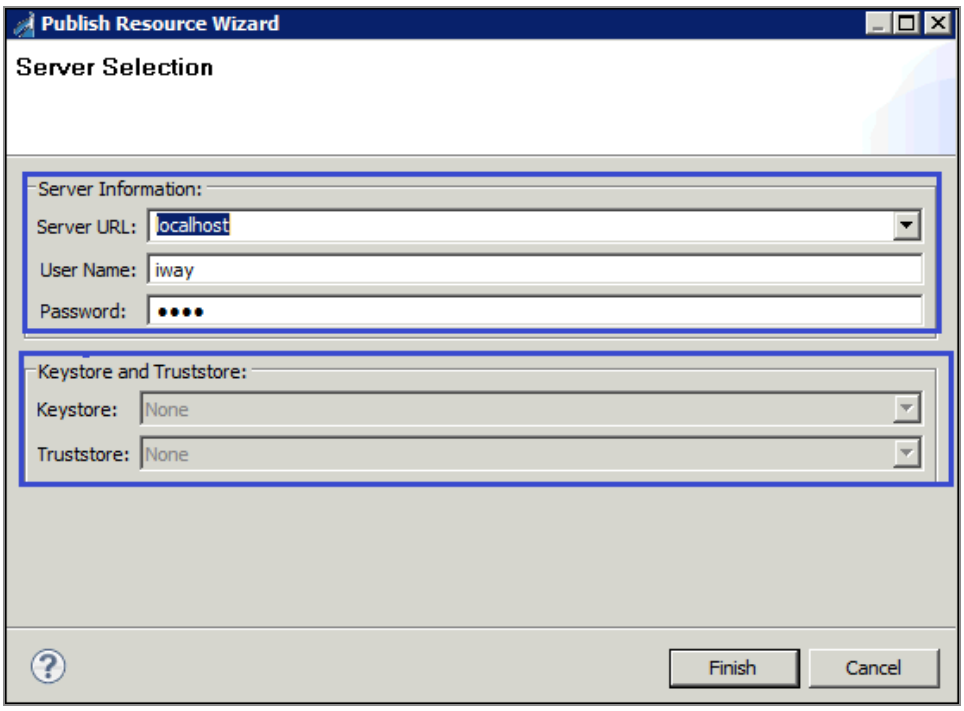
Publish the Transform Project

Procedure

1. From the integration explorer, right-click on a transform project, select **Integration Tools**, and then click **Publish to** from the context menu.

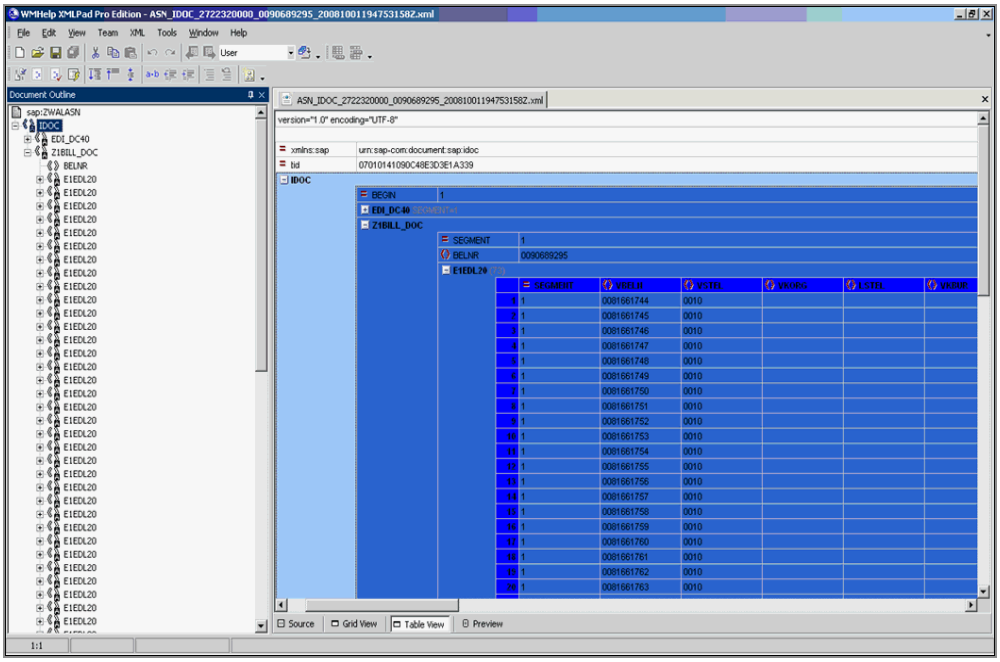


2. Provide the Server URL, User Name, and Password. If the server is HTTPS, then provide the Keystore and Truststore information, and then click **Finish**.



IDoc Structure

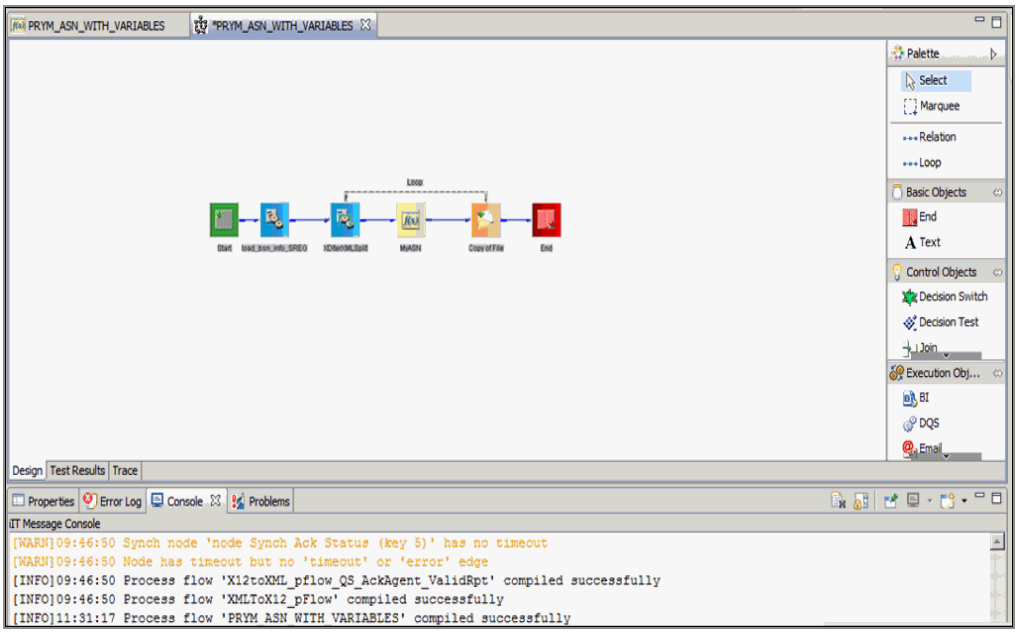
As illustrated below, IDocs are structured to contain multiple units of work. Each unit of work initiates with an E1EDL20 segment and also contains one instance of EDI_DC40 (the IDoc header), a Z1BILL_DOC segment, and a BELNR segment. The invoice number (BELNR) is the common factor of all of these shipments. The shipments are all on the same truck bound for the same distribution center. Each shipment is marked for the destination store. The accompanying invoice is sent to the trading partner in summary form and it details all items sent with a total quantity, regardless of the store breakdown. The invoice mimics the purchase order with the omission of the SDQ segments.



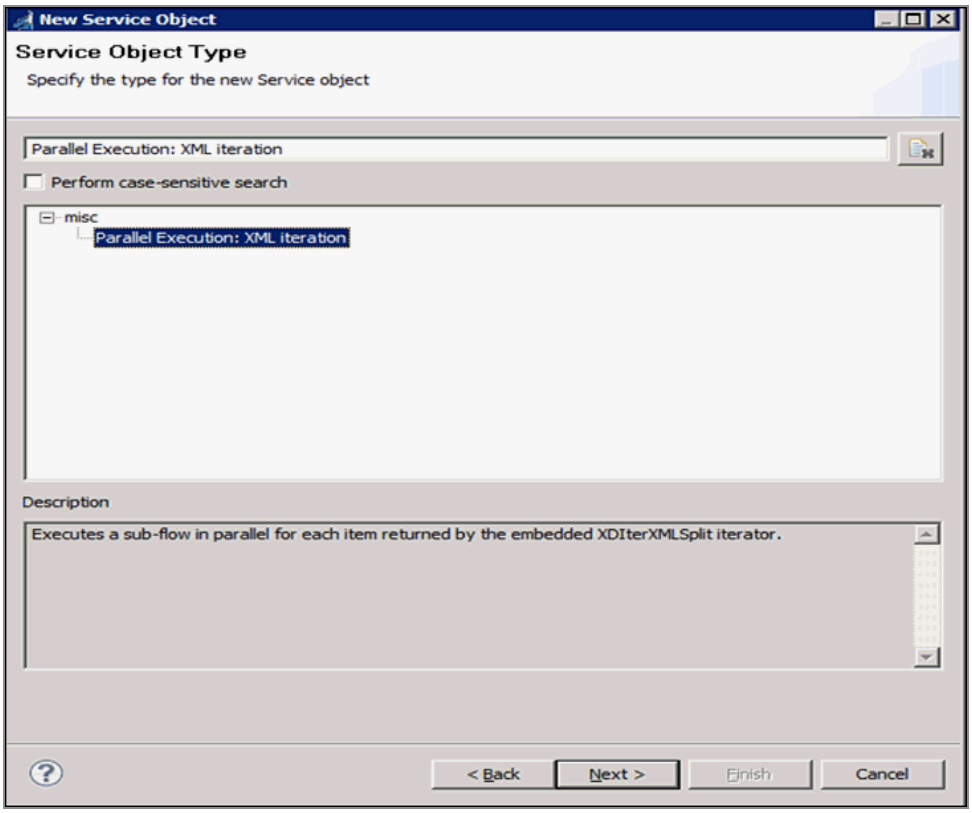
ASN Workflow

To create an ASN for each unit of work, you can pre-process the IDocs as part of the workflow. Create a channel that contains a process flow to do this. A general discussion on channel creation within the context of iWay Service Manager can be found in the **iWay Service Manager User's Guide**.

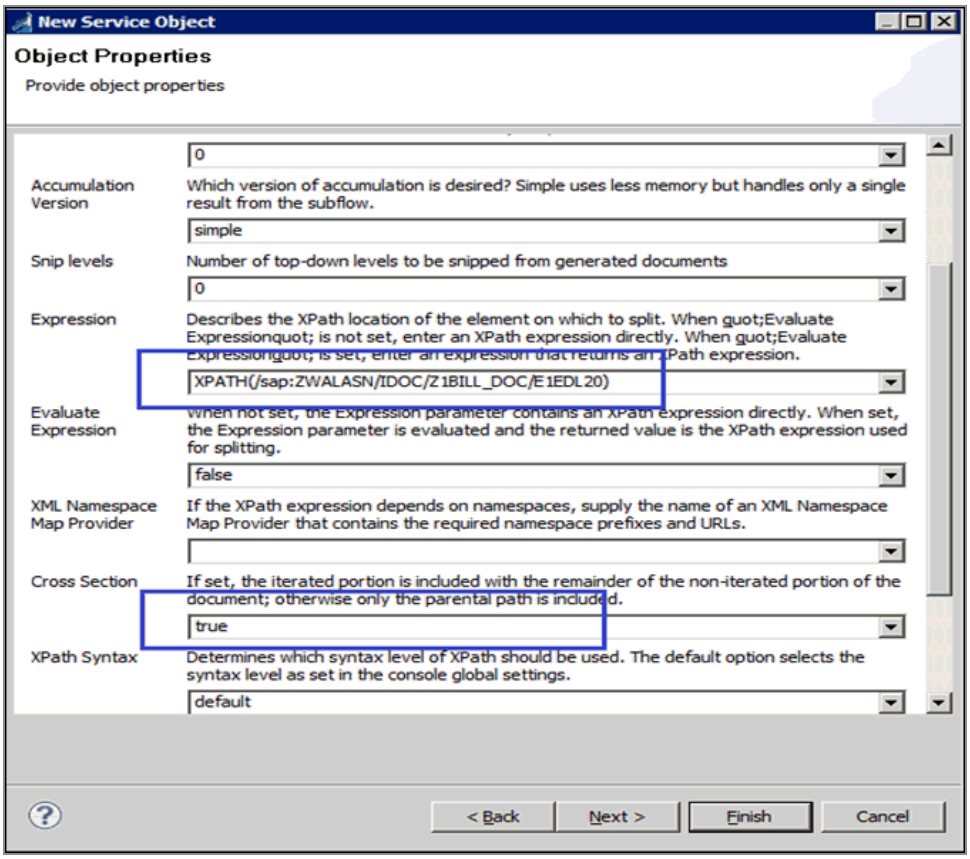
The following image is an example flow.



This process flow contains a simple loop and the same XML splitter agent that was used in the SDQ Splitter example. Then, an iterator object to repeat the same process if multiple units of IDoc occur. This process flow iterates through the IDoc and split multiple IDocs into separate units and then sends those units that were split, into transformation.



Here are the splits.



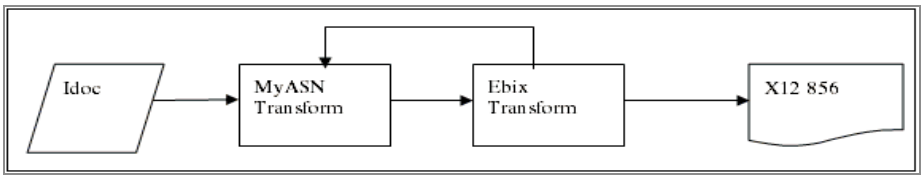
When you are done creating your process flow, publish it to the registry under a meaningful name.

ASN Transformation

There are two different styles to building the channels required for this transformation, a one channel approach and a two channel approach. Both use the "standard EDI outbound channel" as documented in the **EDI manual**. Channel building is also covered in the **iSM manual**.

The One Channel Approach

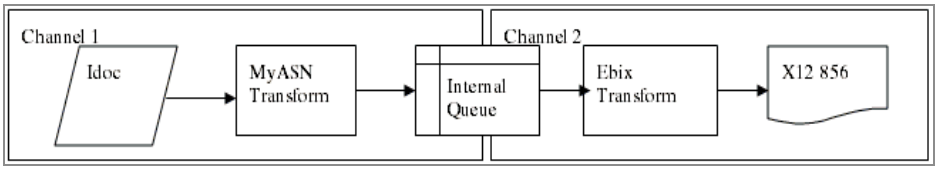
Copy the sample standard channel and all of its components, renaming so they all have unique names. This is done to protect the components from inadvertent changes later on. Add your process flow to the route. Point the listener to the directory that SAP is dropping the IDocs into, and emit to the proper location.



You may have multiple channels, one for each trading partner. Your channel will also run in a serial fashion, split, transform #1, transform #2, next split.

The Two Channel Approach

Create a new channel that just contains the process flow in the route. Use your favorite style to link the two channels, internal listener, directory, and queue. Use the standard channel to run the Ebix transformation.



In this example, channel #1 and channel #2 each run independently and simultaneously. Channel #2 is a standard component that can be reused for many flows. ASN will start to create while the IDoc is still being processed.

The X12 ADN Mapping Final Results

Regardless of the channel method used, you receive an X12 formatted 856 (ASN) document at the end. The following is one sample document from the IDoc.

```
ST*856*0001
BSN*00*0081661744*20081001*1800*0001
HL*1**S
TD1*CTN*2****G*4.398*LB
REF*BM*GRN0571922104
REF*CN*GRN0571922104
DTM*011*20081001
FOB*PP
N1*ST*WAL MART DC 6023D DSDC DEPT 19*UL*0078742029764
N3*21504 COX ROAD
N4*SUTHERLAND*VA*23885
N1*SF*YOUR COMPANY NAME*ZZ*123456789
N3*FIRST ROAD
N4*CITY*NY*01234
```

```
HL*2*1*O
PRF*7400371944***20080930
REF*DP*00052
REF*MR*0073
REF*IA*461244191
REF*IV*0090689295
N1*BY*WAL-MART STORE 01-1399 DEPT 19*UL*0078742013671
HL*3*2*P
MAN*GM*00007287900027501947
HL*4*3*I
LIN**UP*72879250916
SN1**6*EA
HL*5*3*I
LIN**UP*72879252651
SN1**6*EA
HL*6*3*I
LIN**UP*73650991219
SN1**6*EA
<extraneous lines omitted>
HL*15*3*I
LIN**UP*72879870480
SN1**3*EA
HL*16*2*P
MAN*GM*00007287900027502010
HL*17*16*I
LIN**UP*72879250916
SN1**6*EA
HL*18*16*I
LIN**UP*72879252651
SN1**6*EA
HL*19*16*I
LIN**UP*73650991219
SN1**6*EA
```

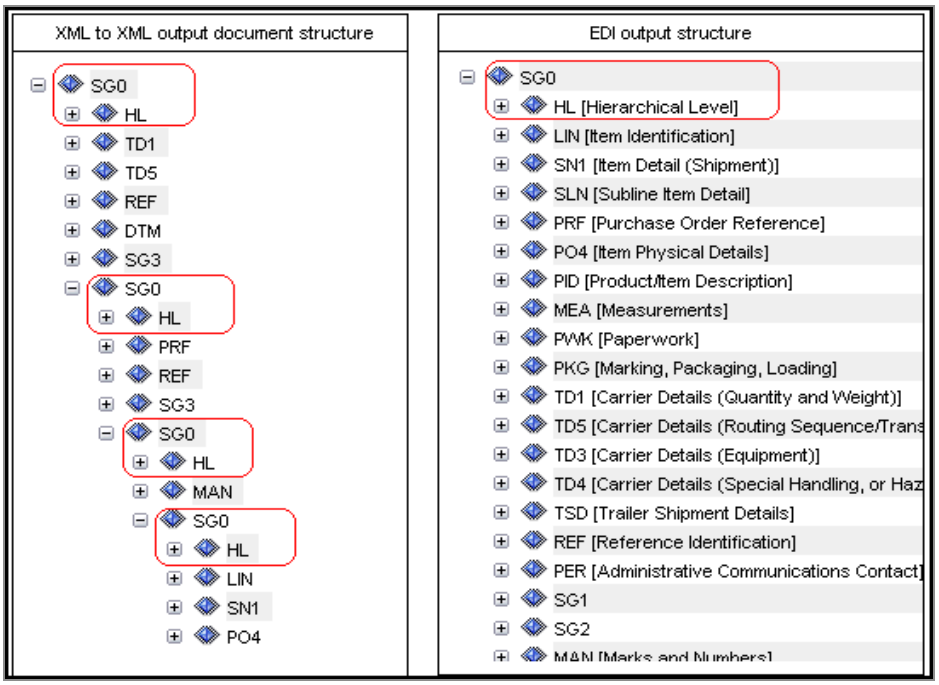
```
<extraneous lines omitted>
```

```
HL*28*16*I
LIN**UP*72879870480
SN1**3*EA
CTT*28
SE*61*0001
```

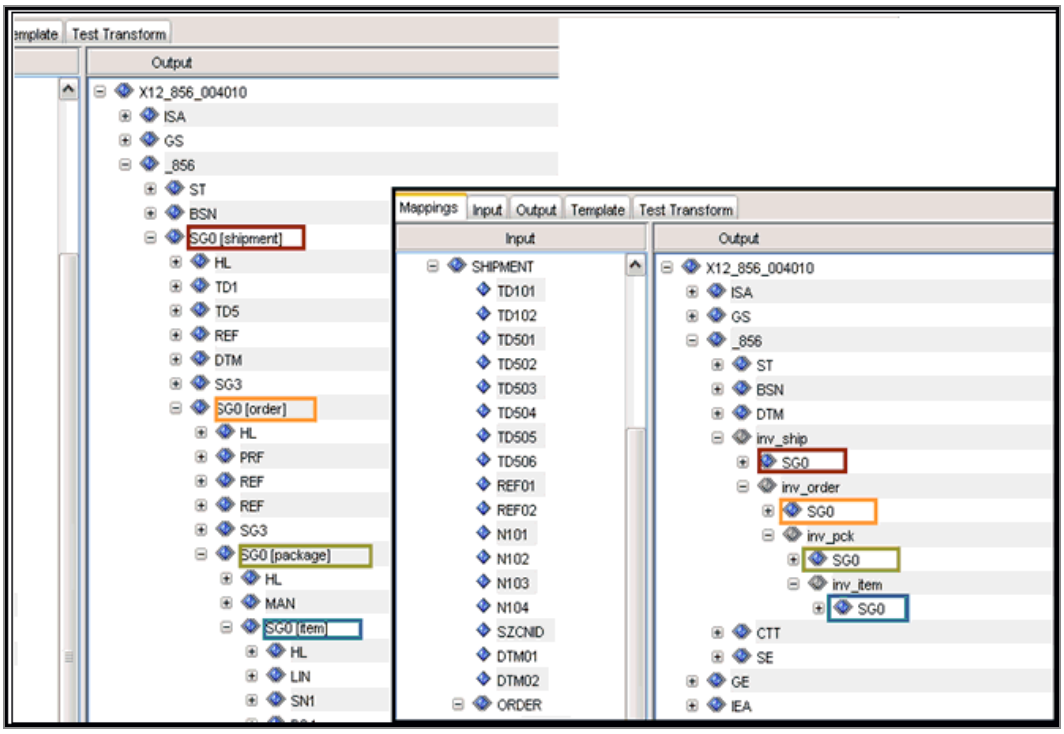
Flattening the Output Structure

The *normal* ANSI X12 looping structure for an ASN consists of a header, a looping detail (HL) and a trailer section. The example actually consists of an embedded looping structure, Orders within Shipments, Packs within Orders, and Items within Packs. The technique discussed previously creates this looping structure, but also causes the output XML to come out in a normalized or *flattened* state.

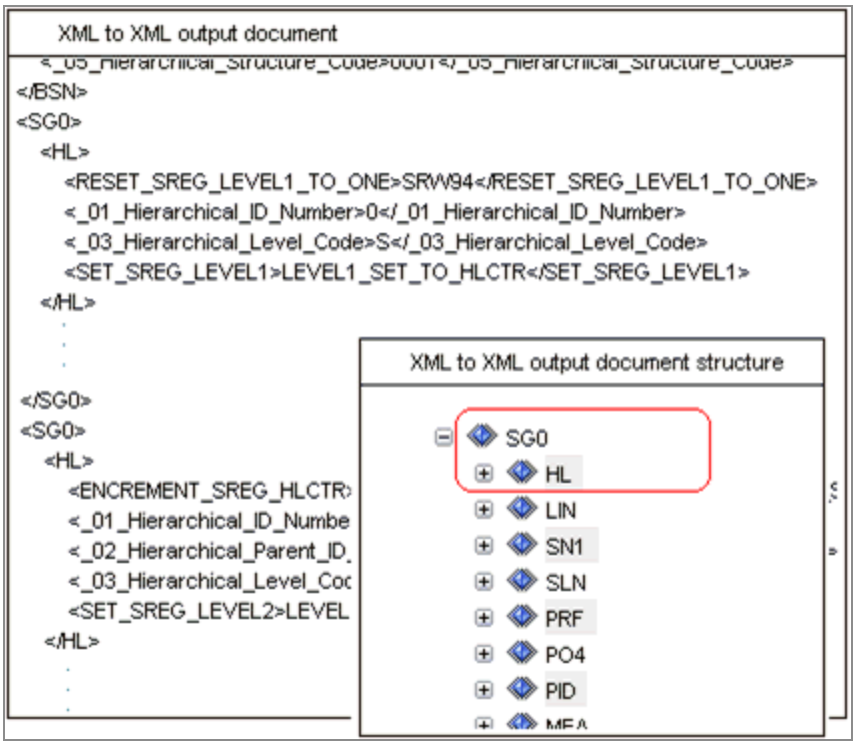
A typical XML to XML transformation produces an embedded SG0/HL structure. One-to-one mapping to EDI structure is not possible.



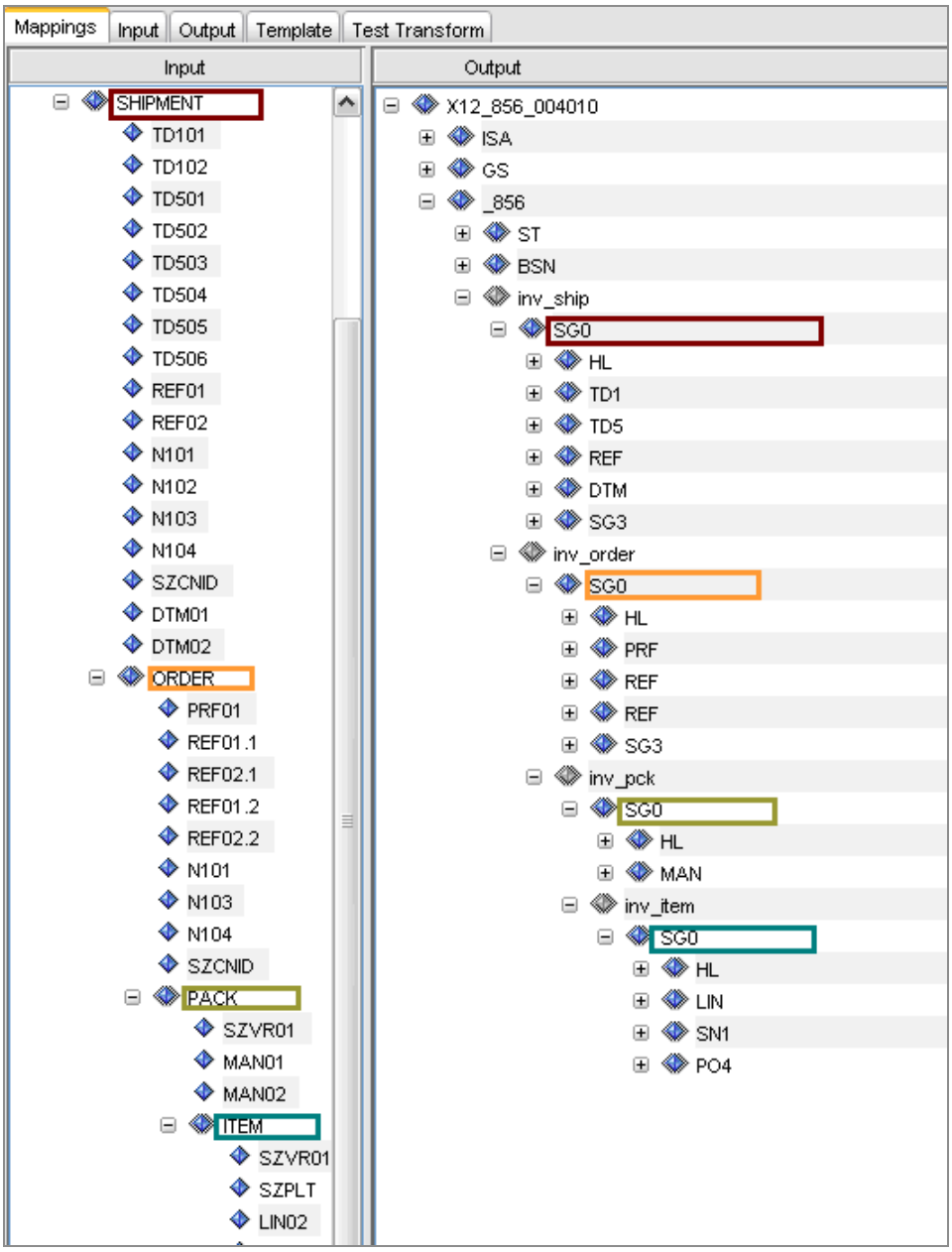
To produce flat SG0/HL structure but still keep the embedded structure for calculation of HL IDs, you can update XML to XML mappings to use the invisible group method to have subsequent SG0 on the same level.



This produces flat SG0/HL structure.



This is an illustration of how your loops may appear:


Mappings
Input
Output
Template
Test Transform
Input
SHIPMENT
TD101
TD102
TD501
TD502
TD503
TD504
TD505
TD506
REF01
REF02
N101
N102
N103
N104
SZCNID
DTM01
DTM02
ORDER
PRF01
REF01.1
REF02.1
REF01.2
REF02.2
N101
N103
N104
SZCNID
PACK
SZVR01
MAN01
MAN02
ITEM
SZVR01
SZPLT
LIN02
Output
X12_856_004010
ISA
GS
_856
ST
BSN
inv_ship
SG0
HL
TD1
TD5
REF
DTM
SG3
inv_order
SG0
HL
PRF
REF
REF
SG3
inv_pck
SG0
HL
MAN
inv_item
SG0
HL
LIN
SN1
PO4

Tutorial: Adding a Detail Line Counter to a Purchase Order Transform

This section provides a tutorial that describes how to add a detail line counter, such as a variable, to a purchase order transform. You will add a variable to the transform will count the total number of detail lines and then insert that total into the document trailer.

Configuring the Required Variables

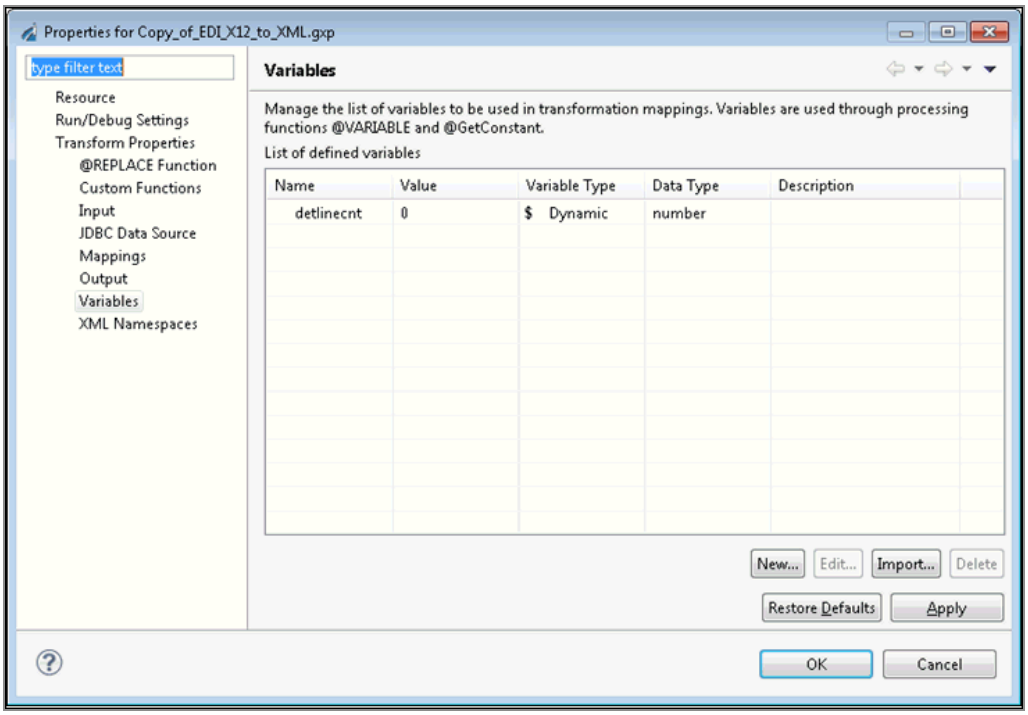
This section describes how to configure a variable and then add this variable to a root node.

Configure a Variable

To configure a variable:

Procedure

1. In Integration explorer, right-click the transform name and select **Properties**.
2. Select the variables and then click **New**.
3. Enter the variable Name, Value, Variable Type, and Data Type, as shown in the following image.

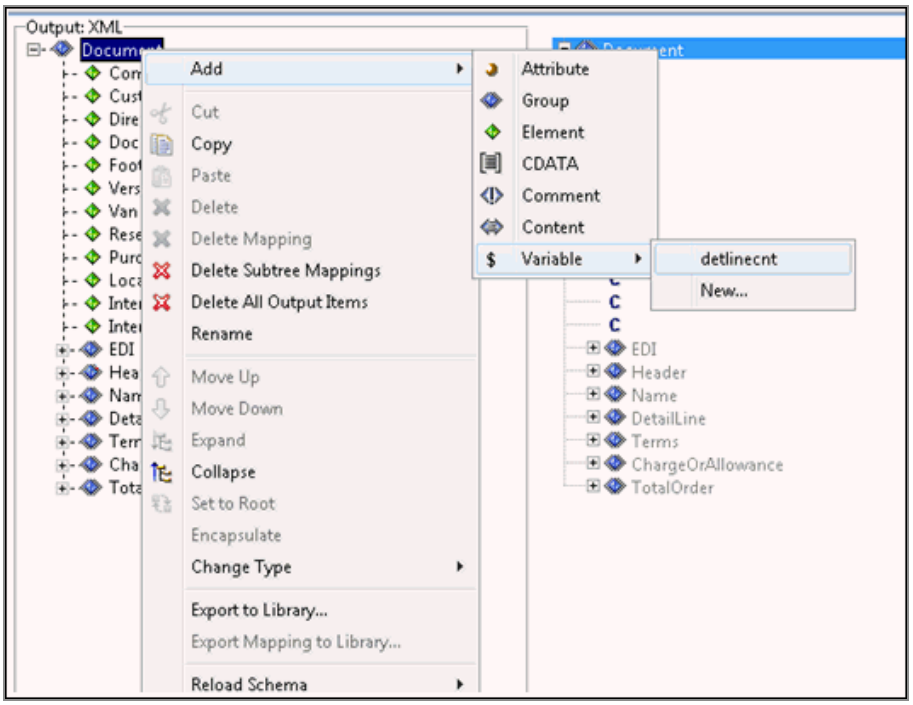


Add a Variable to a Root Node

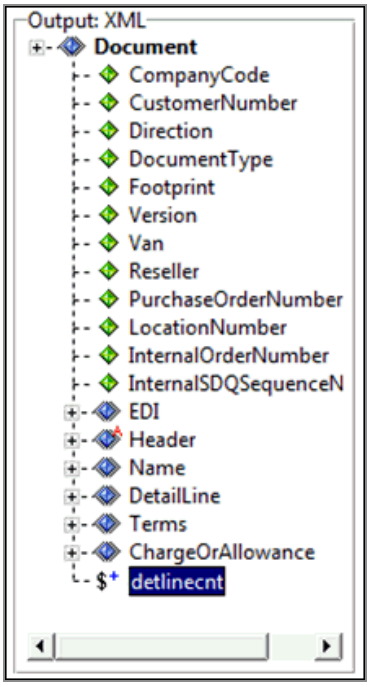
To add a variable to a root node (for example, Document):

Procedure

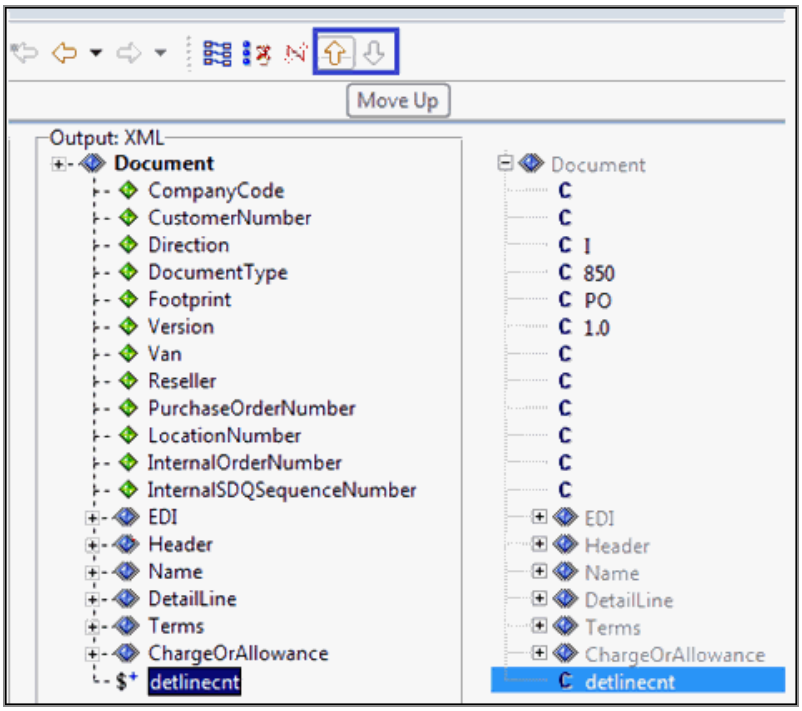
1. Right-click the document root node, click **Add**, select **Variable**, and then click on any newly created variable to add into the Document root tag, for example **detlinecnt**.

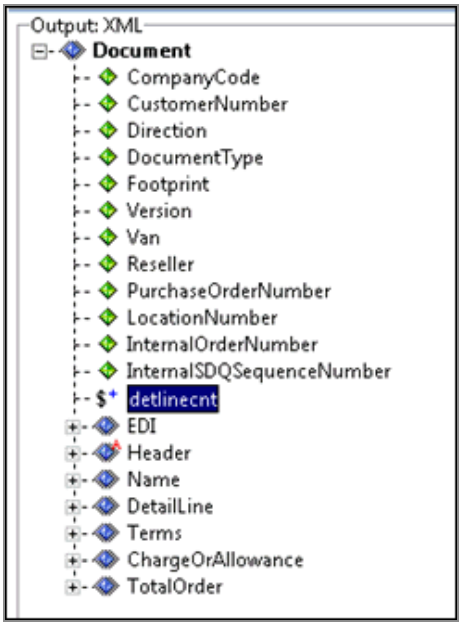


The variable appears in the Output: XML pane, as shown in the following image.



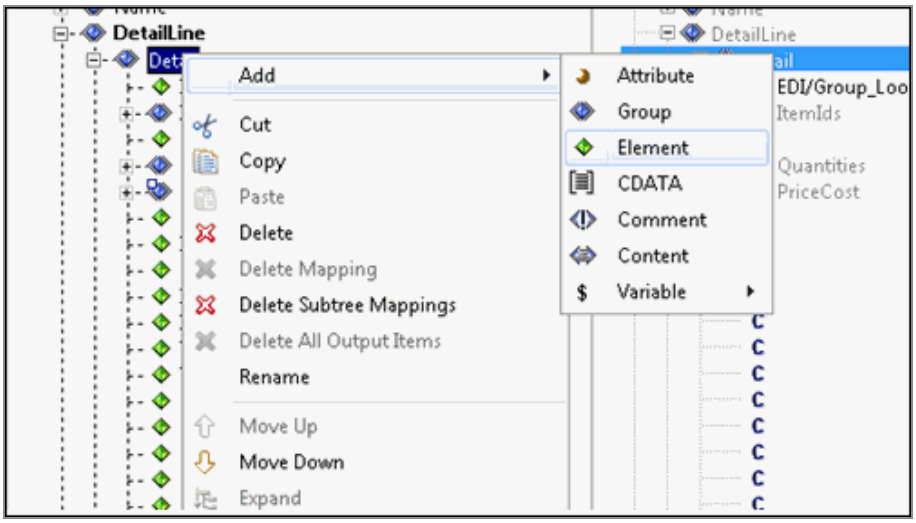
2. Using the up arrow on the button bar, move the newly added variable up.





The counter should be initialized to zero for each document prior to the detail line loop (you must set the counter to 0).

3. Expand the detail line group and detail group, then right-click on the group name, and add a work element that will contain the Line Count Value in the output XML.



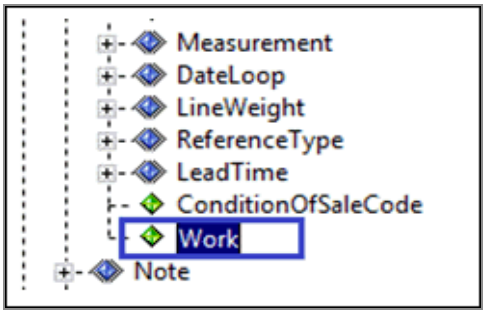
Using the Graphical Mapping Builder

This section describes how to use the Graphical Mapping Builder to manage the mapping of the output node.

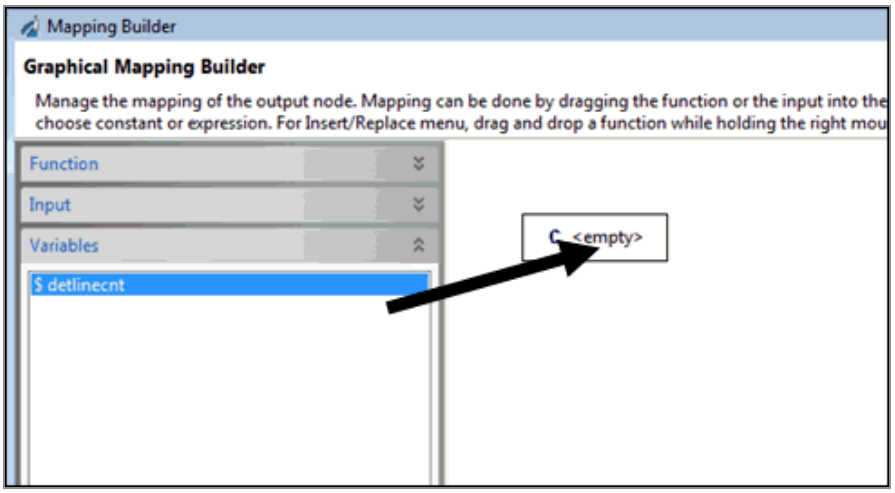
Use the Graphical Mapping Builder

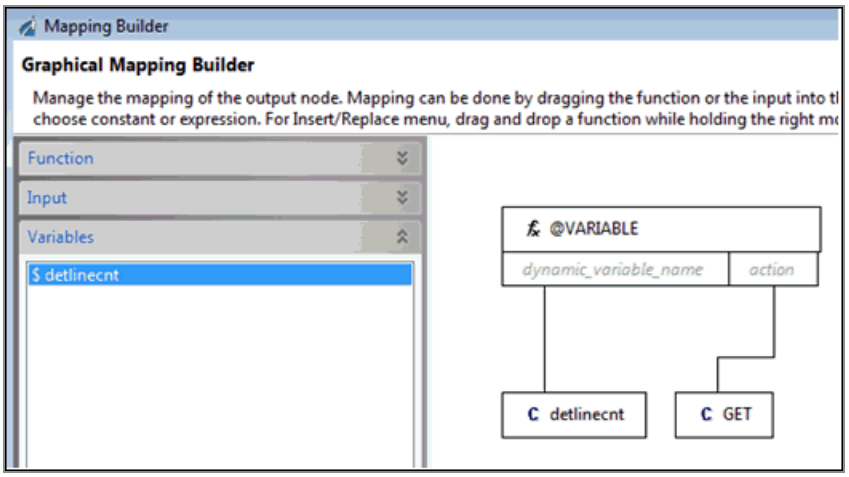
Procedure

1. Double-click the **Work** element to open Graphical Mapping Builder.

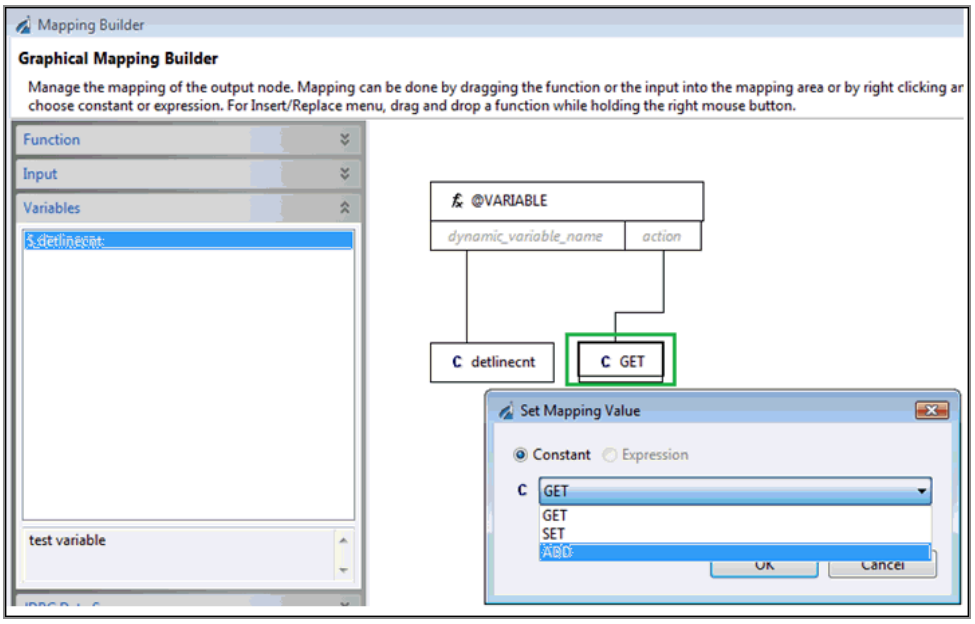


2. Drag the **$detlinecnt** variable from the Variables pane and drop it in Graphical Mapping Builder workspace, as shown in the following image.

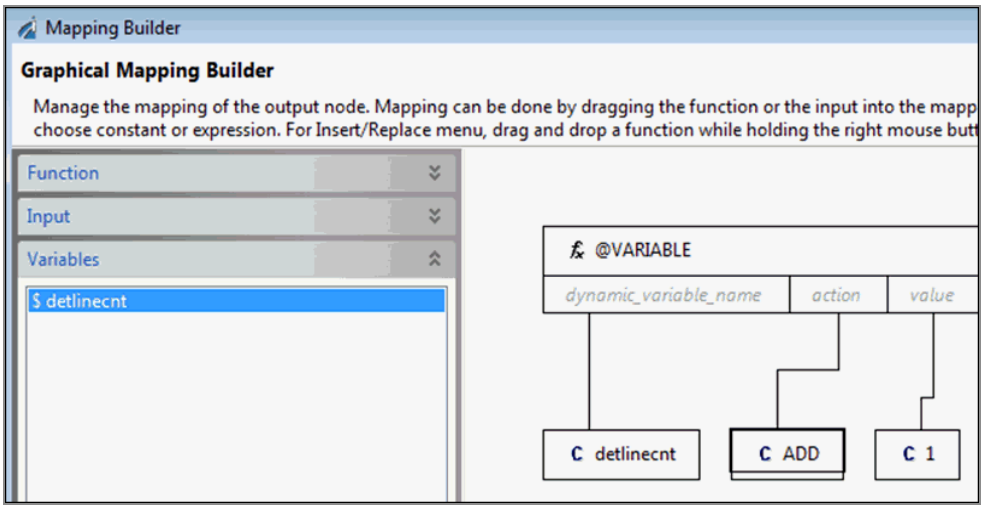




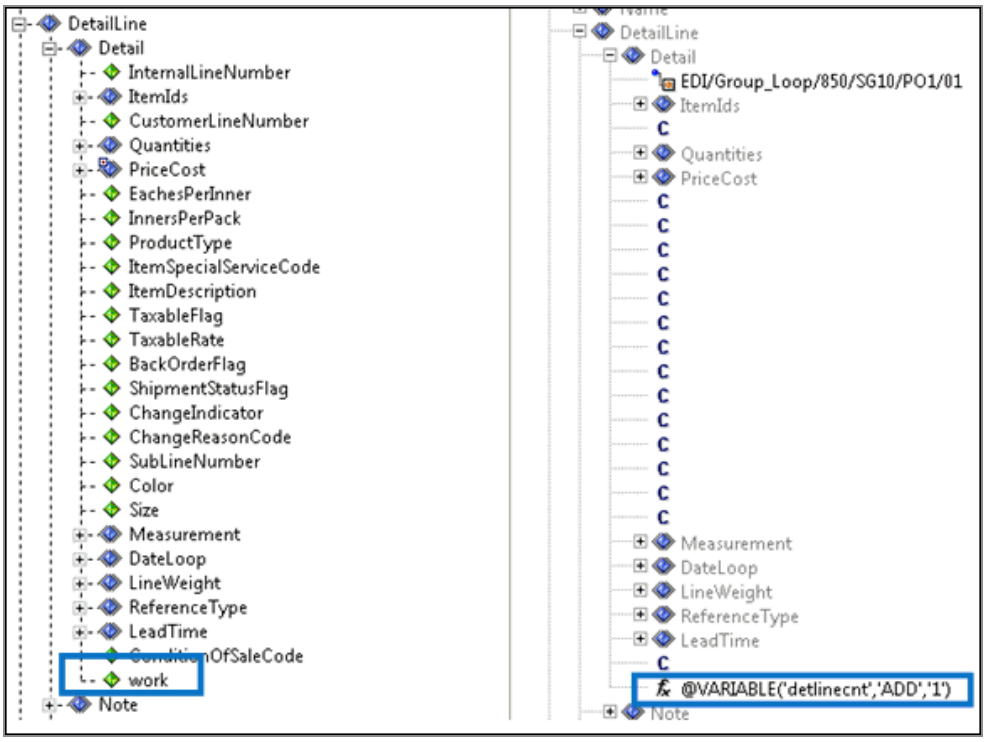
3. Double-click the **GET** action support box, select **ADD** from drop-down list, and click **OK**, as shown in the following image.



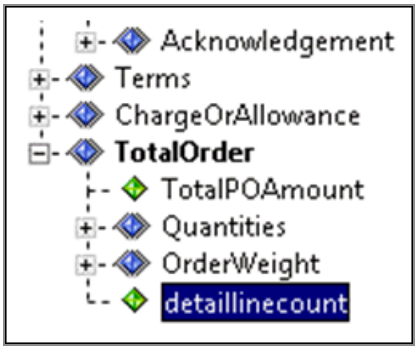
The updated variable appears.



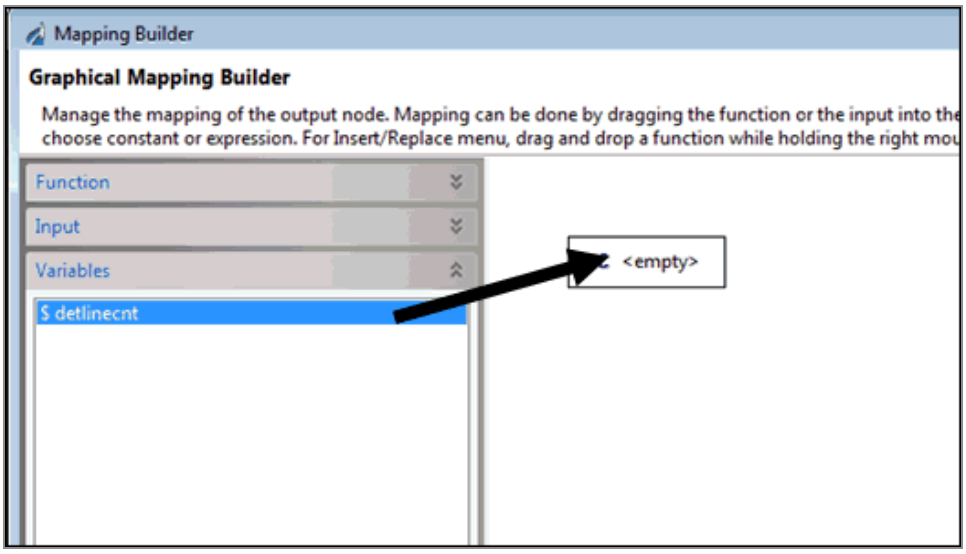
The following screen appears in the transform.

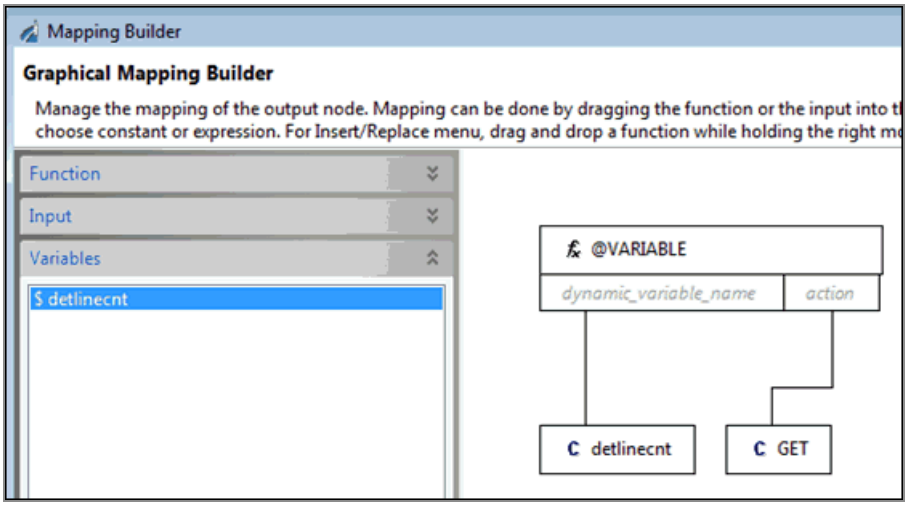


The TotalOrder group already contains the element, **detaillinecount**, to contain the counter, as shown in the following image.



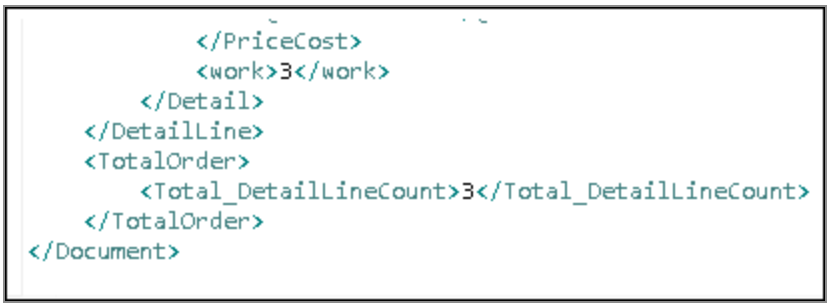
4. Double-click the **detaillinecount** element to open the Graphical Mapping Builder.
5. Drag the **$detlinecnt** variable from the Variables pane and drop it in the Graphical Mapping Builder workspace.





6. Click **OK** and then save your transform.
7. Test run your transform.

   The following example shows **3** as the total number of detail lines appearing in the node.


</PriceCost>
<work>3</work>
</Detail>
</DetailLine>
<TotalOrder>
<Total_DetailLineCount>3</Total_DetailLineCount>
</TotalOrder>
</Document>

ibi Documentation and Support Services

For information about this product, you can read the documentation, contact Support, and join Community.

How to Access ibi Documentation

Documentation for ibi products is available on the Product Documentation website, mainly in HTML and PDF formats.

The Product Documentation website is updated frequently and is more current than any other documentation included with the product.

Product-Specific Documentation

The documentation for this product is available on the ibi™ iWay® Service Manager Documentation page.

How to Contact Support for ibi Products

You can contact the Support team in the following ways:

- To access the Support Knowledge Base and getting personalized content about products you are interested in, visit our product Support website.
- To create a Support case, you must have a valid maintenance or support contract with a Cloud Software Group entity. You also need a username and password to log in to the product Support website. If you do not have a username, you can request one by clicking **Register** on the website.

How to Join ibi Community

ibi Community is the official channel for ibi customers, partners, and employee subject matter experts to share and access their collective experience. ibi Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from ibi products. For a free registration, go to ibi Community.

Legal and Third-Party Notices

SOME CLOUD SOFTWARE GROUP, INC. (“CLOUD SG”) SOFTWARE AND CLOUD SERVICES EMBED, BUNDLE, OR OTHERWISE INCLUDE OTHER SOFTWARE, INCLUDING OTHER CLOUD SG SOFTWARE (COLLECTIVELY, “INCLUDED SOFTWARE”). USE OF INCLUDED SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED CLOUD SG SOFTWARE AND/OR CLOUD SERVICES. THE INCLUDED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER CLOUD SG SOFTWARE AND/OR CLOUD SERVICES OR FOR ANY OTHER PURPOSE.

USE OF CLOUD SG SOFTWARE AND CLOUD SERVICES IS SUBJECT TO THE TERMS AND CONDITIONS OF AN AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER AGREEMENT WHICH IS DISPLAYED WHEN ACCESSING, DOWNLOADING, OR INSTALLING THE SOFTWARE OR CLOUD SERVICES (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH LICENSE AGREEMENT OR CLICKWRAP END USER AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE SAME TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

ibi, the ibi logo, iWay,and TIBCO are either registered trademarks or trademarks of Cloud Software Group, Inc. in the United States and/or other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only. You acknowledge that all rights to these third party marks are the exclusive property of their respective owners. Please refer to Cloud SG’s Third Party Trademark Notices (https://www.cloud.com/legal) for more information.

This document includes fonts that are licensed under the SIL Open Font License, Version 1.1, which is available at: https://scripts.sil.org/OFL

Copyright (c) Paul D. Hunt, with Reserved Font Name Source Sans Pro and Source Code Pro.

Cloud SG software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the “readme” file for the availability of a specific version of Cloud SG software on a specific operating system platform.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. CLOUD SG MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S), THE PROGRAM(S), AND/OR THE SERVICES DESCRIBED IN THIS DOCUMENT AT ANY TIME WITHOUT NOTICE.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "README" FILES.

This and other products of Cloud SG may be covered by registered patents. For details, please refer to the Virtual Patent Marking document located at https://www.cloud.com/legal.