Information Builders

iWay

iWay Integration Solution for EDIHL7 User's Guide

Version 7.0.x and Higher

August 16, 2018

Contents

Preface

This documentation describes how to configure and use the iWay Integration Solution for EDIHL7. It is intended for integration specialists who wish to integrate healthcare enterprise systems by parsing, validating, and storing HL7 messages.

Note: This Release 7.0.x content is currently being updated to support iWay Release 8.0.x software. In the meantime, it can serve as a reference for your use of iWay Release 8. If you have any questions, please contact *Customer_Success@ibi.com*.

How This Manual Is Organized

This manual includes the following chapters:

	Chapter/Appendix	Contents
1	Introducing the iWay Integration Solution for EDIHL7	Describes the mandate of the Health Level Seven (HL7) protocol and how the components of the iWay Integration Solution for EDIHL7 streamline the flow of information between healthcare partners. Provides a roadmap to information on other products used with the iWay Integration Solution for EDIHL7.
2	Deployment Information for Your iWay Integration Solution	Describes the iWay products used with your iWay Integration Solution for EDIHL7 and provides a roadmap to full information on those products. Introduces the concept of a channel for the construction of a message flow in iWay Service Manager.
3	Downloading, Extracting, and Importing HL7 Sample Data	Describes how to download, extract, and import HL7 sample data.
4	Configuring the EDI Activity Driver	Describes how to configure the EDI Activity Driver using iWay Service Manager.
5	Inbound Processing: HL7 to XML	Describes how to configure a basic inbound message flow for the iWay Integration Solution for EDIHL7.
6	Inbound Processing: HL7 to XML (Using MLLP)	Describes how to configure a basic inbound message flow for the iWay Integration Solution for EDIHL7 using the Minimal Lower Layer Protocol (MLLP).

	Chapter/Appendix	Contents
7	Outbound Processing: XML to HL7	Describes how to configure a basic outbound message flow for the iWay Integration Solution for EDIHL7.
8	Outbound Processing: XML to HL7 (Using MLLP)	Describes how to configure a basic outbound message flow for the iWay Integration Solution for EDIHL7 using the Minimal Lower Layer Protocol (MLLP).
A	Supported HL7 Versions	Summarizes the HL7 versions that are currently supported by the iWay Integration Solution for EDIHL7.
B	Using HL7 Separators and Terminators	Includes a list of separators and terminators that are allowed.
C	Using iWay Integration Tools to Configure an Ebix for HL7	Describes how to use iWay Integration Tools (iIT) to configure an Ebix for HL7.

Documentation Conventions

The following table describes the documentation conventions that are used in this manual.

Convention	Description
THIS TYPEFACE or this typeface	Denotes syntax that you must enter exactly as shown.
this typeface	Represents a placeholder (or variable), a cross-reference, or an important term. It may also indicate a button, menu item, or dialog box option that you can click or select.
underscore	Indicates a default setting.
Key + Key	Indicates keys that you must press simultaneously.
{ }	Indicates two or three choices. Type one of them, not the braces.
|	Separates mutually exclusive choices in syntax. Type one of them, not the symbol.

Convention	Description
...	Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis (...).
. . .	Indicates that there are (or could be) intervening or additional commands.

Related Publications

Visit our Technical Documentation Library at *http://documentation.informationbuilders.com*. You can also contact the Publications Order Department at (800) 969-4636.

Customer Support

Do you have any questions about this product?

Join the Focal Point community. Focal Point is our online developer center and more than a message board. It is an interactive network of more than 3,000 developers from almost every profession and industry, collaborating on solutions and sharing tips and techniques. Access Focal Point at *http://forums.informationbuilders.com/eve/forums*.

You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our website, *http://www.informationbuilders.com*. It connects you to the tracking system and known-problem database at the Information Builders support center. Registered users can open, update, and view the status of cases in the tracking system and read descriptions of reported software issues. New users can register immediately for this service. The technical support section of *http://www.informationbuilders.com* also provides usage techniques, diagnostic tips, and answers to frequently asked questions.

Call Information Builders Customer Support Services (CSS) at (800) 736-6130 or (212) 736-6130. Customer Support Consultants are available Monday through Friday between 8:00 a.m. and 8:00 p.m. EST to address all your questions. Information Builders consultants can also give you general guidance regarding product capabilities and documentation. Please be ready to provide your six-digit site code number (*xxxx.xx*) when you call.

To learn about the full range of available support services, ask your Information Builders representative about InfoResponse Online, or call (800) 969-INFO.

Help Us to Serve You Better

To help our consultants answer your questions effectively, be prepared to provide specifications and sample files and to answer questions about errors and problems.

The following tables list the environment information our consultants require.

Platform	
Operating System	
OS Version	
JVM Vendor	
JVM Version	

The following table lists the deployment information our consultants require.

Adapter Deployment	For example, iWay Business Services Provider, iWay Service Manager
Container	For example, WebSphere
Version	
Enterprise Information System (EIS) - if any	
EIS Release Level	
EIS Service Pack	
EIS Platform	

The following table lists iWay-related information needed by our consultants.

iWay Adapter	
iWay Release Level	
iWay Patch	

The following table lists additional questions to help us serve you better.

Request/Question	Error/Problem Details or Information
Did the problem arise through a service or event?	
Provide usage scenarios or summarize the application that produces the problem.	
When did the problem start?	
Can you reproduce this problem consistently?	
Describe the problem.	
Describe the steps to reproduce the problem.	
Specify the error message(s).	
Any change in the application environment: software configuration, EIS/database configuration, application, and so forth?	
Under what circumstance does the problem *not* occur?	

The following is a list of error/problem files that might be applicable.

- ❑ Input documents (XML instance, XML schema, non-XML documents)
- ❑ Transformation files
- ❑ Error screen shots
- ❑ Error output files
- ❑ Trace files

- Service Manager package to reproduce problem
- Custom functions and agents in use
- Diagnostic Zip
- Transaction log

For information on tracing, see the *iWay Service Manager User's Guide*.

User Feedback

In an effort to produce effective documentation, the Technical Content Management staff welcomes your opinions regarding this document. Please use the Reader Comments form at the end of this document to communicate your feedback to us or to suggest changes that will support improvements to our documentation. You can also contact us through our website, *http://documentation.informationbuilders.com/connections.asp*.

Thank you, in advance, for your comments.

Information Builders Consulting and Training

Interested in training? Information Builders Education Department offers a wide variety of training courses for this and other Information Builders products.

For information on course descriptions, locations, and dates, or to register for classes, visit our website (*http://education.informationbuilders.com*) or call (800) 969-INFO to speak to an Education Representative.

Chapter 1

Introducing the iWay Integration Solution for EDIHL7

The iWay Integration Solution for EDIHL7 transforms HL7 messages into standard XML format, or transforms XML representations into HL7 message format.

This section provides an overview of HL7 and describes the features that are provided by the iWay Integration Solution for EDIHL7.

In this chapter:

EDIHL7 Prerequisites

Before you use the iWay Integration Solution for EDIHL7 for inbound (HL7 to XML) and outbound (XML to HL7) processing, 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 is installed.
- iWay Integration Solution for EDIHL7 (Patch) is installed.
- iIT Version 8.0 is installed.
- System and channel Special Registers (SREGs) are updated to match your directory structure.

Understanding the HL7 Protocol

Health Level Seven (HL7) is a standard for information exchange between medical applications. It is the seventh OSI layer protocol for exchanging information in healthcare systems. HL7 defines a protocol for data exchange. It also defines the format and the content of the messages that applications must use when exchanging data with one another in various circumstances. The following descriptions provide an outline for the protocol.

Event Driven Protocol

Real world events, such as the admission of a patient, cause messages to flow between applications. In other words, an application that encounters a real world event sends a message to other applications that need to be aware of this event.

Application to Application Protocol

It defines a communication between two independent applications, rather than between closely coupled, client-server type applications. The scope of interest for HL7 is the message exchange between the applications, rather than the specific role of each application in the health care delivery process.

OSI Level 7 Protocol

The scope of HL7 is the format and content of the data exchanged between the applications, not how it is passed between computers or networks. HL7 does not specify how messages will be delivered between the applications. Usually a TCP/IP connection or FTP file transfer is used to deliver a message. However, within local area networks, the standard is the Lower Layer Protocol.

Exchange Protocol

HL7 specifies the way data exchange between applications will be accomplished. It does not specify how applications store or process this data. In an application, it is recommended that a database structure coincides with HL7 message definitions. However, this is not mandatory.

Standard Protocol

When a proprietary, non-standard link between two systems is made, a message exchange that satisfies your requirements and fits into the data structure of your application can be created. However, the efforts invested in this link are useless when considering a connection to other third-party systems. When you are using HL7, your initial development effort can be reused.

For more information on the HL7 standard, please visit the following website:

http://www.hl7.org

Components of an HL7 Message

HL7 has a message-oriented architecture. This means that the application in which an event occurs will send a message to other applications rather than serving a request.

Note: The application that issues the message is called a Sender or Sending Application, and the addressee (recipient) of the message is called a Receiver or Receiving Application.

The structure of an HL7 Version 2.x message has the following format:

```
Message --> Segments --> Elements
```

Messages

This section provides a use case of a typical HL7 ADT^A04 message. This message is sent when a new patient arrives at the hospital. The demographics of the patient are entered into a hospital information system (HIS) and then the information is communicated to all the other systems to avoid multiple entries of the patient's demographic information.

```
MSH|^~\&|EPIC||SMS||201501011408||ADT^A04|9000123|D|2.7
EVN|A04|201501011408
PID||0493575^^^2^ID1|454721||DOE^JOHN
PV1||O|168~219~C|R
IN1||ABC123|Blue Cross Blue Shield
```

HL7 messages are ASCII messages, and the standard defines them to be human readable. Messages are a defined sequence of segments and/or segment groups. Each segment, group, or message set within a message, can be optional and/or repeating. Each message consists of the segments that are delimited by carriage return characters ("\r" or 0x0D).

Segments

Every line in a message is called a segment and contains information of a specific type. For example, the MSH segment contains information about the Sender and Receiver of the message, the type of the message, a time stamp, and so on. The EVN segment contains information about the type of message, for example, A04 (Register a patient). The PID segment contains demographic information about the patient such as name, ID codes, address, and so on. The PV1 segment contains information regarding the patient's stay in the hospital, such as location assigned, referring doctor, and so on. The IN1 segment contains information regarding the patient’s insurance plan.

The sample message in the previous section contains MSH, EVN, PID, PV1, and IN1 segments. There are more than 183 segments defined in Version 2.7 of the HL7 standard. An HL7 message is a combination of the segments represented in sequence. An HL7 message definition states whether each segment is mandatory or not. Segments consist of elements that may be composites. Elements are delimited by a vertical pipe (|). Each element is defined by the HL7 standard.

Elements

Elements are the building blocks of segments. Elements may be a primitive data type (string, number, and so on), or be made up of other composites. Elements cannot have a recursive reference to themselves. The components of each element are delimited by characters and the sub-components of these components themselves can be delimited using ampersand (&) characters.

Delimiter Characters

An important part of the HL7 protocol is the use of delimiter characters. The following table lists the default delimiter characters used in HL7:

Character	Purpose
0x0D – <cr> (Carriage Return)	Marks the end of each segment
| (Vertical pipe)	Field Delimiter
^ (Caret)	Sub-Field Delimiter
& (Ampersand)	Sub-Sub-Field Delimiter

Two other important characters are the tilde character (~), which is used to separate repeating fields, and the escape character (\).

Escape Characters

Some user data may contain these special delimiter characters. For this reason, HL7 has a system for escaping them. Unlike a language like C, each character in the system has a unique escape sequence. The following table shows the escape sequences for each of the different characters:

Character	Escape Sequence
& (Ampersand)	\T\
^ (Caret)	\S\
| (Vertical pipe)	\F\
~ (Tilde)	\R\
\ (Backward slash)	\E\

Unprintable hex characters are also escaped in the format \0xXX\, unicode characters in the format \UXXXX\, and multi-byte character sequences \MXXXXXX\ for far eastern language support.

Delimiter Redefinition

The delimiter characters can be redefined on the fly in the MSH segment of each message. Each HL7 message starts with a MSH segment. Some HL7 implementations may choose to use different delimiter characters in the message. For example:

```
MSH$^~\&$EPIC$$SMS$$201501011408$$ADT^A04$9000123$D$2.7
```

```
MSH|^~\&|EPIC||SMS||201501011408||ADT^A04|9000123|D|2.7
```

Present but Null

When a field is absent from a message, it should mean that the system does not use that field or that the field has not changed. But what if a system supports the given field, but the value of the field is null? The HL7 protocol requires a method of making this clear. This is done by putting two double quote characters (“”) together in a field like this:

```
ZBR|||||""||||
```

Repetition and Optionality of HL7 Segments

In the message definition, each segment can be either mandatory or optional. Each message starts with an MSH segment that is always mandatory (required). Another example of a mandatory field is PID (Patient identification). Without patient identification data, messages like ADT^A04 (Register Patient) do not have any relevance. Some segments, such as AL1 (Allergies), are optional because patients may or may not have allergies. The message in the example below consists of MSH, EVN, PID, NK1 and PV1. According to the HL7 Version 2.7 definition, in an ADT^A04 message the MSH, EVN, PID and PV1 segments are required and the NK1 segment is optional. DG1, PR1 and AL1 are also optional segments that could be in this message, but are not.

HL7 is order-sensitive. Order is important to both segments and fields inside the segment.

Segments in HL7 messages can also be repeating. For example, NK1 (Next of Kin/Associated Parties) will repeat several times if a person has several next of kin relationships:

```
MSH|^~\&|EPIC||SMS|SMSDT|201501011408||ADT^A04|9000123|D|2.7|
EVN|A04|201501011408
PID||0493575^^^2^ID 1|454721||DOE^JOHN
NK1||CONROY^MARI|SPO||(216)731-4359
NK1||DOE^JOHNNY^^^^|CHD||(216)731-4222
NK1||DOE^ROBERT ^^^^|CHD||(216)731-4222
PV1||O|168 ~219~C~P
```

A group of segments may be optional as a group and may repeat as a group. For example, ORU^R01 message has a group that contains one OBR (Observation request) and 0 to N OBX (Observation result) segments. This group is optional in the message so it may not appear at all. The message will then look like this:

```
MSH PID PD1
```

On the other hand, the message may contain a number of observation groups. Then the OBR-OBX group will repeat and the message will look like this:

```
MSH PID PD1 OBR OBX OBX OBX OBR OBX OBX
```

The first three observation results may belong to the first observation request, and the next two observation results may belong to the second observation request.

```
MSH PID PD1 - header and patient demographics
OBR OBX OBX OBX  - first observation group (e.g. height and weight)
OBR OBX OBX  - second observation group (e.g. lab results)
```

HL7 Version 2.x Backward Compatibility

HL7 2.x is designed to be backward compatible.

A good example to clarify this comes from considering a Patient ID field in a PID segment. Early versions of HL7 declare a Patient ID to be just a simple one field identifier, such as:

```
|PatientID|
```

In more recent versions, the Patient ID has become embellished with composite data types:

```
|PatientID^Check Digit^Assigning Authority NameSpace&...|
```

Most of the time, extra fields are optional in HL7 2.x. When additional fields are not present, the trailing delimiters are not displayed.

```
|234324^^^|
```

is equivalent to:

```
|234324|
```

Withdrawn Segments

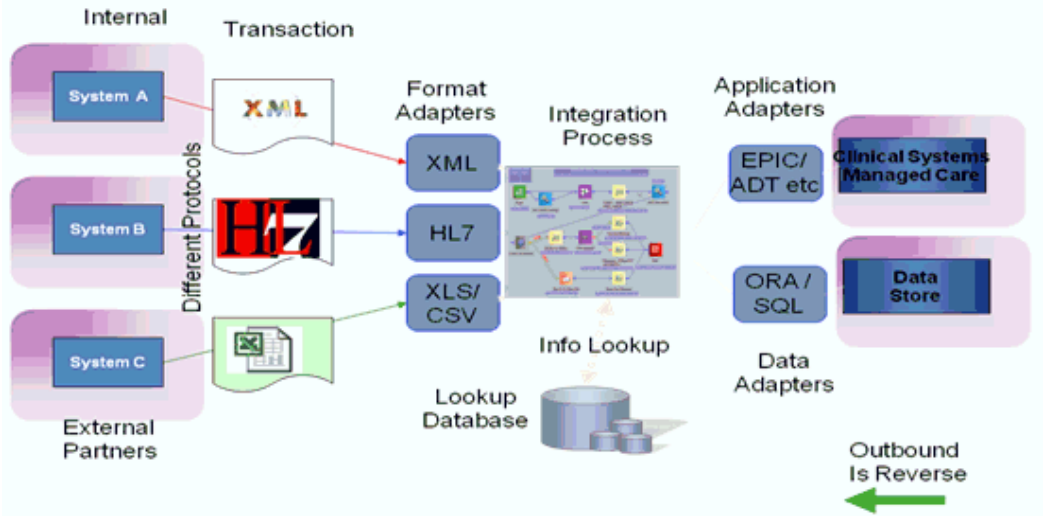
In newer versions of HL7, segments can become obsolete or withdrawn. This is noted by the W data type.

These elements remain in the metadata as placeholders. Their data types have been changed to allow any legacy data to pass through transformation. The user may modify the Ebix to reintroduce validation for these elements if desired.

Sample EDIHL7 Integration Scenario

A hospital needs a number of software applications to efficiently and accurately manage daily functions. A dedicated application system is available to cater to the needs of individual functional entity, such as Patient Registration, Billing, Nursing and Bed Management, Order Management, Pharmacy, and Laboratory. These systems are provided by different vendors and are typically built using different technologies. HL7 plays the vital role of providing mechanism for interconnecting these systems. Each individual application entity has the capability of exchanging information using HL7 messages. They often maintain the same information in different forms and use different codes to represent common values. This requires an integration engine that has HL7 capabilities.

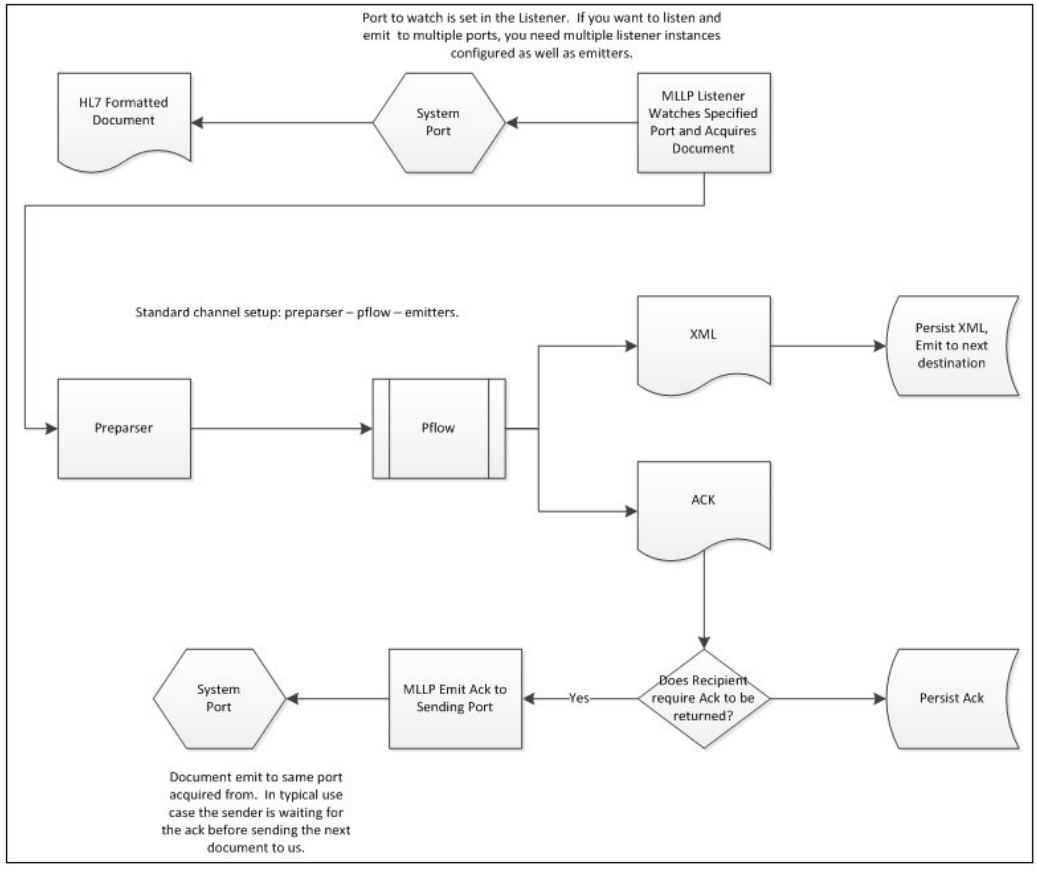
This diagram illustrates a typical integration scenario. Data is exchanged over different interfaces to ensure a connected and functional application system environment.



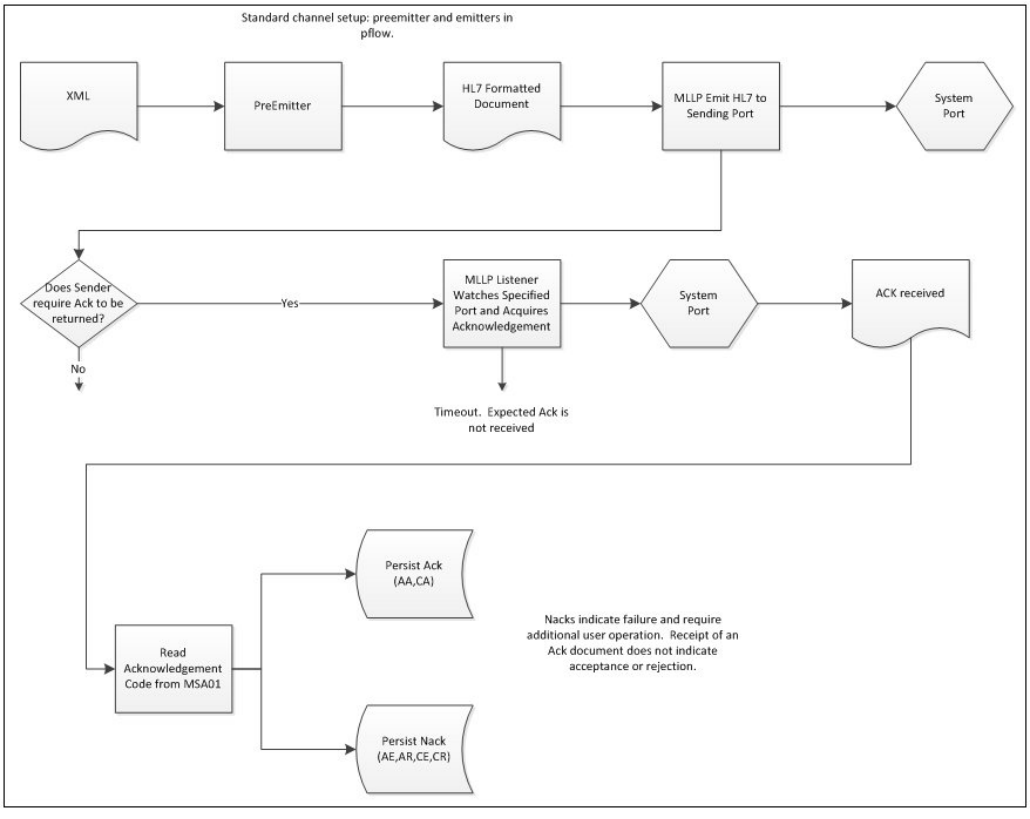
HL7 Inbound and Outbound Flows Using MLLP

This section provides diagrams that illustrate HL7 inbound and outbound flows using MLLP.

HL7 Inbound Flow Using MLLP



HL7 Outbound Flow Using MLLP



Features of the iWay Integration Solution for EDIHL7

The iWay Integration Solution for EDIHL7 reduces the amount of effort required to integrate HL7 messages with your internal enterprise applications. It includes conversion of messages from HL7 to XML format, making it easy to include HL7 messages in your integration projects. Features of the iWay Integration Solution for EDIHL7 include:

- Integration with iWay Service Manager to provide bi-directional (synchronous and asynchronous) conversion of HL7 formats and XML.
- Interactions with application servers, integration brokers, third-party software packages, and messaging services are also supported.
- 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.

- Support for HL7 messages. For more information, see *Supported HL7 Versions* on page 231.

Components of the iWay Integration Solution for EDIHL7

iWay business components used in the construction of a message flow for HL7 messages include:

- Ebix (e-Business Information Exchange) File
- MLLP Listener
- EDIHL7 Batch Splitter Preparser
- EDIHL7 Preparser
- EDIHL7 Validation Report Service
- EDIHL7 Acknowledgement Service
- XML to EDIHL7 Transform Service
- MLLP Emit Service
- Deidentification service

Ebix (e-Business Information Exchange) File

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 EDIHL7 contains several Ebix files, one for each supported HL7 version. An Ebix file for HL7 is named using the following format:

`HL7_version.ebx`

where:

version

Is the HL7 version number. For example, the Ebix file for HL7 Version 2.7 is named HL7_2.7.ebx.

An Ebix is a collection of metadata that defines the structure of data. The Ebix supplied with the iWay Integration Solution for EDIHL7 defines the structure of supported HL7 messages.

Each Ebix includes:

- Pre-built data dictionaries. The structure of each HL7 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 based on the definition of the HL7 standard.
- Pre-built XML schemas that define the structure and content of XML messages in detail.
- Pre-built HL7 to XML transformation templates, and XML to HL7 templates, for the supported HL7 versions.
- Pre-built rule files for each message. The iWay Integration Solution for EDIHL7 uses these rule files to validate inbound and outbound documents.

MLLP Listener

The MLLP listener uses the Minimal Lower Layer Protocol (MLLP) to receive messages within a channel from configured endpoints.

The following table lists and describes the parameters for the MLLP listener.

Parameter	Description
IP Properties	
Port*	The TCP port where MLLP messages are received.
Local Bind Address	Local bind address for multi-homed hosts. This parameter is usually left blank.
Persistent Connection	If set, the connection is maintained until client closes or the Persistence Timeout expires.

Parameter	Description
Maximum Connections	Maximum number of simultaneous connections allowed. When this threshold is reached, new connections will not be accepted until current connections have ended and the total number of connections is below the limit. Leave blank or set to zero for no maximum.
Persistence Timeout value in Minutes	Maximum length of time (in minutes) that a connection can persist with no activity. 0 or blank will default to 60.
Set Response NoDelay	If true, disables Nagle's Algorithm on the response. This will result in faster line turnaround at the expense of an increased number of packets.
Reuse Address	If true, when the connection is closed, immediately make the address available, bypassing the TCP defaults.
Allowable Clients	If supplied, only messages from this list of fully qualified host names and/or IP addresses are accepted. Enter as comma-separated list or use FILE().
Secure Connection	Use a secure connection over SSL.
SSL Context Provider	iWay Security Provider for SSL Context. If this component is secure and SSL Context Provider is left blank, then the default provider will be used.
MLLP	
Start Block Character	The encoding of the Start of Block character in decimal, octal, or hexadecimal format. The default is 0x0B.
End Block Character	The encoding of the End of Block character in decimal, octal, or hexadecimal format. The default is 0x1C.
Maximum Input Size	Maximum number of bytes read before the end of the message is found. This prevents denial of service attacks with very large messages or a large number of bytes between messages.
Tuning	

Parameter	Description
Multithreading	Indicates the number of worker threads (documents or requests) that iWay Service Manager can handle in parallel. Setting this to a value of greater than 1 enables the listener to handle a second request while an earlier request is still being processed. The total throughput of a system can be affected by the number of threads operating. Increasing the number of parallel operations may not necessarily improve throughput. The default is 1. The max value is 99.
Maximum Threads	The parallel threads can grow to this count automatically on demand. Over time, the worker count will decrease back to the multithreading level. Use this parameter to respond to bursts of activity.
Optimize Favoring	Use this option to customize how the listener performs. For smaller transactions, select *performance*. For large input documents that could monopolize the amount of memory used by iWay Service Manager, select *memory*.
Events	
Failed ReplyTo Flow	Name of a published process flow to run if a message cannot be emitted on an address in its reply address list.
Dead Letter Flow	Name of a published process flow to run if an error cannot be emitted on an address in its error address list.
Channel Failure Flow	Name of a published process flow to run if this channel cannot start or fails during message handling. iWay Service Manager will attempt to call this process flow during channel shut down due to the error.
Parse Failure Flow	The name of a published process flow to run if XML parsing fails for the incoming message.

Parameter	Description
Channel Startup Flow	The name of a published process flow to run prior to starting the channel.
Channel Shutdown Flow	The name of published process flow to run when the channel is shut down.
Other	
Whitespace Normalization	Specifies how the parser treats whitespace in Element content. Select *preserve* to turn off all normalization as prescribed by the XML Specification. Select *condense* to remove extra whitespace in pretty printed documents and for compatibility with earlier versions.
Accepts non-XML (flat) only	If set to *true*, the input data is sent directly to the business logic step. The data is not preparsed, parsed, or validated. This flag is used primarily to send non-XML to the business logic or replyTo without processing it.
Execution Time Limit	The maximum time that a request may take to complete. Used to prevent runaway requests. Any request that takes longer to complete than this value will be attempted to be terminated.
Default Java File Encoding	The default encoding if the incoming message is not self-declaring (that is, XML).

Parameter	Description
Agent Precedence	Sets the order by which iWay Service Manager selects agents. iWay Service Manager selects the agent or agents to process the document by searching through the configuration dictionary. Usually, it looks for a document entry in the configuration and when a match is found, the agent specified in that document entry is selected. If a matching document entry is not found, or no agent is specified, the engine looks in the input protocol configuration (listener). To have the processing agent taken directly from the listener (thus ignoring the document entry), use <listener> overrides <document>. Possible values are <document> overrides <listener> and <listener> overrides <document>. The default value is <document> overrides <listener>.
Error Documents treated normally	If set to *true*, error documents are processed by any configured preemitters.
Listener is Transaction Manager	If set to *true*, agents run within a local transaction.
Record in Activity Log(s)	If set to *true*, activity on this channel will be recorded in the activity logs, otherwise the activity will not be recorded.
AES Key	If the channel will receive encrypted AFTI messages, set the AES key (maximum 16 characters) to be used for decrypting.
Startup Dependencies	A comma-separated list of channel names that must be started before this one is called.

EDIHL7 Batch Splitter Preparser

The EDIHL7 Batch Splitter preparser (com.ibi.preparsers.EDIHL7BatchSplitter) splits the HL7 batch into individual HL7 messages. It splits on the Message Header (MSH) prior to parsing the data into XML. The EDIHL7 Batch Splitter preparser should only be used in conjunction with the EDIHL7 preparser. In addition, the EDIHL7 Batch Splitter preparser should be placed in order first if multiple preparsers are used.

The EDIHL7 Batch Splitter preparser will extract all messages from a batch. It does not write the following format to your output XML document: FHS - BHS - MSH - BTS - FTS

The following table lists and describes the parameters for the EDIHL7 Batch Splitter preparser.

Parameter	Description
timestamp	Select *true* from the drop-down list to write a timestamp to the log file. By default, this parameter is set to *false*.

EDIHL7 Preparser

The EDIHL7 preparser (com.ibi.preparsers.XDEDIHL7PreParser) is available for the iWay Integration Solution for EDIHL7. The preparser for the iWay Integration Solution for EDIHL7 converts an incoming HL7 formatted document to iWay XML format.

The following table lists and describes the parameters for the EDIHL7 preparser.

Parameter	Description

Template	The pattern used to lookup a document inside the Ebix. The following format is used:
	`HL7_%_^toXML.xch`
	where:
	`%`
	Represents the message type.
	`^`
	Represents the version.

Timestamp	Select *true* from the drop-down list to write a timestamp to the log file. By default, this parameter is set to *false*.

Parameter	Description
Z-segment	Controls how Z-segments should be formatted in the output XML document. If this parameter is set to *false*, then the fields and components in the segment are not parsed. By default, this parameter is set to *false*. For more information, see *Support for Z-Segments* on page 30.
Segment	Select *true* from the drop-down list to add the *name* attribute to the segment. By default, this parameter is set to *false*.
Composite	Select *true* from the drop-down list to add the *name* attribute to the composite. By default, this parameter is set to *false*.
Element	Select *true* from the drop-down list to add the *name* attribute to the element. By default, this parameter is set to *false*.

Support for Z-Segments

HL7 is a proprietary message structure that allows you to define your own structure. Since HL7 messages can be customized, the HL7 committee provides a framework for the addition of custom information into HL7 messages.

You can define a custom segment in any HL7 message using iWay Integration Tools (iIT). These segments must be named starting with the letter Z.

The EDIHL7 preparser (com.ibi.preparsers.XDEDIHL7PreParser) provides support for Z-segments. The *Make floating structure* parameter controls how Z-segments should be formatted in the output XML document. By default, this parameter is set to *false*.

For example, consider the following segment:

```
ZMS|20150401|FOL|100
```

If the *Make floating structure* parameter is set to *false*, then the fields and components in the segment are not parsed. The resulting output in the XML document is:

```
<Z>ZMS|20150401|FOL|100</Z>
```

If the *Make floating structure* parameter is set to *true*, then the fields and components are parsed. The resulting output in the XML document is:

```
<Z segment-name="ZMS">
     <_01_0 name="">20150401</_01_0>
     <_02_0 name="">FOL</_02_0>
     <_03_0 name="">100</_03_0>
</Z>
```

EDIHL7 Validation Report Service (com.ibi.agents.XDEDIHL7ValidationReportAgent)

Syntax:

```
com.ibi.agents.XDEDIHL7ValidationReportAgent
```

Description:

The EDIHL7 Validation Report service is used to generate a validation report after validating an inbound HL7 message or an outbound XML message.

Parameters:

Parameter	Description
Input Message	Select *true* to add the input message to the generated validation report.
Output Message	Select *true* to add the output message to the generated validation report.

EDIHL7 Acknowledgement Service (com.ibi.agents.XDEDIHL7AckAgent)

Syntax:

```
com.ibi.agents.XDEDIHL7AckAgent
```

Description:

The EDIHL7 Acknowledgement service is used to generate acknowledgement messages using predefined rules. The service chooses ACK / NACK status based on the validity and verification status of the received HL7 message. The generated acknowledgement message can then be sent using the default path of the listener.

Parameters:

Parameter	Description
Sending Application *	The name of the sending application.
Sending Facility *	The name of the sending facility.
Control ID *	The Control ID for the message.
Mode *	Select one of the following acknowledgment modes from the drop-down list: ❏ Original ❏ Enhanced The default value is *Original*.
Acknowledgment *	Specify when an acknowledgement message should be generated by selecting the corresponding value from the drop-down list: ❏ Always ❏ On Error ❏ On Success The default value is *Always*.

XML to EDIHL7 Transform Service (com.ibi.agents.XMLtoEDIHL7TransformAgent)

Syntax:

```
com.ibi.agents.XMLtoEDIHL7TransformAgent
```

Description:

This service is used in outbound processing to convert the XML-formatted HL7 document to an HL7 formatted document.

Parameters:

Parameter	Description
timestamp	Select *true* from the drop-down list to write a timestamp to the log file. By default, this parameter is set to *false*.

MLLP Emit Service (com.ibi.agents.XDMLLPEmitAgent)

Syntax:

```
com.ibi.agents.XDMLLPEmitAgent
```

Description:

This service emits a message using the Minimal Lower Layer Protocol (MLLP). This protocol allows you to wrap an HL7 message with a header and footer to ensure you know where a message starts, where a message stops, and where the next message starts.

Parameters:

Parameter	Description
Host *	The machine name or IP address of the MLLP destination.
Port *	The designated TCP/IP port that is being used to receive MLLP messages.
Secure Connection	Select *true* from the drop-down list if you want use a secure connection through Secure Sockets Layer (SSL). By default, this parameter is set to *false*.
SSL Context Provider	If configured, specify the name of an available iWay Security Provider for SSL Context. If the Secure Connection parameter is set to *true* and the SSL Context Provider field is blank, then the default provider will be used.
Set TCP No Delay	If set to *true*, then Nagle's Algorithm on the client socket will be disabled. This will result in faster line turnaround at the expense of an increased number of packets. By default, this parameter is set to *false*.

Parameter	Description
Socket Timeout	The timeout value in seconds. When a non-zero timeout value is specified, a read() function call on the socket will block for only the amount of time specified (in seconds). If the timeout expires, a *java.net.SocketTimeoutException* is generated. The default timeout is dependent on the operating system being used.
Persistence	If set to *true*, then iWay Service Manager (iSM) is instructed to maintain the connection.
Persistence Timeout value in Minutes	The maximum length of time (in minutes) that a connection can persist with no activity. If this parameter value is zero (0) or left blank, then the default is set to 60 minutes.
Retry Count	The number of times to try to send the message after an initial failure is encountered. By default, this parameter is set to *0*.
Retry Pause	The amount of time (in milliseconds) to wait between retry attempts. By default, this parameter is set to *1000*.
Start Block Character	The encoding of the Start of Block character in decimal, octal, or hexadecimal format. The default is *0x0B*.
End Block Character	The encoding of the End of Block character in decimal, octal, or hexadecimal format. The default is *0x1C*.
Maximum Message Size	The maximum size of a message that can be sent or received through this emitter. If this parameter value is zero (0) or left blank, then the default is set to 256KB.
Output Document	Determines whether the output document is a *response* document, *status* document, or the original *input* document. By default, this parameter is set to *response*.

Deidentification Service (com.ibi.agents.XDDeidentifyAgent)

Syntax:

```
com.ibi.agents.XDDeidentifyAgent
```

Description:

The Deidentification service provides algorithms that can be used to implement the deidentification of protected health information in accordance with the Health Insurance Portability and Accountability Act (HIPAA) Privacy Rule. Multiple algorithms can be configured since a combination of algorithms will be needed to deidentify the data correctly.

The Deidentification service takes an XML document as input. The first configured algorithm takes this document as input and modifies it in place. The result is fed into the next configured algorithm and so on. The result of the last configured algorithm is the XML document returned by the service.

For more information on configuring and using the Deidentification service, see the *iWay Service Manager Component Reference Guide*.

Chapter 2

Deployment Information for Your iWay Integration Solution

This topic describes the iWay products used with your iWay Integration Solution for EDIHL7 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.

In this chapter:

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 *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 *iWay Service Manager User's Guide* and the *iWay Service Manager Programmer's Guide*.

Using a Channel to Construct a EDIHL7 Message Flow

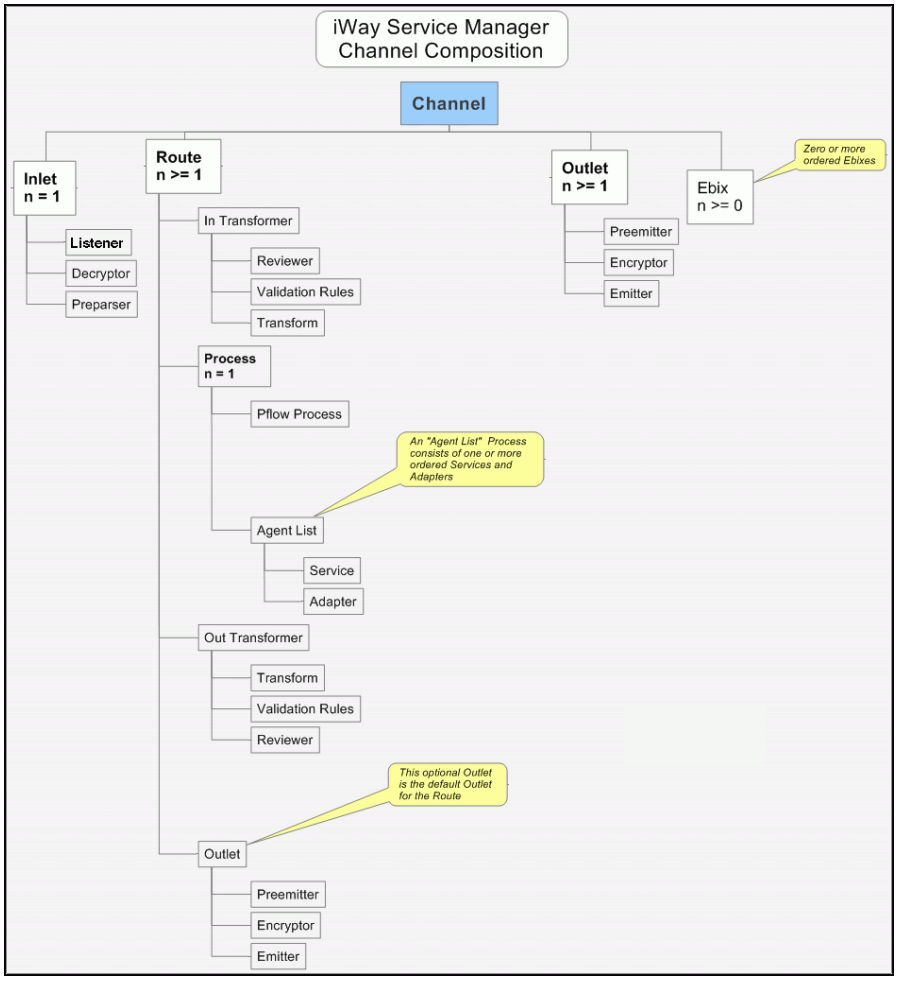
The use of iWay Service Manager is centered on a channel. A channel is a container for all the iWay business components used in an EDI message flow.

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 *iWay Service Manager User's Guide*.

Chapter 3

Downloading, Extracting, and Importing HL7 Sample Data

This chapter describes how to download, extract, and import HL7 sample data.

In this chapter:

Downloading and Extracting HL7 Sample Data

This section describes how to download and extract HL7 sample data.

Procedure: How to Download and Extract User Samples for HL7

1. Download the HL7_usr_samples.zip file containing HL7 user sample workspace from the following website:

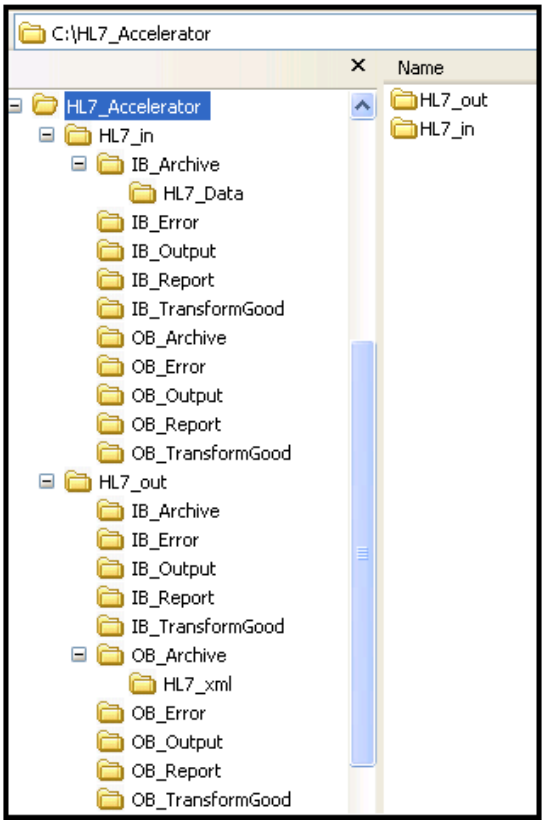
   http://techsupport.informationbuilders.com

   The downloaded HL7_usr_samples.zip contains the following files:

   - HL7_Accelerator.zip
   - HL7_usr_samples_iIT_workspace.zip

2. Save the HL7_usr_samples_iIT_workspace.zip file to a folder on your local drive.

3. Save and extract the HL7_Accelerator.zip file to a location where you want to store your data, as shown in the following image.



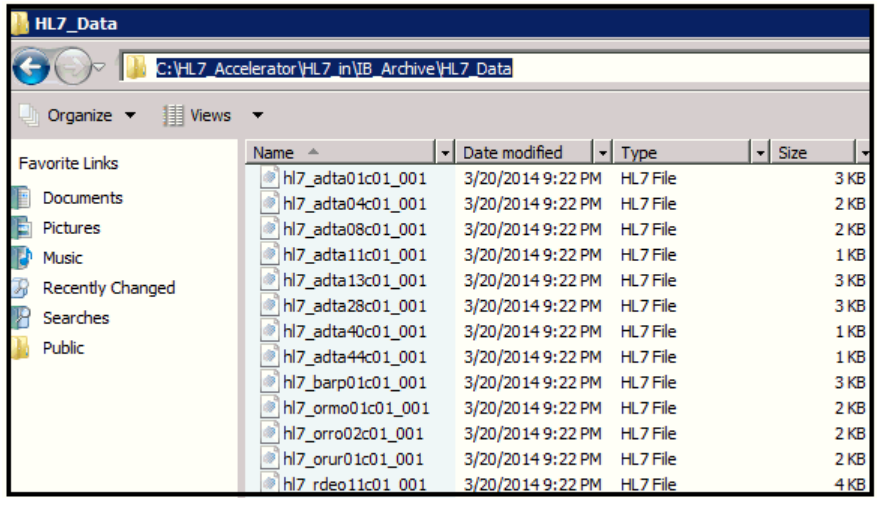
4. The HL7_Accelerator.zip file contains sample input and output data that you can use.

   ❏ Inbound test data is located in the following folder:

   `\HL7_Accelerator\HL7_in\IB_Archive`

   There is a folder called HL7_Data.

For example:

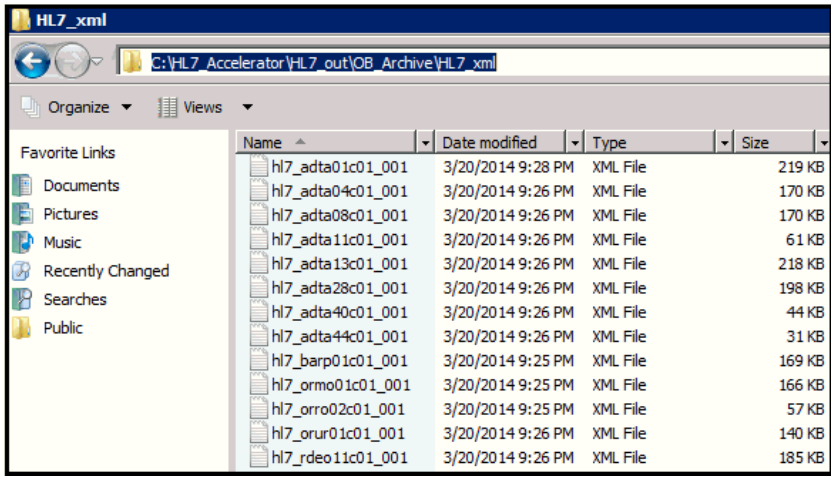


- Outbound test data is located in the following folder:

  `\HL7_Accelerator\HL7_out\OB_Archive`

  There is a folder called HL7_Xml.

  For example:

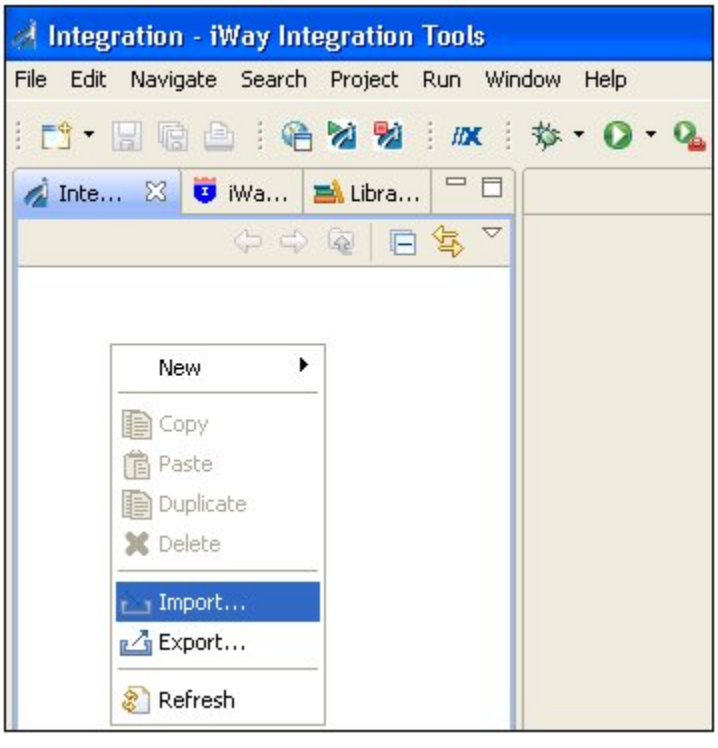


Importing HL7 Sample Data to iWay Integration Tools as a Workspace

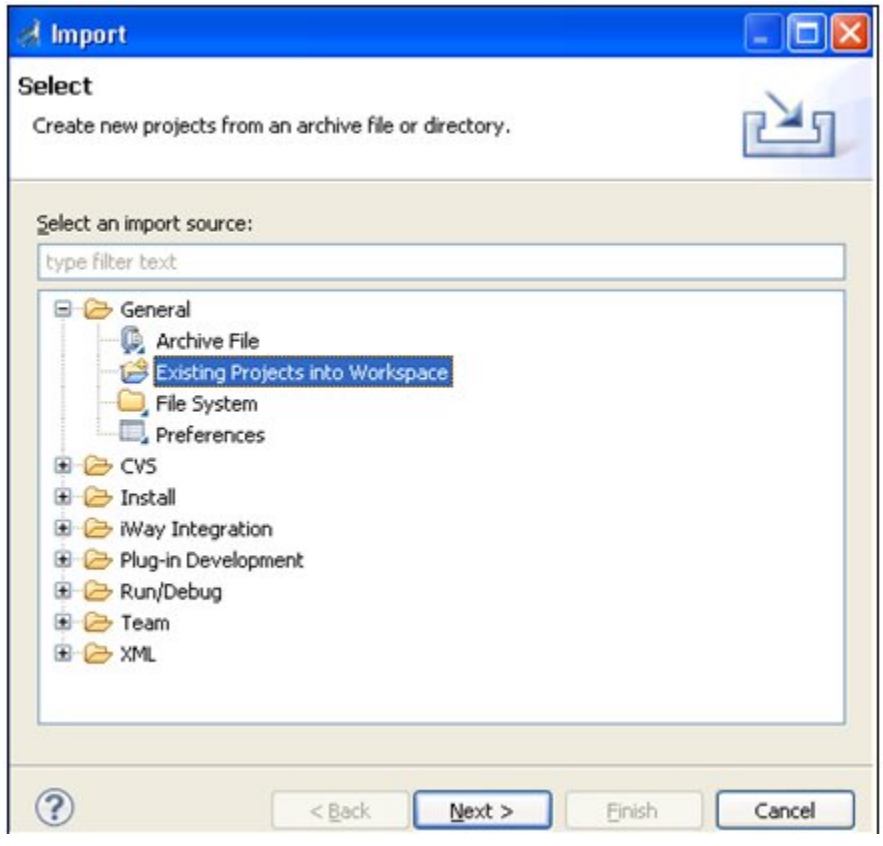
This section describes how to import HL7 sample data to iWay Integration Tools (iIT) as a workspace.

Procedure: How to Import HL7 User Samples to iWay Integration Tools as a Workspace

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.

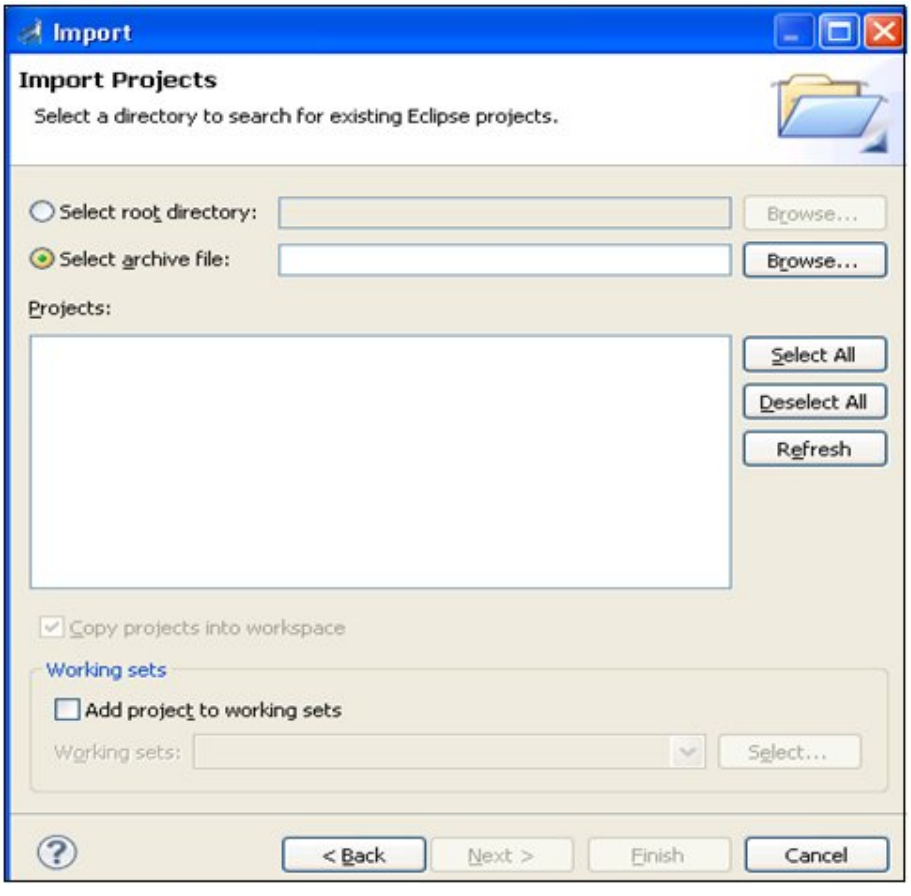


The Import dialog opens, as shown in the following image.



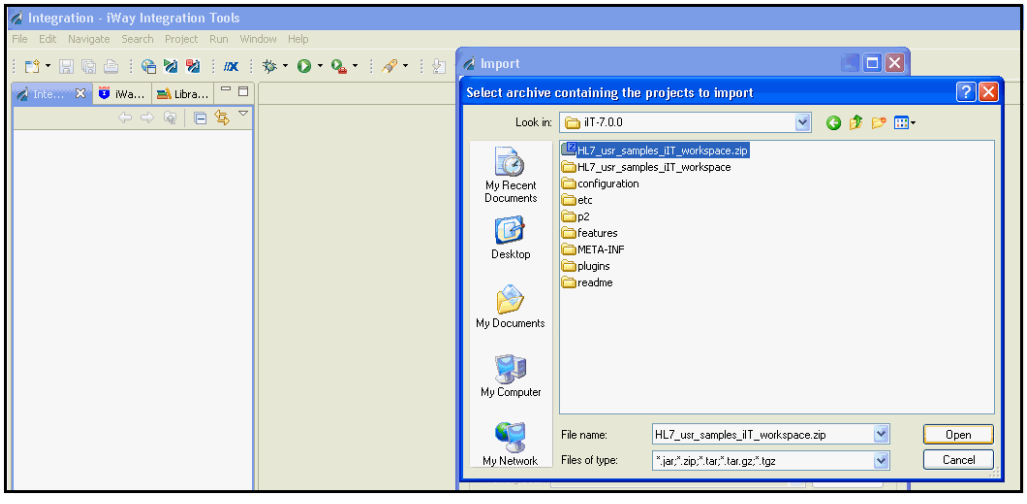
3. Expand the *General* folder, select *Existing Projects into Workspace*, and then click *Next*.

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



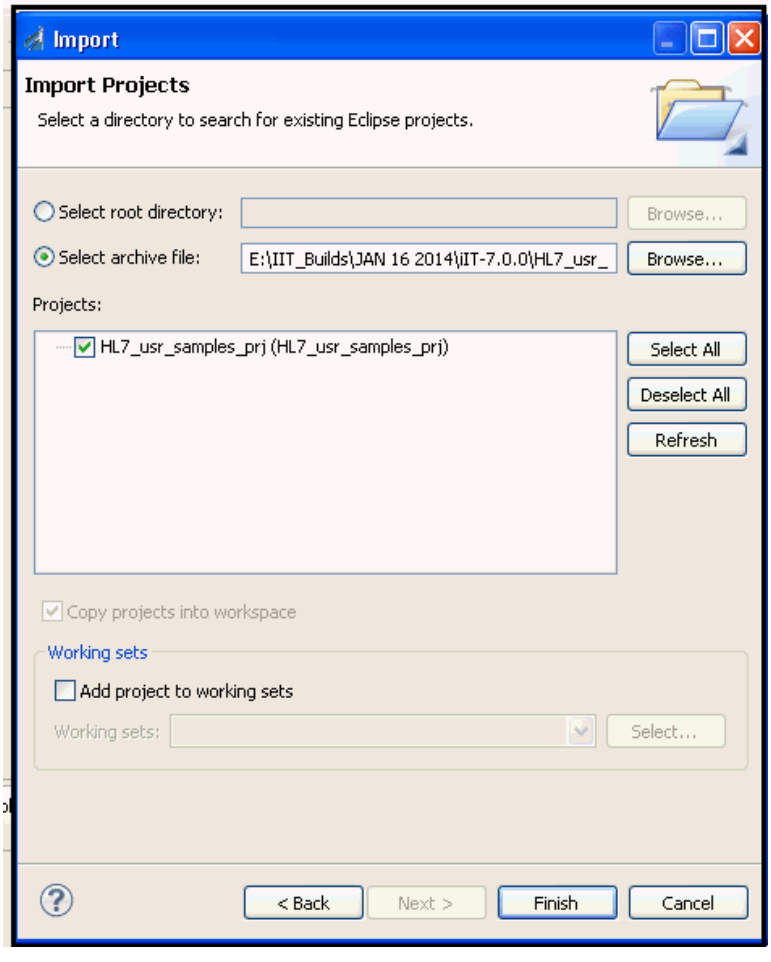
4. Click *Select archive file* and then click *Browse*.

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



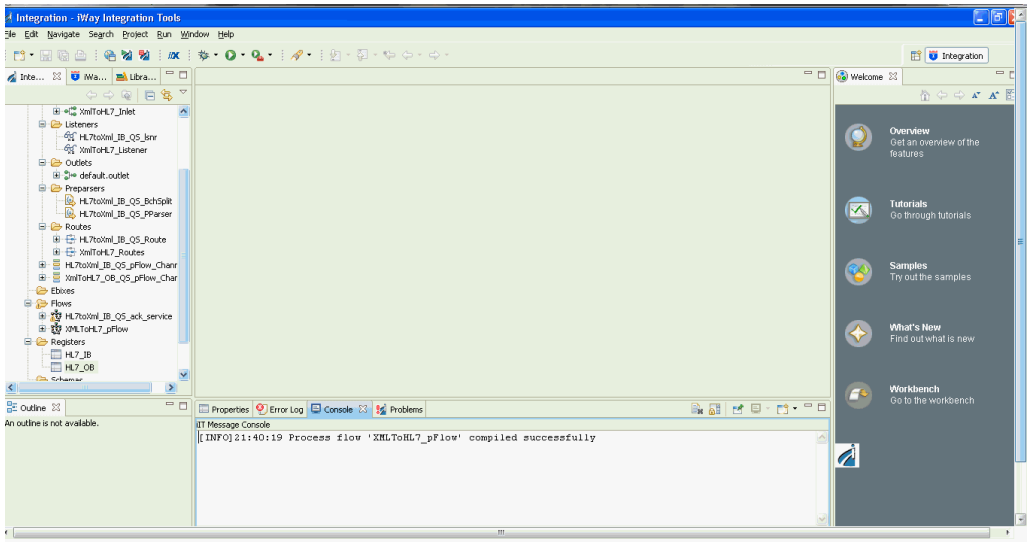
5. Select the *HL7_usr_samples_iIT_workspace.zip* file and click *Open*.

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



6. Click *Finish*.

The HL7 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.

Chapter 4

Configuring the EDI Activity Driver

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

In this chapter:

EDIHL7 EDI Activity Driver 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 *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

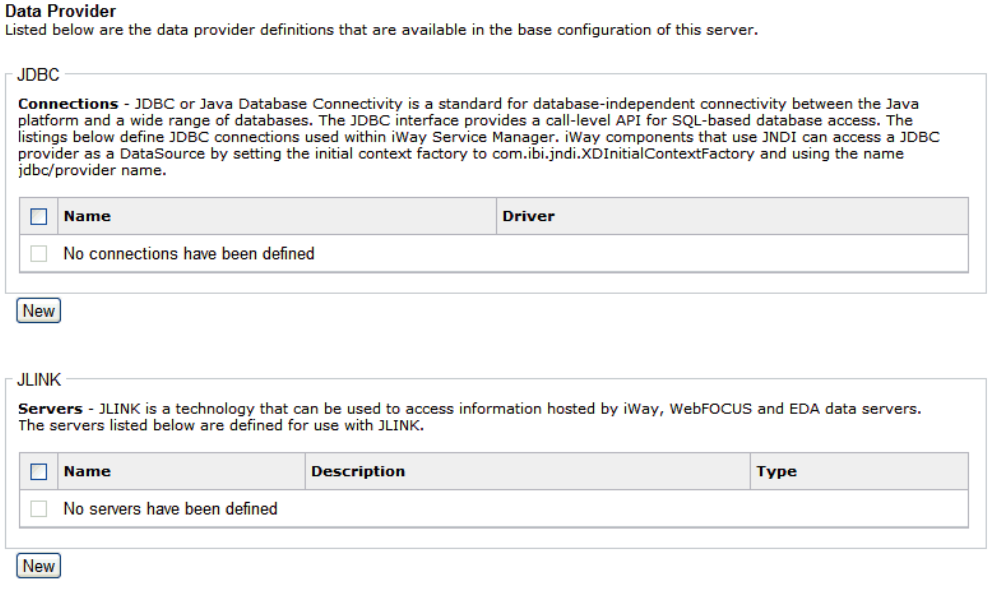
This section describes how to configure the EDI data provider.

Procedure: How to Configure the EDI Data Provider

To configure the EDI data provider:

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

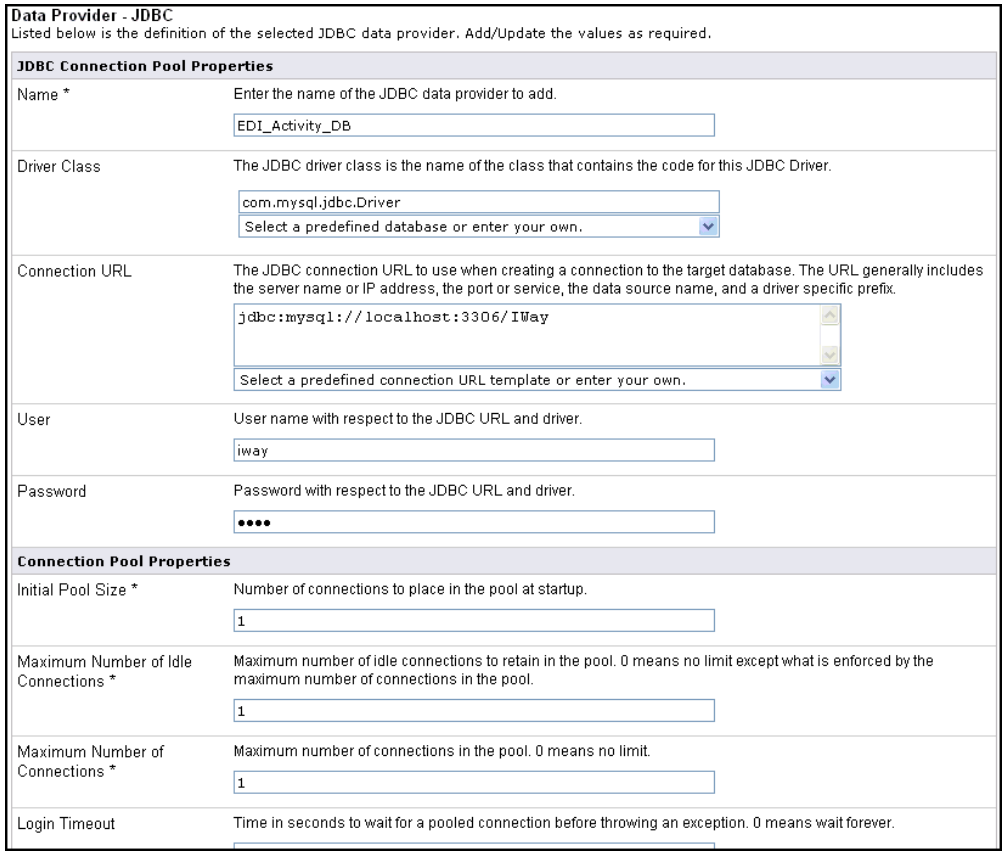
   The Data Provider pane opens.



   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.

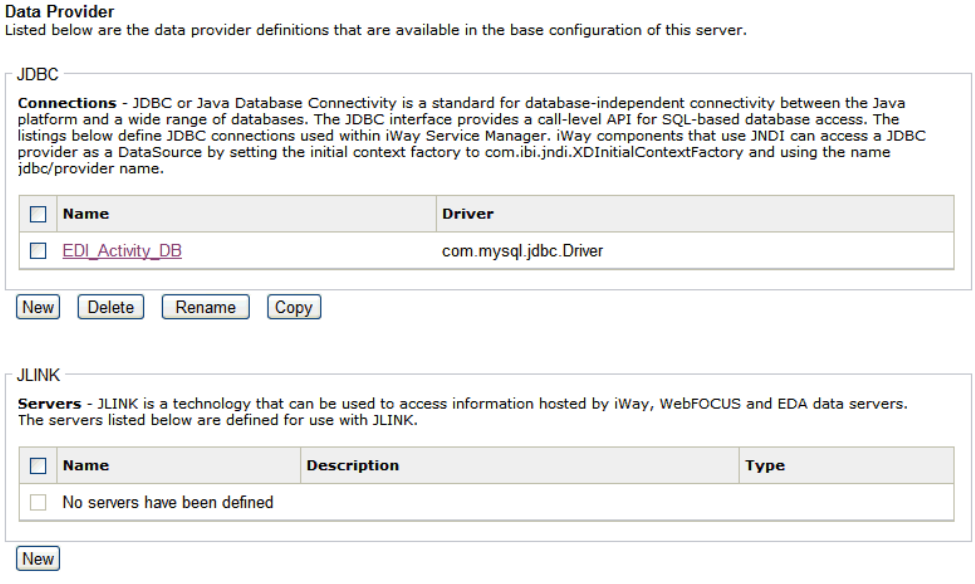
   A value of zero means that there is no limit, except what is enforced by the maximum number of connections in the connection pool.

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.

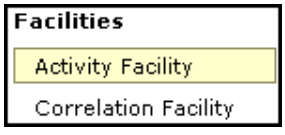


Configuring the EDI Activity Driver Using iWay Service Manager

This section describes how to configure the EDI Activity Driver.

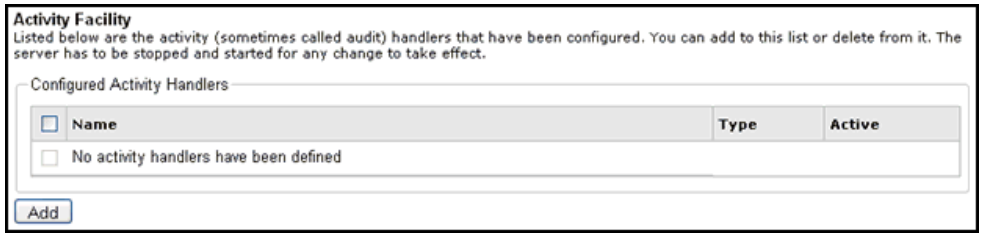
Procedure: How to Configure the EDI Activity Driver

To configure the EDI Activity Driver:



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

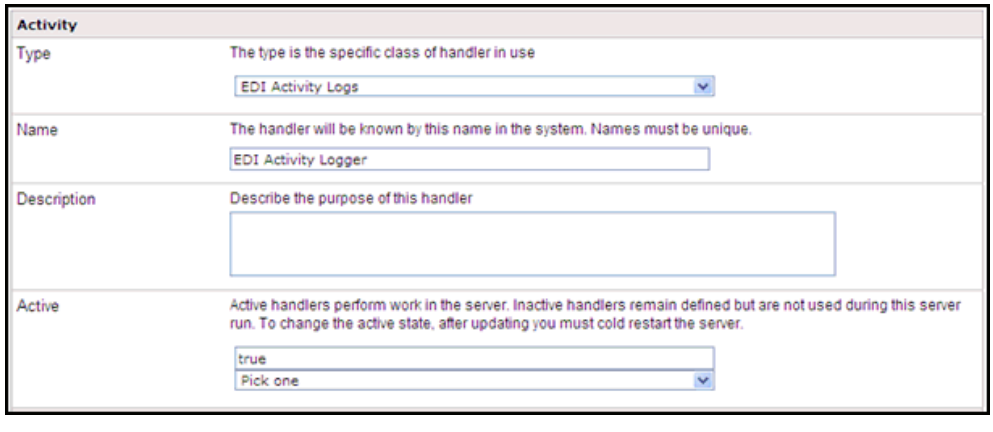
   The Activity Facility pane opens.



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

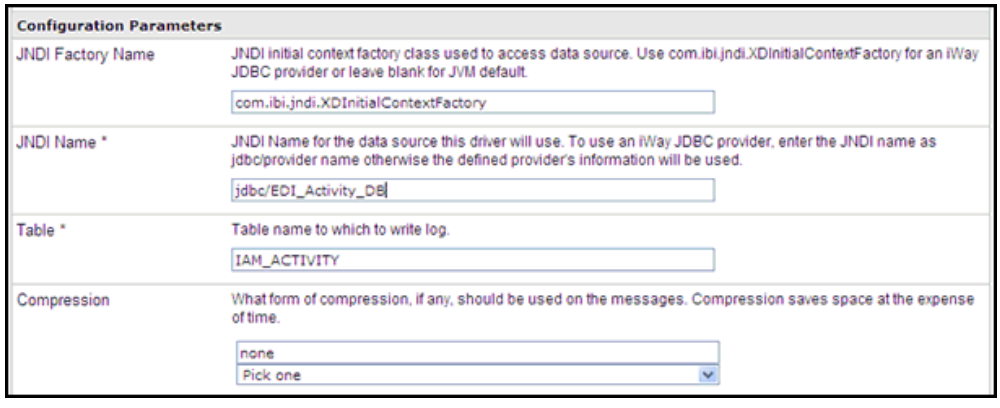
2. Click *Add* to configure a new Activity Facility handler.

   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 the console.

Parameter Name	Type	Description
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.

Database Field	Description
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	The Message Control ID assigned to this message.
tid	The 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.
text	Message text for business errors (rules system violations).

Database Field	Description
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	The name of the receiving partner.
partner_from	The name of the sending partner.
encoding	Encoding of the listener that obtained the document.
mac	Not used in this version.
Driver version	1.0 in 8.0 SM

Chapter 5

Inbound Processing: HL7 to XML

This section describes how to configure a basic inbound message flow for the iWay Integration Solution for EDIHL7. The message flow represents the movement and tasks in the conversion of a message from HL7 format to XML format.

In this chapter:

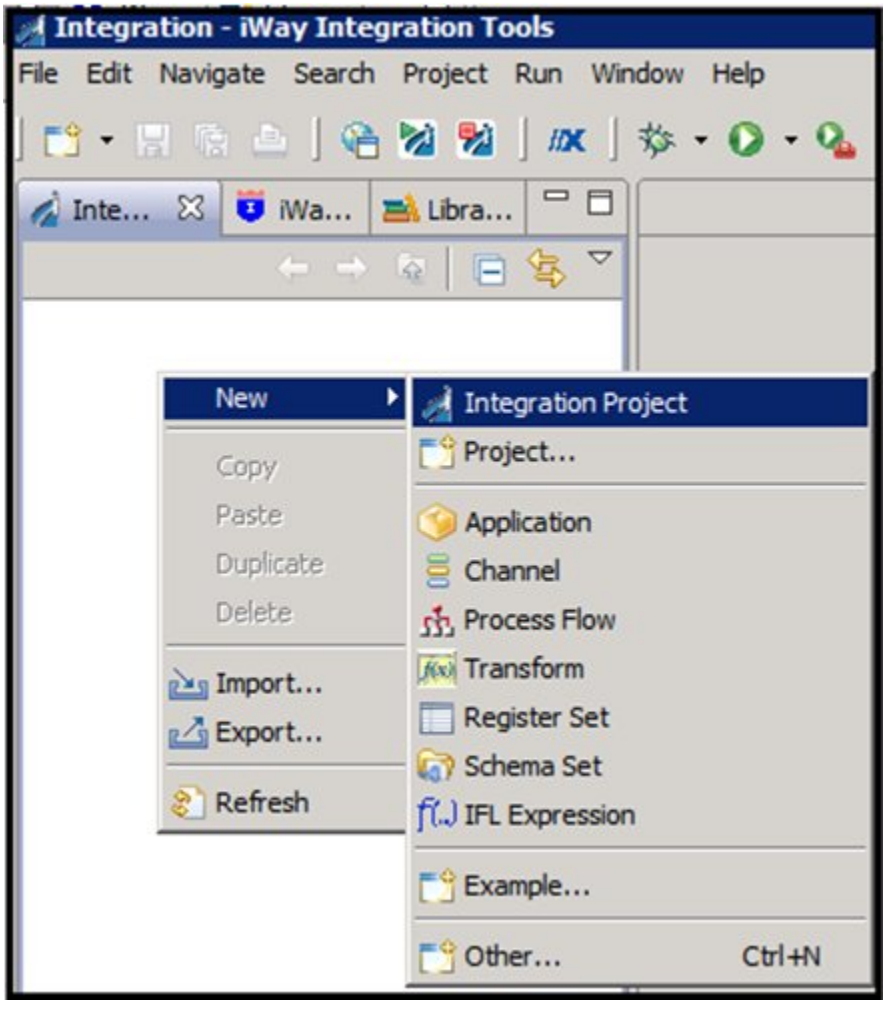
Configuring a Channel for HL7 Inbound Processing

The inbound channel creates an XML representation of a HL7 inbound message, and an acknowledgement message. The documents are routed to designated folders based on the success or failure results of the transformation and HL7 rule validation.

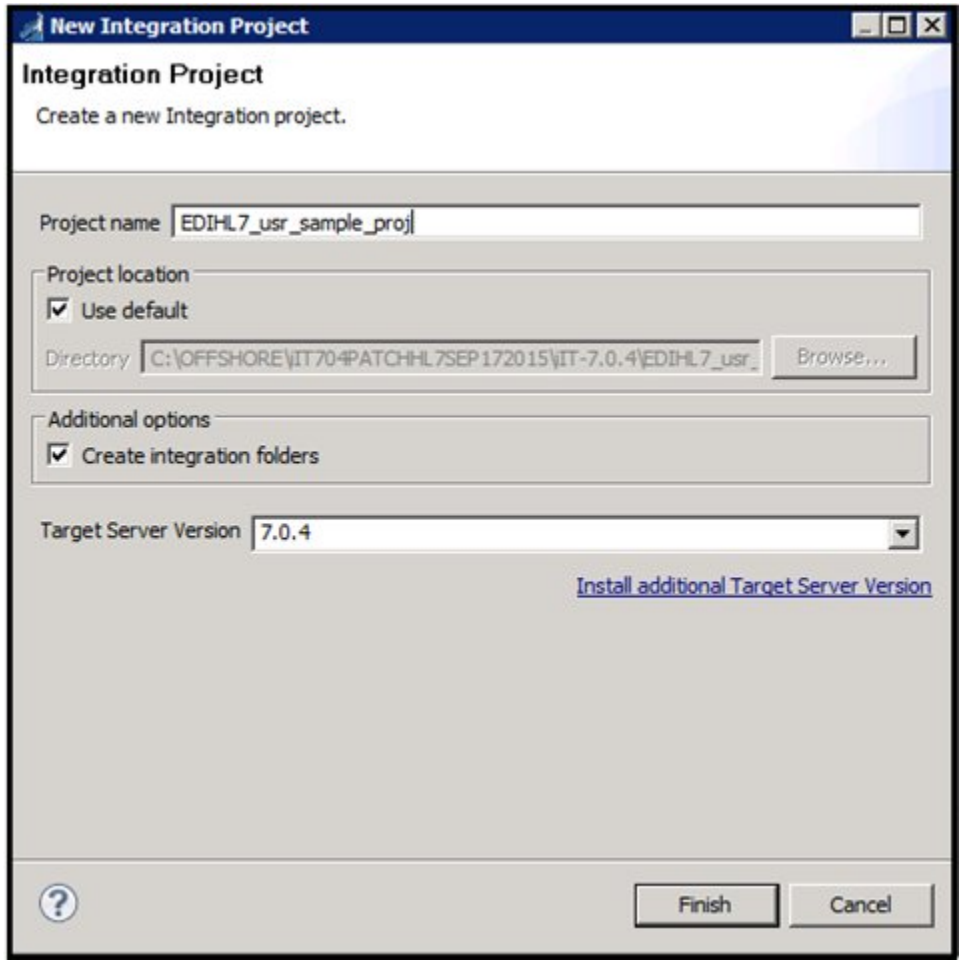
Procedure: How to Create a Channel for Inbound Processing

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.



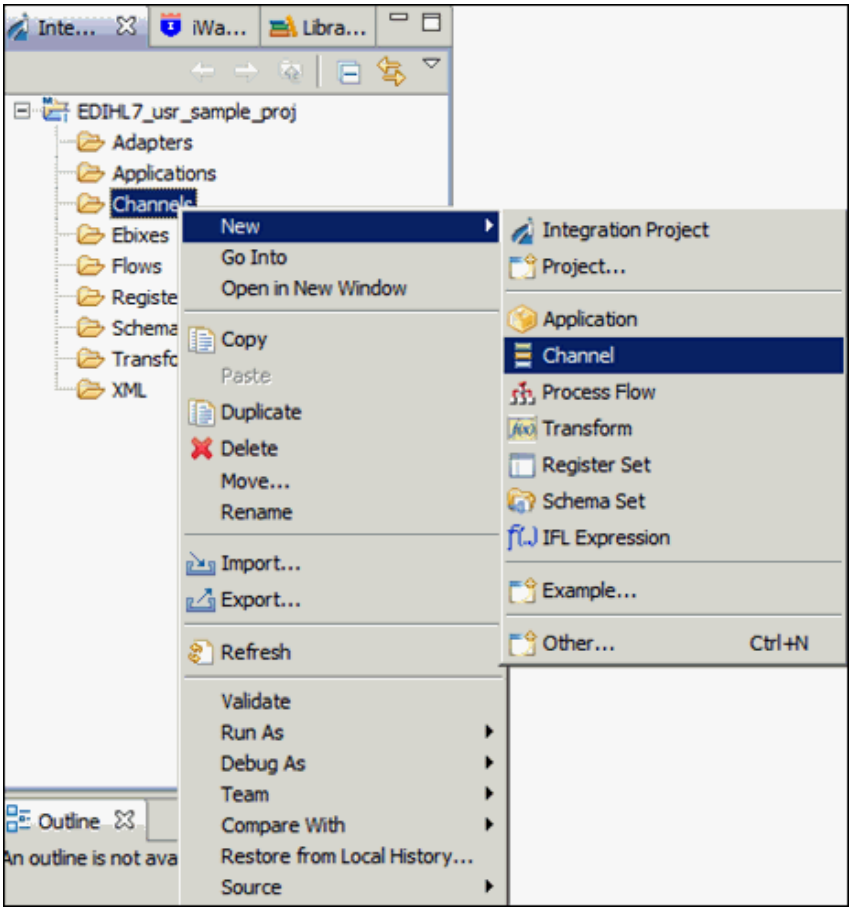
The New Integration Project dialog box opens, as shown in the following image.



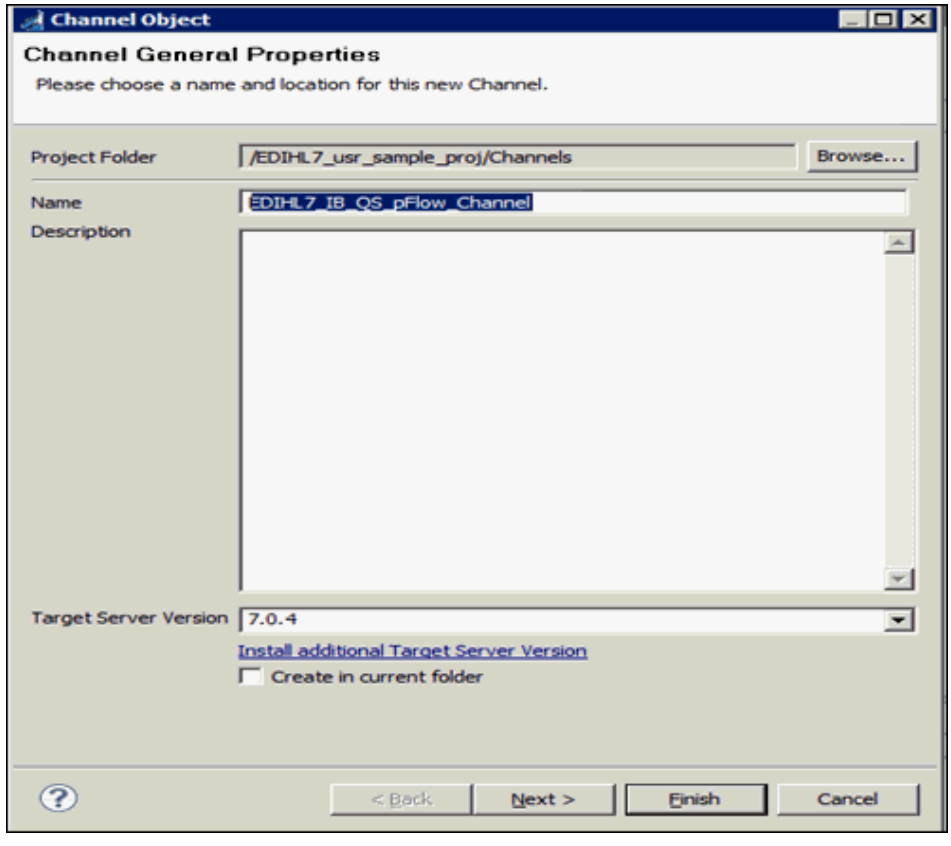
3. Enter a name for the Integration Project (for example, *EDIHL7_usr_sample_proj*) in the Project name field, and then click *Finish*.

   A new Integration Project node called *EDIHL7_usr_sample_proj* is added to the Integration Explorer.

4. Expand the *EDIHL7_usr_sample_proj* Integration Project node, right-click the *Channels* folder, select *New*, and then click *Channel* from the context menu, as shown in the following image.

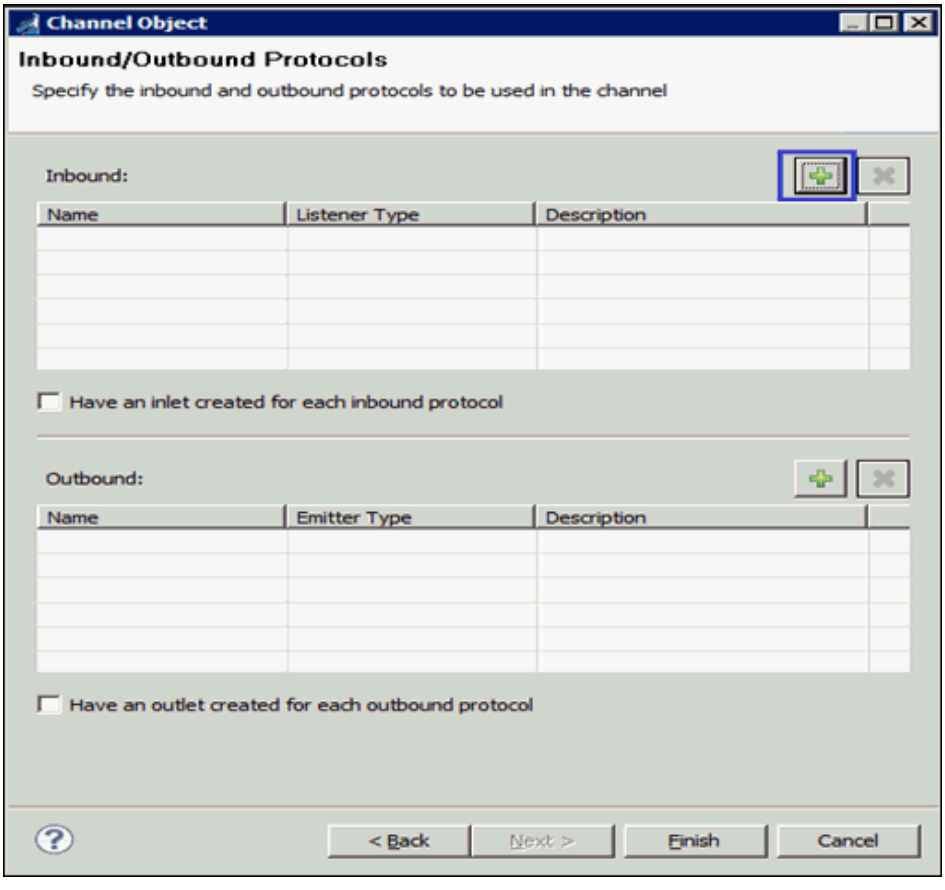


5. Enter a name for the Channel (for example, EDIHL7_IB_QS_pFlow_Channel), and then click *Next*, as shown in the following image.

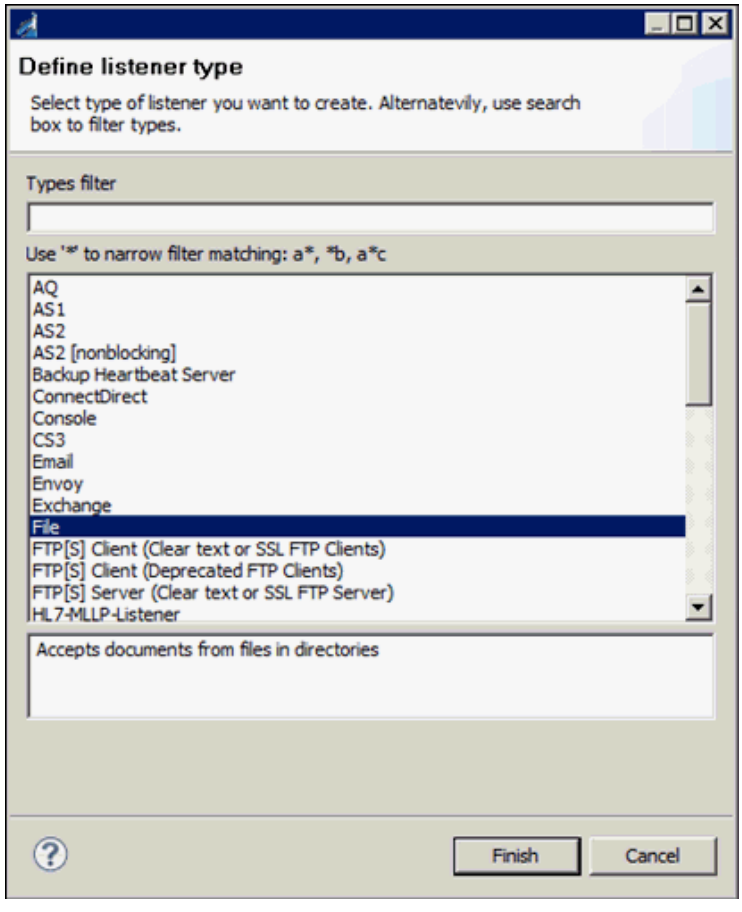


The Inbound/Outbound Protocols pane appears.

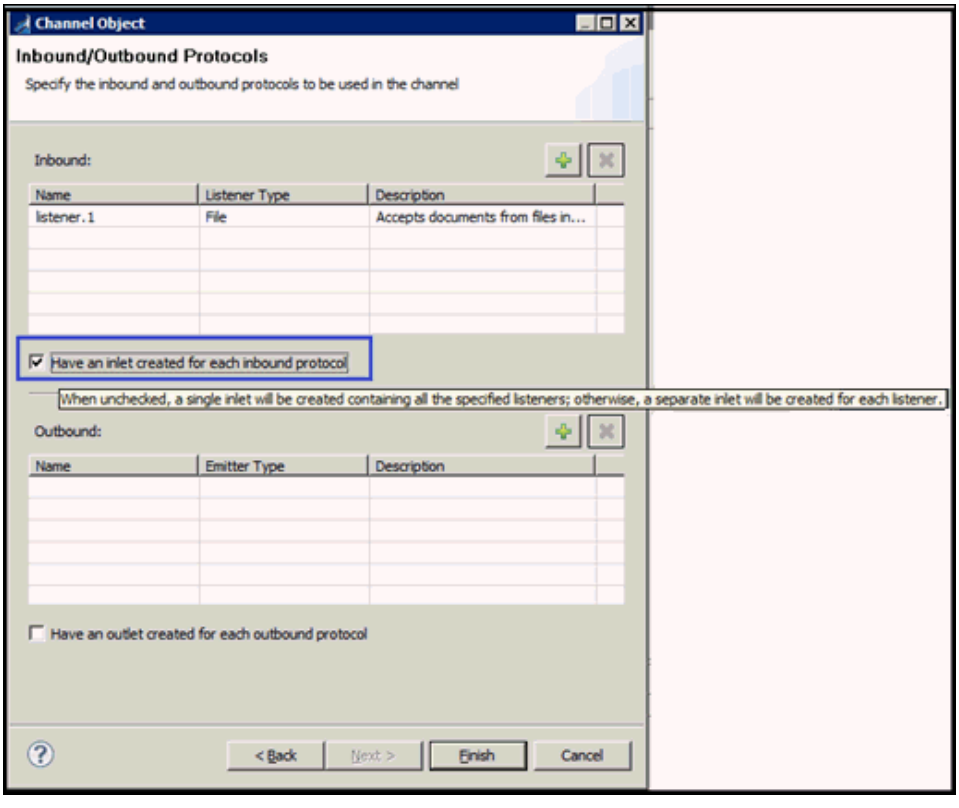
6. Click the *Add* button to add a File Listener in the Inbound section, as shown in the following image.



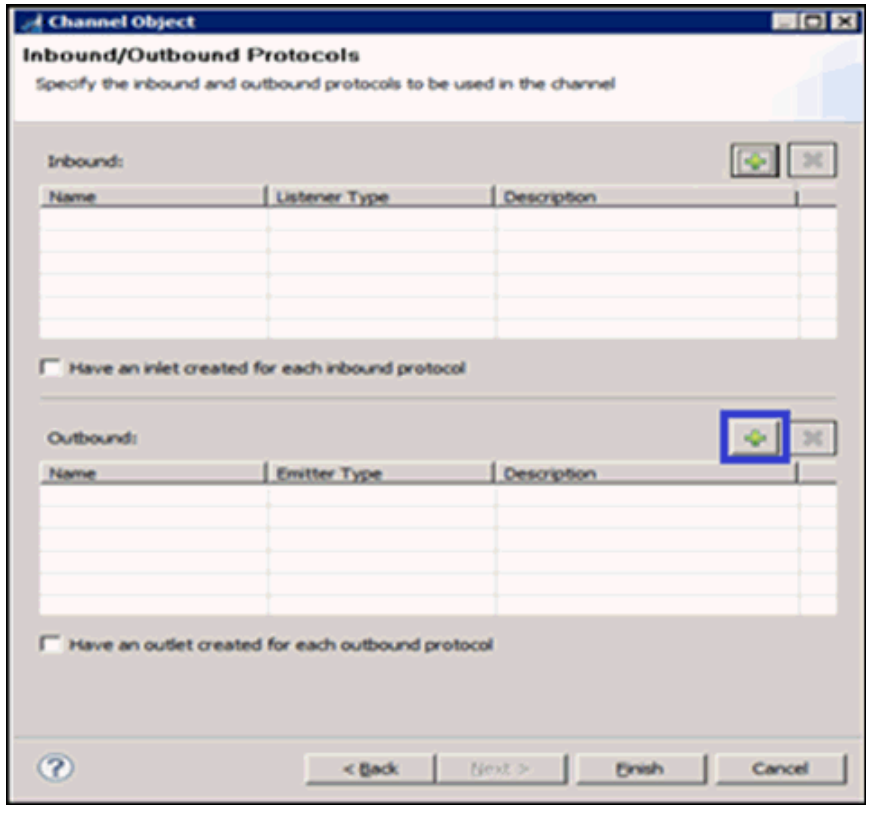
7. From the Types Filter list, select *File* and then click *Finish*, as shown in the following image.



8. Select the *Have an Inlet created for each inbound protocol* check box to create an Inlet for the channel, as shown in the following image.

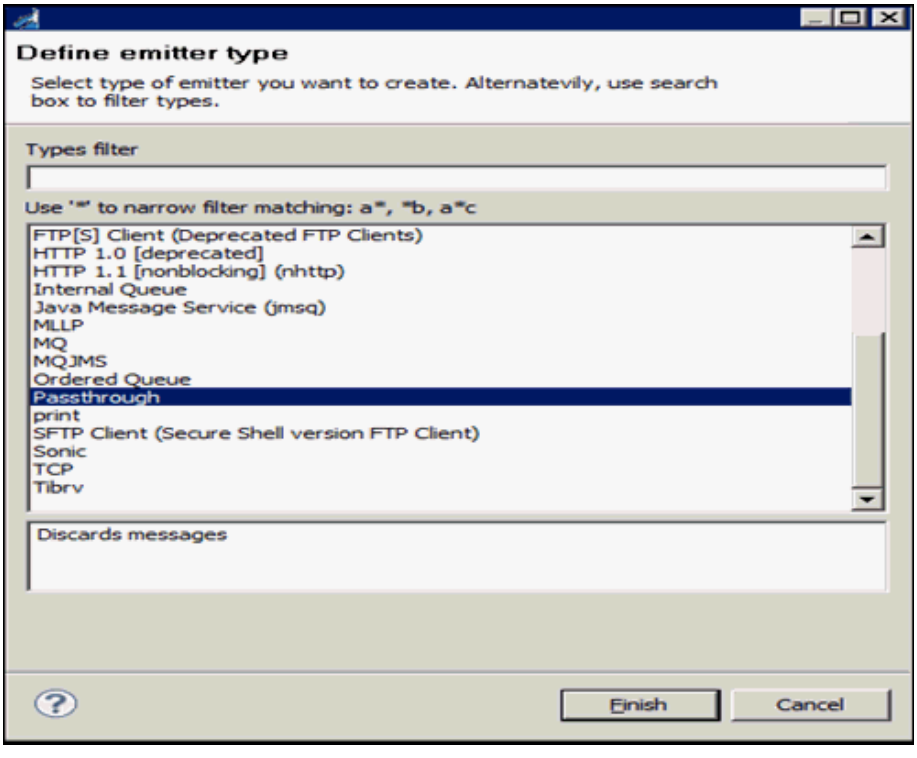


9. Click the *Add* button in the Outbound section to define an emitter.



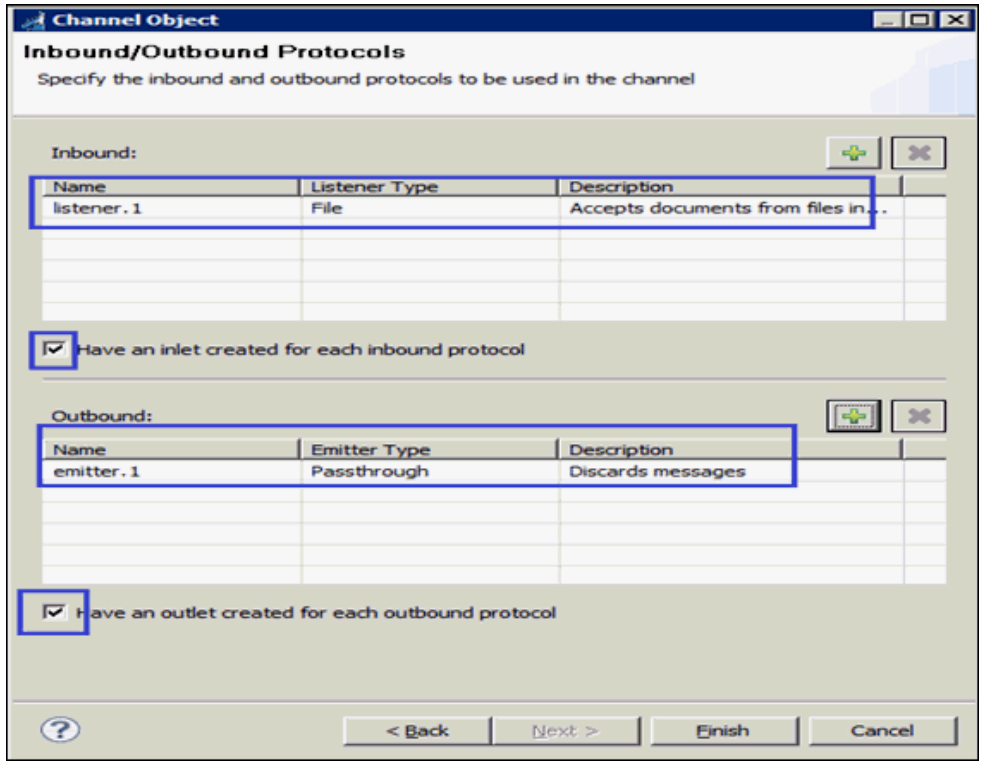
The Define emitter type pane appears.

10. In the Types filter list, select *Passthrough* to define the passthrough emitter, and then click *Finish*, as shown in the following image.

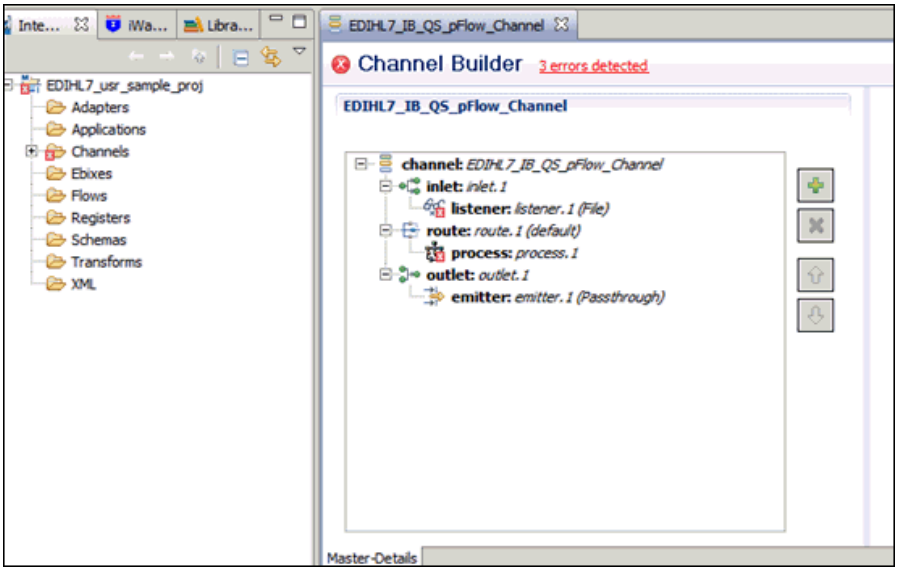


Note: Passthrough does not emit data from the listener. Instead, it just passes the control here and does nothing.

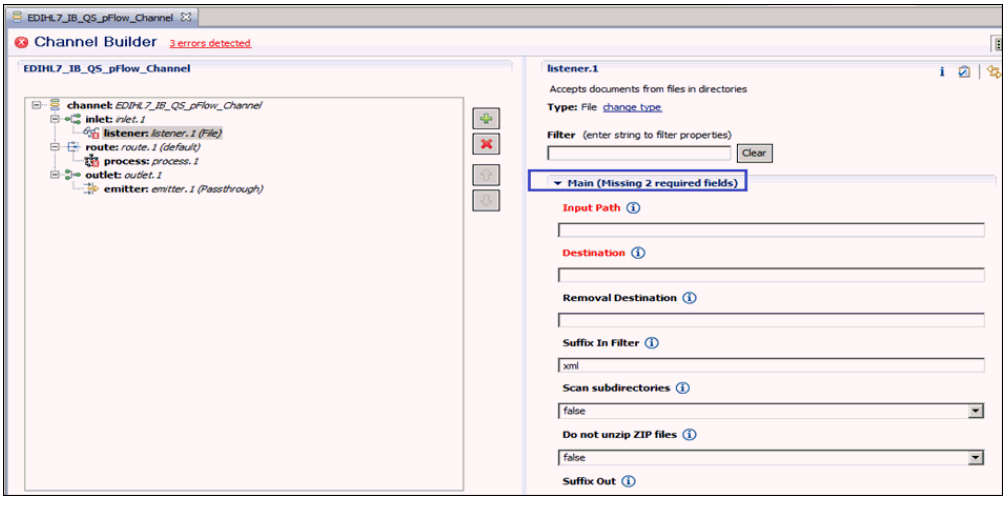
11. Select the *Have an outlet created for each outbound protocol* check box to create an outlet for the channel, and then click *Finish*, as shown in the following image.



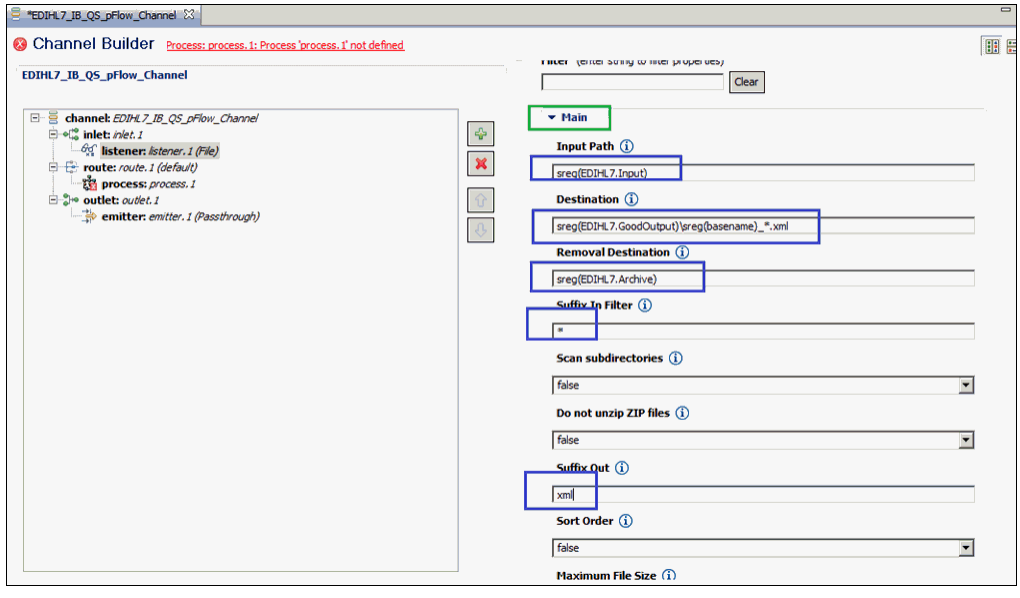
The Channel Builder pane appears, as shown in the following image.



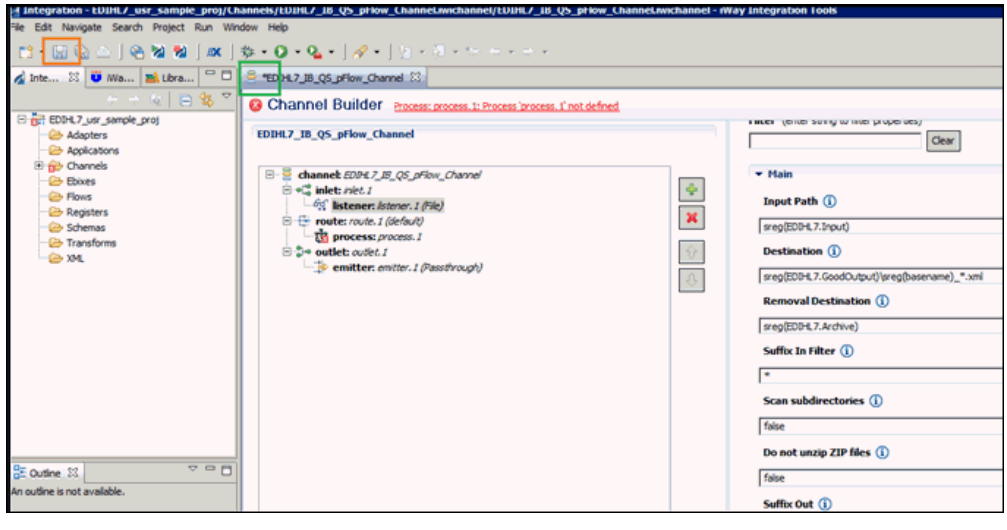
12. Under *inlet: inlet.1*, click *listener: listener.1 (File)* and then expand the *Main (Missing 2 required fields)* configuration parameter on the right pane, as shown in the following image.



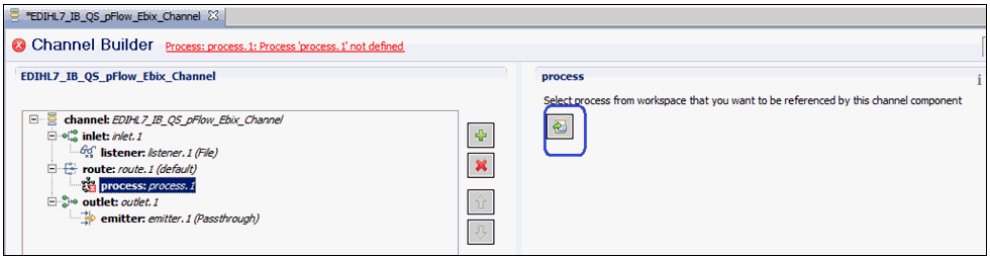
13. Enter a valid Input Path, Destination, and Removal Destination (optional) path, then change the Suffix In Filter parameter from *xml* to * to receive all types of input files (for example, hl7 and txt), and then set the Suffix Out parameter value to *xml*, as shown in the following image.



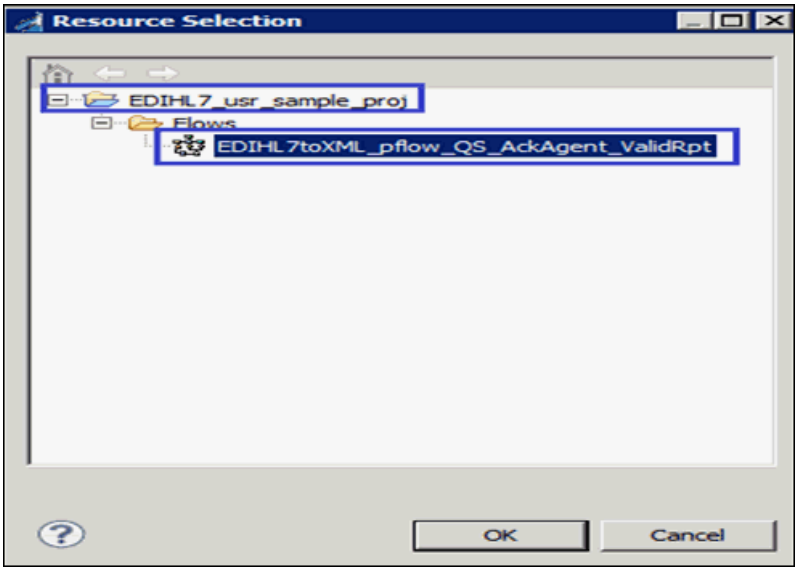
14. Click the *Save* icon near the File menu to save your edited listener configuration. You can also press the shortcut key (Ctrl+S) to save your work if you are using a Windows environment.



15. Import the *EDIHL7toXML_pflow_QS_AckAgent_ValidRpt* process flow from the local drive or create one in the EDIHL7_usr_sample_proj directory in the Flows folder.
16. Under the *route: route.1(default)* node, select *process: process.1* and click the process icon on the right panel to reference the process flow into your channel, as shown in the following image.



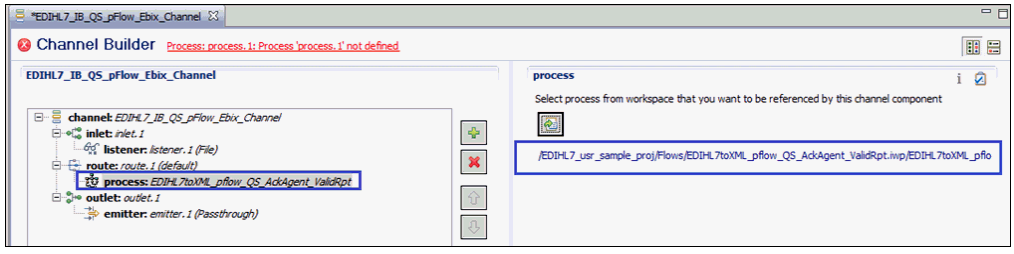
17. Select a process flow from the integration project and then click *OK*.



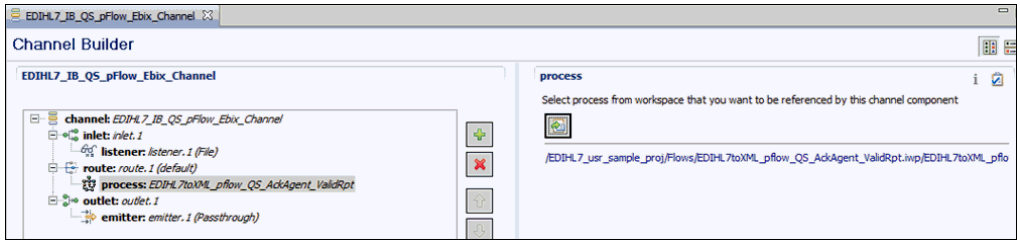
Note: Process flows should already be built and available in the iIT integration project. They can be found in the EDIHL7_usr_sample_proj directory inside the Flows folder.

For more information, see *Process Flow Used for Inbound Processing: HL7 to XML* on page 81.

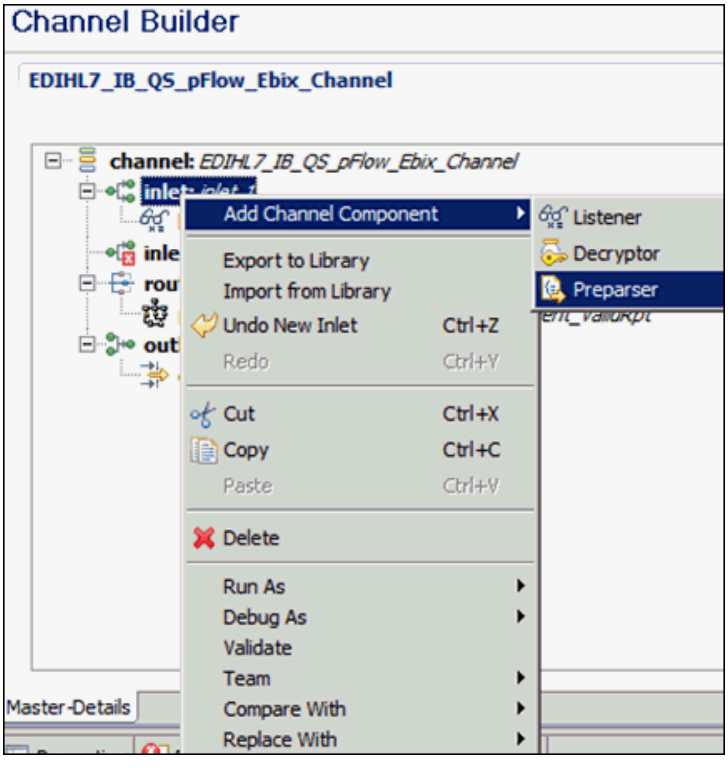
Your screen should resemble the following image.



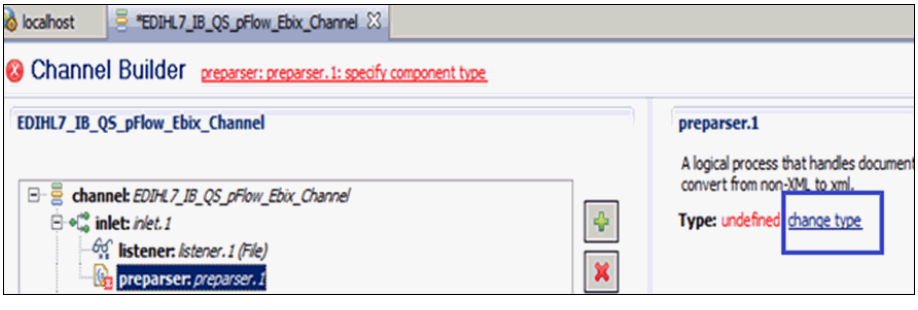
18. Click the *Save* icon near the File menu to save your edited listener configuration. You can also press the shortcut key (Ctrl+S) to save your work if you are using a Windows environment.



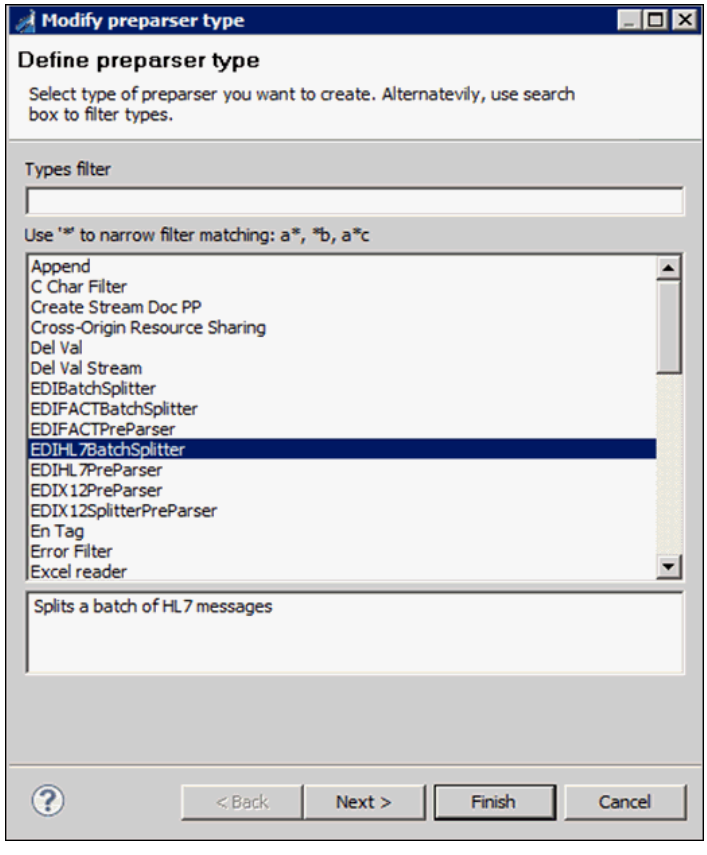
19. Right-click *inlet: inlet.1*, select *Add Channel Component*, and then click *Preparser* from the sub context menu, as shown in the following image.



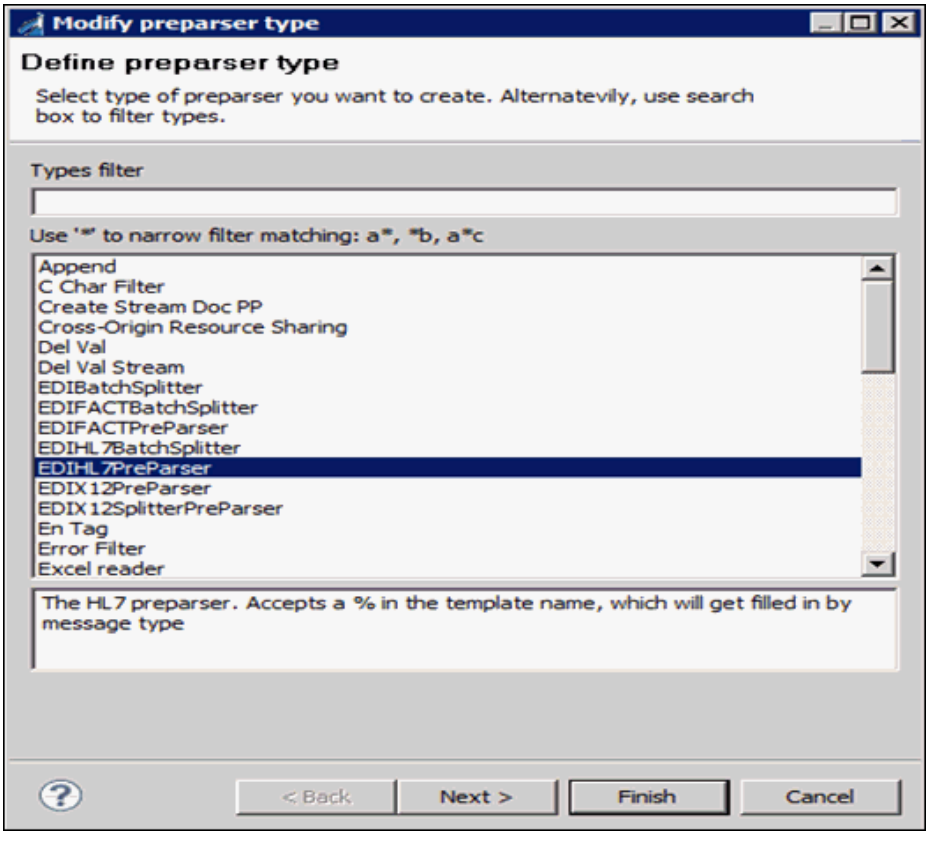
20. Click the *change type* link provided in the preparser.1 configuration section, as shown in the following image.



21. From the Types filter list of the Define preparser type pane, select *EDIHL7BatchSplitter* and click *Finish*, as shown in the following image.

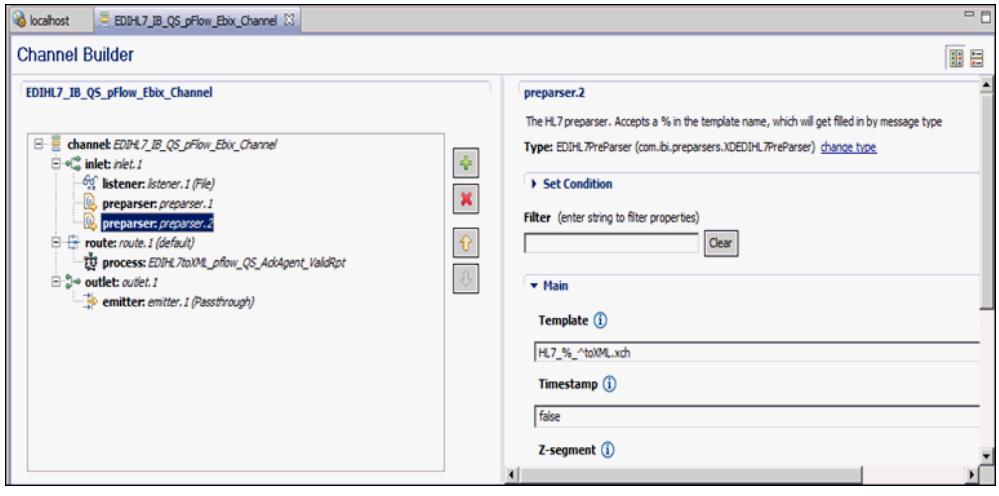


22. Repeat steps 20 and 21 and choose the *EDIHL7PreParser* preparser type when you are in step 21, as shown in the following image.



23. Click *Finish*.

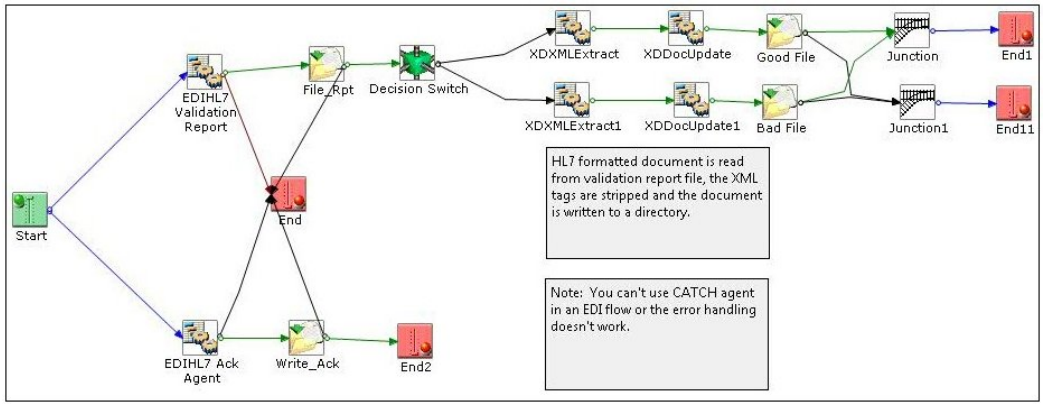
Your screen should resemble the following image.



Reference: Process Flow Used for Inbound Processing: HL7 to XML

This section provides an overview of the process flow used for inbound processing: HL7 to XML. This process flow (EDIHL7toXML_pflow_QS_AckAgent_ValidRpt) is already built and available in the iIT integration project. It is located under the *EDIHL7_usr_sample_proj* node inside the Flows subfolder.

The following image shows the entire inbound process flow, including all of the nodes that are used and their connections.

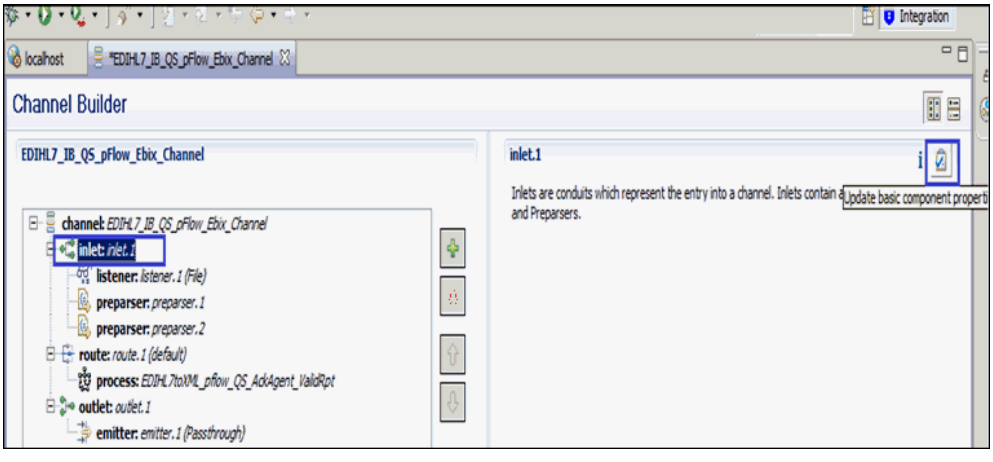


In this process flow, an HL7 formatted document is read from a validation report file. The XML tags are stripped and the document is written to a directory.

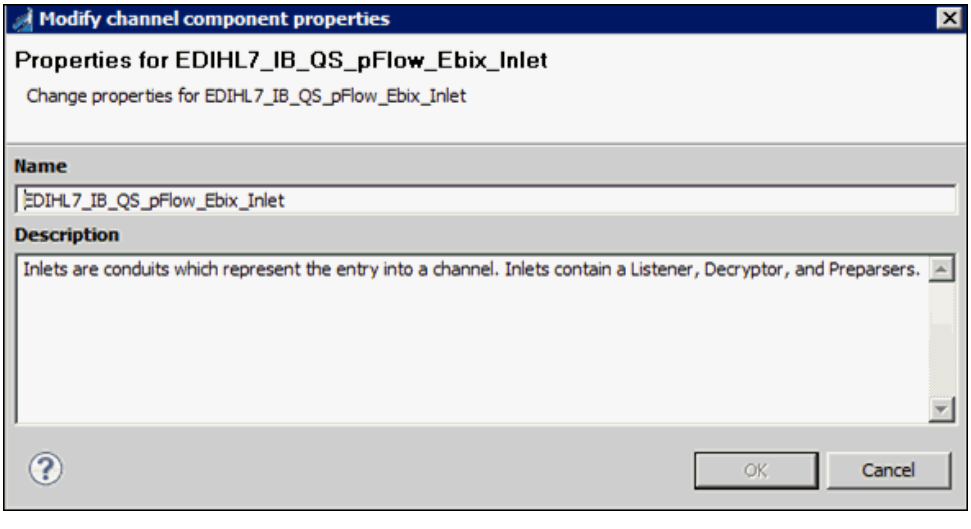
Note: Using a Catch service (com.ibi.agents.XDCatchAgent) in an EDI flow is not supported. The error handling does not work as a result.

Procedure: How to Update Basic Details for Channel Components

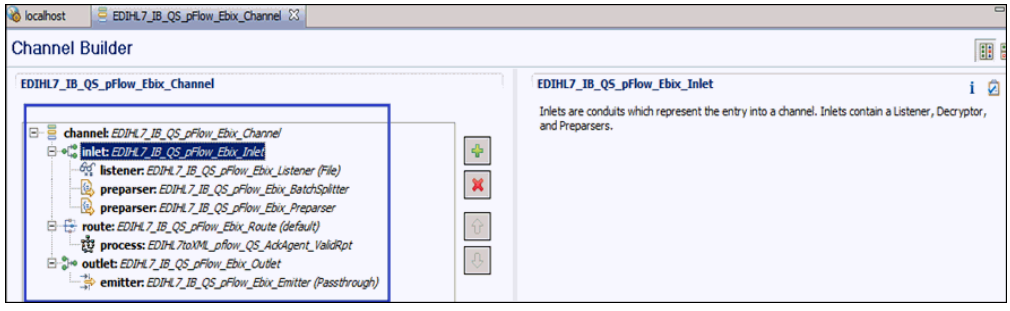
1. Select the channel component *inlet:inlet.1* from the channel and click the update channel component icon on the top right side panel, as shown in the following image.



2. Rename or update the inlet details (for example, XMLEDIHL7_IB_QS_pFlow_Ebix_Inlet), and click *OK*, as shown in the following image.



After renaming or changing the channel components, your iIT Channel Builder should resemble the following image.

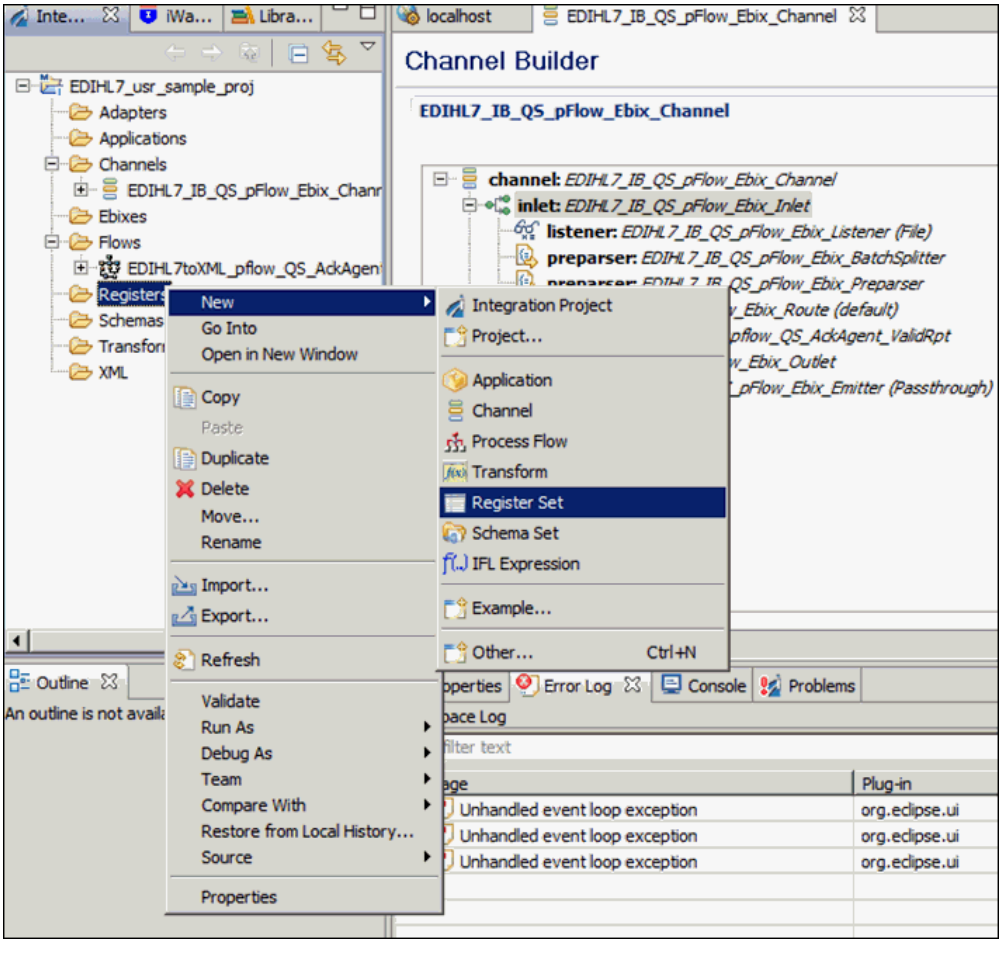


Configuring Register Sets and Registers

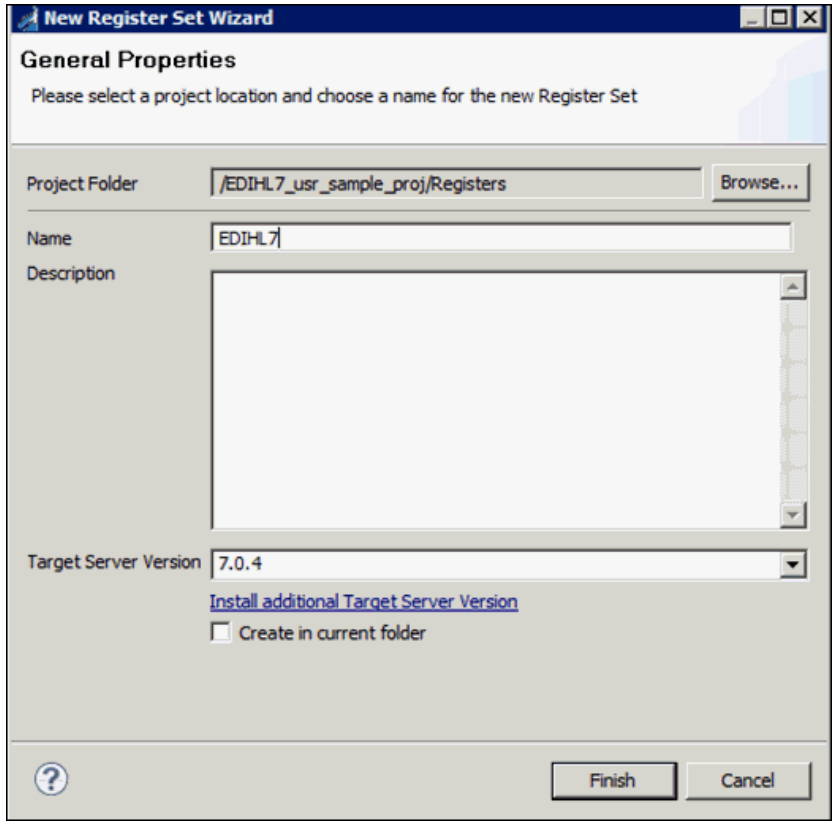
This section describes how to configure register sets and registers using iWay Integration Tools (iIT).

Procedure: How to Configure Register Sets and Registers

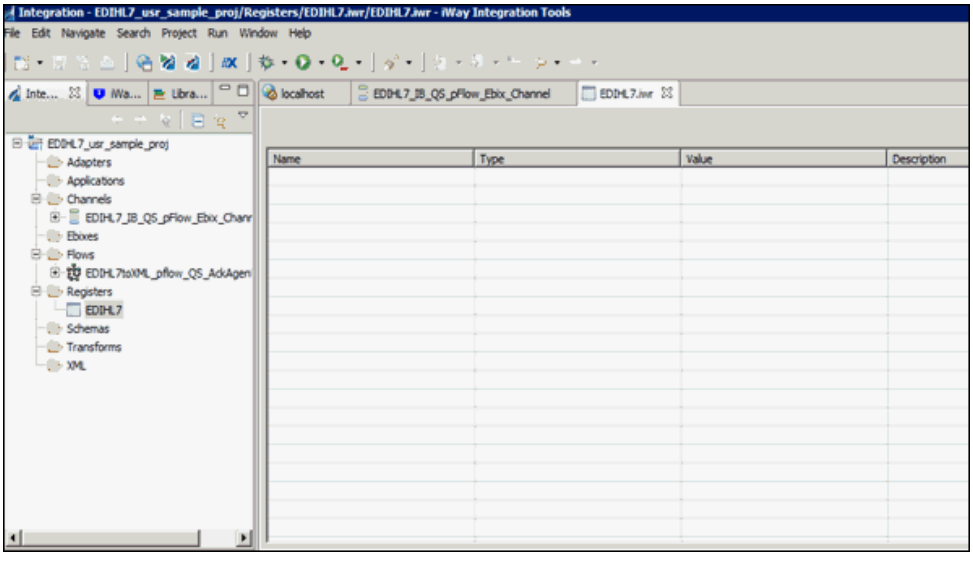
1. In the EDIHL7_usr_sample_proj project tree, right-click the Registers folder, and select *New*, and then click *Register Set* from the context menu, as shown in the following image.



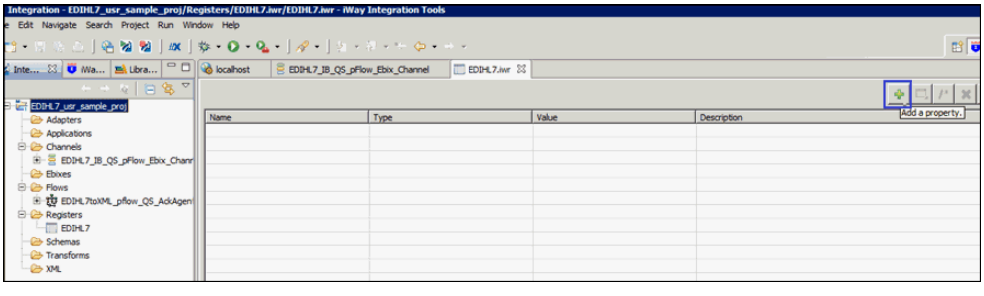
2. In the Name field, enter a name for the register set and click *Finish*, as shown in the following image.



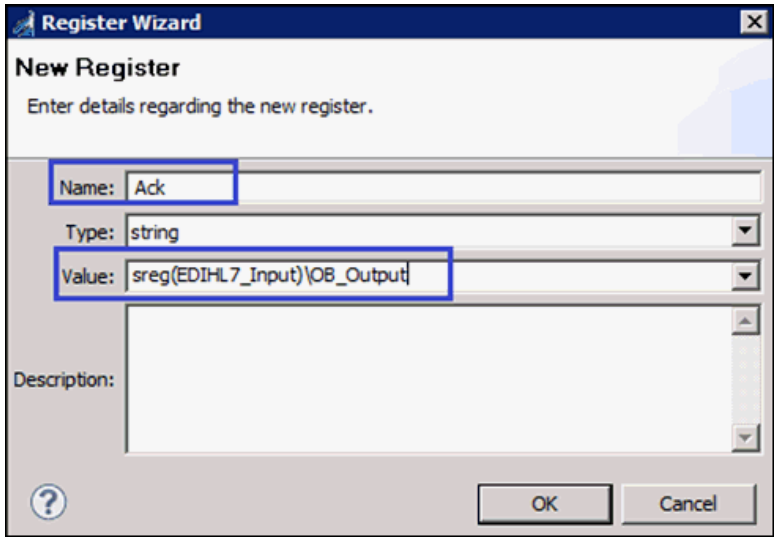
The new register set appears under Registers in the Registers folder, as shown in the following image.



3. Click the *Add a property* icon to add a register to the register set, as shown in the following image.



4. Enter a name for the new register in the Name field, select a register type from the Type drop-down list (set to *string* by default), and then enter a value in the Value field, as shown in the following image.

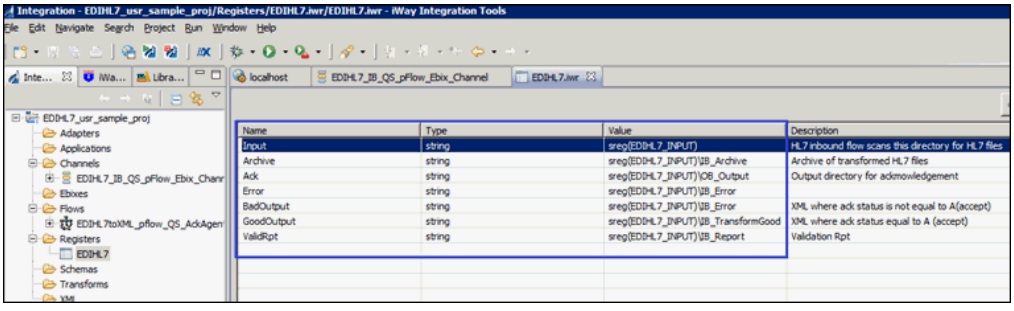


5. Click *OK*.

6. Create the following registers under the EDIHL7 registers set you just created, along with the values shown in the table below.

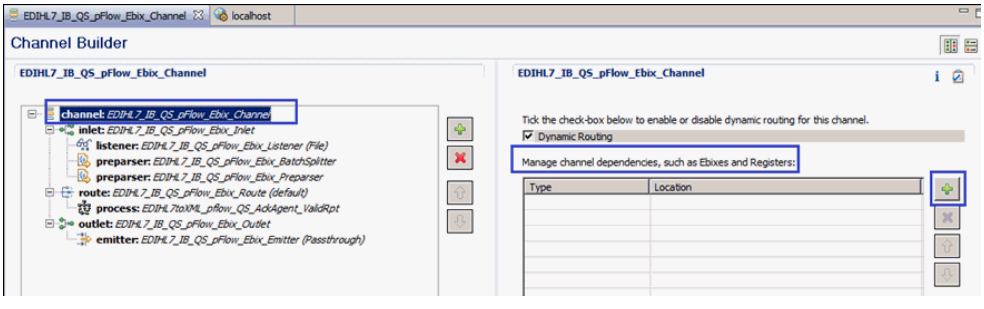
Register Name	Value
Input	Name=Input, Value=sreg(EDIHL7_INPUT)
Error	Name=Error, Value=sreg(EDIHL7_INPUT)\IB_Error
GoodOutput	Name=Output, Value=sreg(EDIHL7_INPUT)\IB_TransformGood
BadOutput	Name=BadOutput, Value=sreg(EDIHL7_INPUT)\IB_Error
ValidReport	Name=ValidReport, Value=sreg(EDIHL7_INPUT)\IB_Report
Archive	Name=Archive, Value=sreg(EDIHL7_INPUT)\IB_Archive

After creating all of the required registers in register set, your iIT screen should resemble the following image.

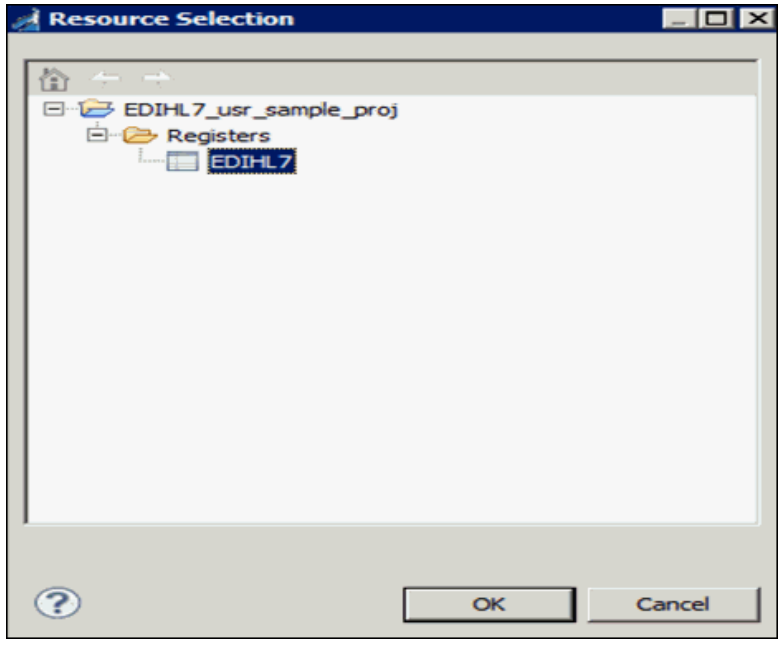


Procedure: How to Add a Register Set to an Inbound Channel as a Dependency

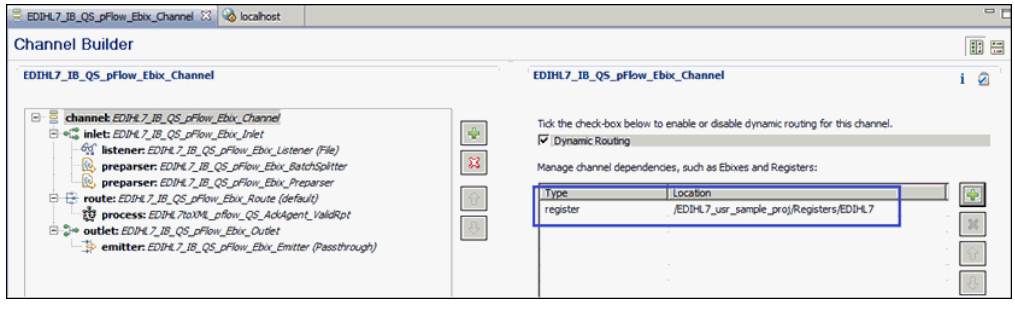
1. Click the *XMLEDIHL7_IB_QS_pFlow_Ebix_Channel* node and then click the Add dependency icon on the far right side panel of the channel properties pane, as shown in the following image.



2. From the Registers folder under the integration folder, select *EDIHL7* and click *OK*.



The iIT page should resemble the following image.



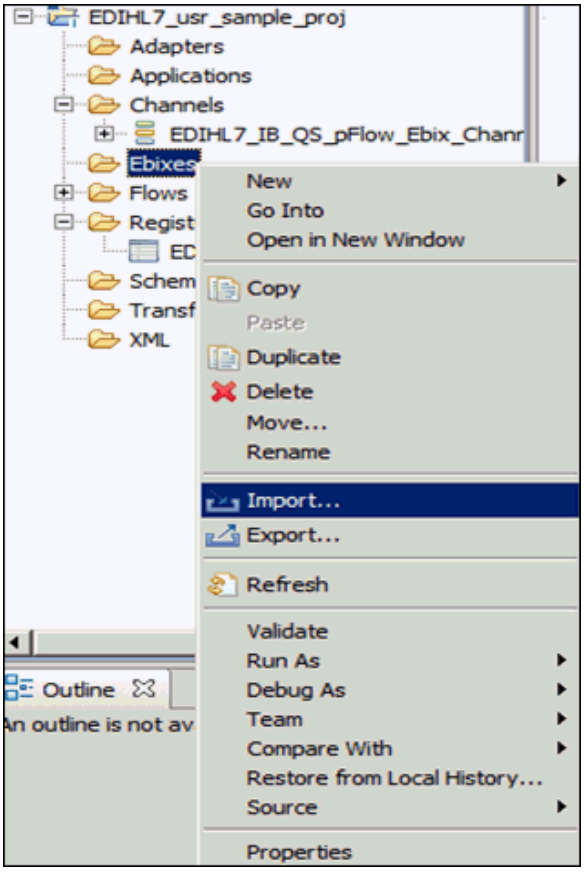
3. Click the *Save* icon to save your changes. You can also use the keyboard shortcut (Ctrl+S) if you are using a Windows environment.

Importing an Ebix Into the Workspace

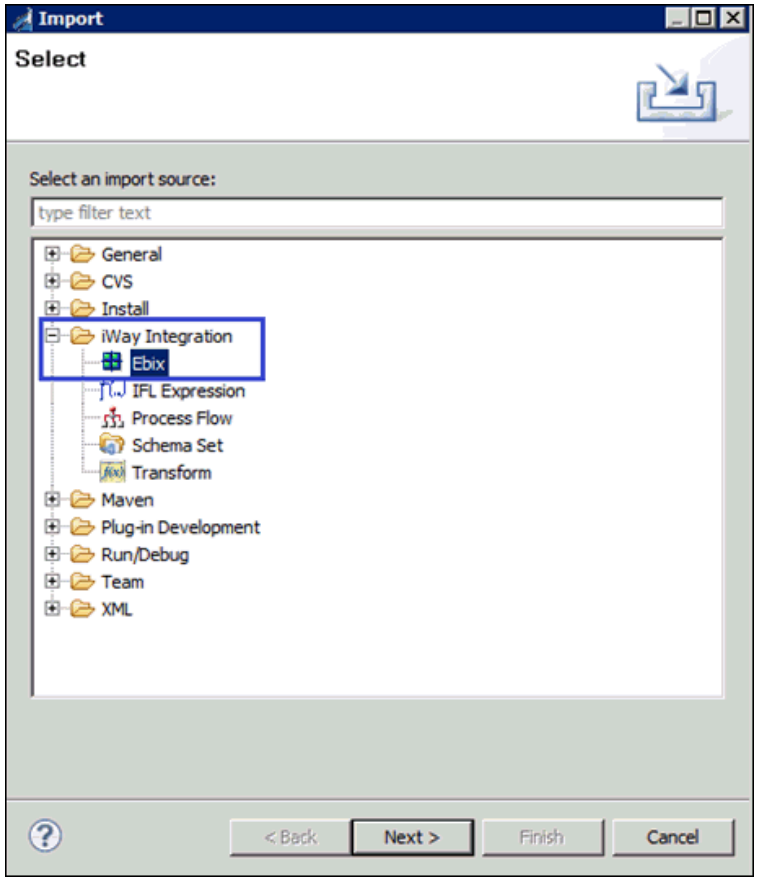
This section describes how to import an Ebix into the workspace using iWay Integration Tools (iIT).

Procedure: How to Import an Ebix Into the Workspace

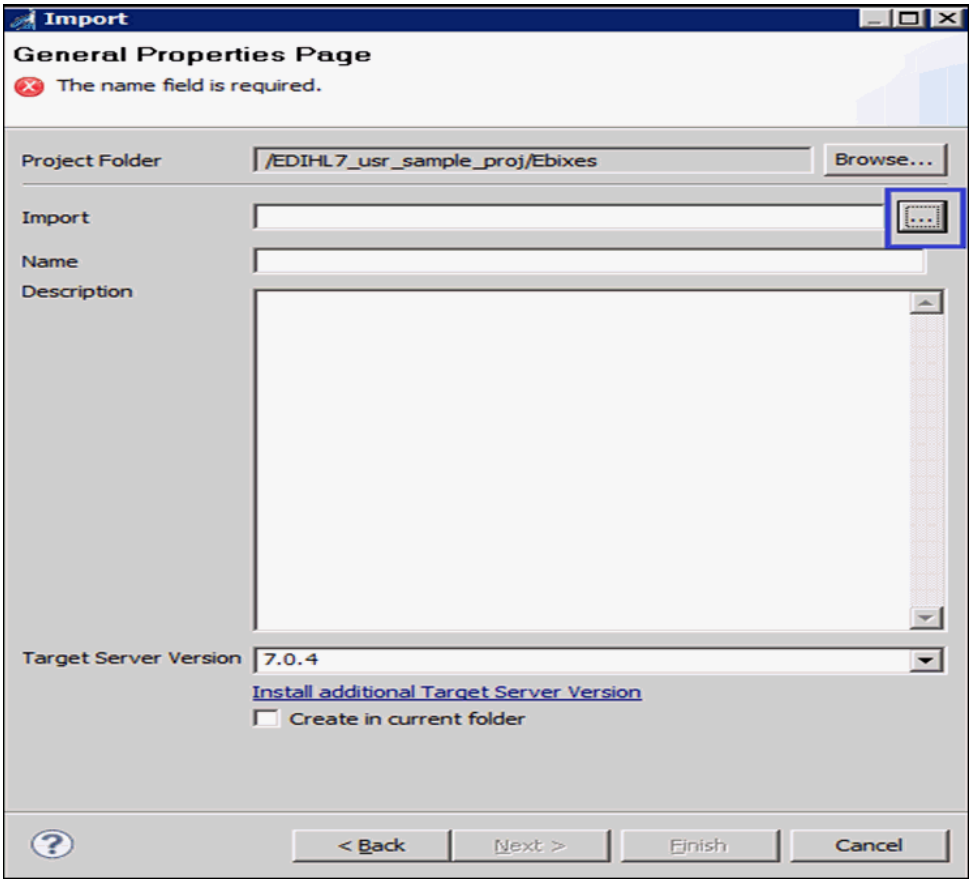
1. In the integration project EDIHL7_usr_sample_proj, right-click the Ebixes folder and then select *Import* from the context menu, as shown in the following image.



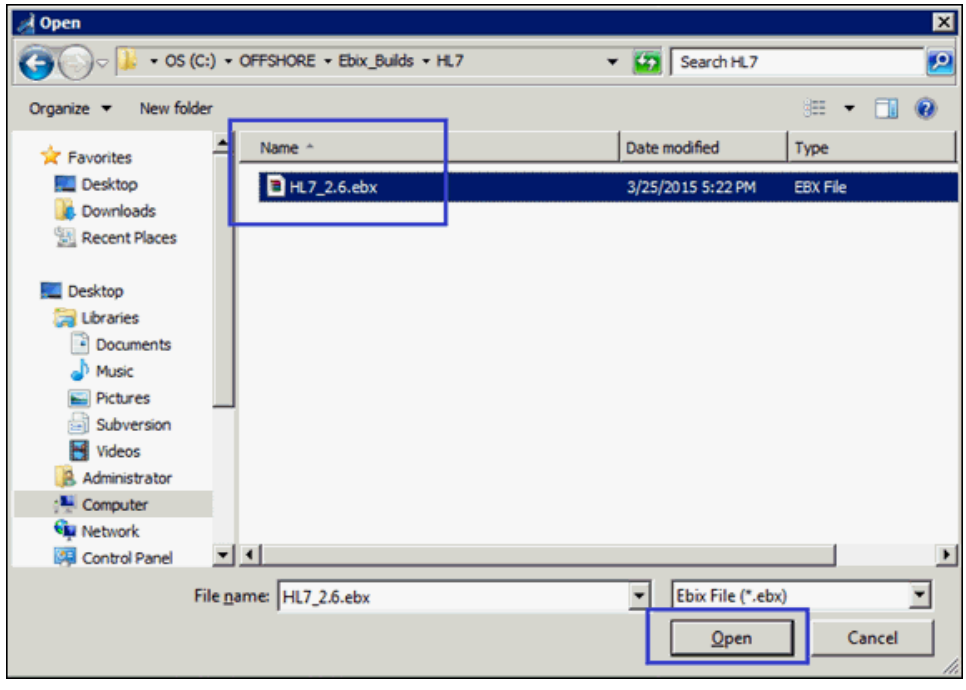
2. Expand the iWay Integration folder, select *Ebix*, and then click *Next*, as shown in the following image.



3. Click the ellipses (...) button to browse and import the ebix from a specific folder location, as shown in the following image.

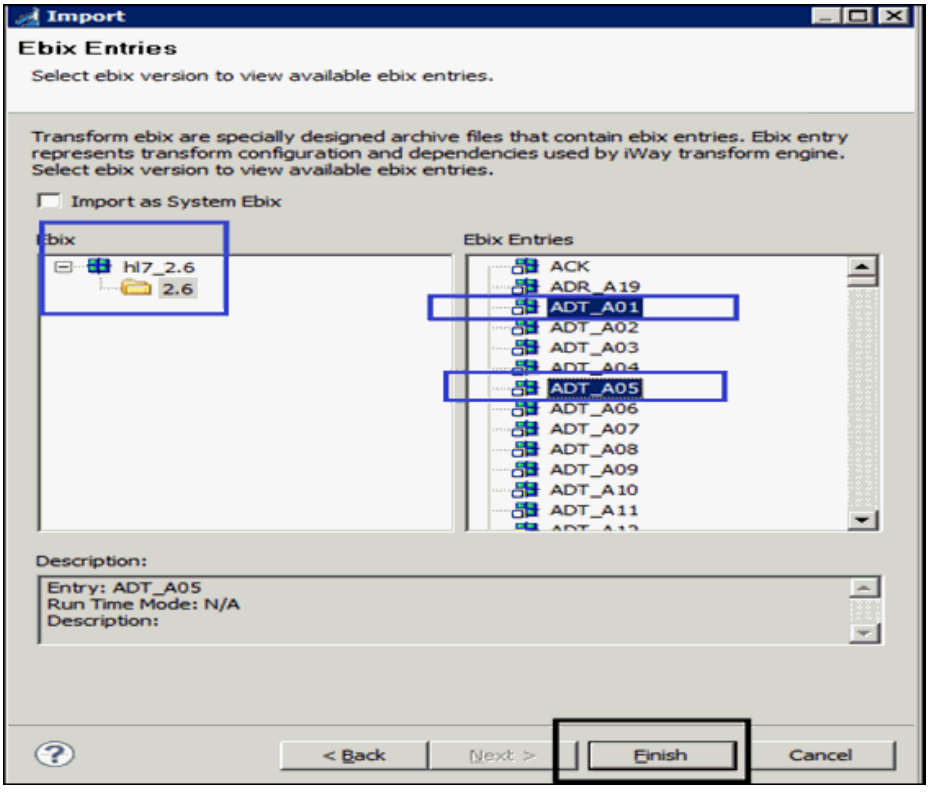


4. Select *HL7_2.6 ebix* from the folder location and click *Open*, as shown in the following image.

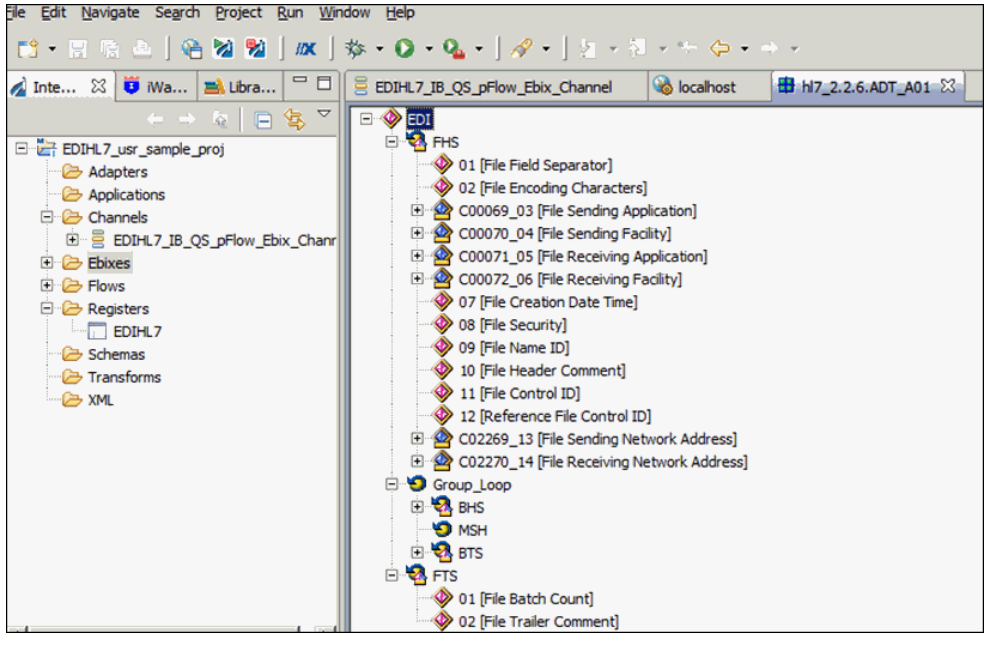


5. In the Import wizard, click *Next*.
6. Expand the hl7_2.6 ebix from the left panel and select the 2.6 folder.

7. Select the messages you wish from the right side panel and click *Finish*, as shown in the following image.

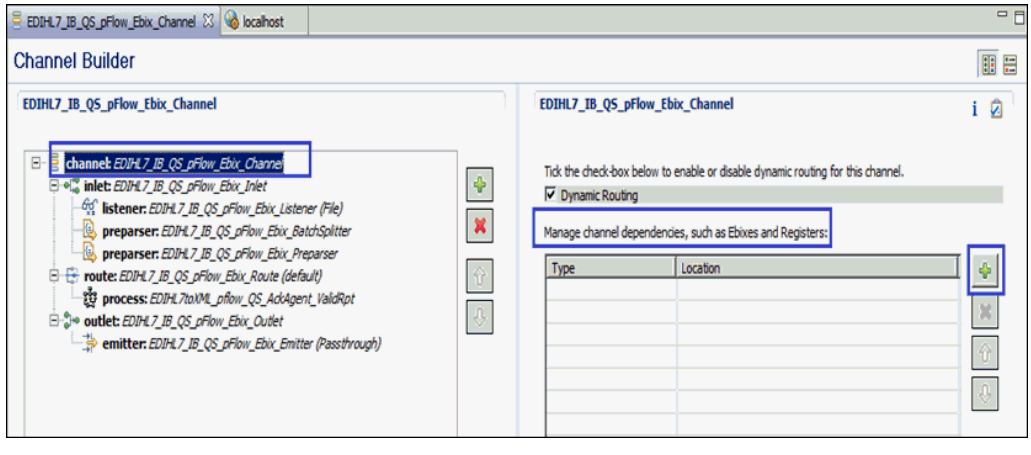


The iIT page should resemble the following image.

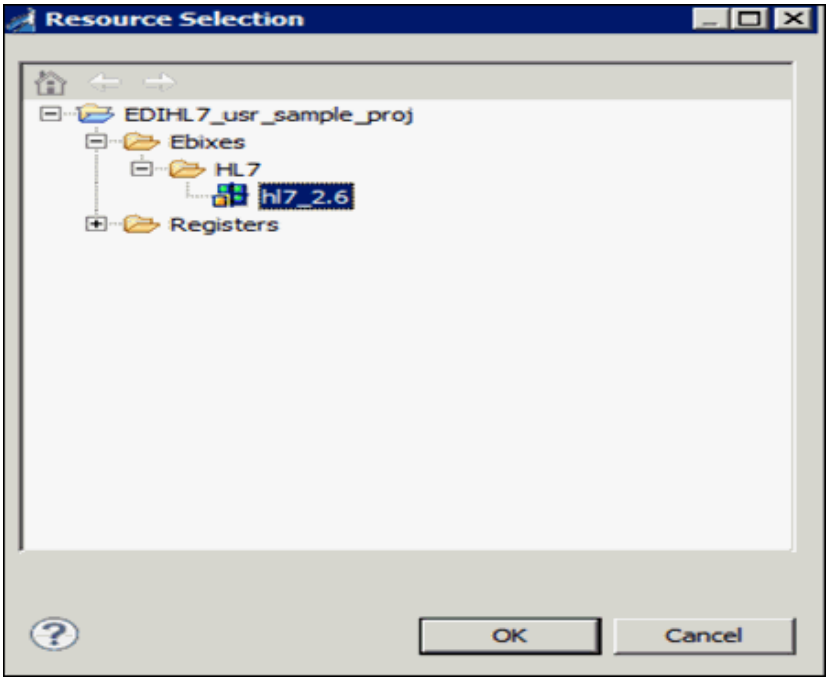


Procedure: How to Add an Ebix to an Inbound Channel as a Dependency

1. Click the *channel: XMLEDIHL7_IB_QS_pFlow_Ebix_Channel* node and then click the *Add dependency* icon on the right side panel, as shown in the following image.

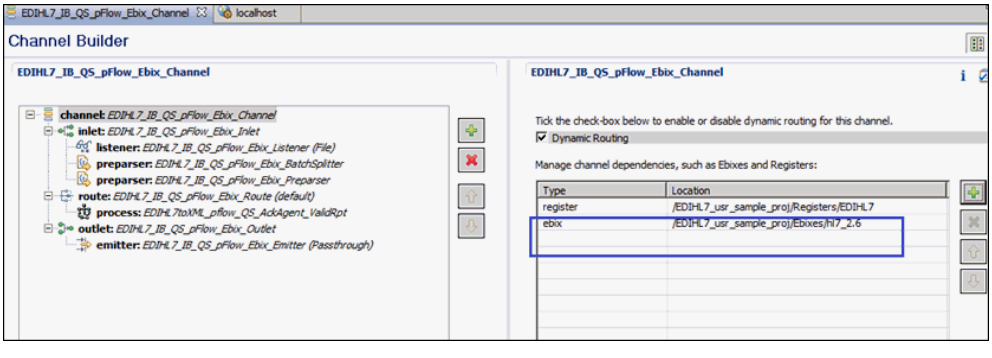


2. Expand *EDIHL7_usr_sample_proj*, *Ebixes*, *HL7*, and then select ebix *hl7_2.6* and click *OK*, as shown in the following image.



3. Click the *Save* button to save your changes or press the keyboard shortcut (Ctrl+S) if you are using a Windows environment.

   Your iIT workspace should resemble the following image.

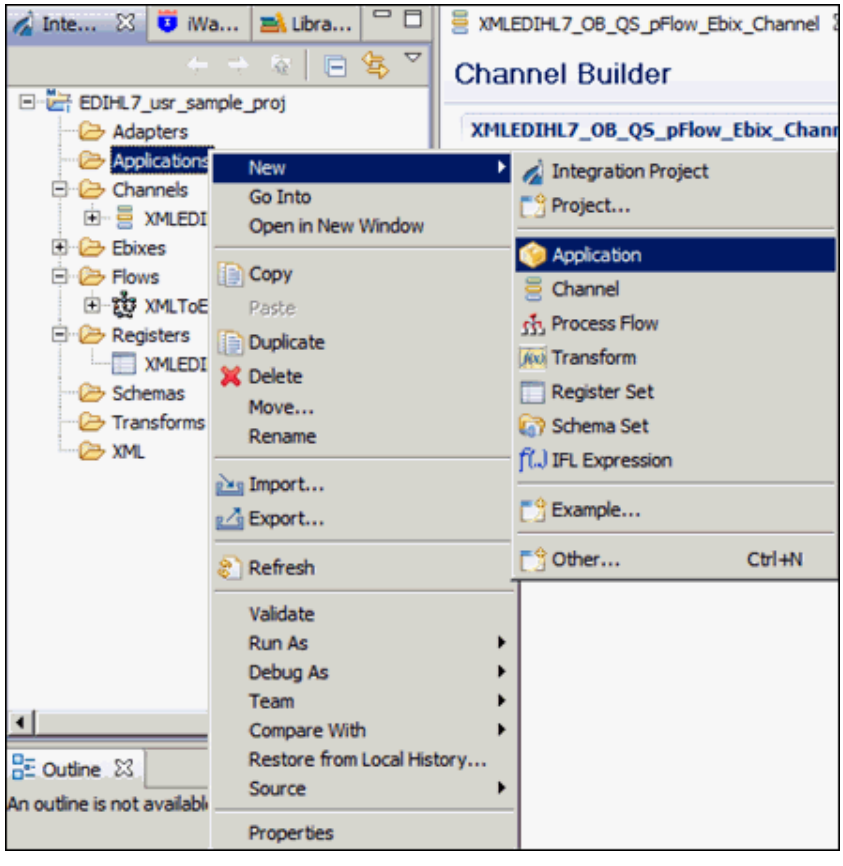


Configuring an iWay Integration Application for Inbound Processing

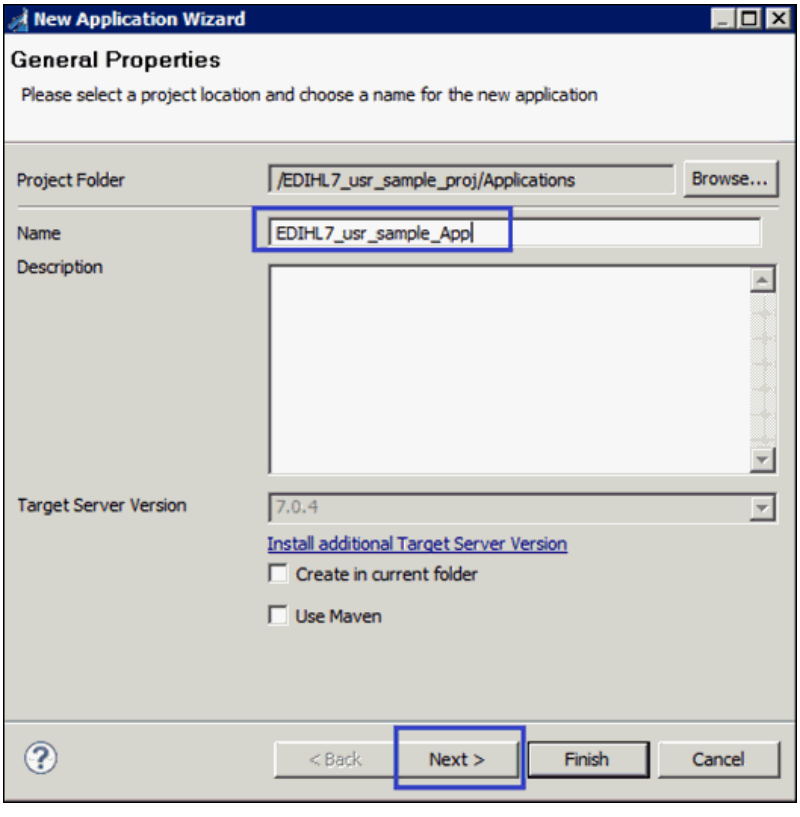
This section describes how to configure an iWay Integration Application (iIA) for inbound processing using iWay Integration Tools (iIT).

Procedure: How to Configure an iWay Integration Application for Inbound Processing

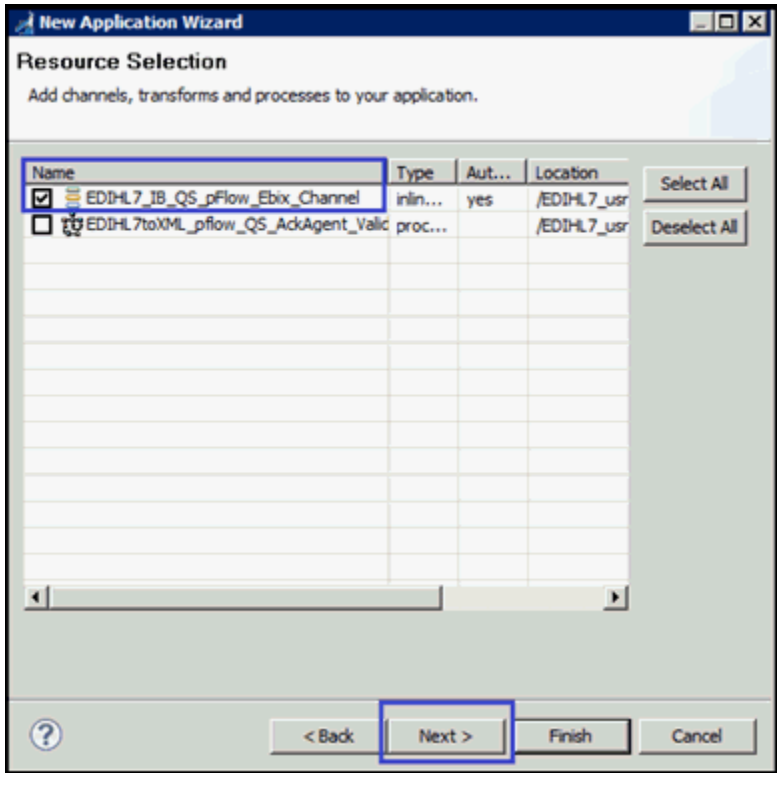
1. Right-click the Applications folder under the EDIHL7_usr_sample_proj integration project, select *New*, and then click *Application* from the context menu, as shown in the following image.



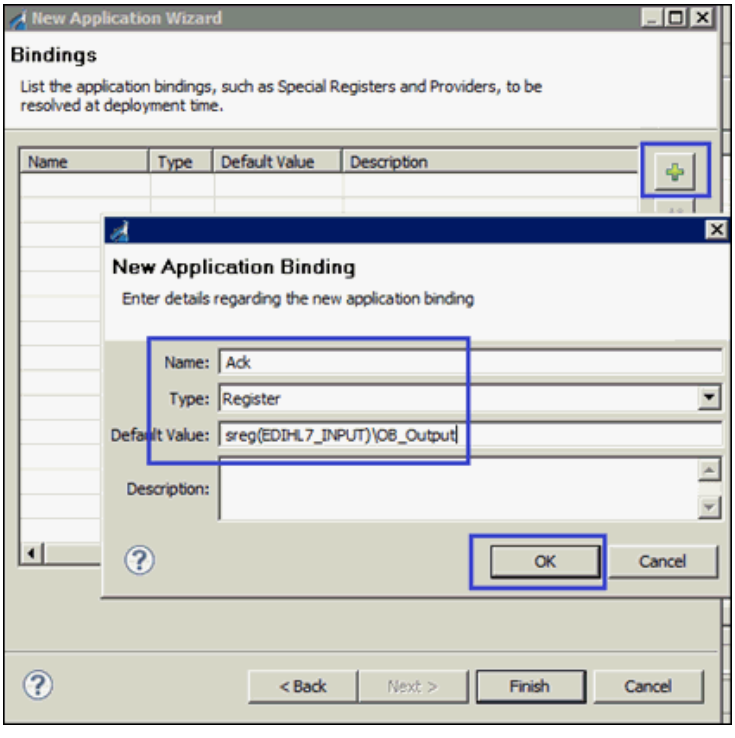
2. Enter a value in the Name field (for example, *EDIHL7_usr_sample_App*), and click *Next*, as shown in the following image.



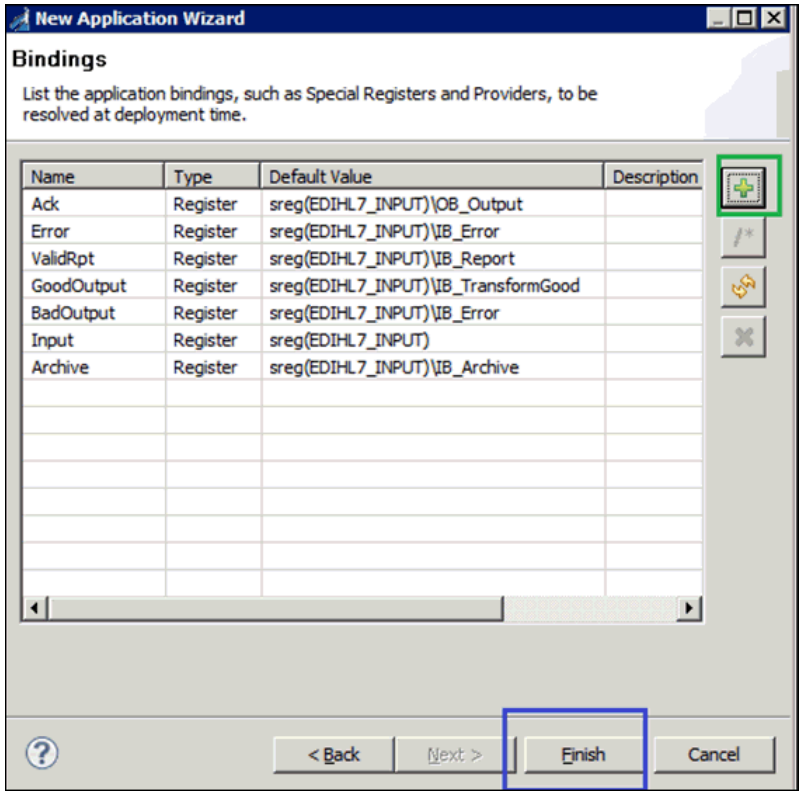
3. Select the *XMLEDIHL7_IB_QS_pFlow_Ebix_Channel* check box from Resource Selection pane and keep clicking *Next* until you get to the Bindings pane, as shown in the following image.



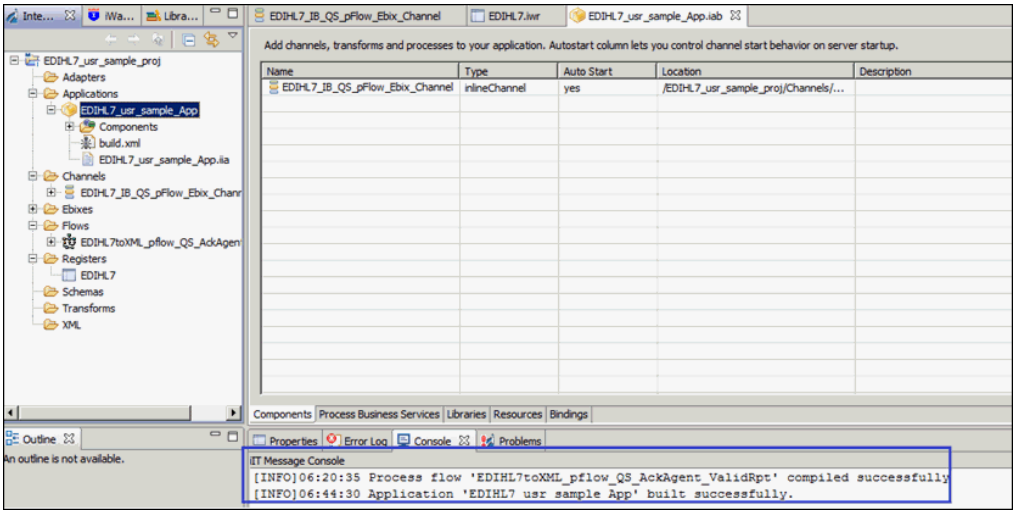
4. In the Bindings pane, add Bindings for Registers you have added in the XMLEDIHL7 Register Set to the iIA EDIHL7_usr_sample_App and click *Finish*, as shown in the following image.



After adding all register bindings, your iIT screen should resemble the following image.



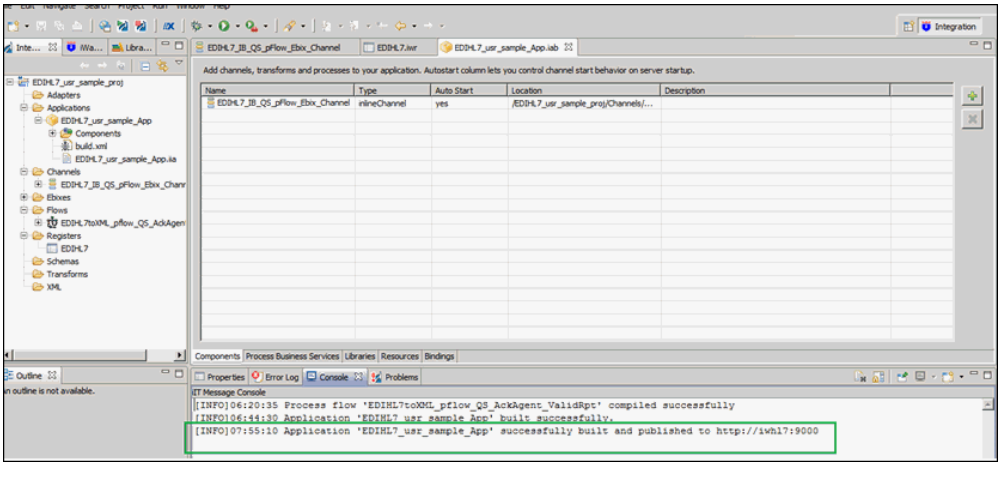
5. To build the application, right-click *EDIHL7_usr_sample_App*, select *Integration Tools*, and then click *Build* from the context menu, as shown in the following image.



6. To publish the iIA, right-click EDIHL7_usr_sample_App, select *Integration Tools*, and then click *Publish to* from the context menu.

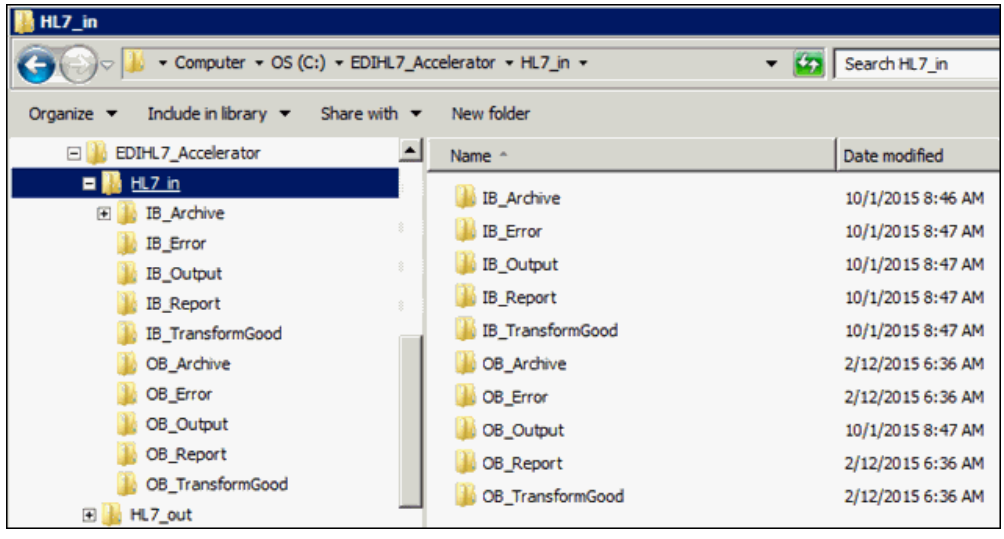
7. Provide the iSM server details in the Server URL text box and the other credentials, then click *Finish*.

   The iIT page should resemble the following image.



8. Deploy the iIA application on the iSM registry.

9. Have the following folder structure created before starting your application in the iSM console, as shown in the following image.

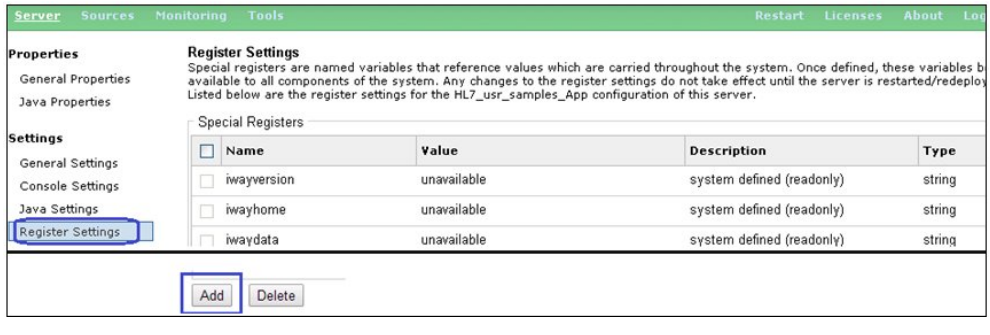


Setting HL7 System Registers

This section describes how to set system registers using the iWay Service Manager (iSM) Administration Console.

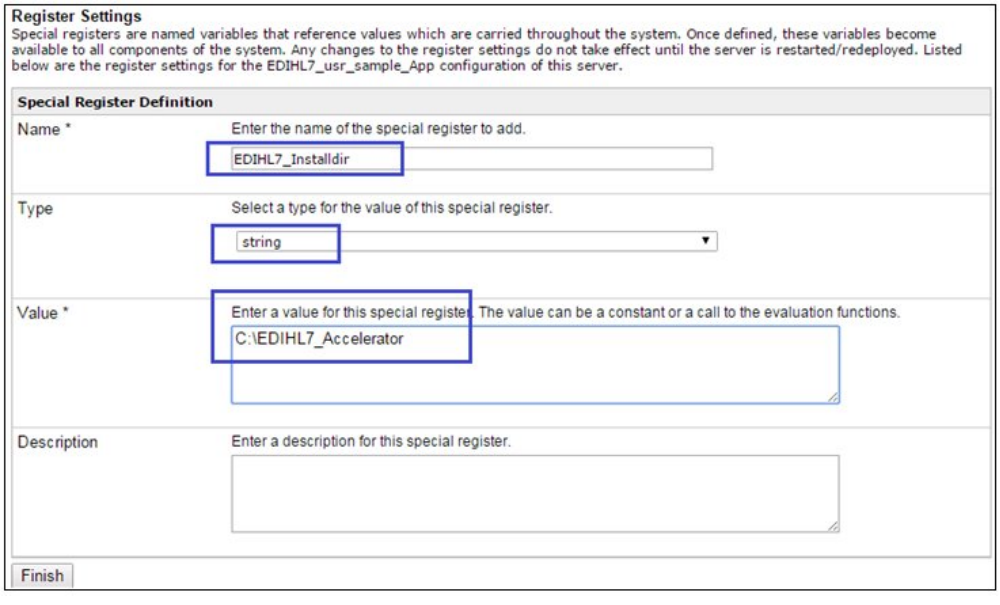
Procedure: How to Set System Registers

1. Open the iSM Administration Console and select *EDIHL7_usr_sample_App_IB [down]* from the Management drop-down list.
2. From the console bar, click *Server*, *Register Settings*, and then click *Add*, as shown in the following image.

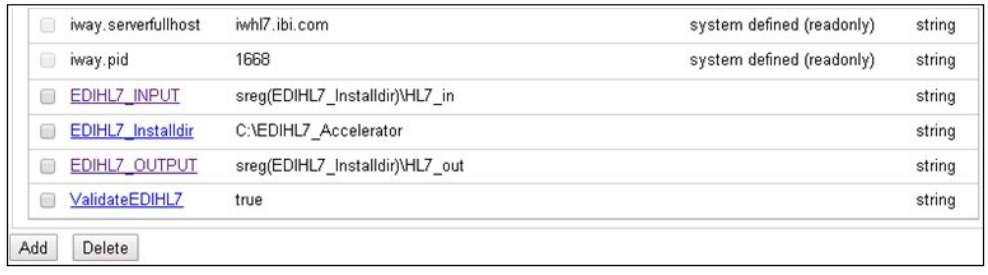


3. For the deployed EDIHL7_usr_sample_App_IB application, define the system registers as listed in the following table:

Register Name	Value
EDIHL7_INPUT	`sreg(EDIHL7_Installdir)\HL7_in`
EDIHL7_installdir	`C:\EDIHL7_Accelerator`
EDIHL7_OUTPUT	`sreg(EDIHL7_Installdir)\HL7_out`
ValidateEDIHL7	`true`



After adding the registers, your Register Settings page in the iSM Administration Console should resemble the following image.



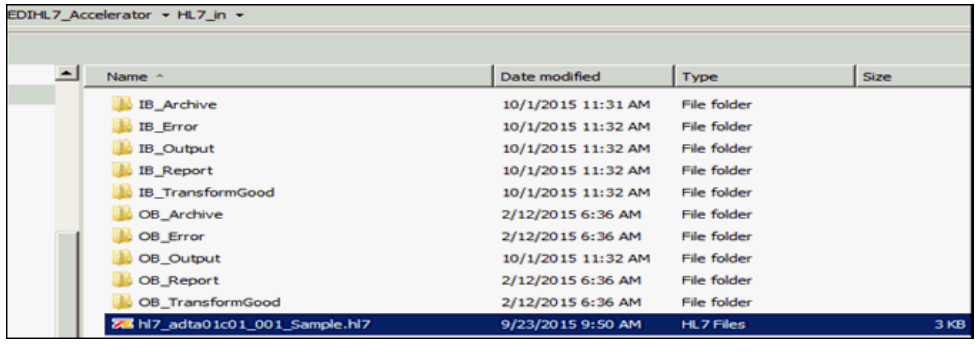
4. Start the deployed application in the iSM Administration Console and ensure that the channel is up and running in the Monitoring section of the console.

Testing the EDIHL7 Inbound Channel Application

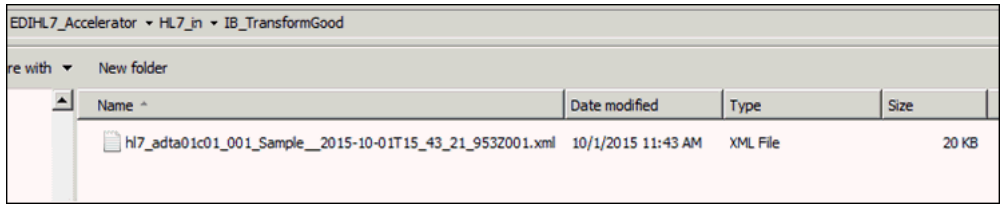
This section describes how to test the inbound channel application.

Procedure: How to Test the Inbound Channel Application

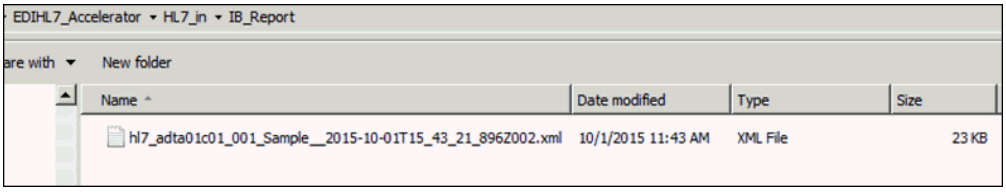
1. Place your input files under *EDIHL7_Accelerator\HL7_in*, as shown in the following image.



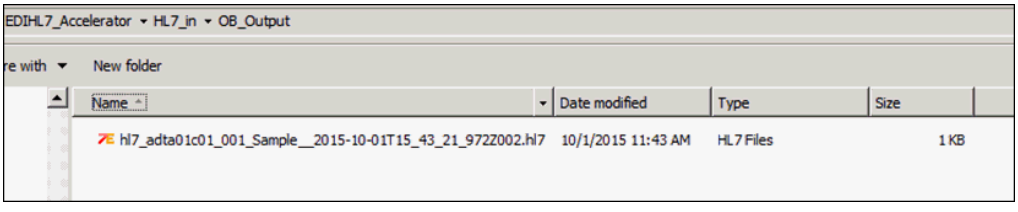
2. Observe the transformed XML output under *EDIHL7_Accelerator\HL7_in \IB_TransformGood*, as shown in the following image.



3. Monitor the reports under *EDIHL7_Accelerator\HL7_in\IB_report*, as shown in the following image.



4. Observe the Acknowledgement under *EDIHL7_Accelerator\HL7_in\OB_Output*, as shown in the following image.



5. If the input data contained any errors, you can review this error data in the output directory that you have configured for error handling (for example, EDIHL7_Accelerator\HL7_in \IB_Error).

6. After inbound processing has completed, a copy of the input data that was used during the transformation is stored under the archive directory that you specified (for example, EDIHL7_Accelerator\HL7_in\IB_Archive).

Chapter 6

Inbound Processing: HL7 to XML (Using MLLP)

This section describes how to configure a basic inbound message flow for the iWay Integration Solution for EDIHL7 using the Minimal Lower Layer Protocol (MLLP). The message flow represents the movement and tasks in the conversion of a message from HL7 format to XML format.

In this chapter:

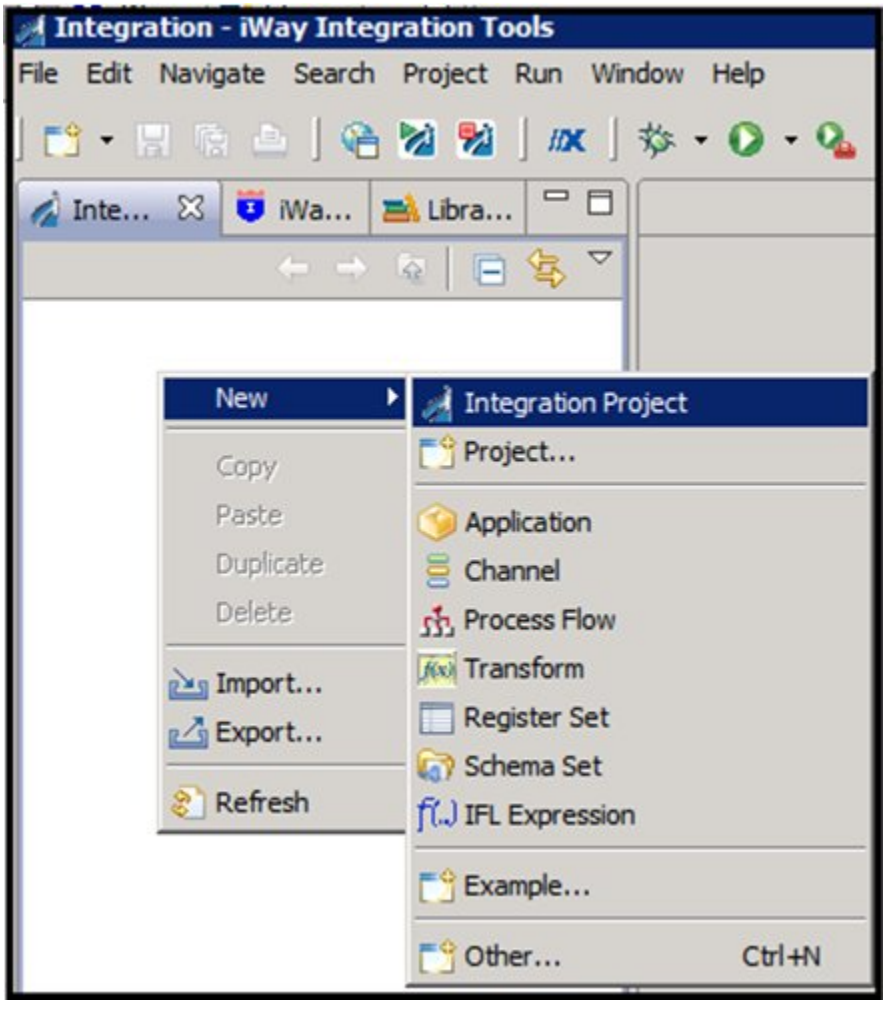
Configuring a Channel for HL7 Inbound Processing Using MLLP

The inbound channel creates an XML representation of a HL7 inbound message, and an acknowledgement message. The documents are routed to designated folders based on the success or failure results of the transformation and HL7 rule validation.

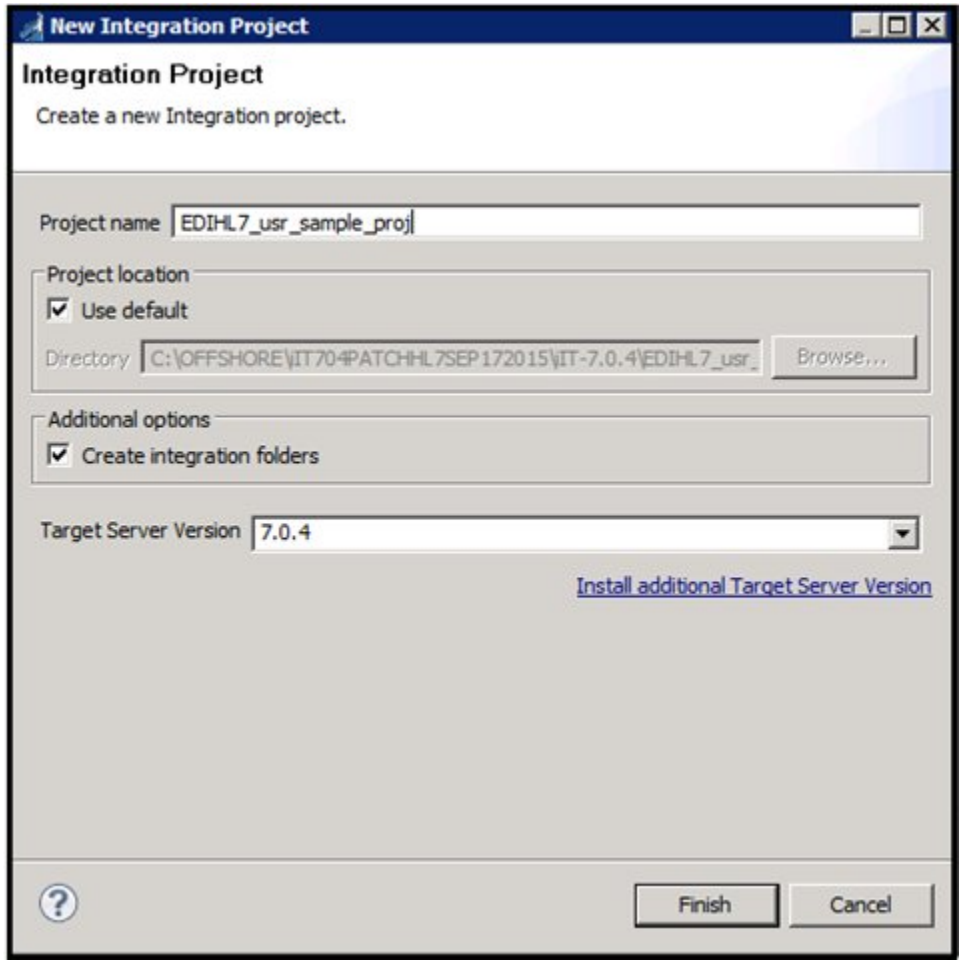
Procedure: How to Create a Channel for Inbound Processing

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.



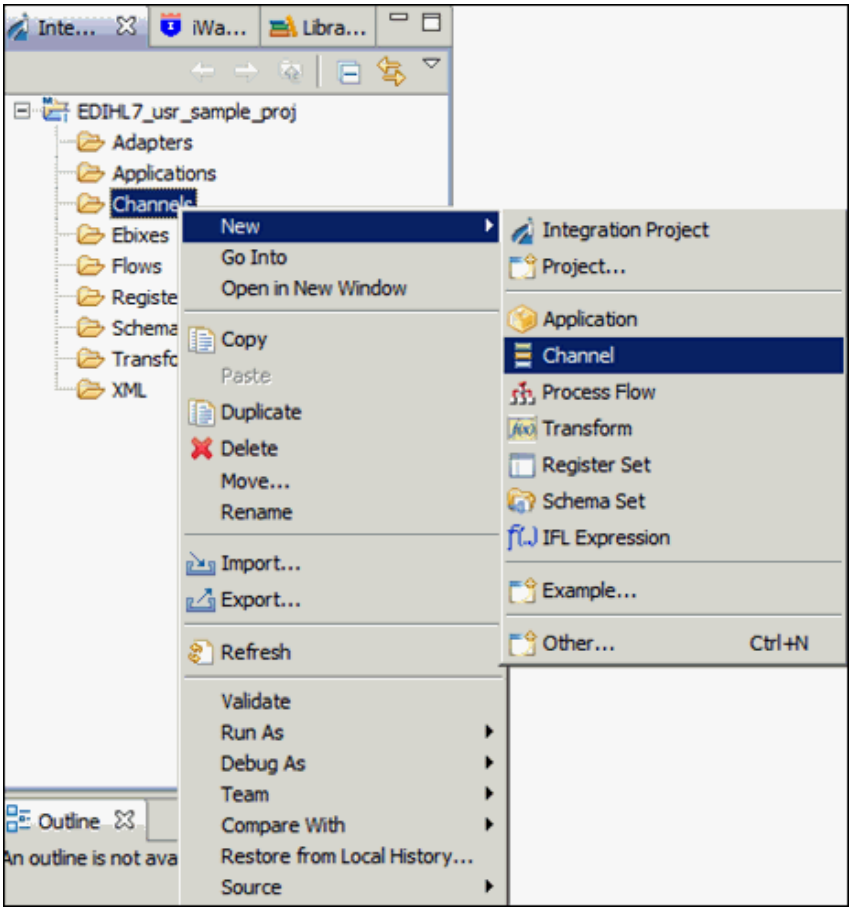
The New Integration Project dialog box opens, as shown in the following image.



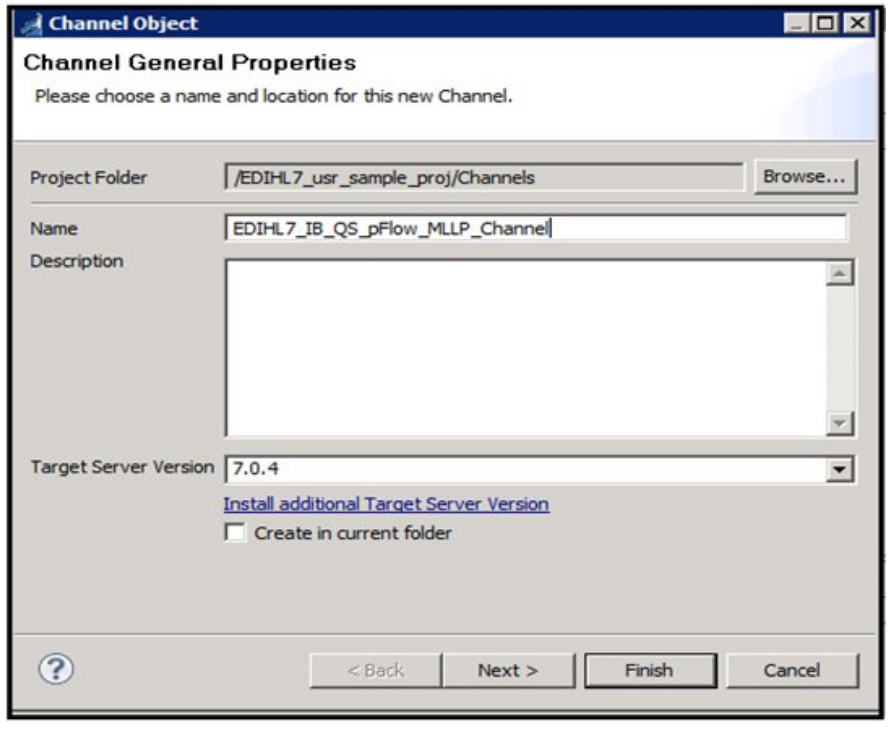
3. Enter a name for the Integration Project (for example, *EDIHL7_usr_sample_proj*) in the Project name field, and then click *Finish*.

   A new Integration Project node called *EDIHL7_usr_sample_proj* is added to the Integration Explorer.

4. Expand the *EDIHL7_usr_sample_proj* Integration Project node, right-click the *Channels* folder, select *New*, and then click *Channel* from the context menu, as shown in the following image.

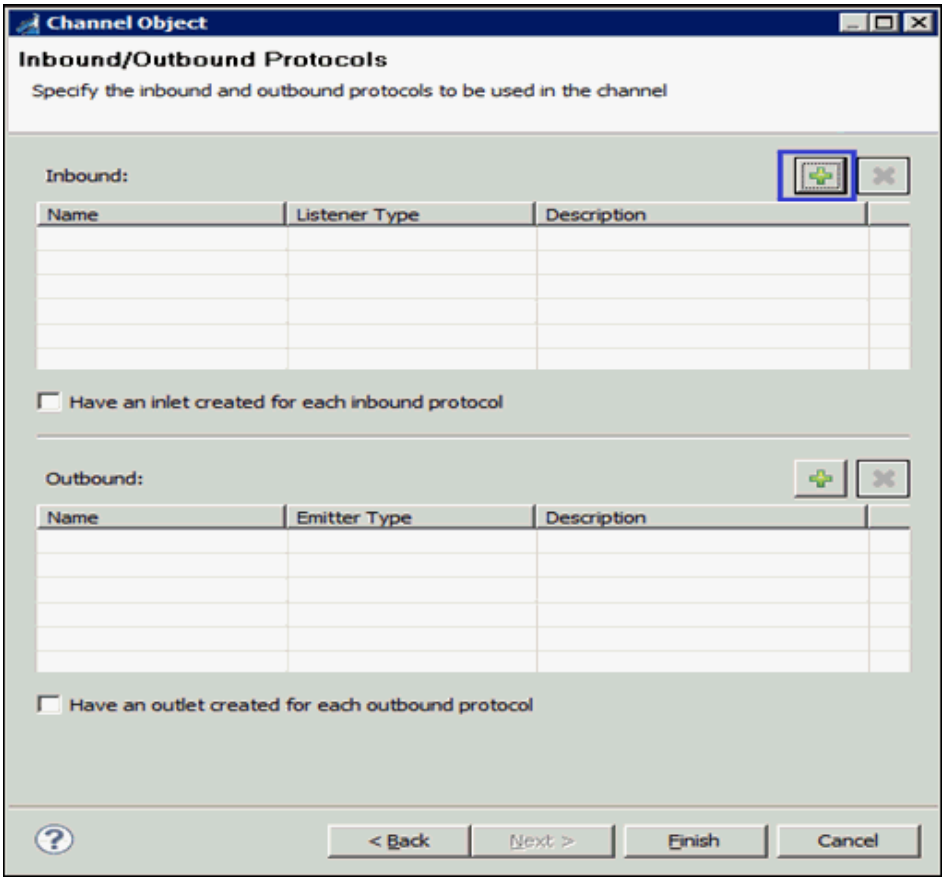


5. Enter a name for the Channel (for example, XMLEDIHL7_IB_QS_pFlow_MLLP_Channel), and then click *Next*, as shown in the following image.

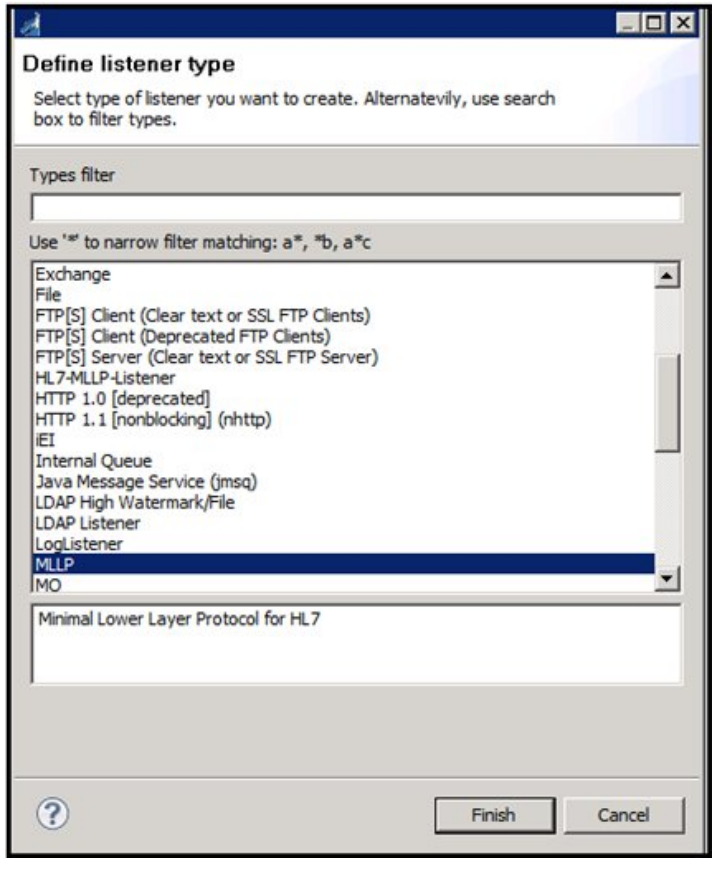


The Inbound/Outbound Protocols pane appears.

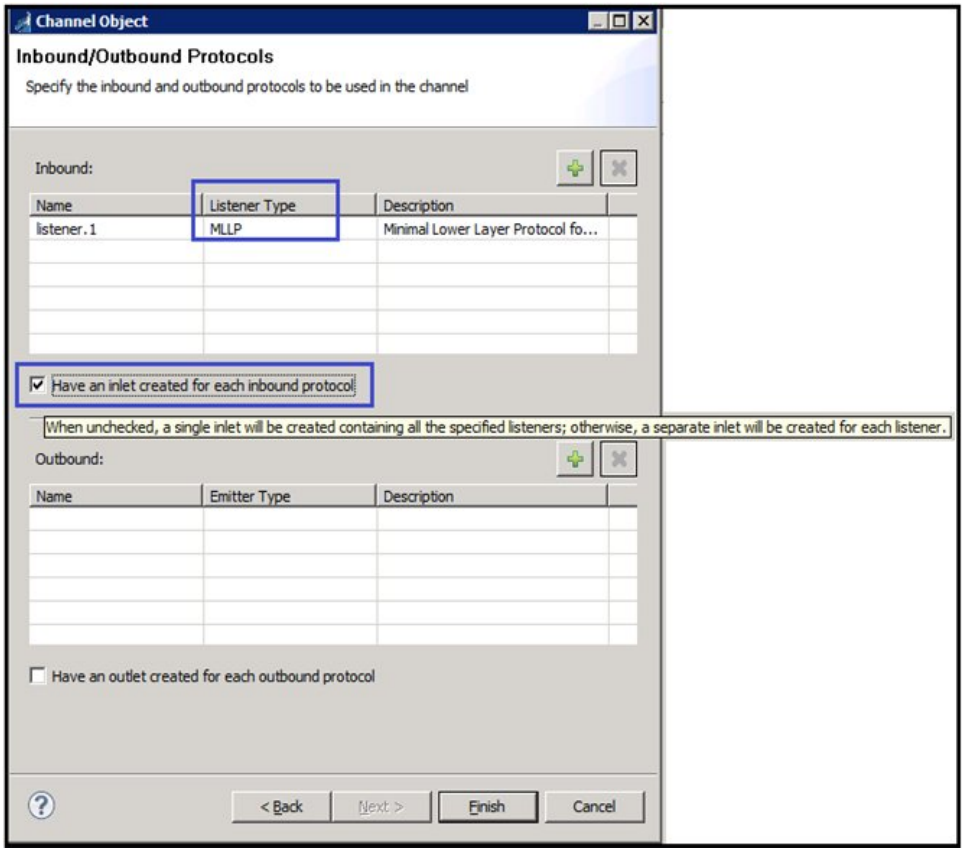
6. Click the *Add* button to add a File Listener in the Inbound section, as shown in the following image.



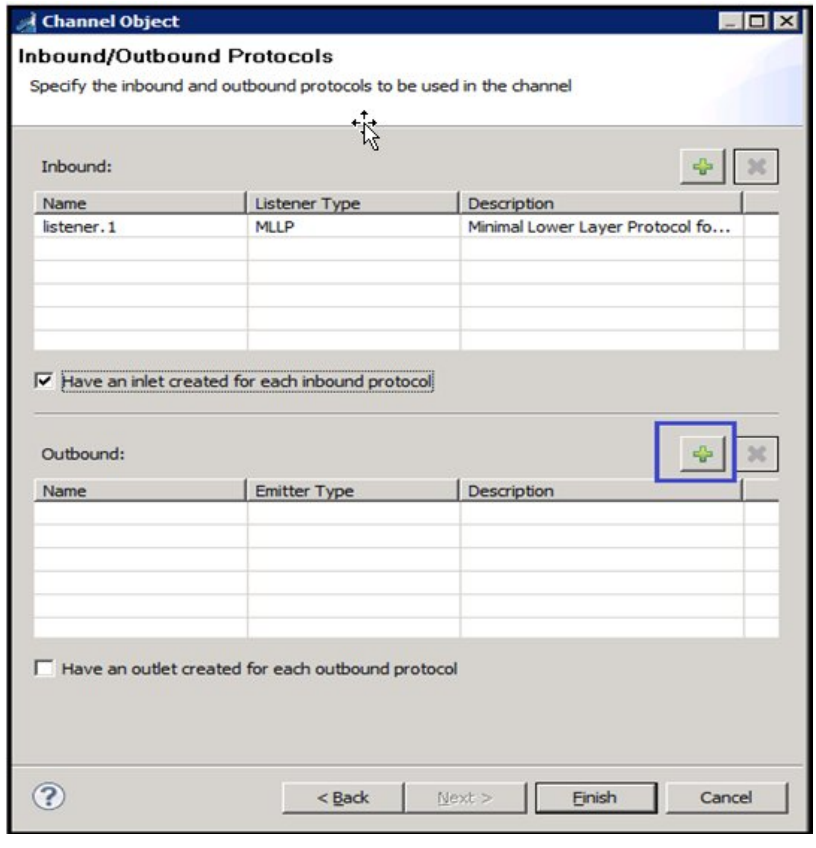
7. From the Types Filter list, select *MLLP* and then click *Finish*, as shown in the following image.



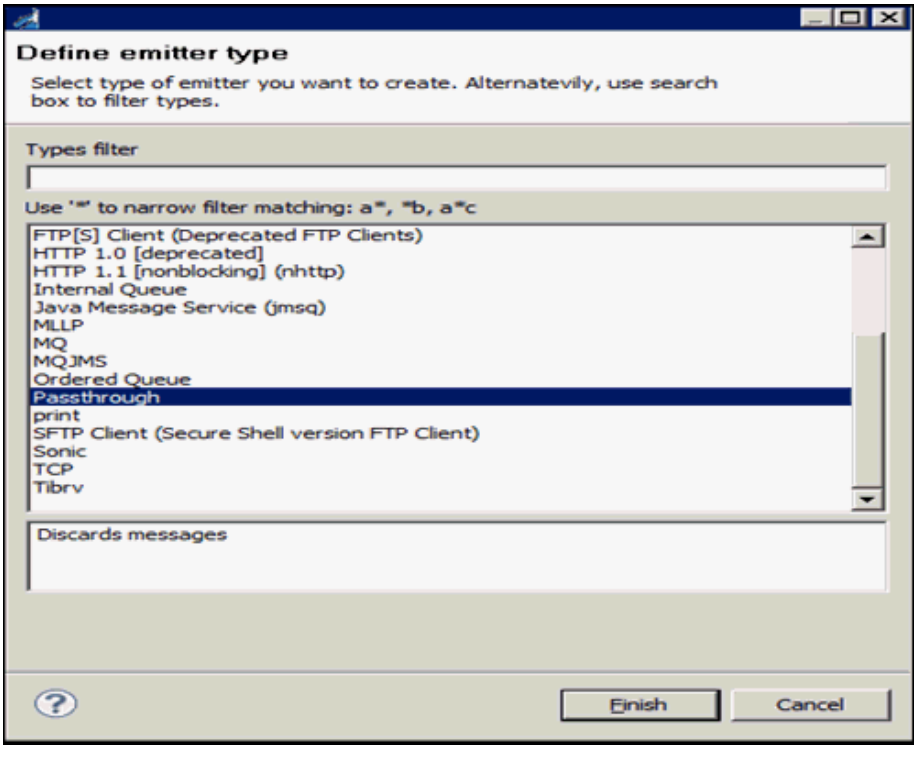
8. Select the *Have an Inlet created for each inbound protocol* check box to create an Inlet for the channel, as shown in the following image.



9. Click the *Add* button in the Outbound section to define an emitter, as shown in the following image.

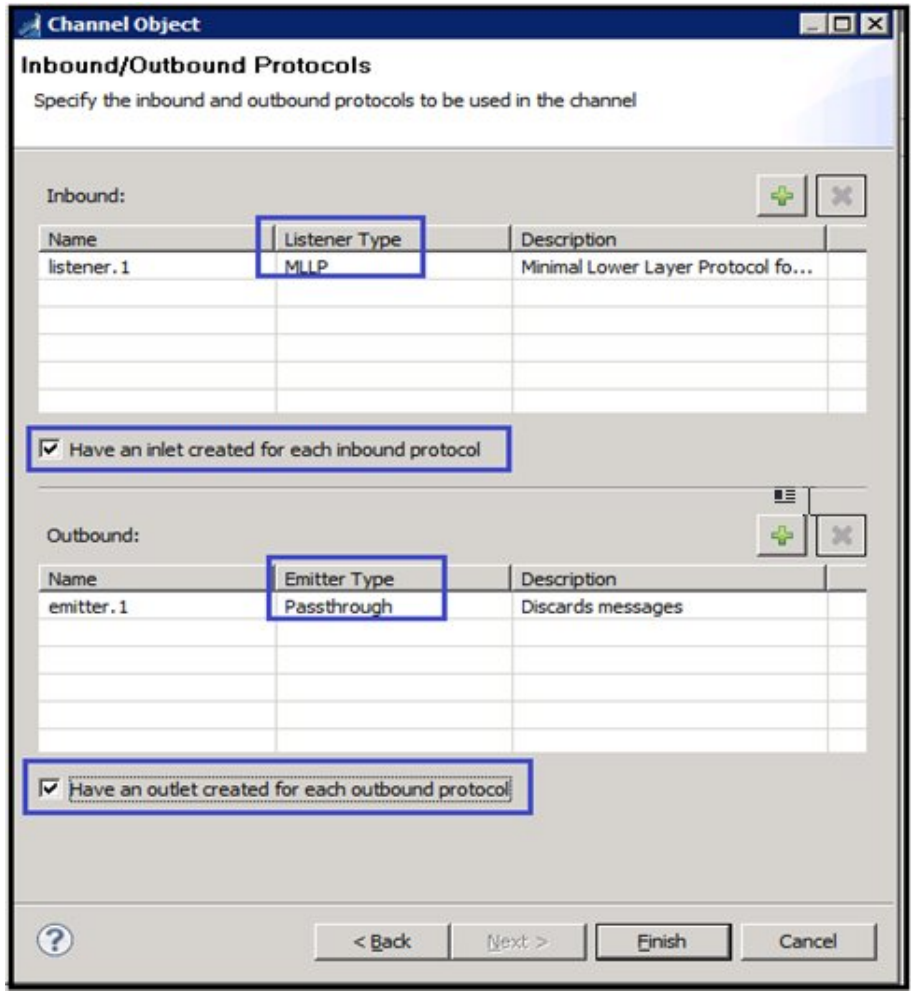


10. In the Types filter list, select *Passthrough* to define the passthrough emitter, and then click *Finish*, as shown in the following image.

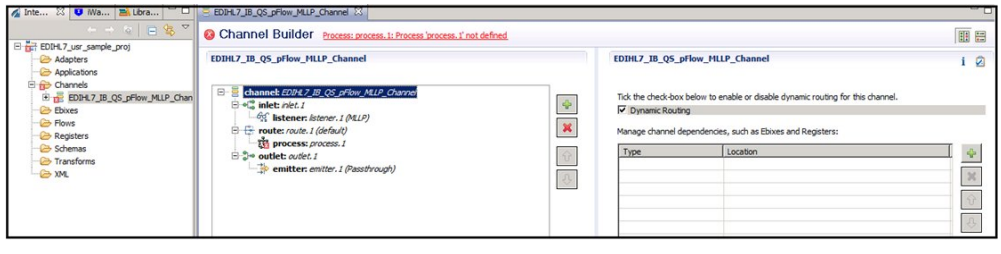


Note: Passthrough does not emit data from the listener. Instead, it just passes the control here and does nothing.

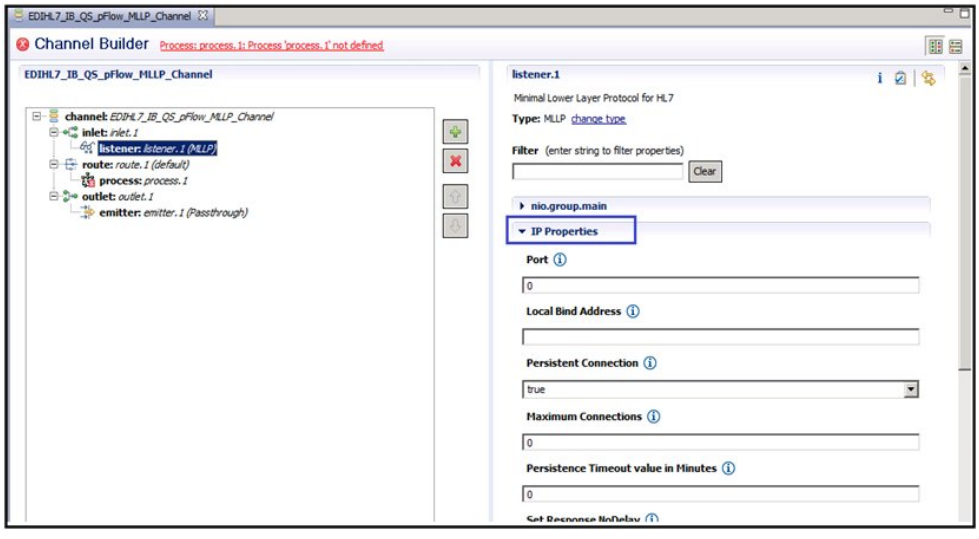
11. Select the *Have an outlet created for each outbound protocol* check box to create an outlet for the channel, and then click *Finish*, as shown in the following image.



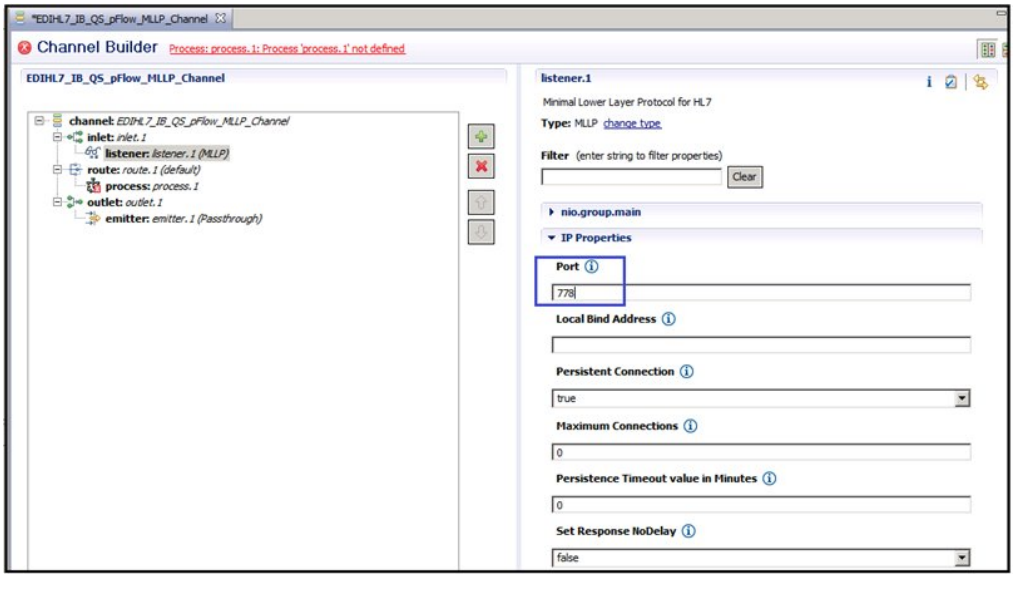
The Channel Builder pane appears.



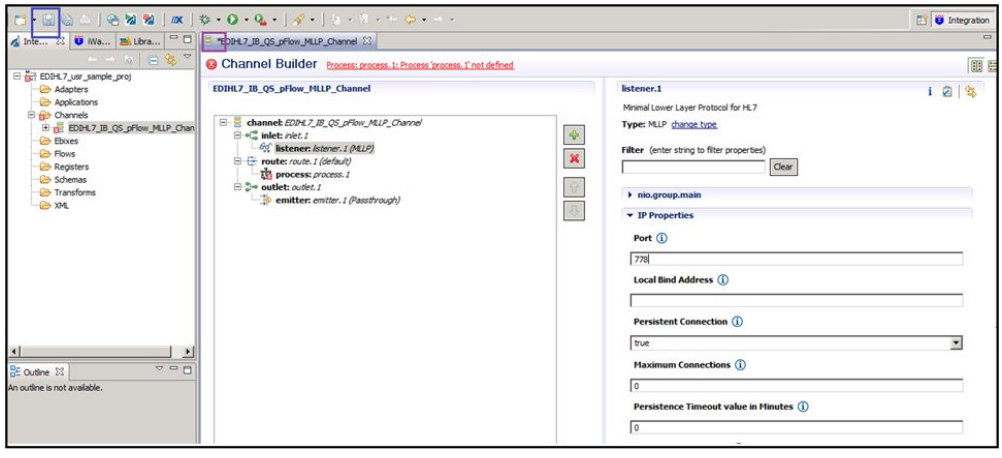
12. Under *inlet: inlet.1*, click *listener: listener.1 (MLLP)* and then expand the *IP Properties* configuration parameter on the right pane, as shown in the following image.



13. Enter a valid Port number (for example, 778) to listen for the input HL7 file, as shown in the following image.

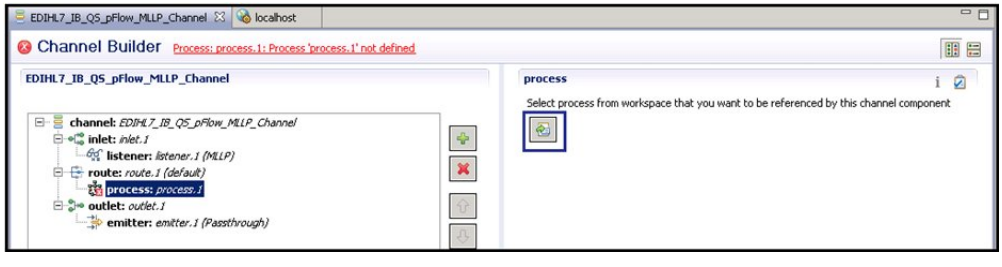


14. Click the *Save* icon near the File menu to save your edited listener configuration. You can also press the shortcut key (Ctrl+S) to save your work if you are using a Windows environment.

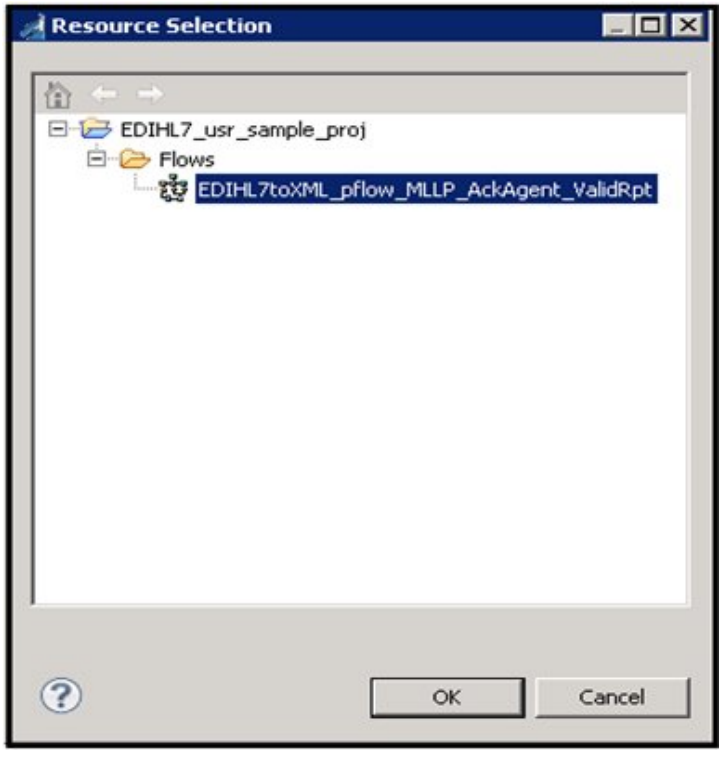


15. Import the *XMLToEDIHL7_Ebix_2_MLLP_AckAgent_ValidRpt* Process Flow from the local drive or create the one in the EDIHL7_usr_sample_proj directory in the Flows folder.

16. Under the *route: route.1(default)* node, select *process: process.1* and click the process icon on the right panel to reference the process flow into your channel, as shown in the following image.



17. Select a process flow from the integration project and then click *OK*.

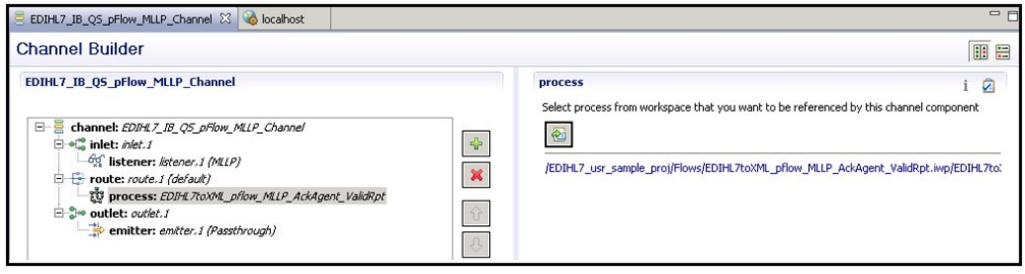


Note: Process flows should already be built and available in the iIT integration project. They can be found in the EDIHL7_usr_sample_proj directory inside the Flows folder.

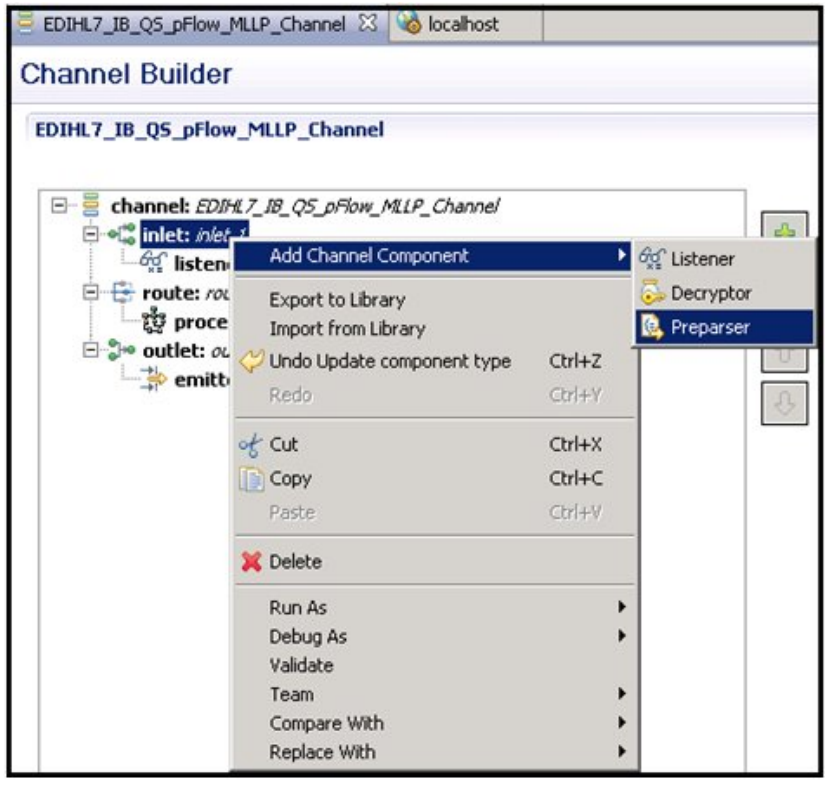
For more information, see *Process Flow Used for Inbound Processing: HL7 to XML (MLLP)* on page 123.

Your screen should resemble the following image.

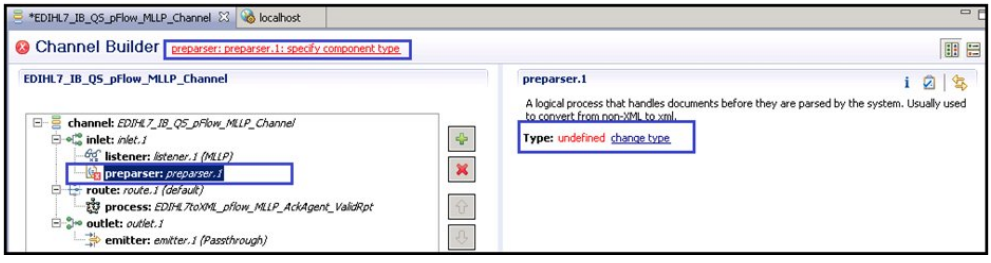
18. Click the *Save* icon near the File menu to save your edited listener configuration. You can also press the shortcut key (Ctrl+S) to save your work if you are using a Windows environment.



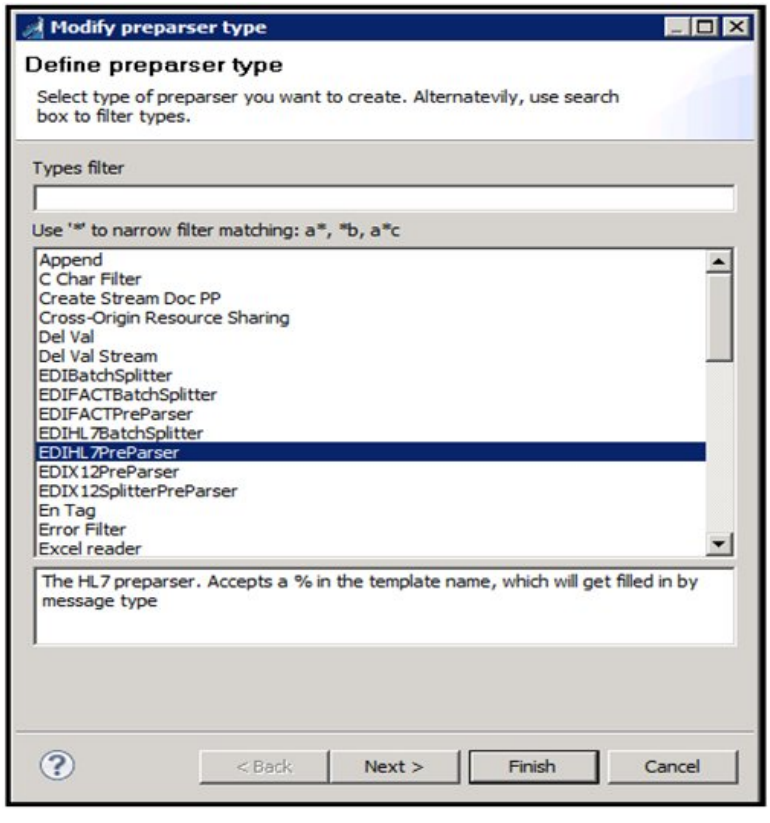
19. Right-click *inlet: inlet.1*, select *Add Channel Component*, and then click *Preparser* from the context menu, as shown in the following image.



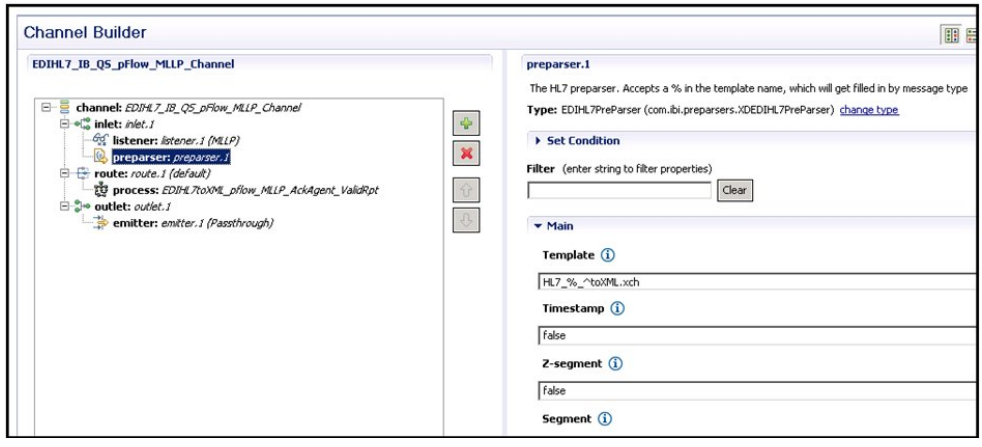
You iIT screen should resemble the following image.



20. In the preparser.1 configuration pane, click the *change type* link and select *EDIHL7Preparser* from the Types filter list and click on *Finish*, as shown in the following image.



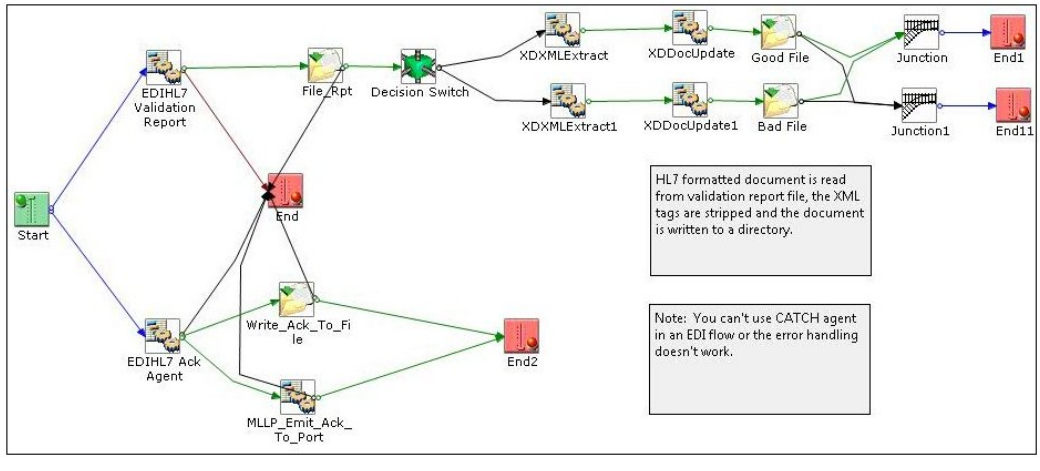
The following image shows the iIT screen.



Reference: Process Flow Used for Inbound Processing: HL7 to XML (MLLP)

This section provides an overview of the process flow used for inbound processing: HL7 to XML (MLLP). This process flow (XMLToEDIHL7_Ebix_2_MLLP_AckAgent_ValidRpt) is already built and available in the iIT integration project. It is located under the *EDIHL7_usr_sample_proj* node inside the Flows subfolder.

The following image shows the entire inbound process flow, including all of the nodes that are used and their connections.

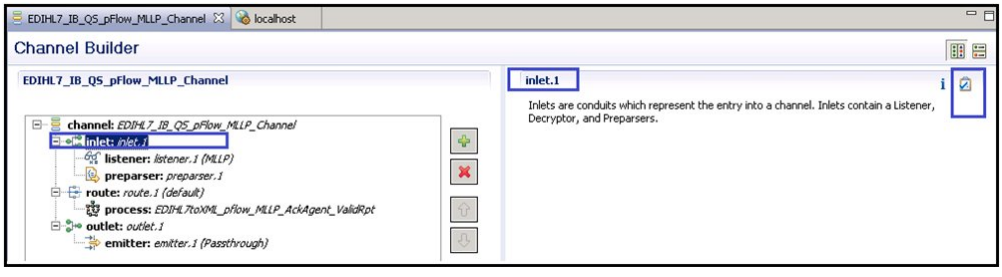


In this process flow, an HL7 formatted document is read from a validation report file. The XML tags are stripped and the document is written to a directory.

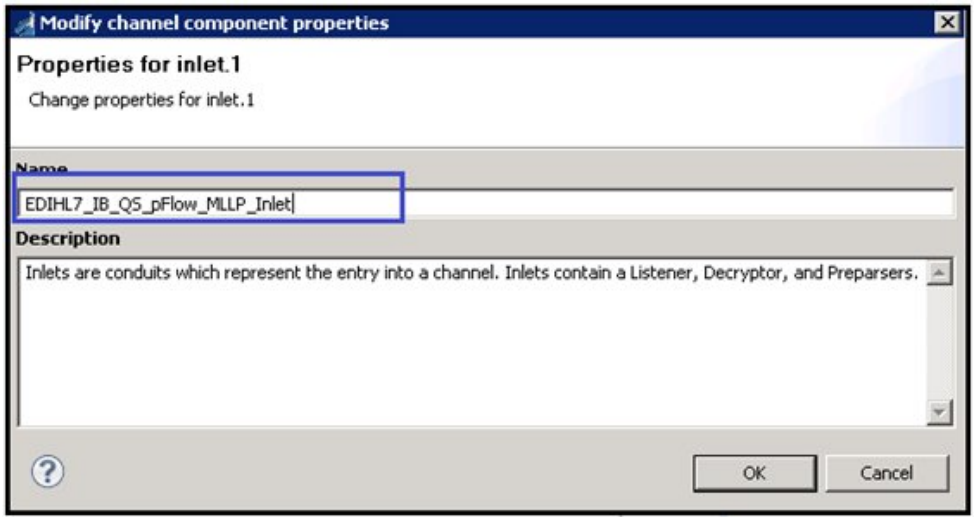
Note: Using a Catch service (com.ibi.agents.XDCatchAgent) in an EDI flow is not supported. The error handling does not work as a result.

Procedure: How to Update Basic Details for Channel Components

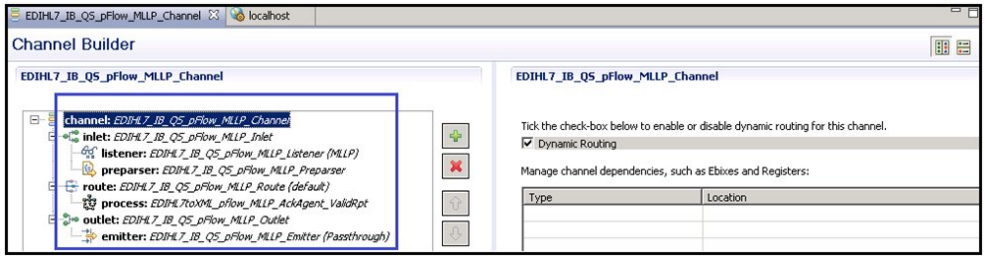
1. Select the channel component *inlet:inlet.1* from the channel and click the update channel component icon on the top right side panel, as shown in the following image.



2. Rename or update the inlet details (for example, EDIHL7_IB_QS_pFlow_MLLP_Inlet), and click *OK*, as shown in the following image.



After renaming or changing the channel components, your iIT Channel Builder should resemble the following image.

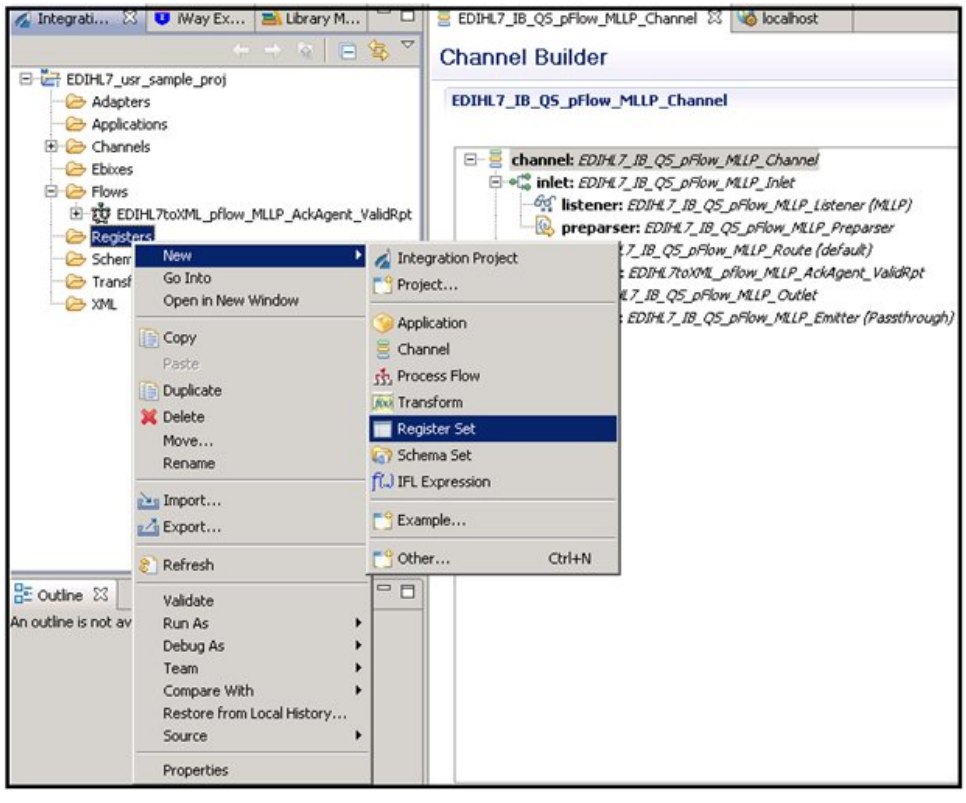


Configuring Register Sets and Registers

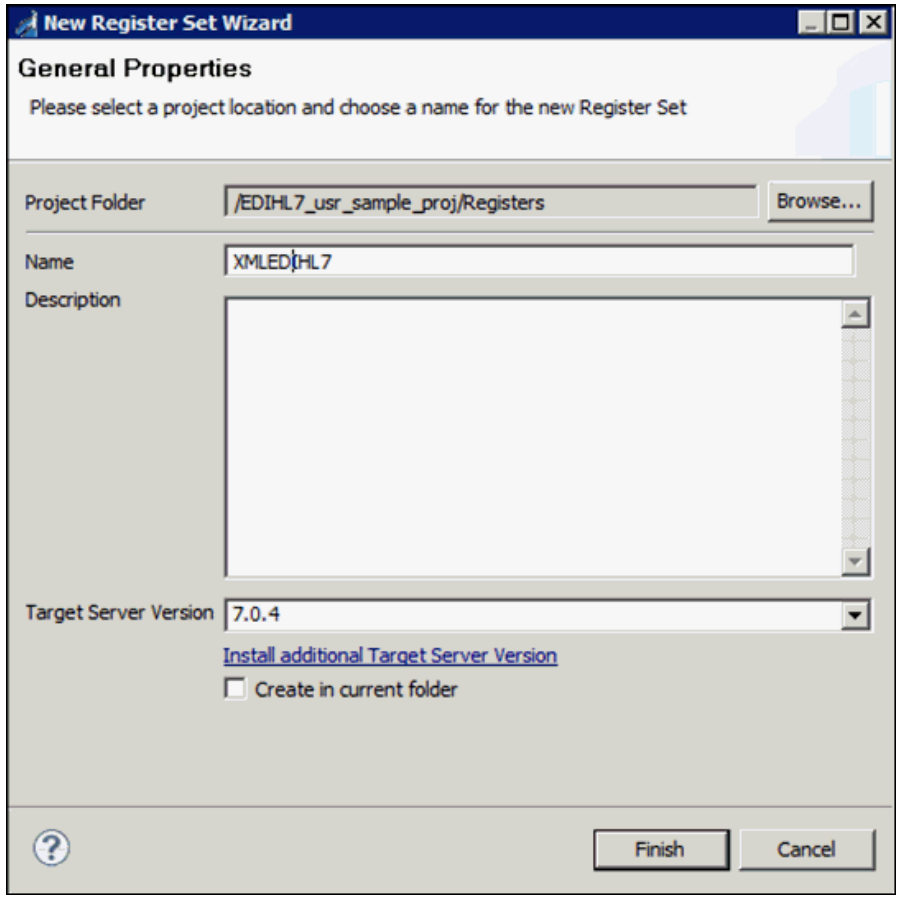
This section describes how to configure register sets and registers using iWay Integration Tools (iIT).

Procedure: How to Configure Register Sets and Registers

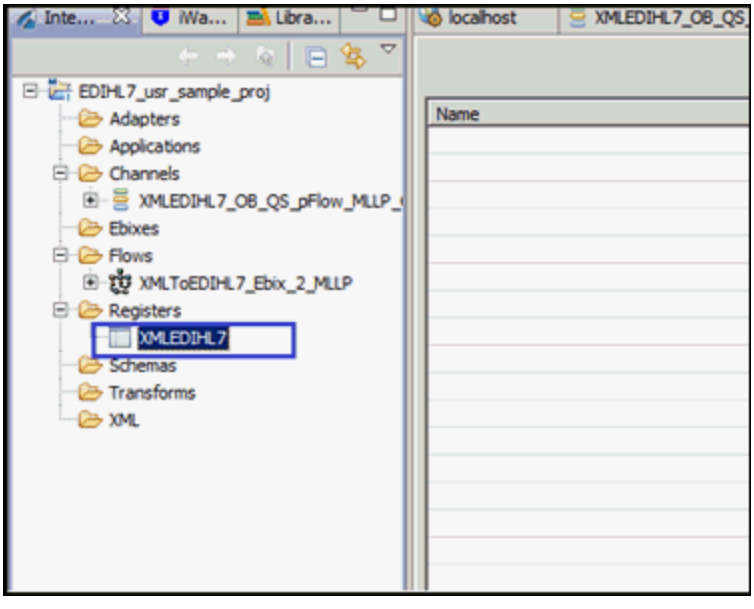
1. In the EDIHL7_usr_sample_proj project tree, right-click the Registers folder, and select *New*, and then click *Register Set* from the context menu, as shown in the following image.



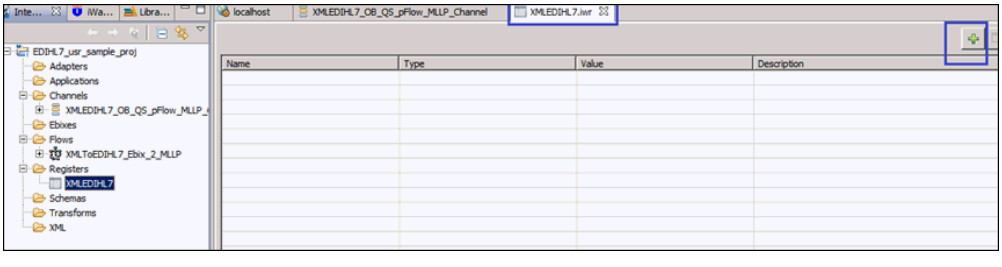
2. In the Name field, enter a name for the register set and click *Finish*, as shown in the following image.



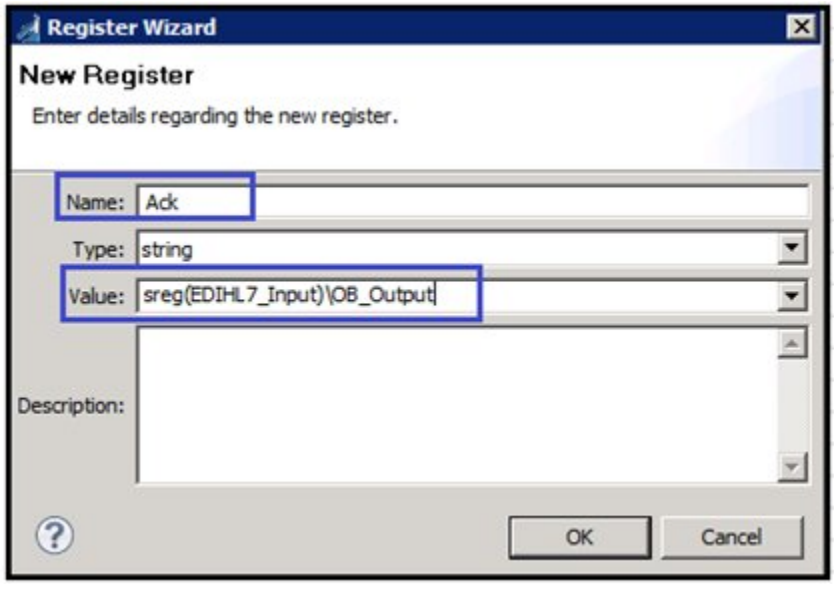
The new register set appears under Registers in the Registers folder, as shown in the following image.



3. Click the *Add a property* icon to add a register to the register set.



4. Enter a name for the new register in the Name field, select a register type from the Type drop-down list (set to string by default), and then enter a value in the Value field, as shown in the following image.

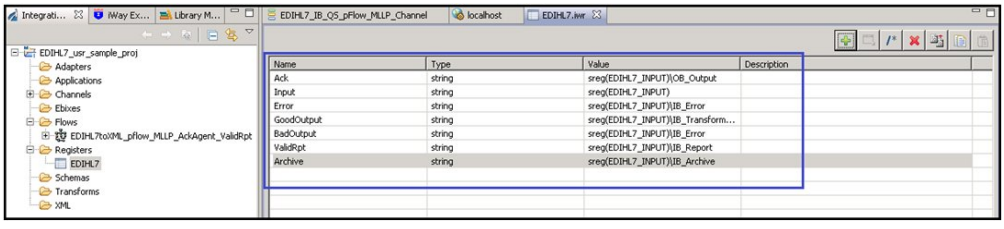


5. Click *OK*.

6. Create the following registers under the XMLEDIHL7 registers set you just created, along with the values shown in the table below:

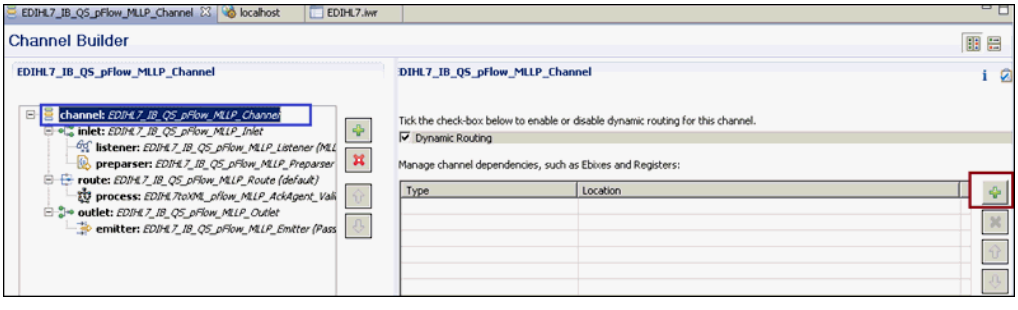
Register Name	Value
Ack	Name=Ack, Value=sreg(EDIHL7_INPUT)\OB_OUTPUT
Input	Name=Input, Value=sreg(EDIHL7_INPUT)
Error	Name=Error, Value=sreg(EDIHL7_INPUT)\IB_Error
GoodOutput	Name=Output, Value=sreg(EDIHL7_INPUT)\IB_TransformGood
BadOutput	Name=GoodOutput, Value=sreg(EDIHL7_INPUT)\IB_Error
ValidRpt	Name=ValidRpt, Value=sreg(EDIHL7_INPUT)\IB_Report

Register Name	Value
Archive	Name=Archive, Value=sreg(EDIHL7_INPUT)\IB_Archive

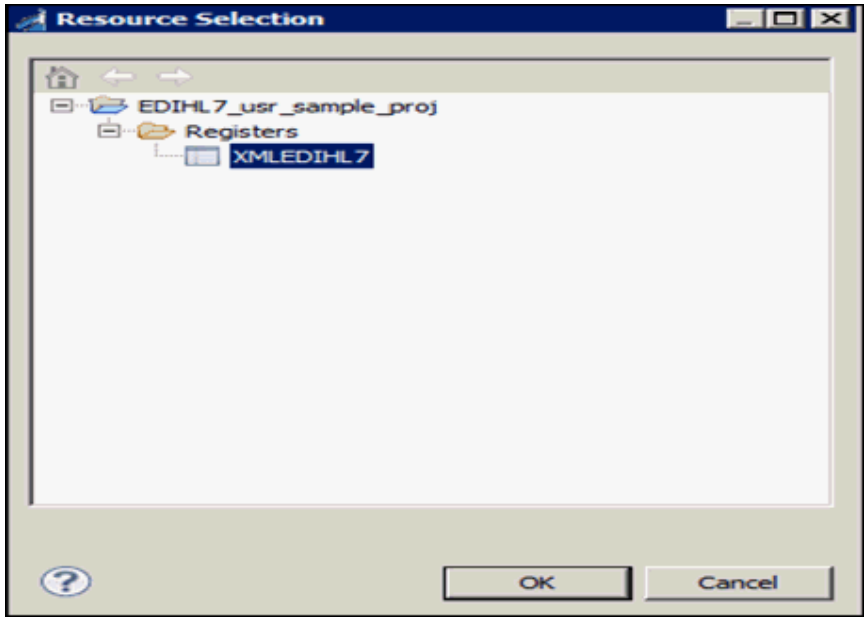


Procedure: How to Add a Register Set to an Inbound Channel as a Dependency

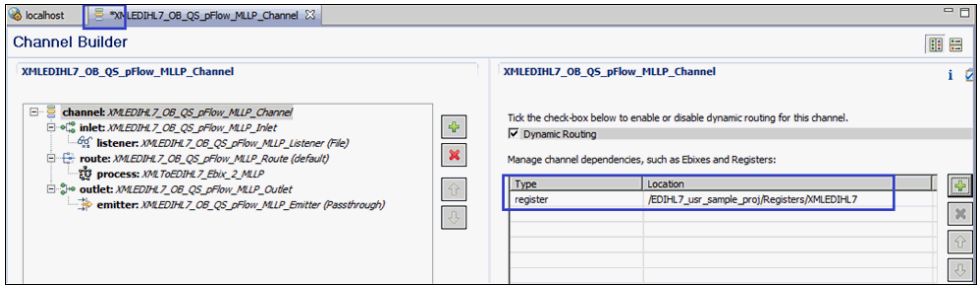
1. Click the *channel: XMLEDIHL7_IB_QS_pFlow_MLLP_Channel* node and then click the Add dependency icon on the far right side panel of the channel properties pane, as shown in the following image.



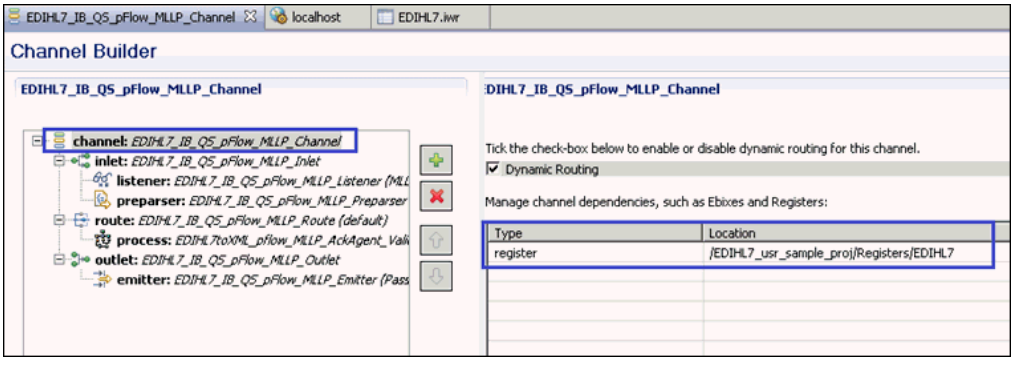
2. From the Registers folder under the integration folder, select *XMLEDIHL7* and click *OK*.



The iIT page should resemble the following image.



3. Click the *Save* icon to save your changes. You can also use the keyboard shortcut (Ctrl+S) if you are using a Windows environment.

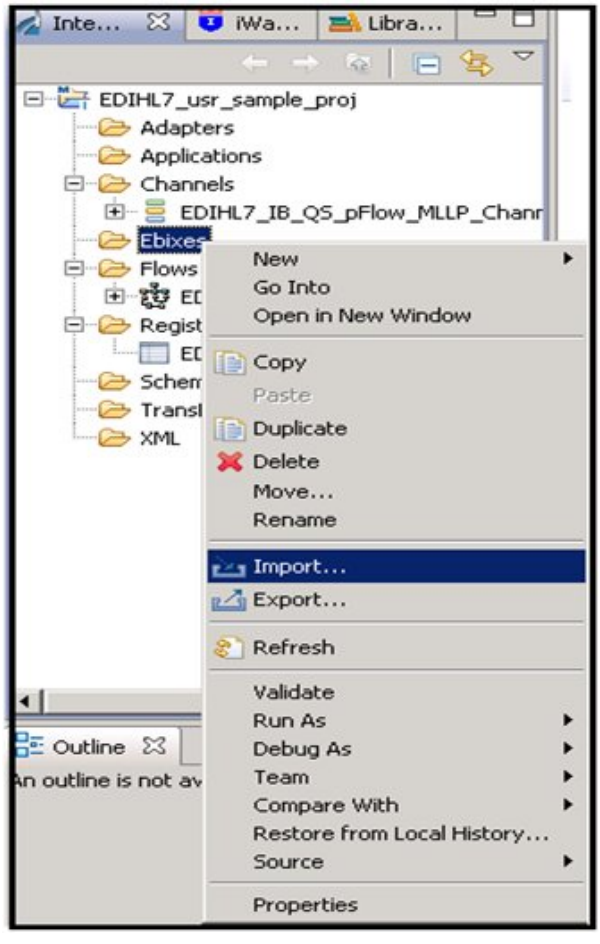


Importing an Ebix Into the Workspace

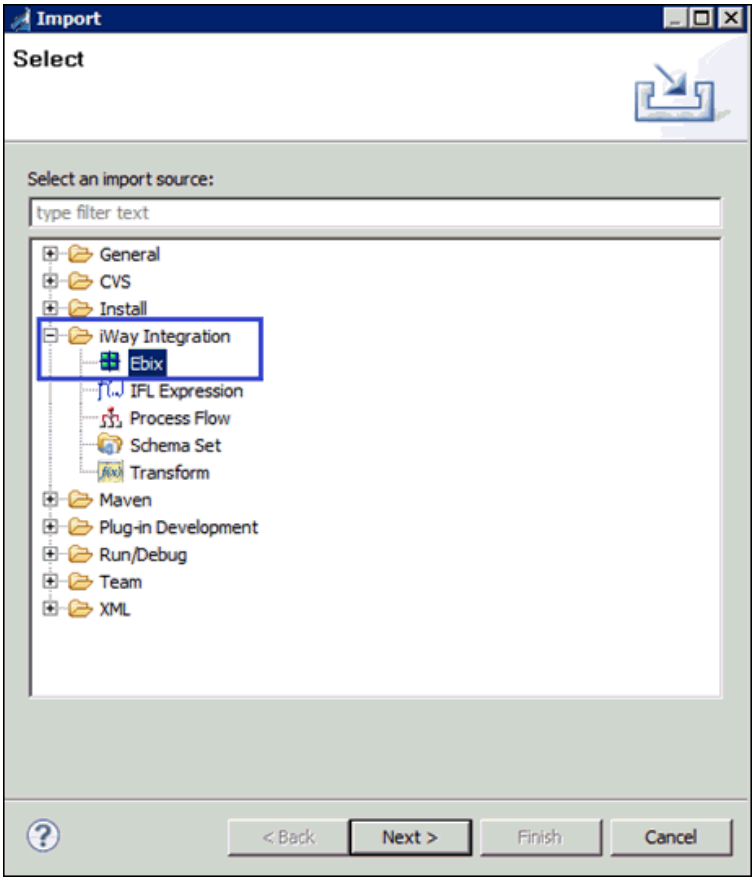
This section describes how to import an Ebix into the workspace using iWay Integration Tools (iIT).

Procedure: How to Import an Ebix Into the Workspace

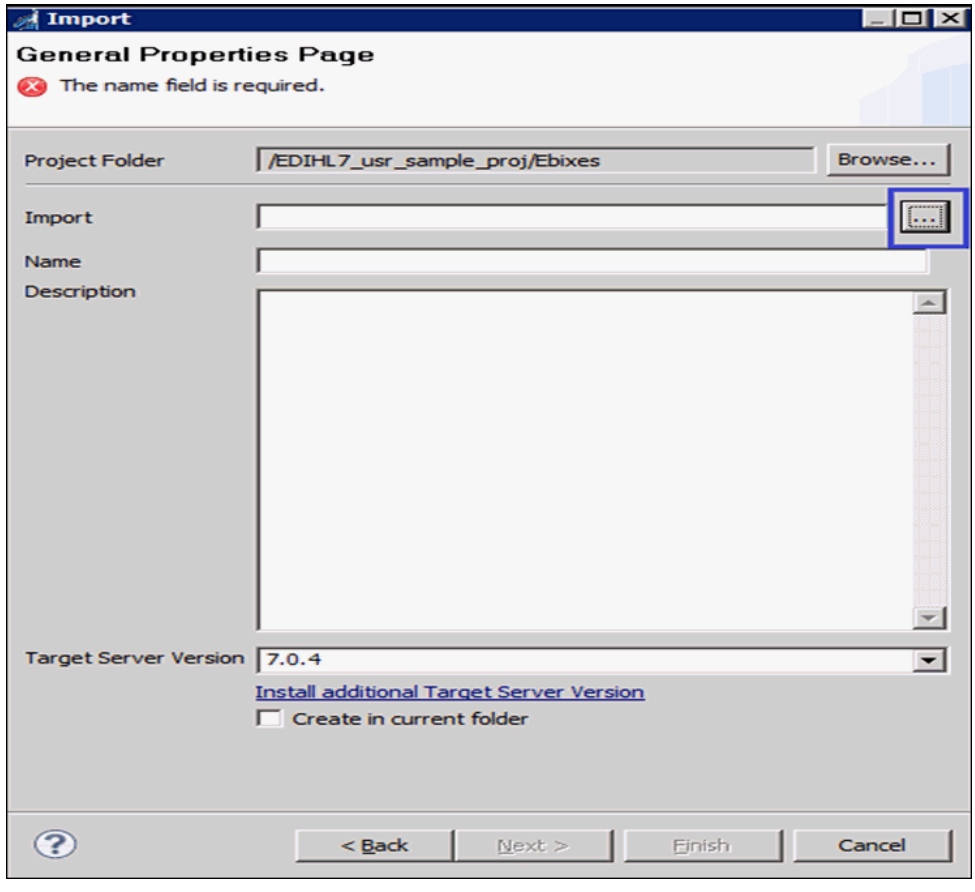
1. In the integration project EDIHL7_usr_sample_proj, right-click the Ebixes folder and then select *Import* from the context menu, as shown in the following image.



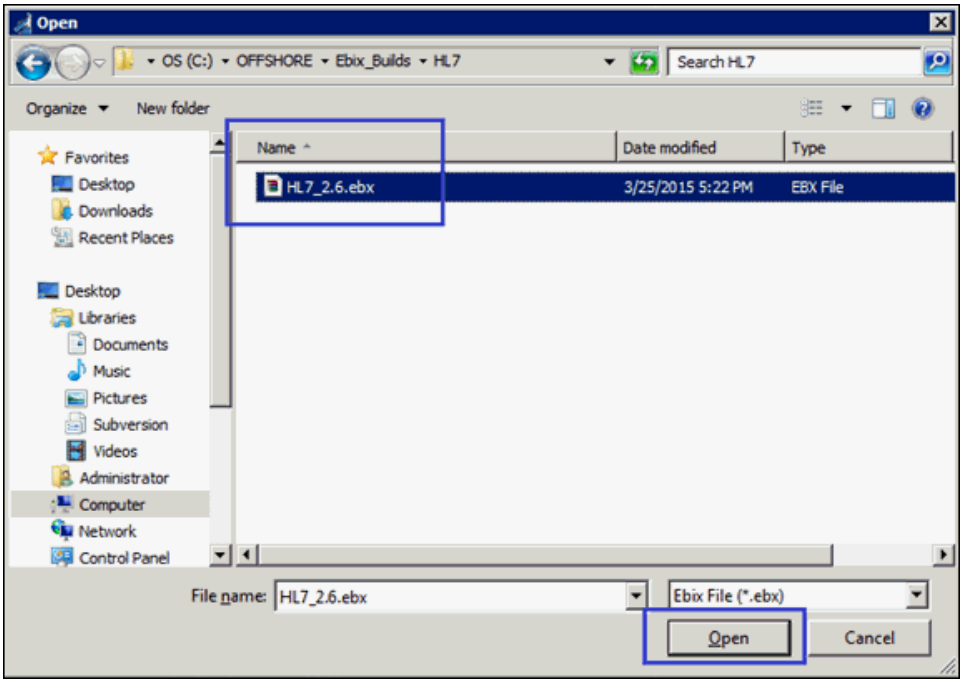
2. Expand the iWay Integration folder, select *Ebix*, and then click *Next*, as shown in the following image.



3. Click the ellipses (...) button to browse and import the ebix from a specific folder location, as shown in the following image.

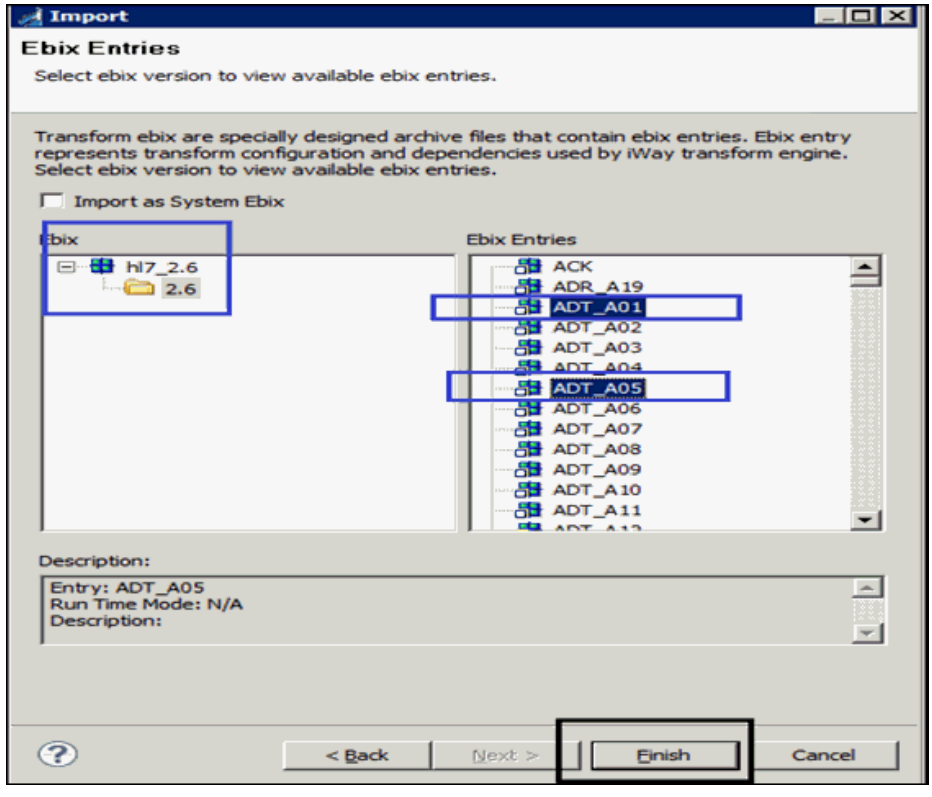


4. Select *HL7_2.6 ebix* from the folder location and click *Open*, as shown in the following image.

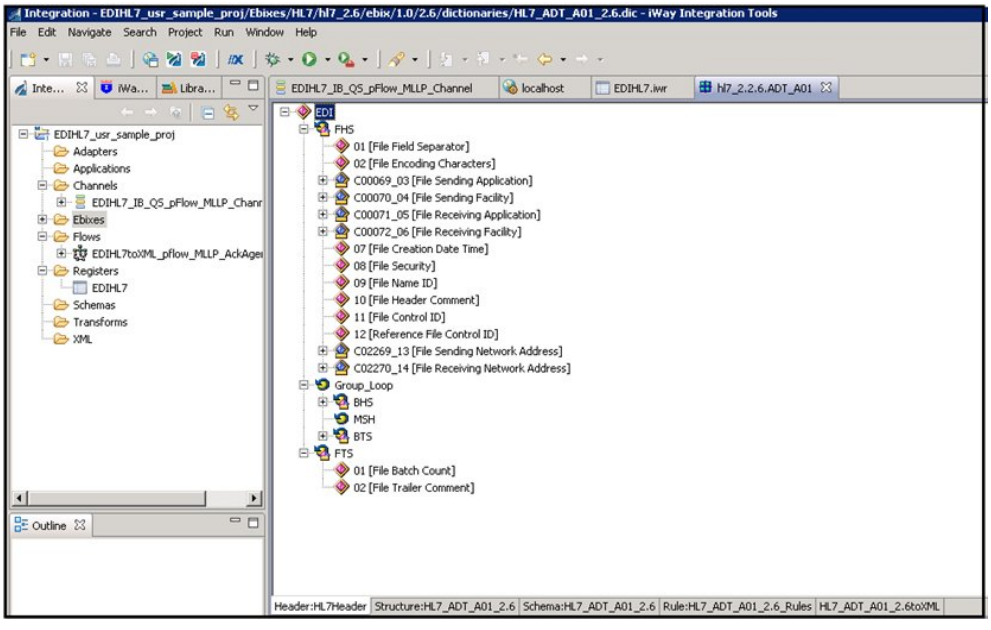


5. In the Import wizard, click *Next*.
6. Expand the hl7_2.6 ebix from the left panel and select the 2.6 folder.

7. Select the messages you wish from the right side panel and click *Finish*, as shown in the following image.

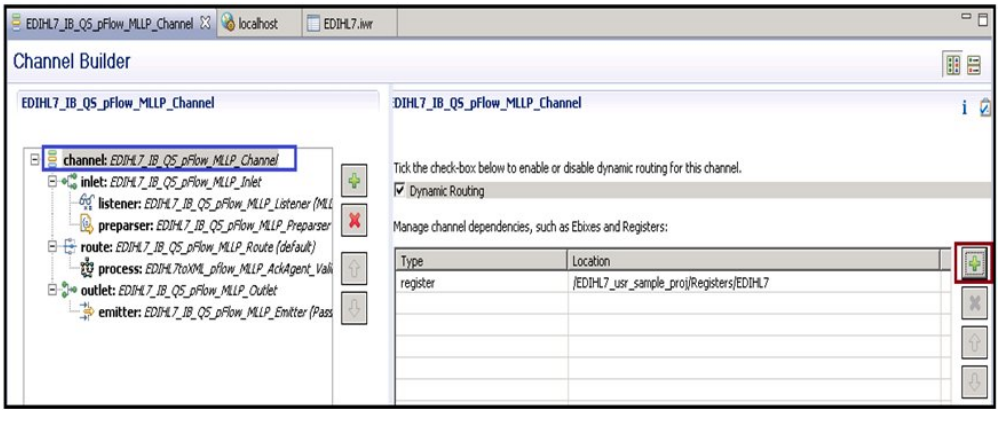


The iIT page should resemble the following image.

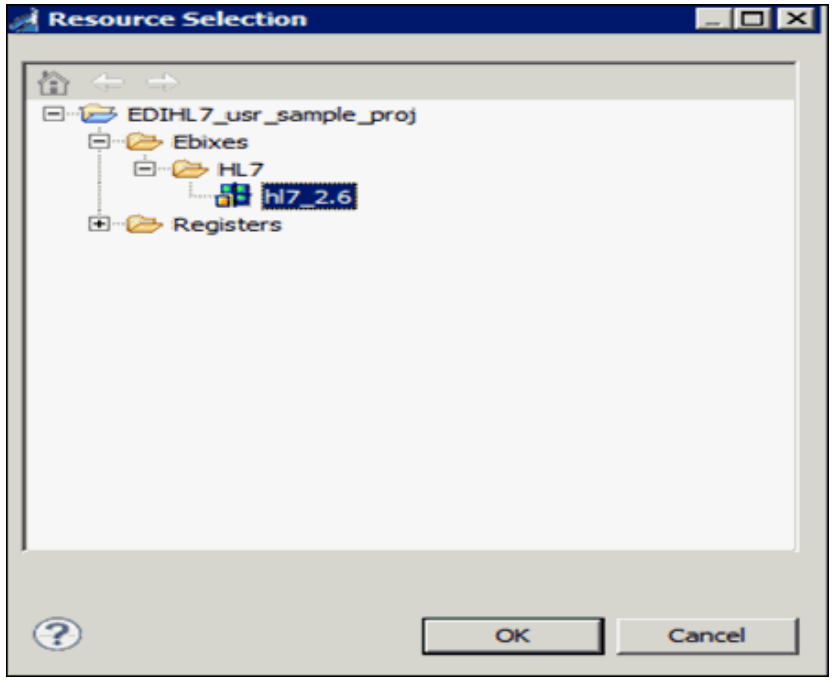


Procedure: How to Add an Ebix to an Inbound Channel as a Dependency

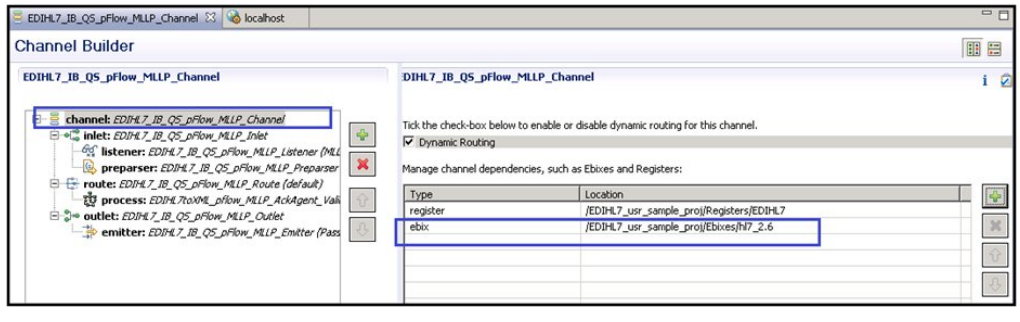
1. Click the *channel: XMLEDIHL7_IB_QS_pFlow_MLLP_Channel* node and then click the *Add dependency* icon on the right side panel, as shown in the following image.



2. Expand *EDIHL7_usr_sample_proj*, *Ebixes*, *HL7*, and then select ebix *hl7_2.6* and click *OK*, as shown in the following image.



3. Click the *Save* button to save your changes or press the keyboard shortcut (Ctrl+S) if you are using a Windows environment.

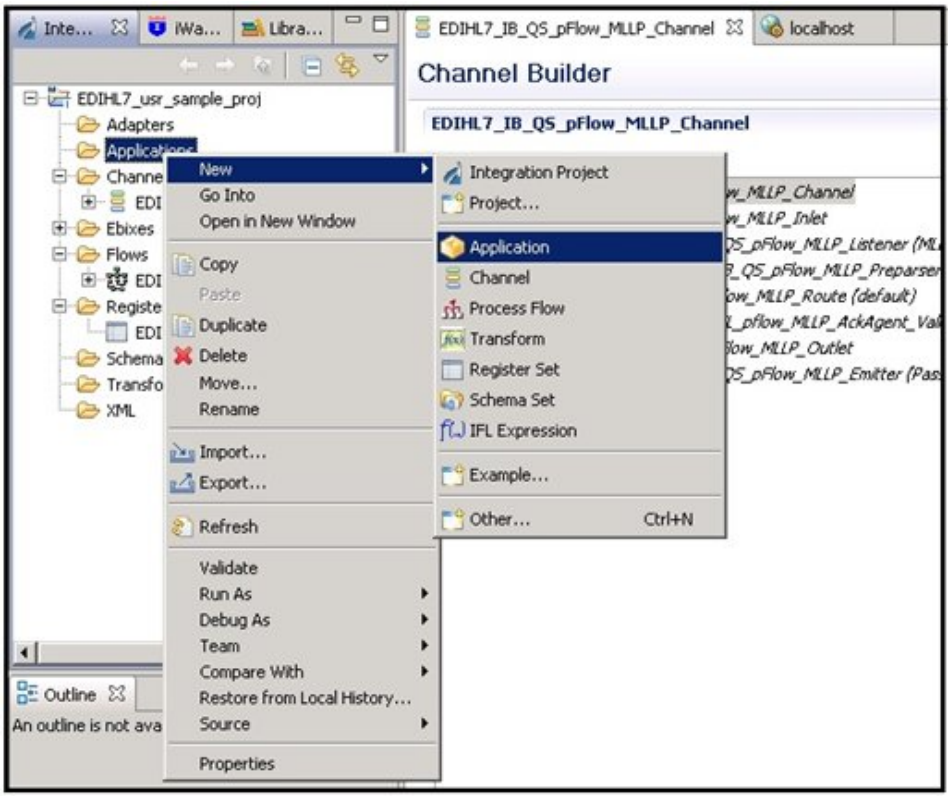


Configuring an iWay Integration Application for Inbound Processing

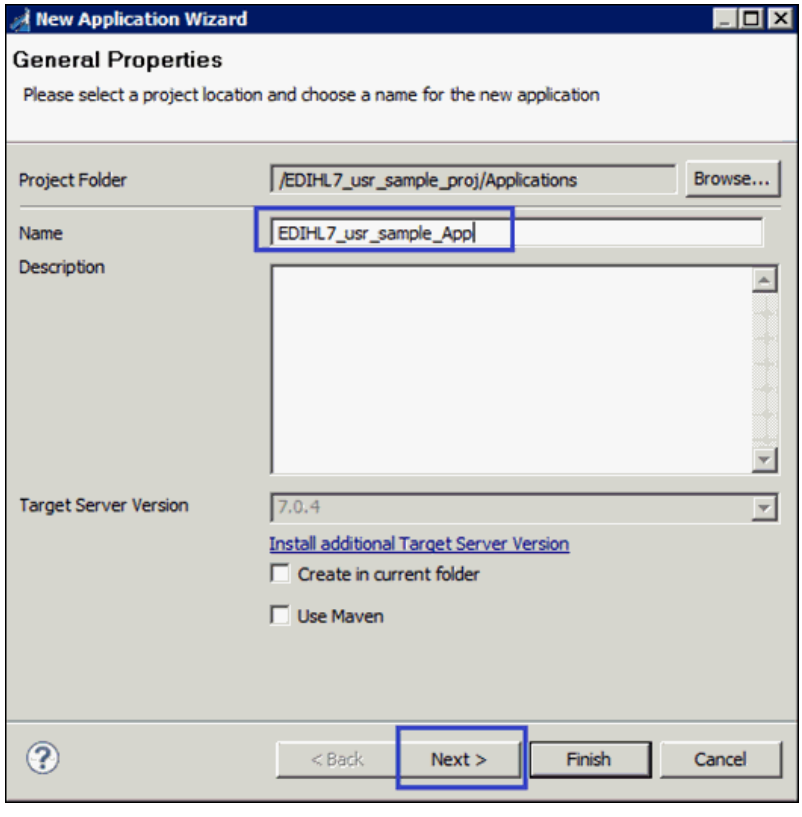
This section describes how to configure an iWay Integration Application (iIA) for inbound processing using iWay Integration Tools (iIT).

Procedure: How to Configure an iWay Integration Application for Inbound Processing

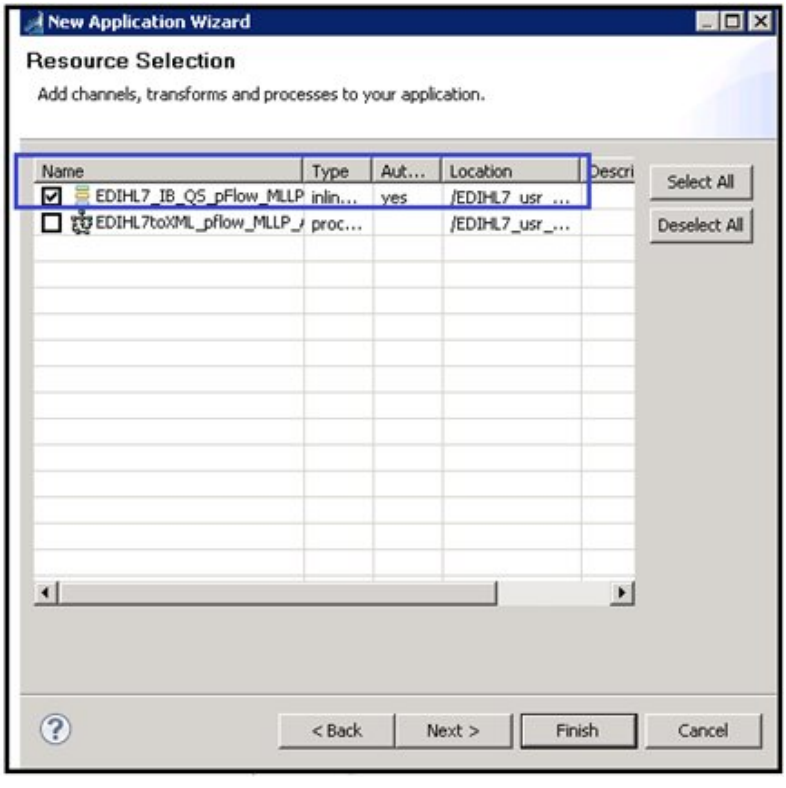
1. Right-click the Applications folder under the EDIHL7_usr_sample_proj integration project, select *New*, and then click *Application* from the context menu, as shown in the following image.



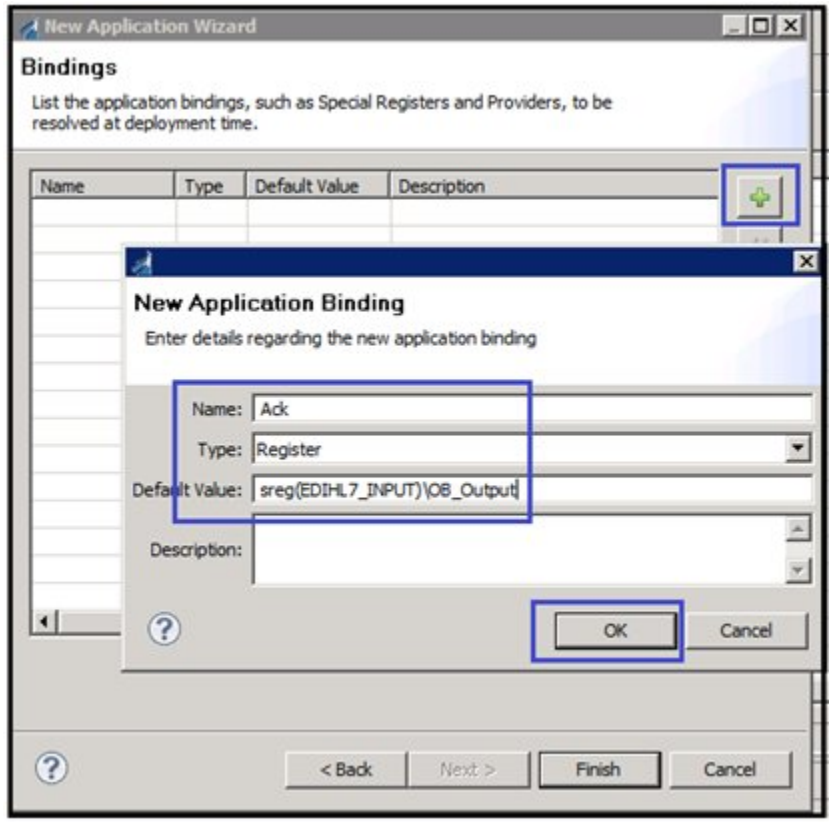
2. Enter a value in Name field (for example, EDIHL7_usr_sample_App), and click *Next*, as shown in the following image.



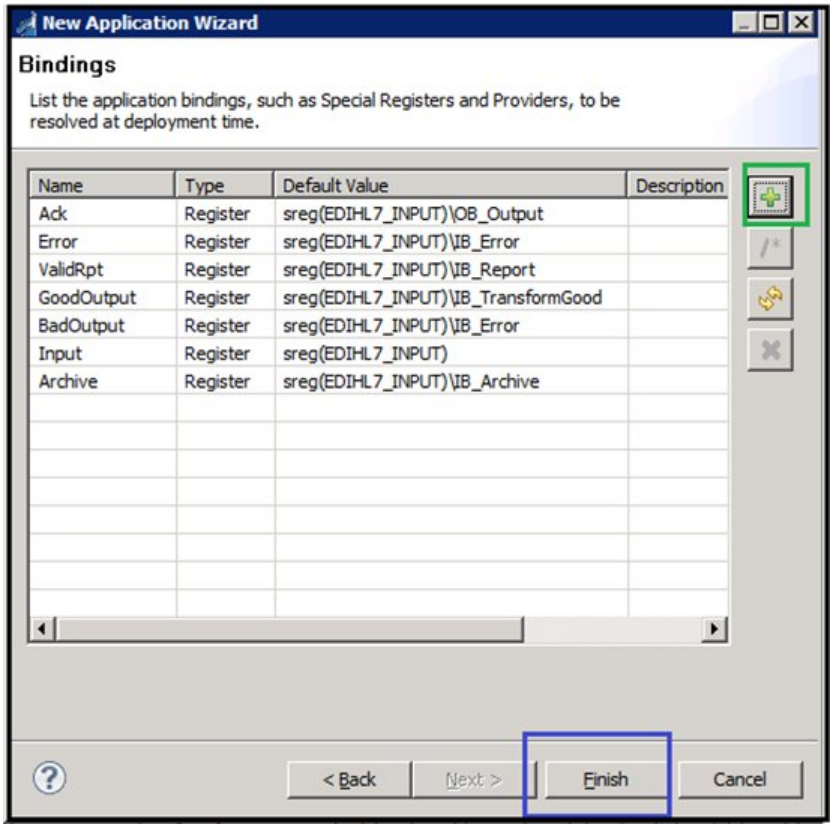
3. Select the *XMLEDIHL7_IB_QS_pFlow_MLLP_Channel* check box from Resource Selection pane and keep clicking *Next* until you get to the Bindings pane, as shown in the following image.



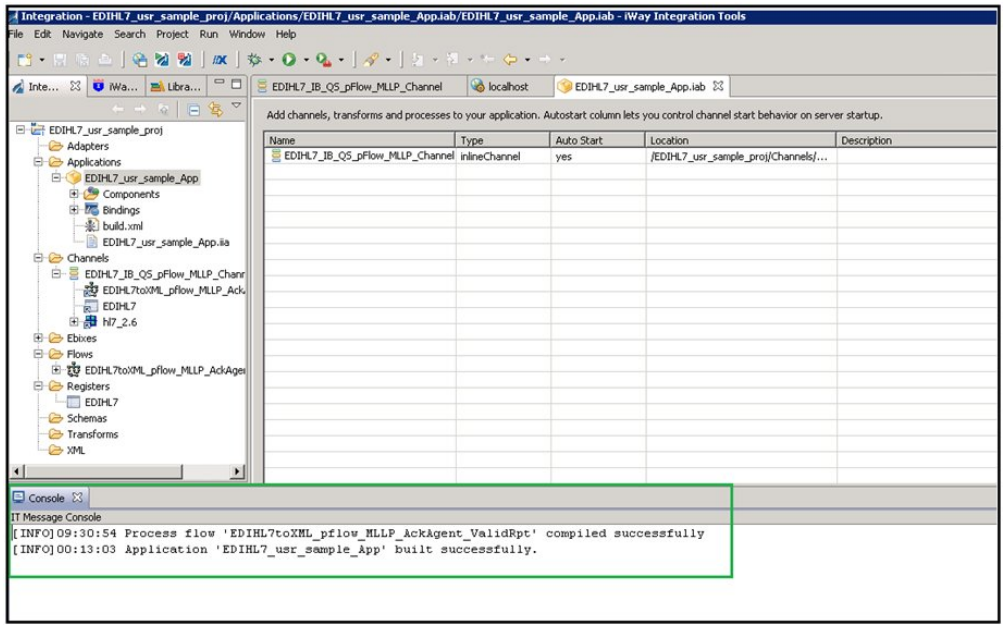
4. In the Bindings pane, add Bindings for Registers you have added in the XMLEDIHL7 Register Set to the iIA EDIHL7_usr_sample_App and click *Finish*, as shown in the following image.



After adding all register bindings, your iIT screen should resemble the following image.



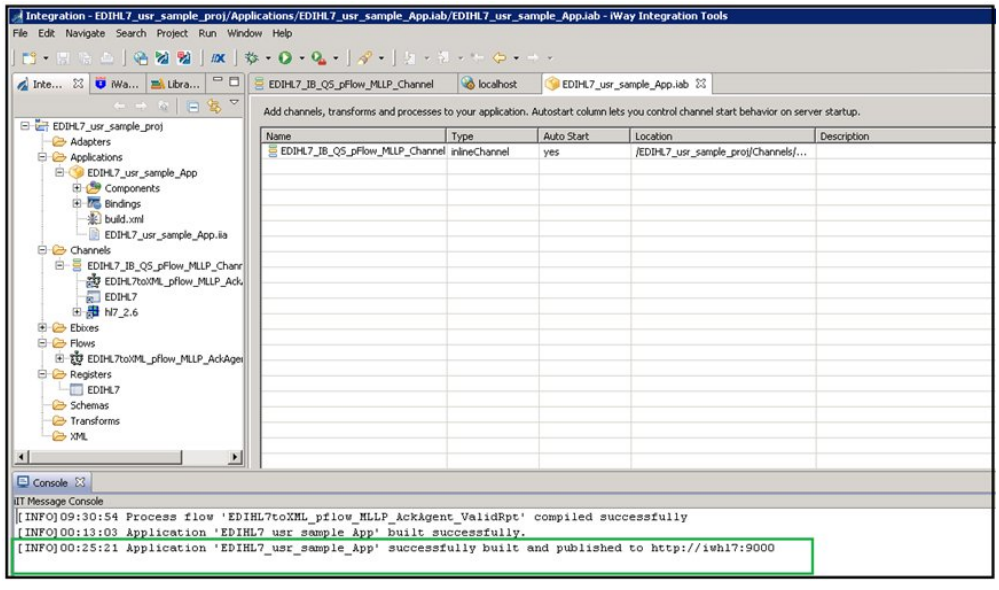
5. To build the application, right-click EDIHL7_usr_sample_App, select *Integration Tools*, and then click *Build* from the context menu, as shown in the following image.



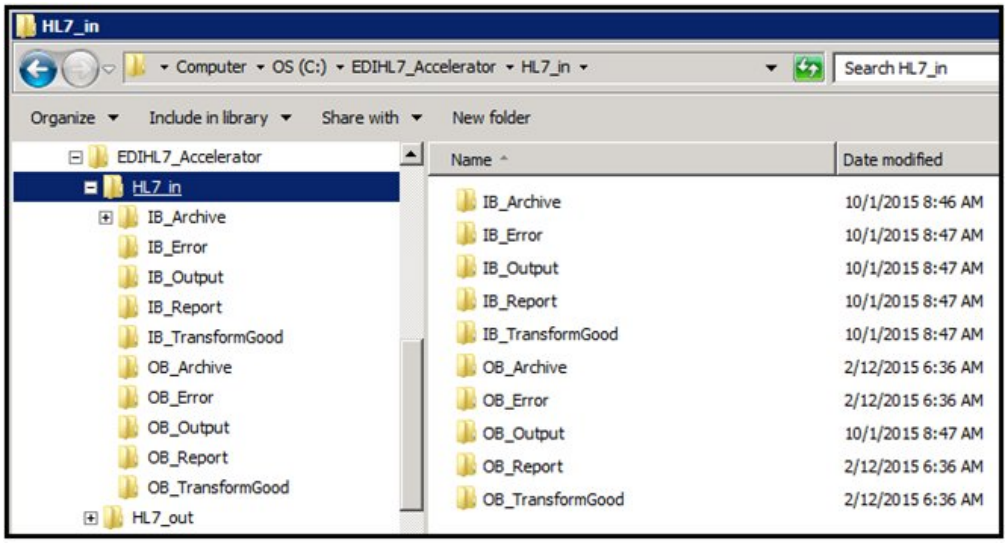
6. To publish the iIA, right-click *EDIHL7_usr_sample_App*, select *Integration Tools*, and then click *Publish to* from the context menu.

7. Provide the iSM server details in the Server URL text box and the other credentials, then click *Finish*.

The iIT page should resemble the following image.



8. Deploy the iIA application on the iSM registry.
9. Have the following folder structure created before starting your application in the iSM console, as shown in the following image.

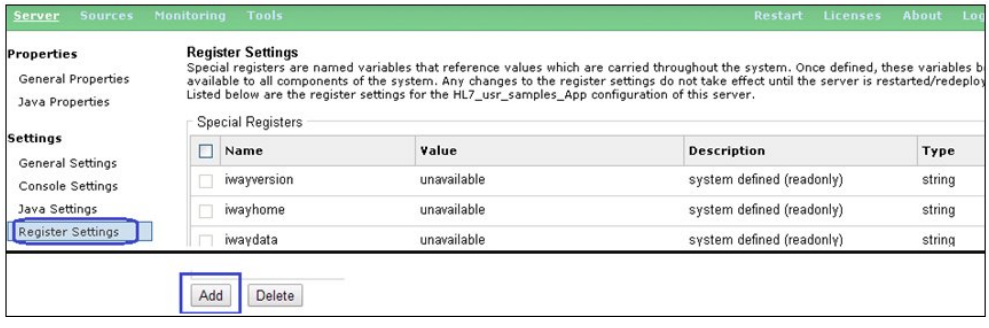


Setting HL7 System Registers Using MLLP

This section describes how to set system registers using the iWay Service Manager (iSM) Administration Console.

Procedure: How to Set System Registers

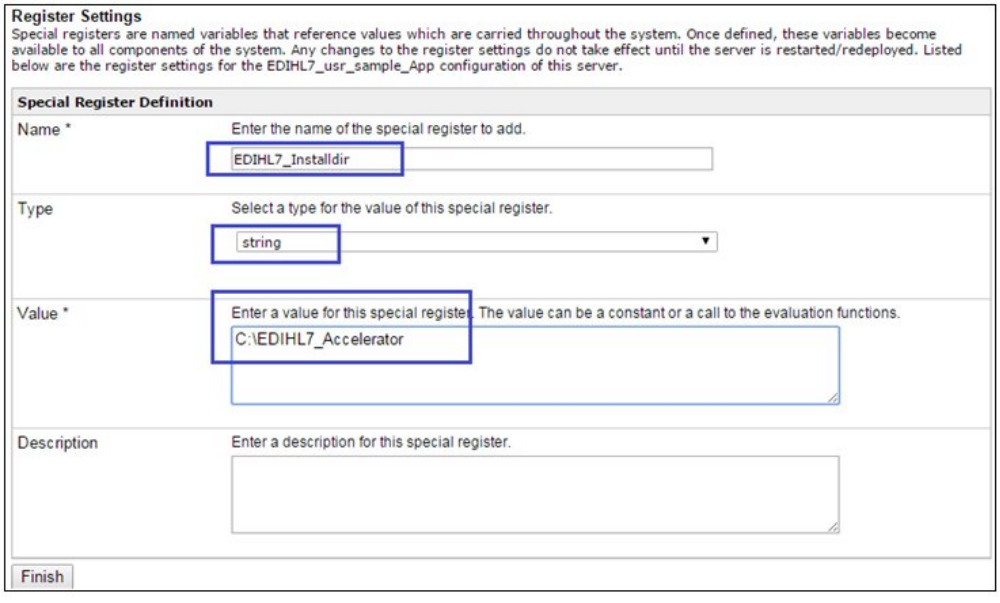
1. Open the iSM Administration Console and select *EDIHL7_usr_sample_App_MLLP_IB [down]* from the Management drop-down list.
2. From the console bar, click *Server*, *Register Settings*, and then click *Add*, as shown in the following image.



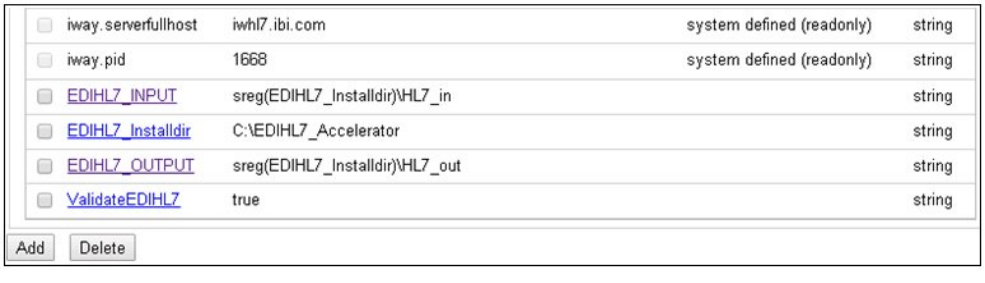
3. For the deployed EDIHL7_usr_sample_App_MLLB_IB application, define the system registers as listed in the following table:

Register Name	Value
EDIHL7_INPUT	`sreg(EDIHL7_Installdir)\HL7_in`
EDIHL7_installdir	`C:\EDIHL7_Accelerator`
EDIHL7_OUTPUT	`sreg(EDIHL7_Installdir)\HL7_out`

Register Name	Value
ValidateEDIHL7	`true`



After adding the registers, your Register Settings page in the iSM Administration Console should resemble the following image.



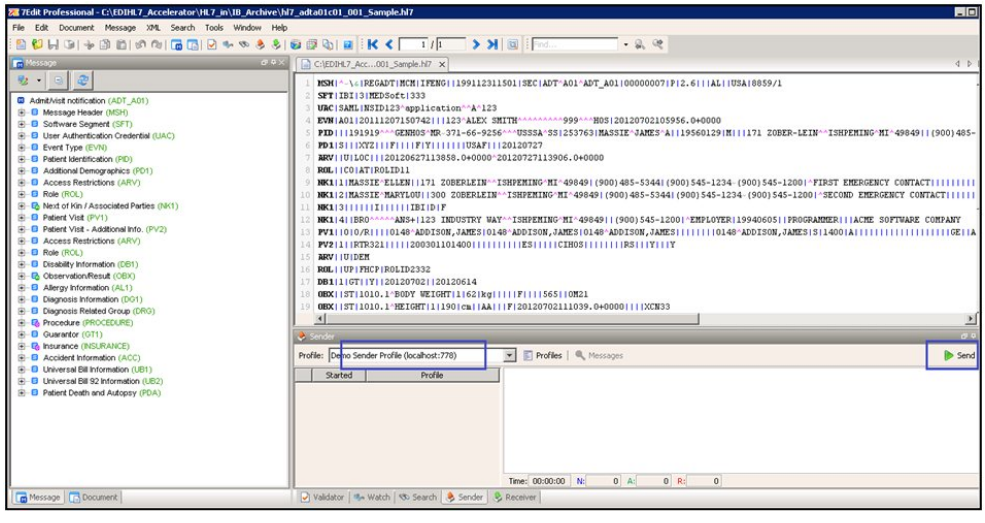
4. Start the deployed application in the iSM Administration Console and ensure that the channel is up and running in the Monitoring section of the console.

Testing the Inbound Channel Application Using MLLP

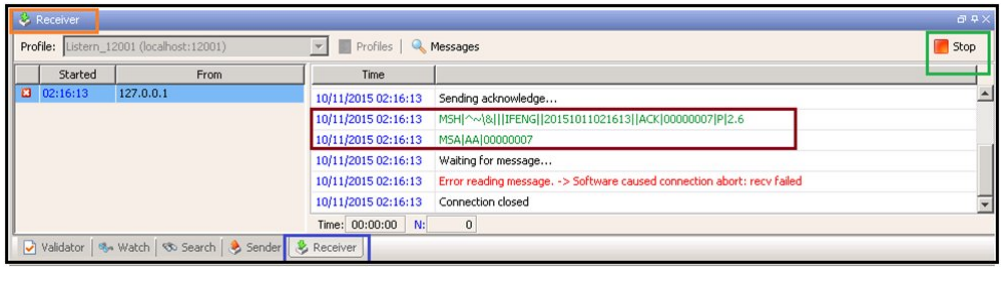
This section describes how to test the inbound channel application using the Minimal Lower Layer Protocol (MLLP).

Procedure: How to Test the Inbound Channel Application Using MLLP

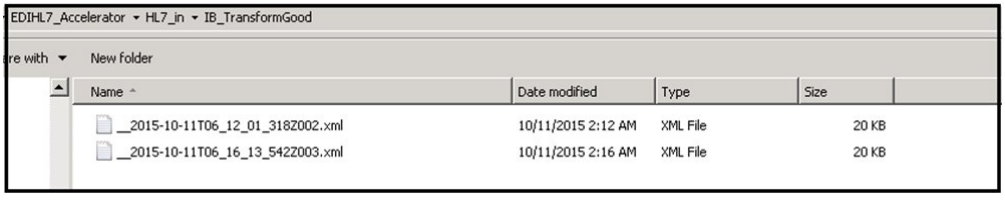
1. Use one of the tools to send messages to a specified port (for example, 7Edit, Telnet Client), set the sending port to *778*, and then send the HL7 message to that port, as shown in the following image.



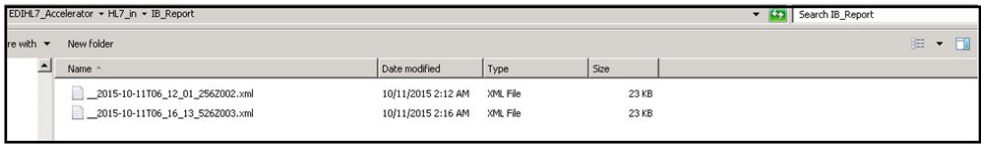
2. Configure a receiver port (for example, *12001*) to receive the ACK file through the MLLP emit, as shown in the following image.



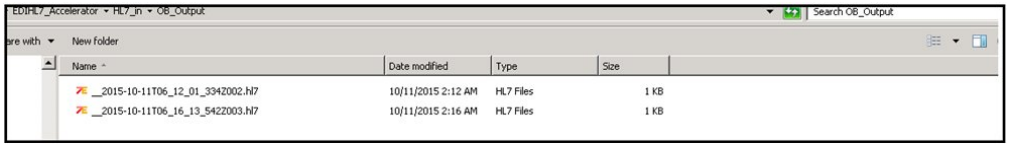
3. Observe the transformed XML output under the EDIHL7_Accelerator\HL7_in \IB_TransformGood location, as shown in the following image.



4. Monitor the reports under the EDIHL7_Accelerator\HL7_in\IB_report, as shown in the following image.



5. Observe the Acknowledgement in the EDIHL7_Accelerator\HL7_in\OB_Output location, as shown in the following image.



6. If the input data contained any errors, you can review this error data in the output directory that you have configured for error handling (for example, EDIHL7_Accelerator\HL7_in \IB_Error).

7. After inbound processing has completed, a copy of the input data that was used during the transformation is stored under the archive directory that you specified (for example, EDIHL7_Accelerator\HL7_in\IB_Archive).

Chapter 7

Outbound Processing: XML to HL7

This section describes how to configure a basic outbound message flow for the iWay Integration Solution for EDIHL7. The message flow represents the movement and tasks in the conversion of a message from XML format to HL7 format.

In this chapter:

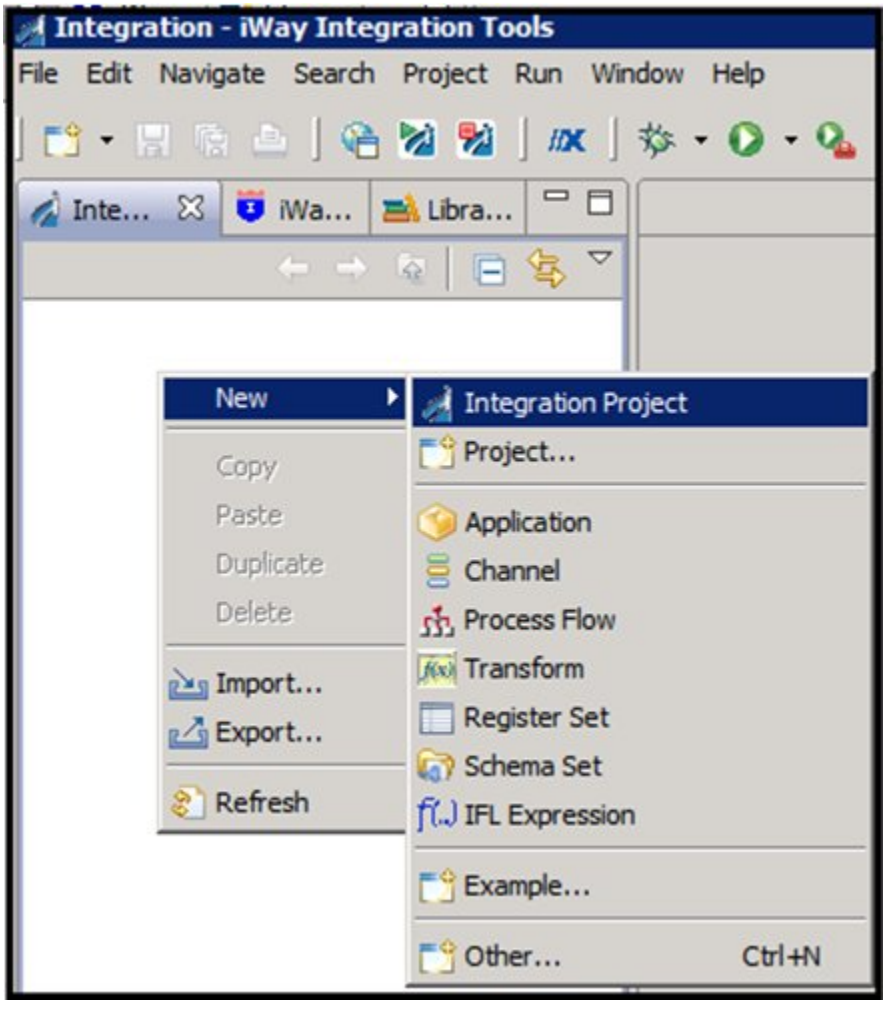
Configuring a Channel for HL7 Outbound Processing

The outbound channel creates an HL7 message from XML and a XML-formatted validation report. The documents are routed to designated folders based on the success or failure results of the transformation.

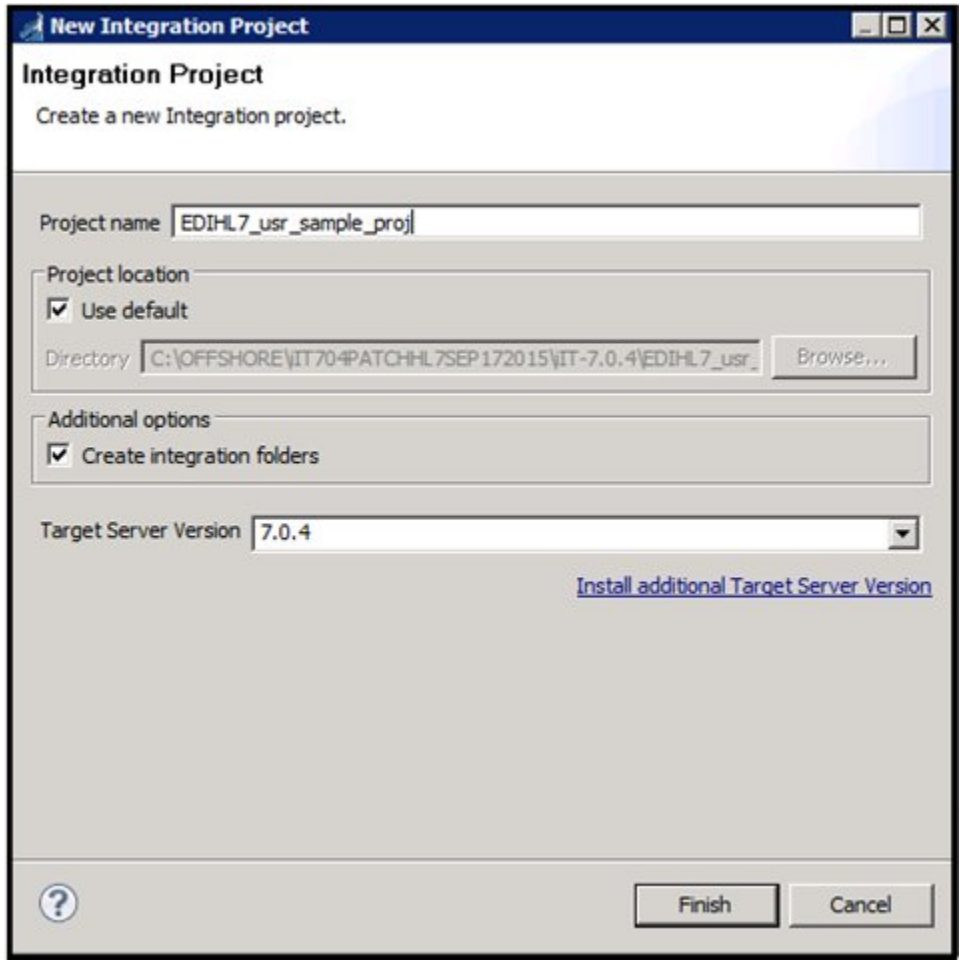
Procedure: How to Create a Channel for Outbound Processing

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.



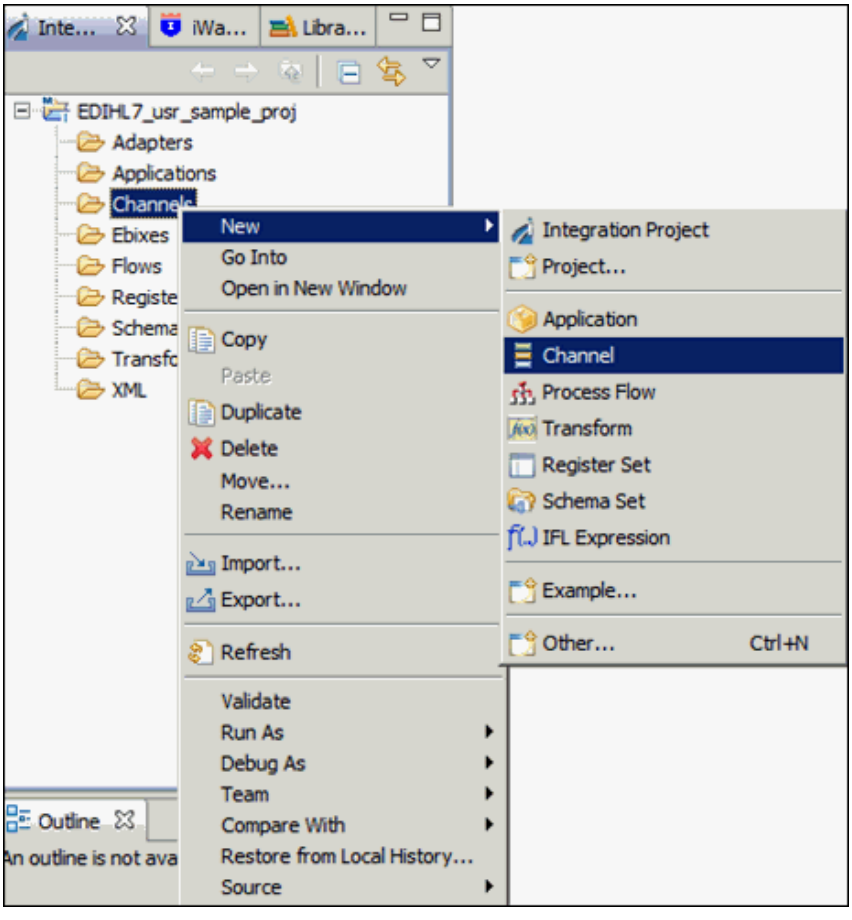
The New Integration Project dialog box opens, as shown in the following image.



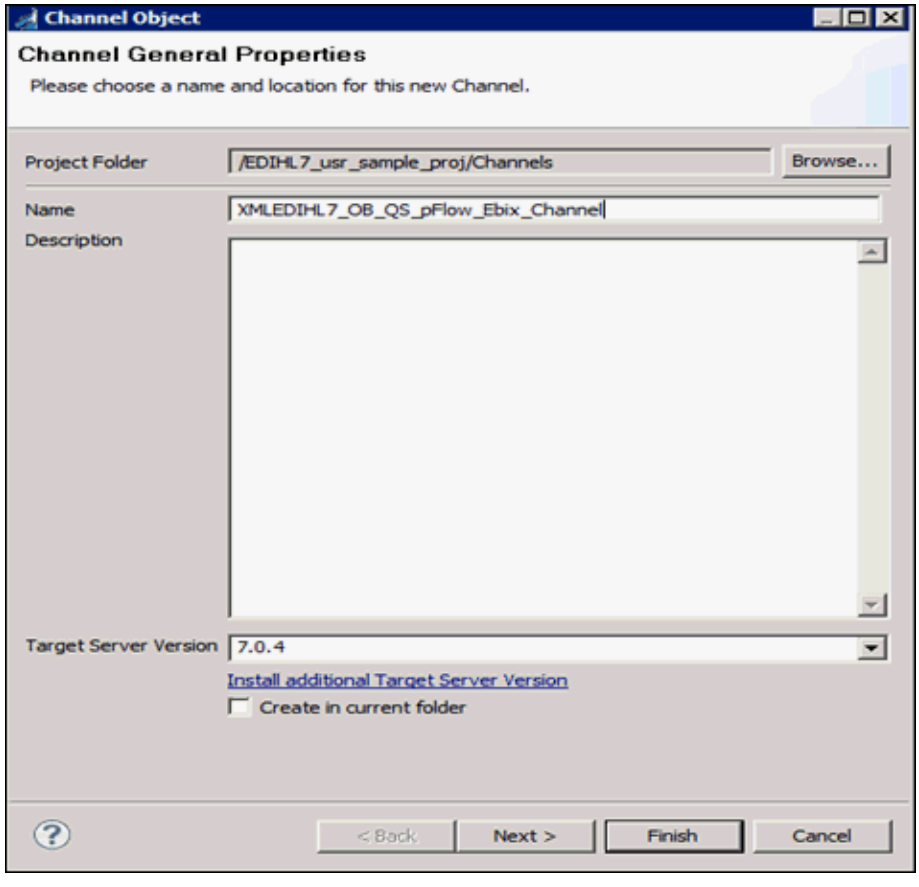
3. Enter a name for the Integration Project (for example, *EDIHL7_usr_sample_proj*) in the Project name field, and then click *Finish*.

   A new Integration Project node called *EDIHL7_usr_sample_proj* is added to the Integration Explorer.

4. Expand the *EDIHL7_usr_sample_proj* Integration Project node, right-click the *Channels* folder, select *New*, and then click *Channel* from the context menu, as shown in the following image.

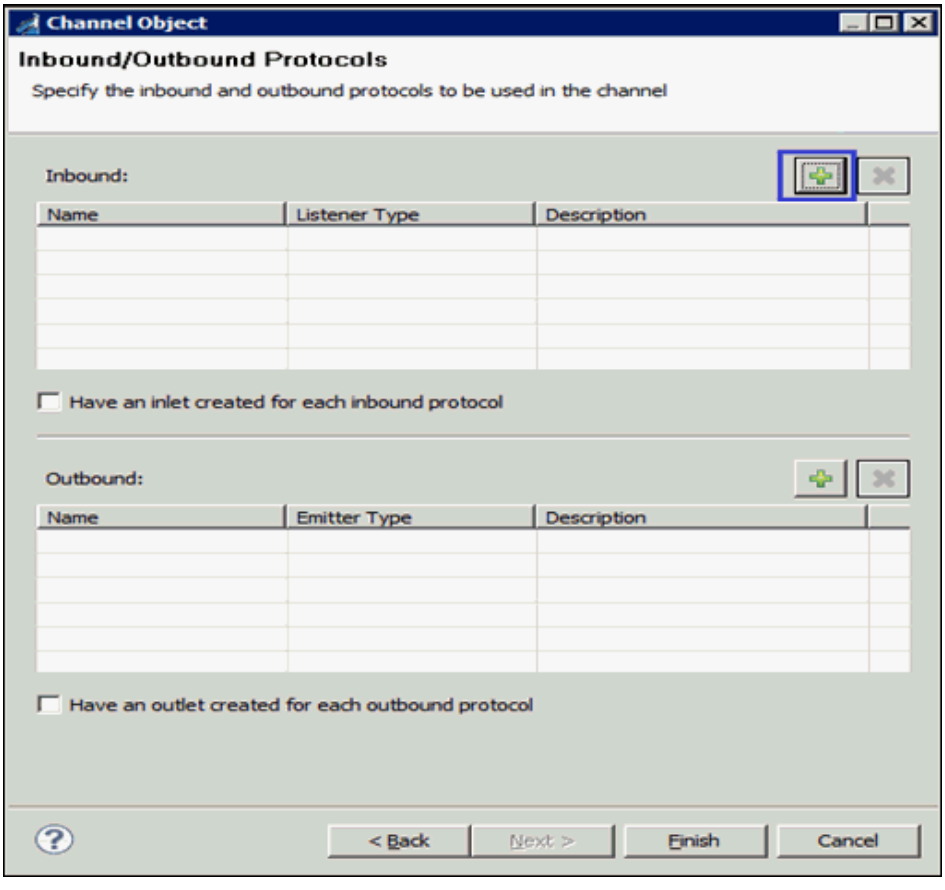


5. Enter a name for the Channel (for example, XMLEDIHL7_OB_QS_pFlow_Ebix_Channel), and then click *Next*, as shown in the following image.

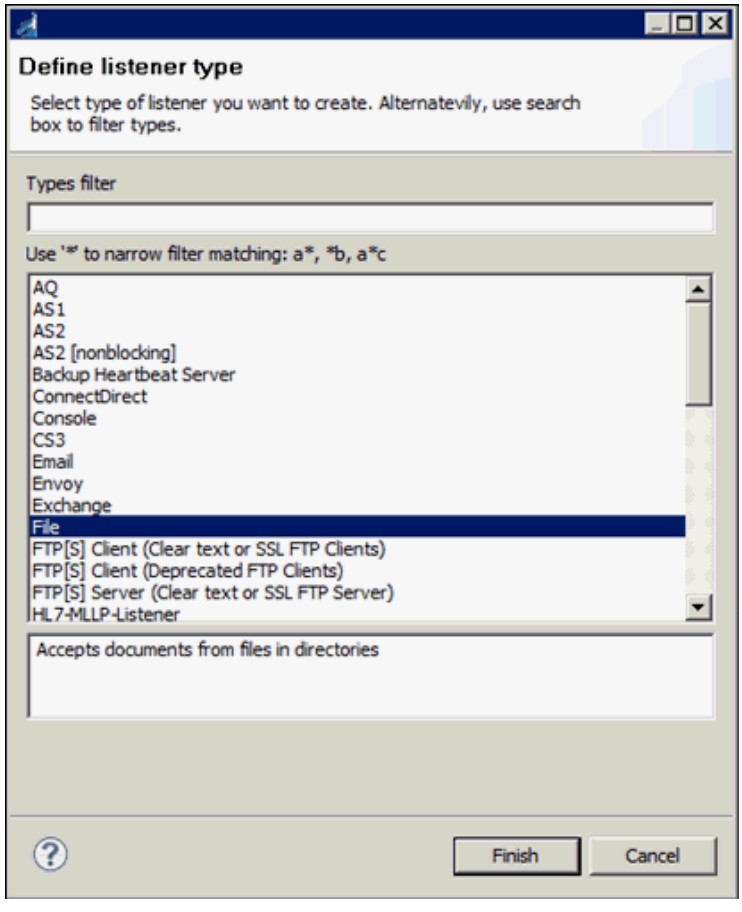


The Inbound/Outbound Protocols pane appears.

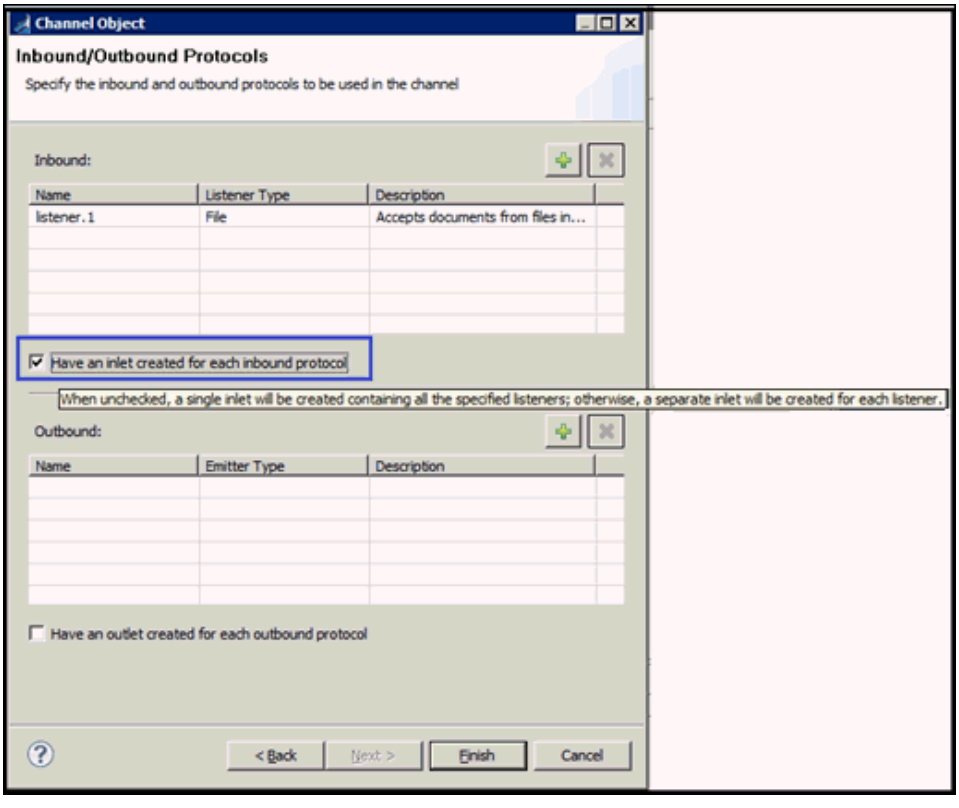
6. Click the *Add* button to add a File Listener in the Inbound section, as shown in the following image.



7. From the Types Filter list, select *File* and then click *Finish*, as shown in the following image.



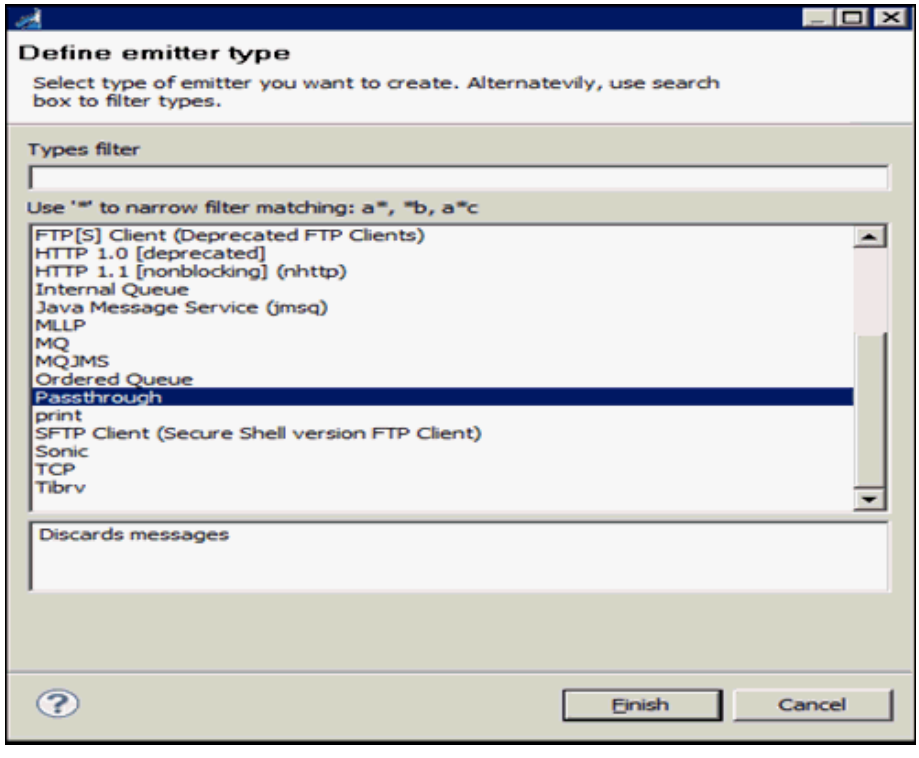
8. Select the *Have an Inlet created for each inbound protocol* check box to create an Inlet for the channel, as shown in the following image.



9. Click the *Add* button in the Outbound section to define an emitter.

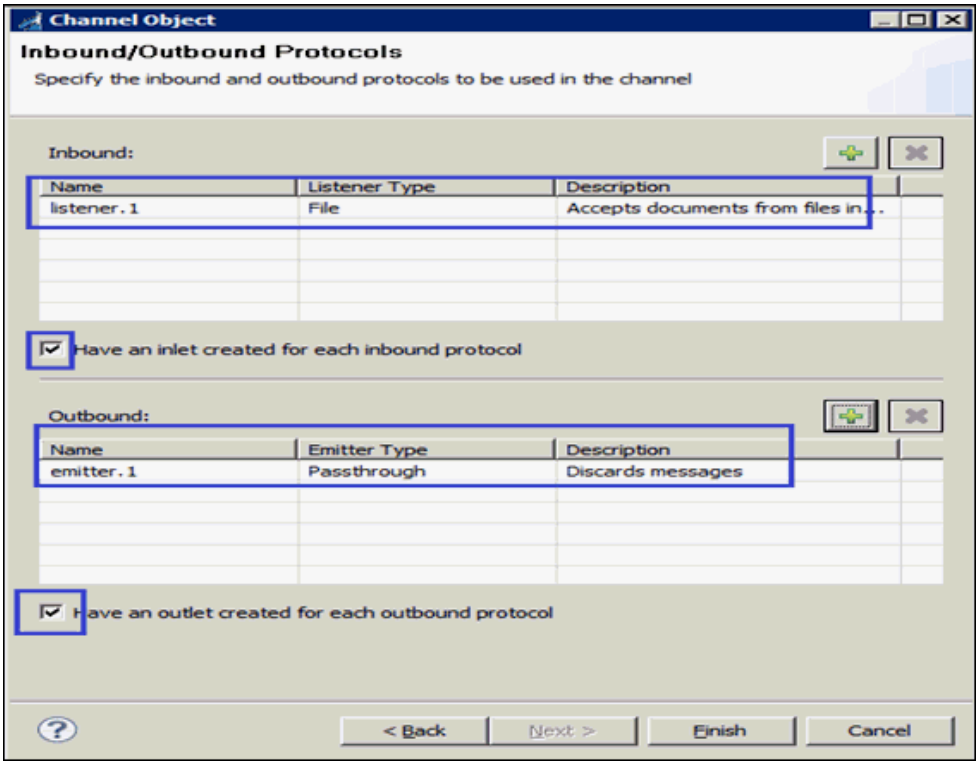
   The Define emitter type pane appears.

10. In the Types filter list, select *Passthrough* to define the passthrough emitter, and then click *Finish*, as shown in the following image.

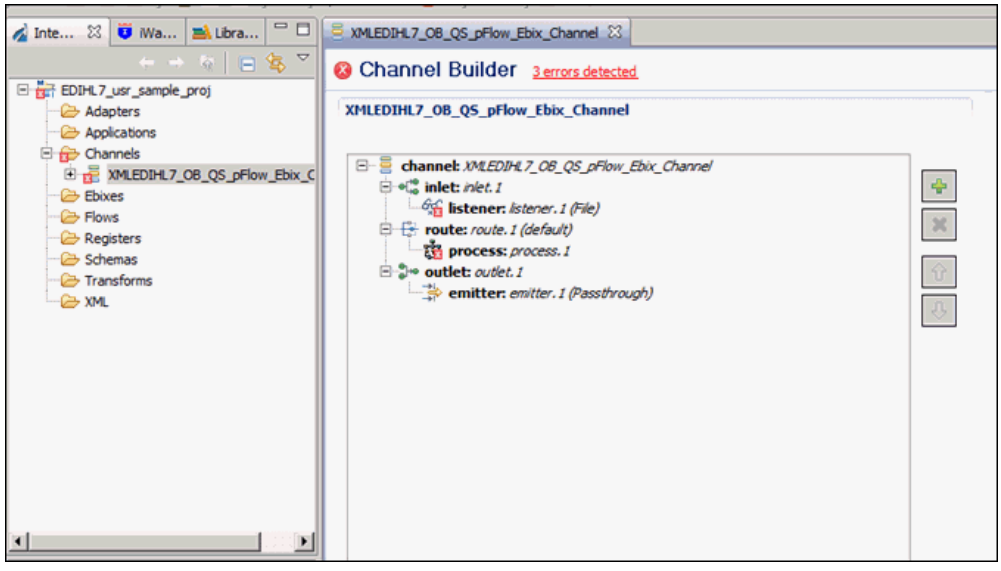


Note: Passthrough does not emit data from the listener. Instead, it just passes the control here and does nothing.

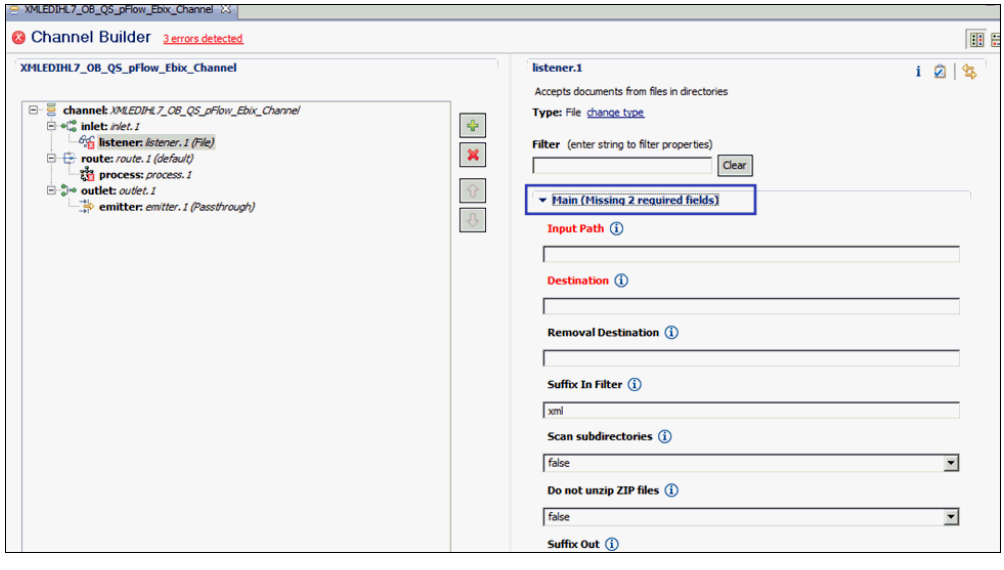
11. Select the *Have an outlet created for each outbound protocol* check box to create an outlet for the channel, and then click *Finish*, as shown in the following image.



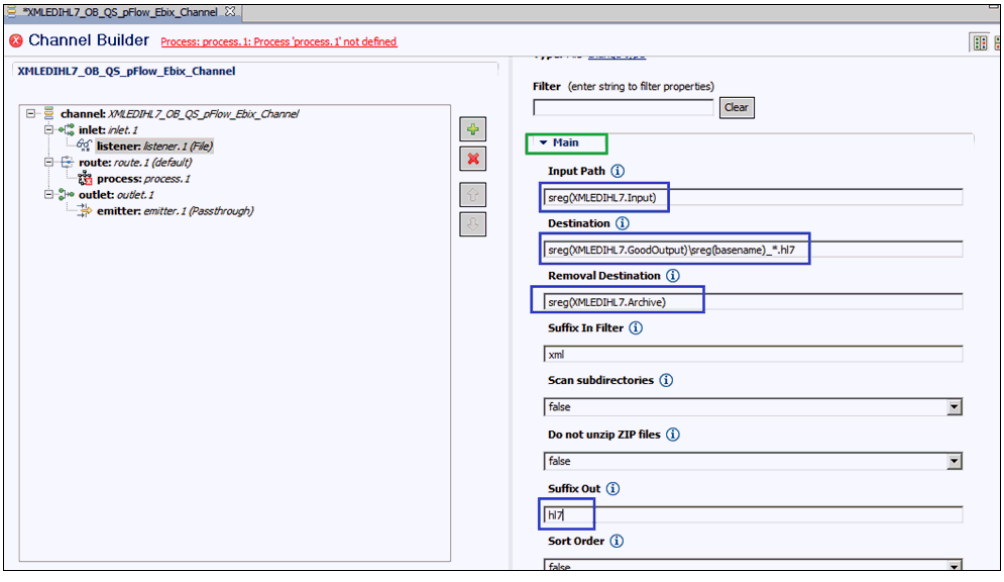
The Channel Builder pane appears.



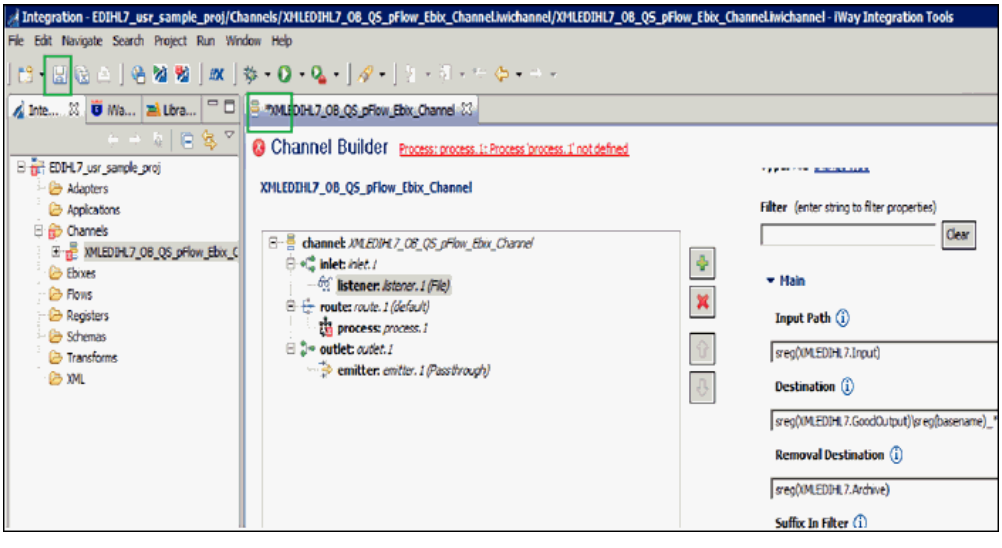
12. Under *inlet: inlet.1*, click *listener: listener.1 (File)* and then expand the *Main (Missing 2 required fields)* configuration parameter on the right pane, as shown in the following image.



13. Enter a valid Input Path, Destination, and Removal Destination (optional) path and then select *hl7* in the Suffix Out drop-down list, as shown in the following image.

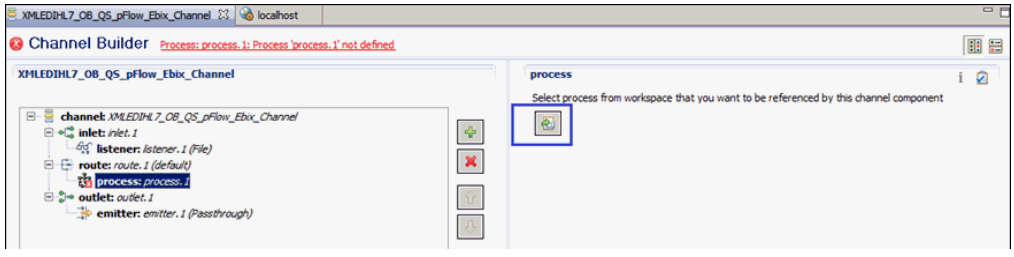


14. Click the *Save* icon near the File menu to save your edited listener configuration. You can also press the shortcut key (Ctrl+S) to save your work if you are using a Windows environment.

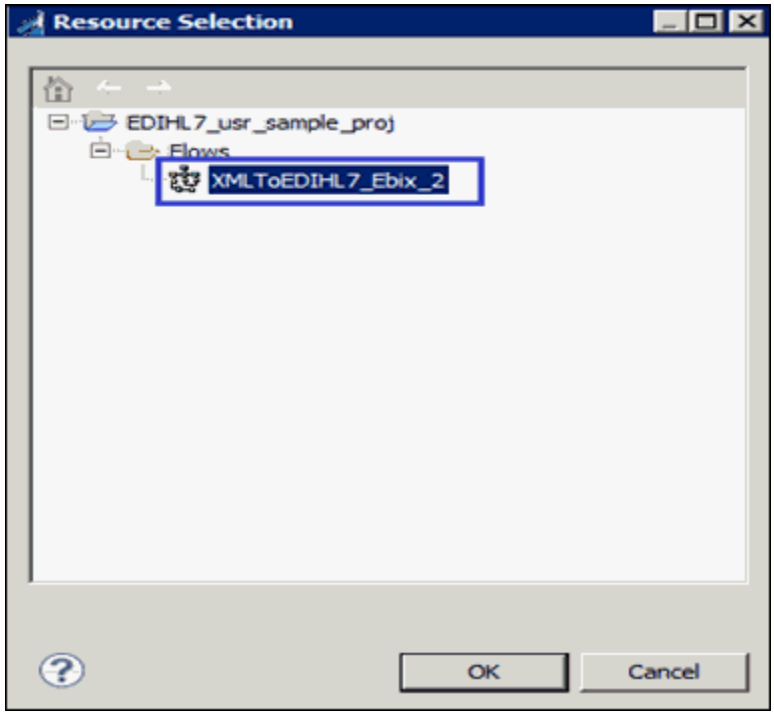


15. Import the *XMLToEDIHL7_Ebix_2_* Process Flow from the local drive or create the one in the EDIHL7_usr_sample_proj directory in the Flows folder.

16. Under the *route: route.1(default)* node, select *process: process.1* and click the process icon on the right panel to reference the process flow into your channel, as shown in the following image.



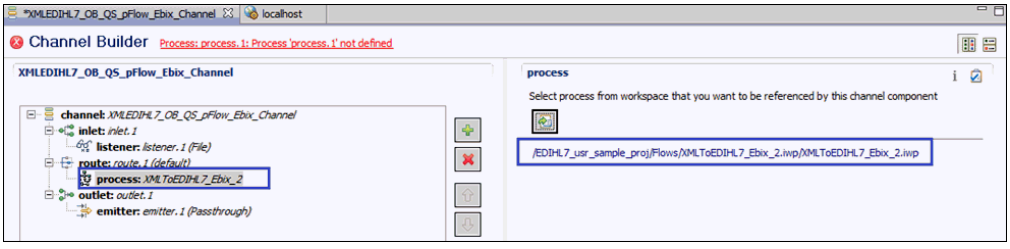
17. Select a process flow from the integration project and then click *OK*.



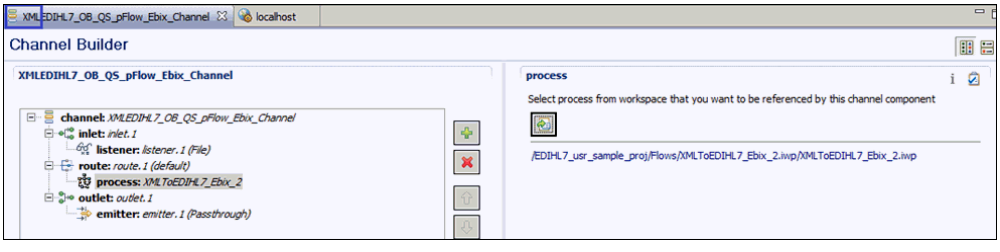
Note: Process flows should already be built and available in the iIT integration project. They can be found in the EDIHL7_usr_sample_proj directory inside the Flows folder.

For more information, see *Process Flow Used for Outbound Processing: XML to HL7* on page 164.

Your screen should resemble the following image.



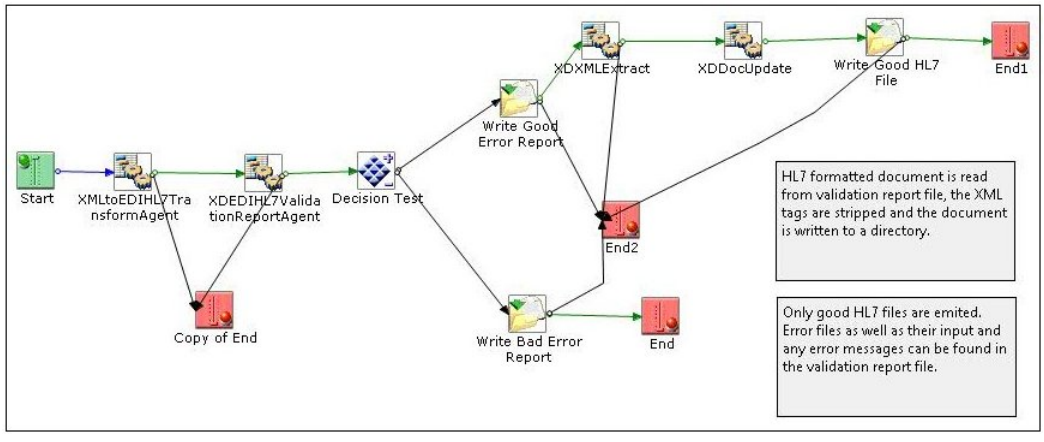
18. Click the *Save* icon near the File menu to save your edited listener configuration. You can also press the shortcut key (Ctrl+S) to save your work if you are using a Windows environment.



Reference: Process Flow Used for Outbound Processing: XML to HL7

This section provides an overview of the process flow used for outbound processing: XML to HL7. This process flow (XMLToEDIHL7_Ebix_2) is already built and available in the iIT integration project. It is located under the *EDIHL7_usr_sample_proj* node inside the Flows subfolder.

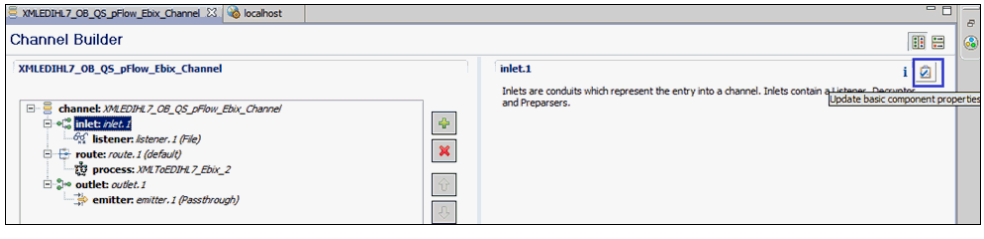
The following image shows the entire outbound process flow, including all of the nodes that are used and their connections.



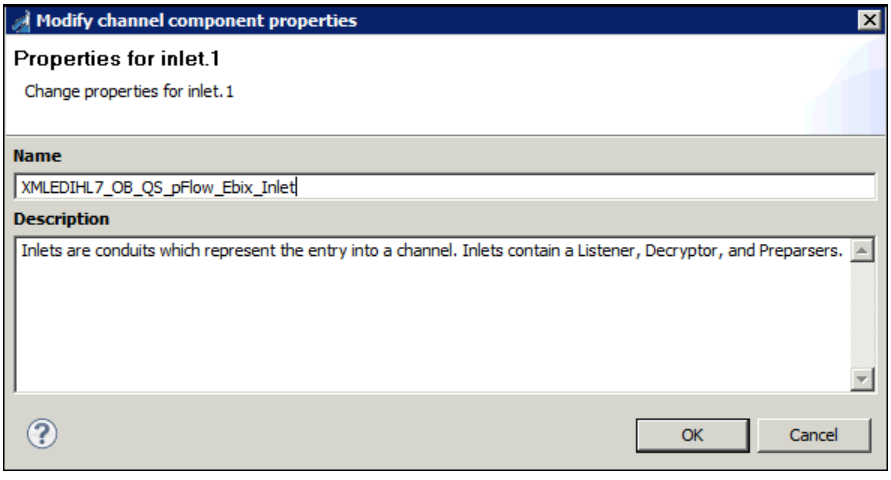
In this process flow, an HL7 formatted document is read from a validation report file. The XML tags are stripped and the document is written to a directory. Only valid HL7 files are emitted. Error files as well as their input and any error messages can be found in the validation report file.

Procedure: How to Update Basic Details for Channel Components

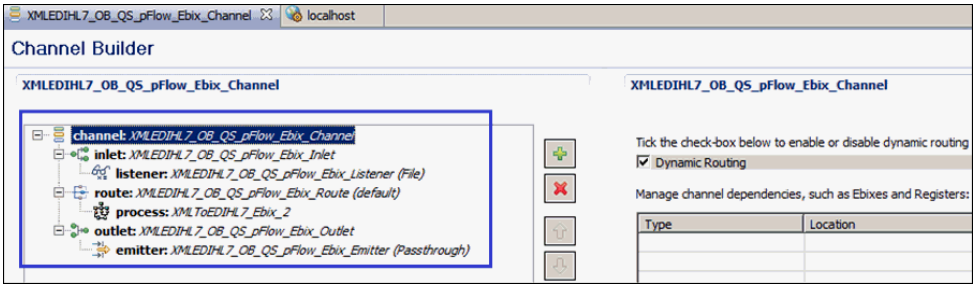
1. Select the channel component *inlet:inlet.1* from the channel and click the update channel component icon on the top right side panel, as shown in the following image.



2. Rename or update the inlet details (for example, XMLEDIHL7_OB_QS_pFlow_Ebix_Inlet), and click *OK*, as shown in the following image.



After renaming or changing the channel components, your iIT Channel Builder should resemble the following image.

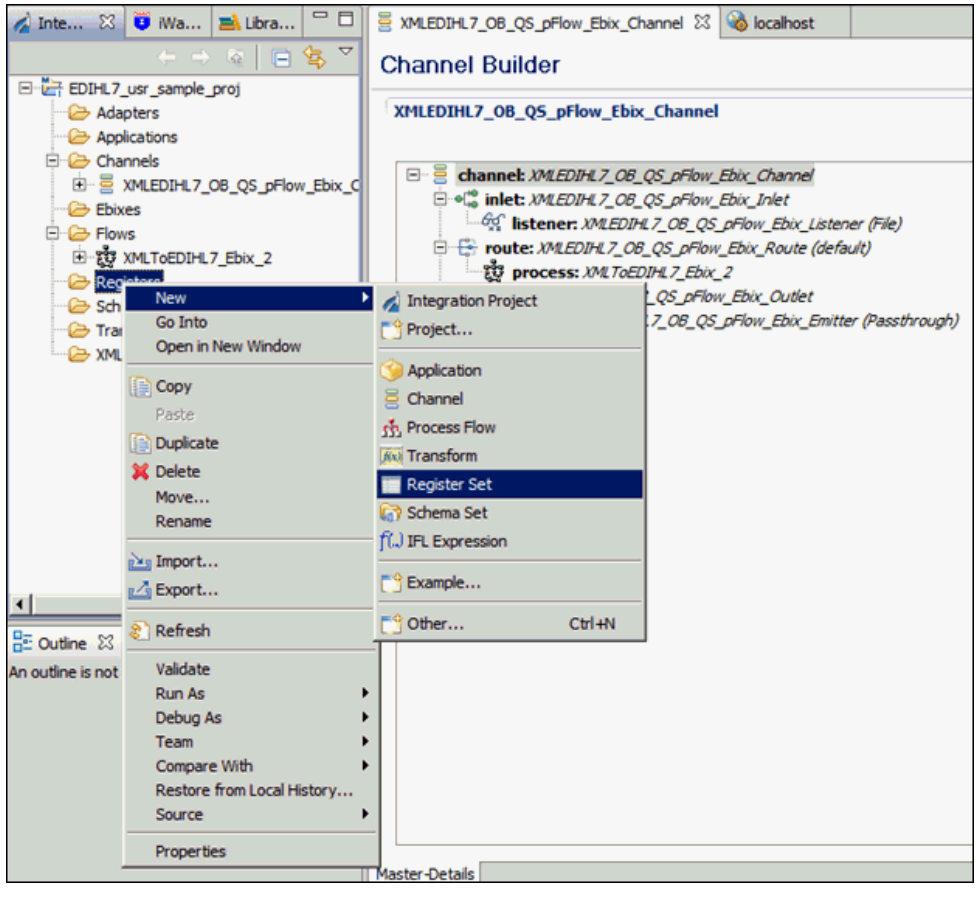


Configuring Register Sets and Registers

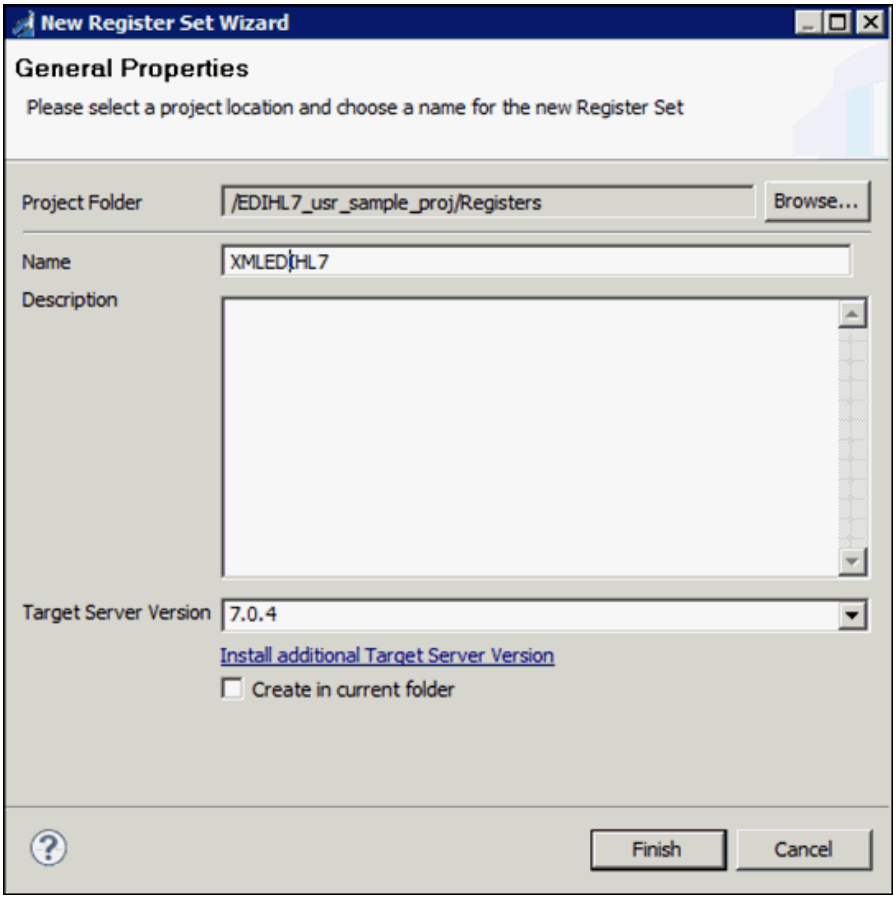
This section describes how to configure register sets and registers using iWay Integration Tools (iIT).

Procedure: How to Configure Register Sets and Registers

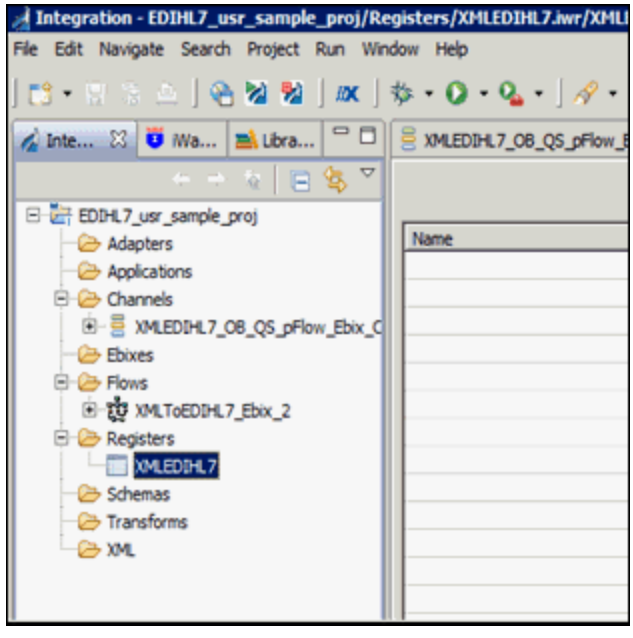
1. In the EDIHL7_usr_sample_proj project tree, right-click the Registers folder, and select *New*, and then click *Register Set* from the context menu, as shown in the following image.



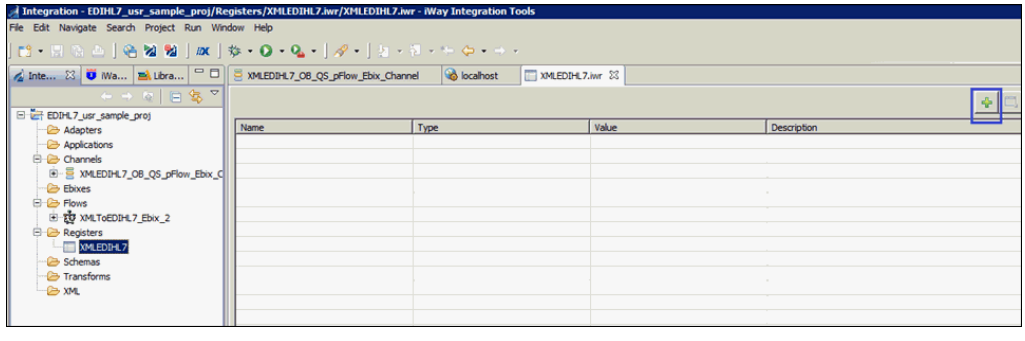
2. In the Name field, enter a name for the register set and click *Finish*, as shown in the following image.



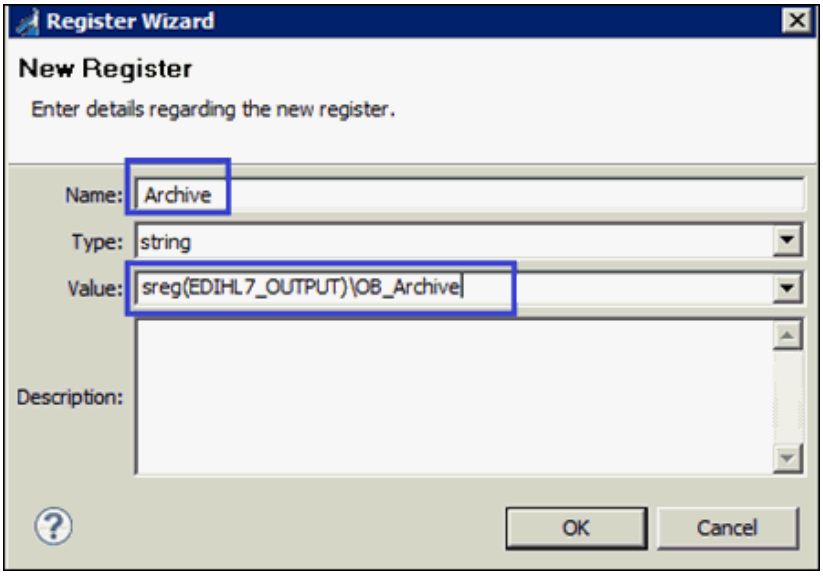
The new register set appears under Registers in the Registers folder, as shown in the following image.



3. Click the *Add a property* icon to add a register to the register set.



4. Enter a name for the new register in the Name field, select a register type from the Type drop-down list (set to string by default), and then enter a value in the Value field, as shown in the following image.

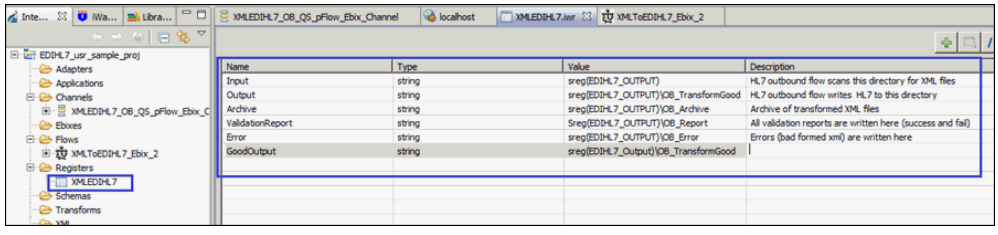


5. Click *OK*.

6. Create the following registers under the XMLEDIHL7 registers set you just created, along with the values shown in the table below.

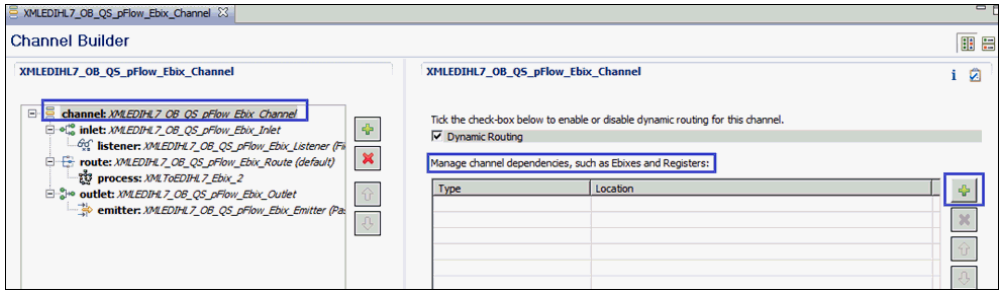
Register Name	Value
Input	Name=Input, Value=sreg(EDIHL7_OUTPUT)
Output	Name=GoodOutput, Value=sreg(EDIHL7_OUTPUT) \OB_TransformGood
Archive	Name=Archive, Value=sreg(EDIHL7_OUTPUT)\OB_Archive
ValidationReport	Name=ValidationReport, Value=sreg(EDIHL7_OUTPUT) \OB_Report
Error	Name=Error, Value=sreg(EDIHL7_OUTPUT)\OB_Error

Register Name	Value
GoodOutput	Name=Output, Value=sreg(EDIHL7_OUTPUT) \OB_TransformGood

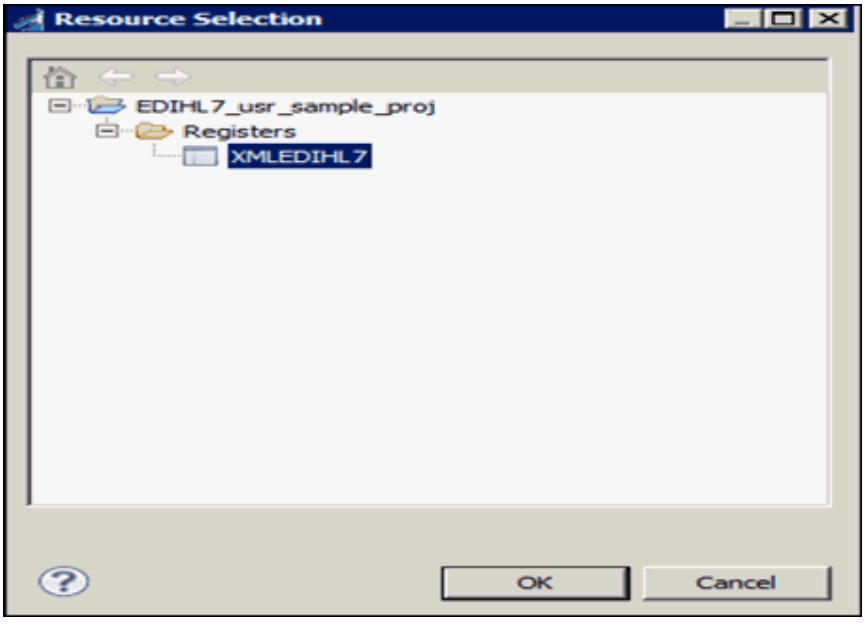


Procedure: How to Add a Register Set to an Outbound Channel as a Dependency

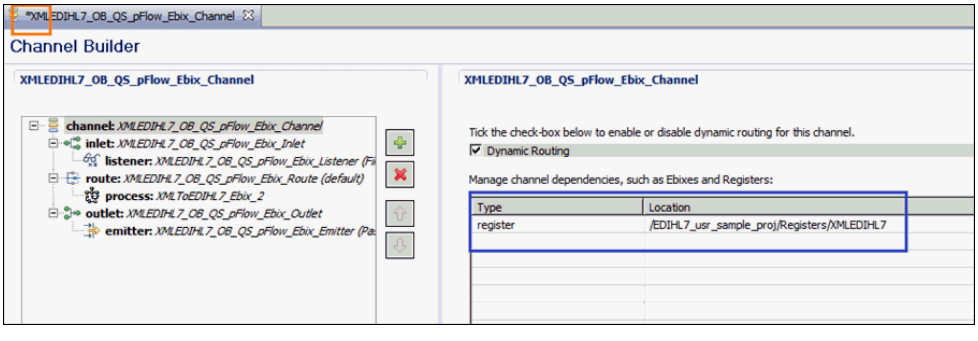
1. Click the *channel: XMLEDIHL7_OB_QS_pFlow_Ebix_Channel* node and then click the Add dependency icon on the far right side panel of the channel properties pane, as shown in the following image.



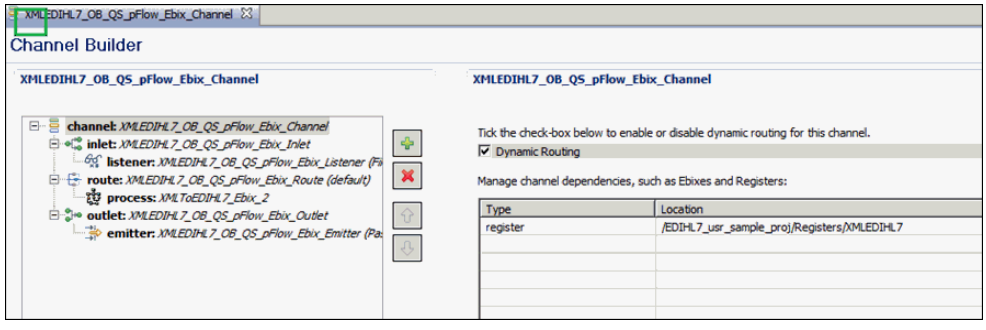
2. From the Registers folder under the integration folder, select *XMLEDIHL7* and click *OK*.



The iIT page should resemble the following image.



3. Click the *Save* icon to save your changes. You can also use the keyboard shortcut (Ctrl+S) if you are using a Windows environment.

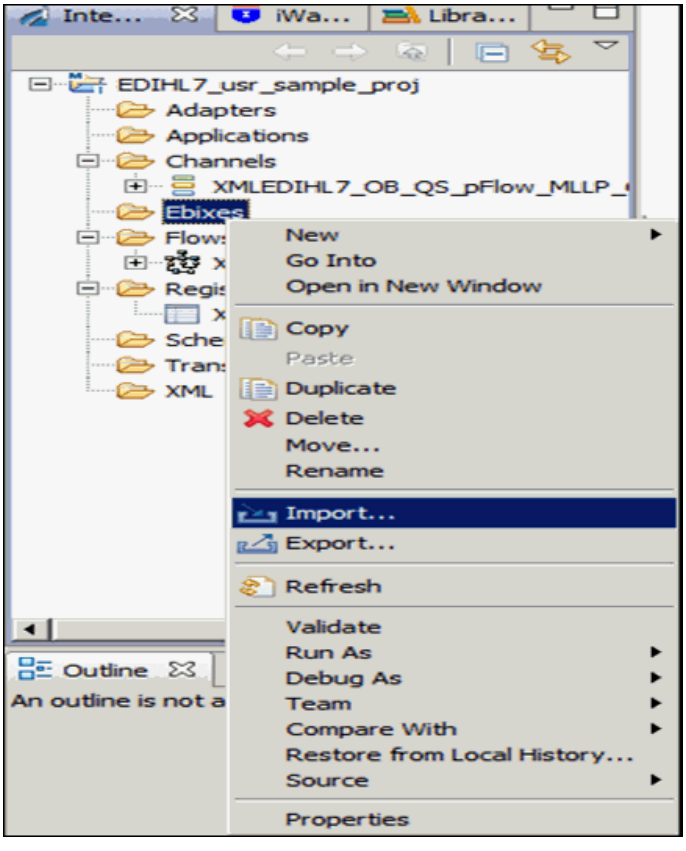


Importing an Ebix Into the Workspace

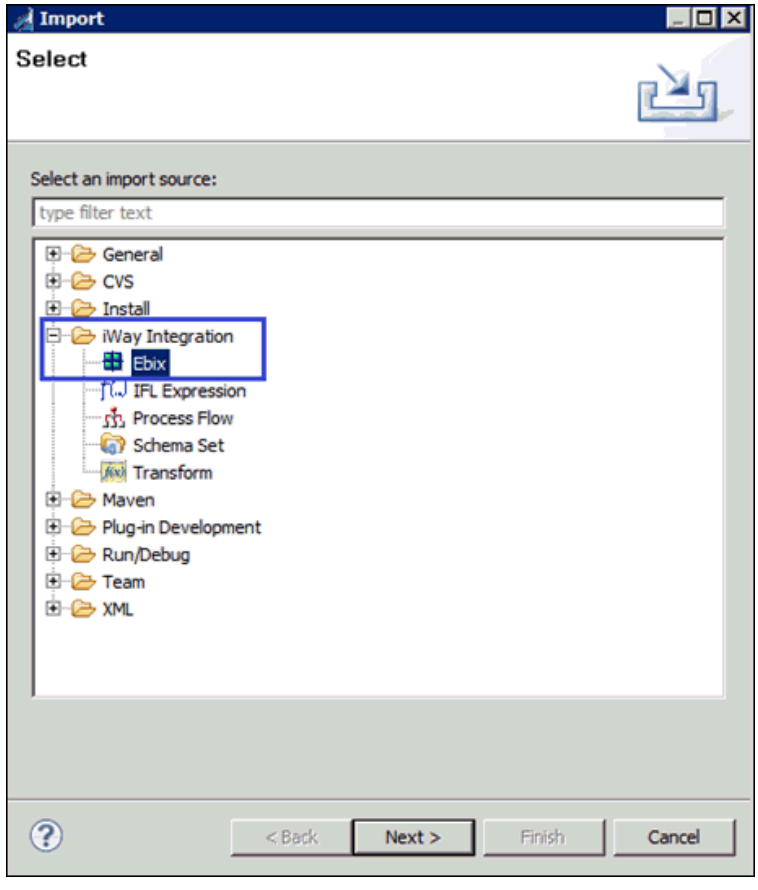
This section describes how to import an Ebix into the workspace using iWay Integration Tools (iIT).

Procedure: How to Import an Ebix Into the Workspace

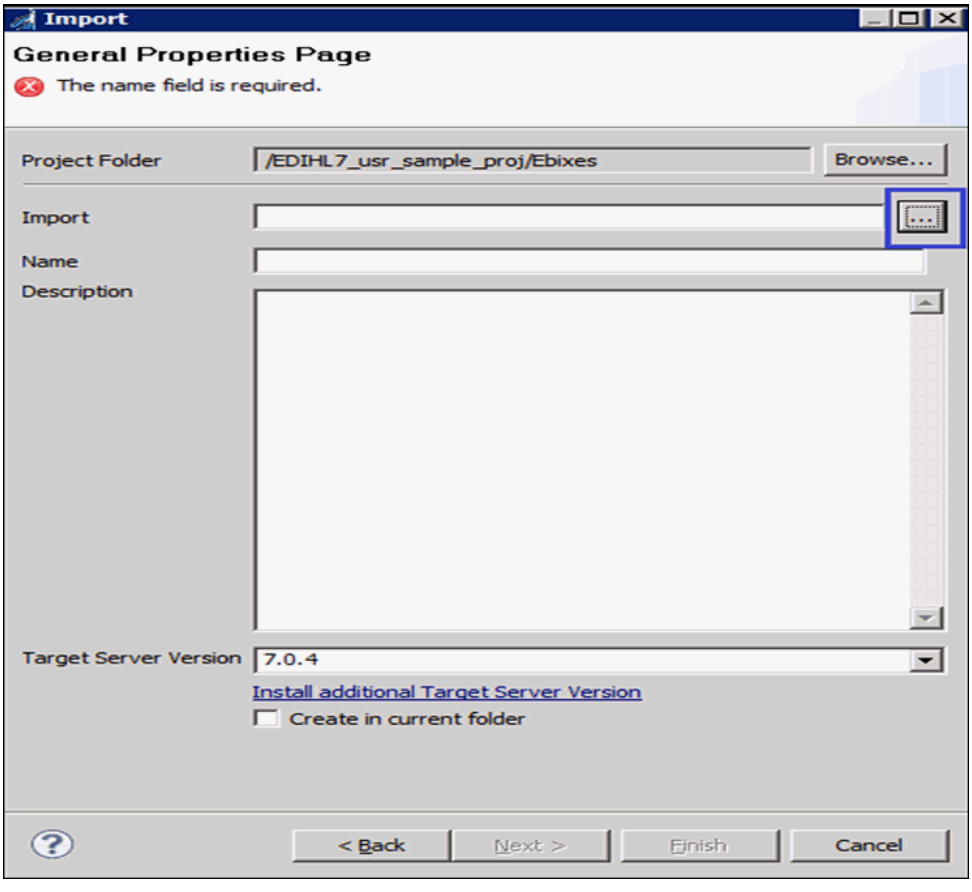
1. In the integration project EDIHL7_usr_sample_proj, right-click the Ebixes folder and then select *Import* from the context menu, as shown in the following image.



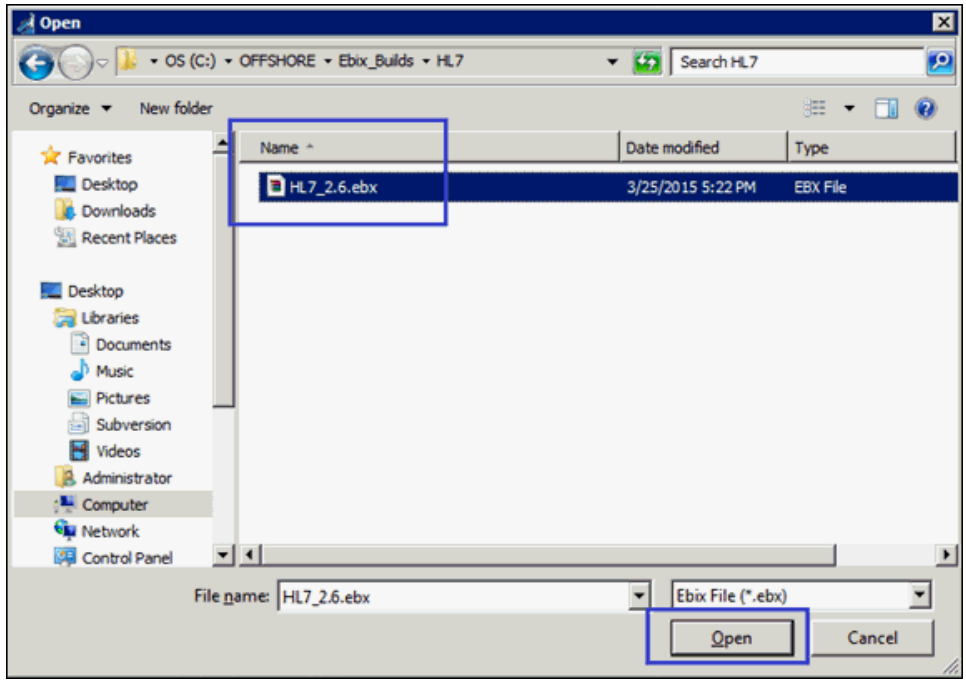
2. Expand the iWay Integration folder, select *Ebix*, and then click *Next*, as shown in the following image.



3. Click the ellipses (...) button to browse and import the ebix from a specific folder location, as shown in the following image.

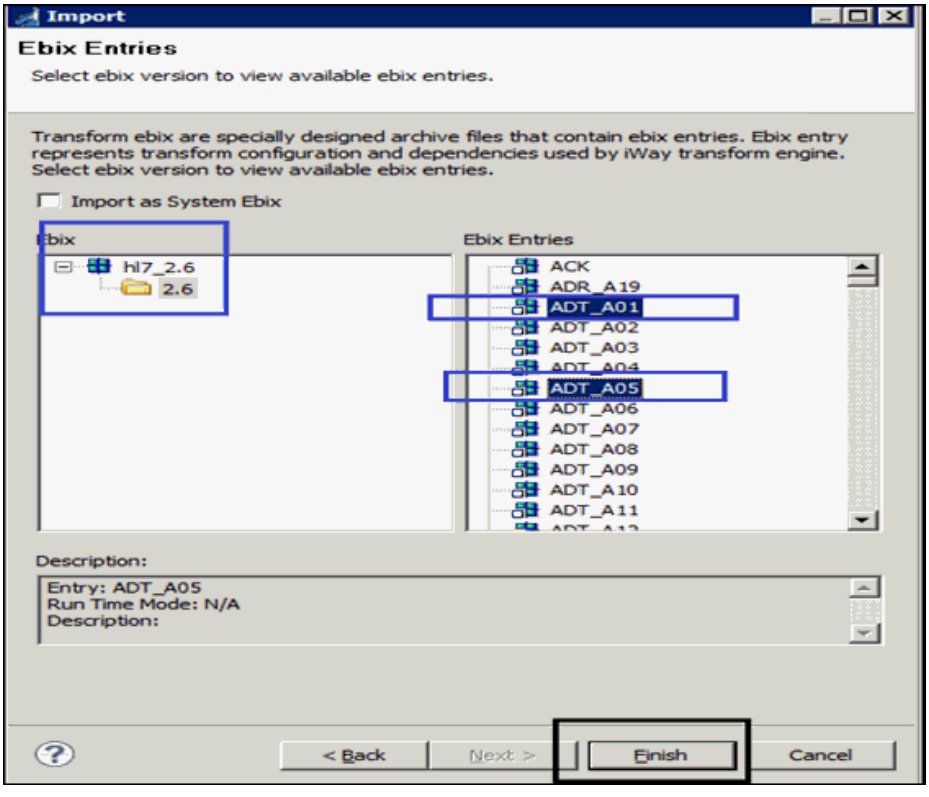


4. Select *HL7_2.6 ebix* from the folder location and click *Open*, as shown in the following image.

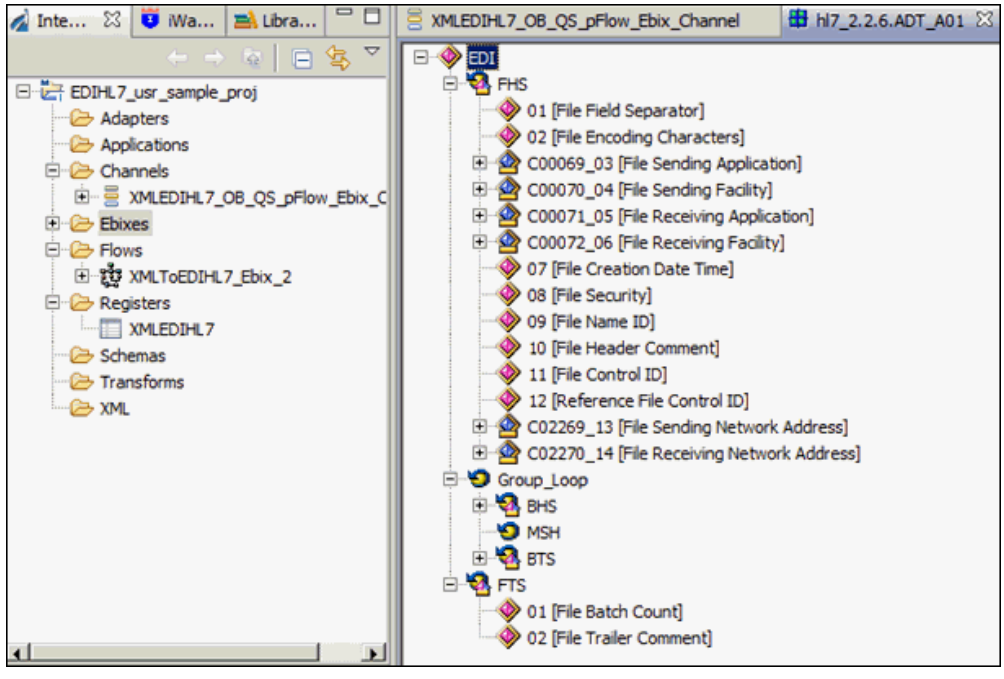


5. In the Import wizard, click *Next*.
6. Expand the hl7_2.6 ebix from the left panel and select the 2.6 folder.

7. Select the messages you wish from the right side panel and click *Finish*, as shown in the following image.

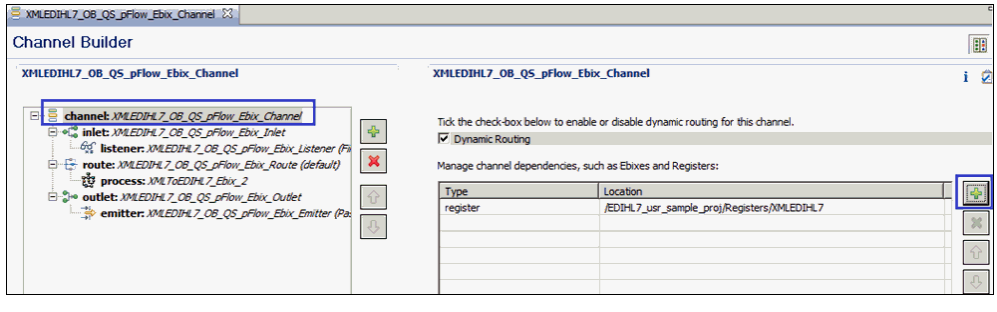


The iIT page should resemble the following image.

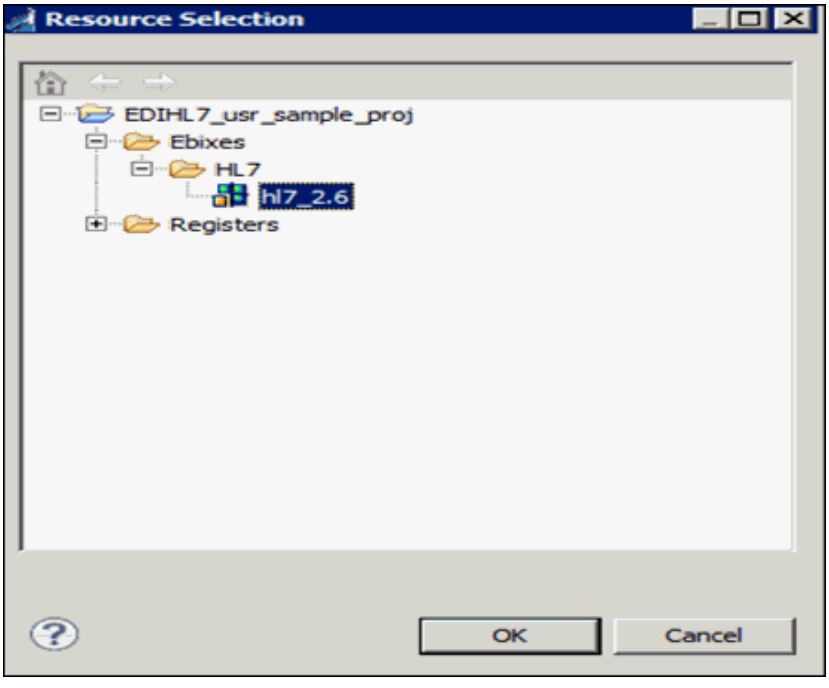


Procedure: How to Add an Ebix to an Outbound Channel as a Dependency

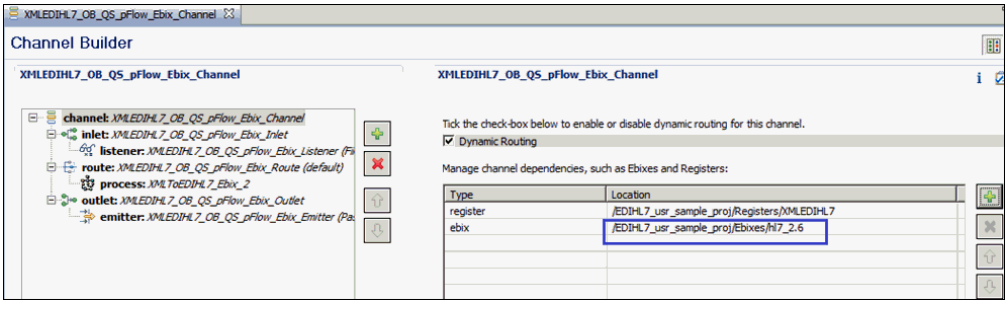
1. Click the *channel: XMLEDIHL7_OB_QS_pFlow_Ebix_Channel* node and then click the *Add dependency* icon on the right side panel, as shown in the following image.



2. Expand *EDIHL7_usr_sample_proj*, *Ebixes*, *HL7*, and then select ebix *hl7_2.6* and click *OK*, as shown in the following image.



3. Click the *Save* button to save your changes or press the keyboard shortcut (Ctrl+S) if you are using a Windows environment.

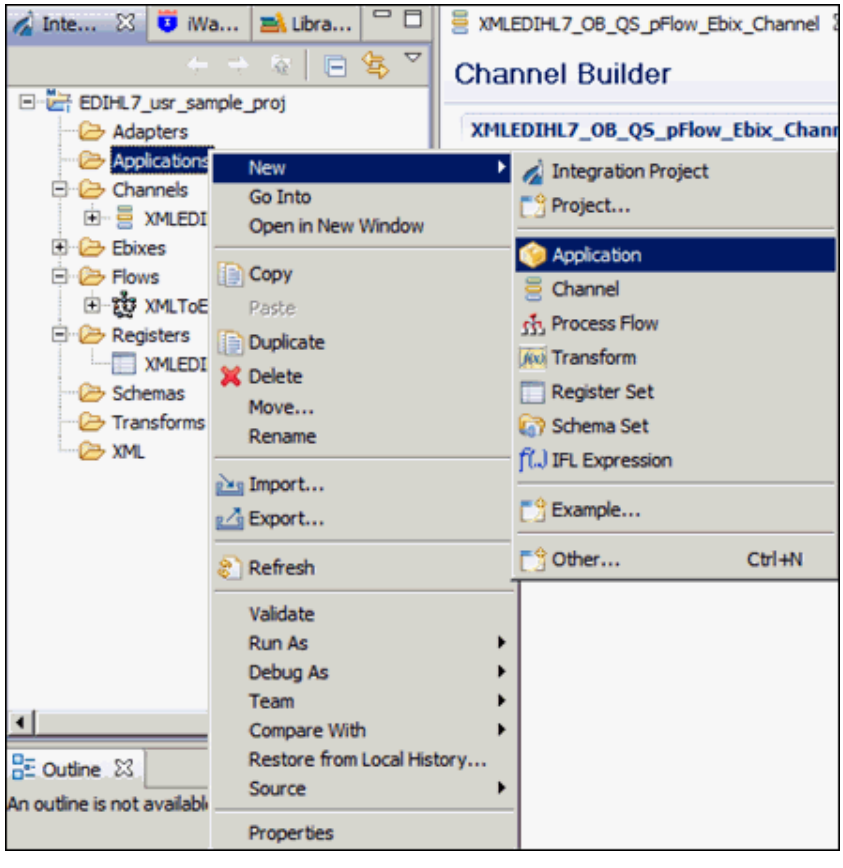


Configuring an iWay Integration Application for Outbound Processing

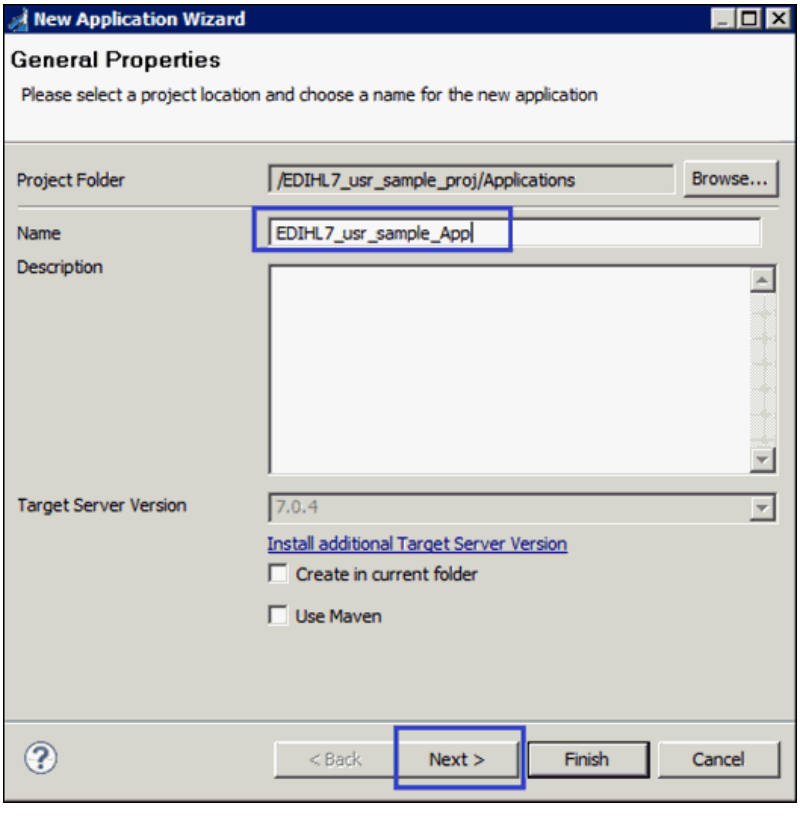
This section describes how to configure an iWay Integration Application (iIA) for outbound processing using iWay Integration Tools (iIT).

Procedure: How to Configure an iWay Integration Application for Outbound Processing

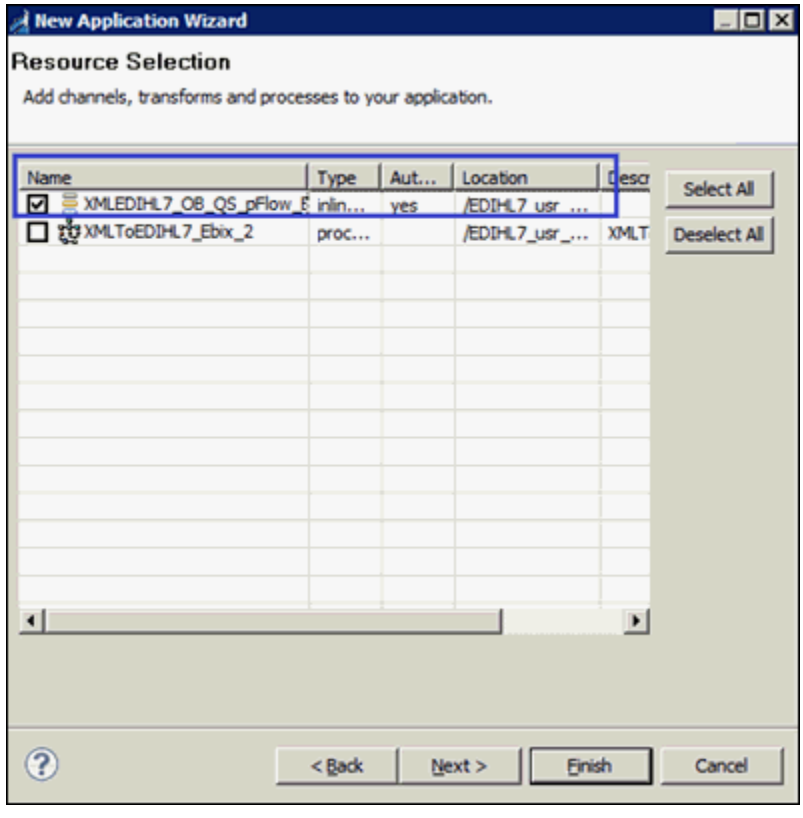
1. Right-click the Applications folder under the EDIHL7_usr_sample_proj integration project, select *New*, and then click *Application* from the context menu, as shown in the following image.



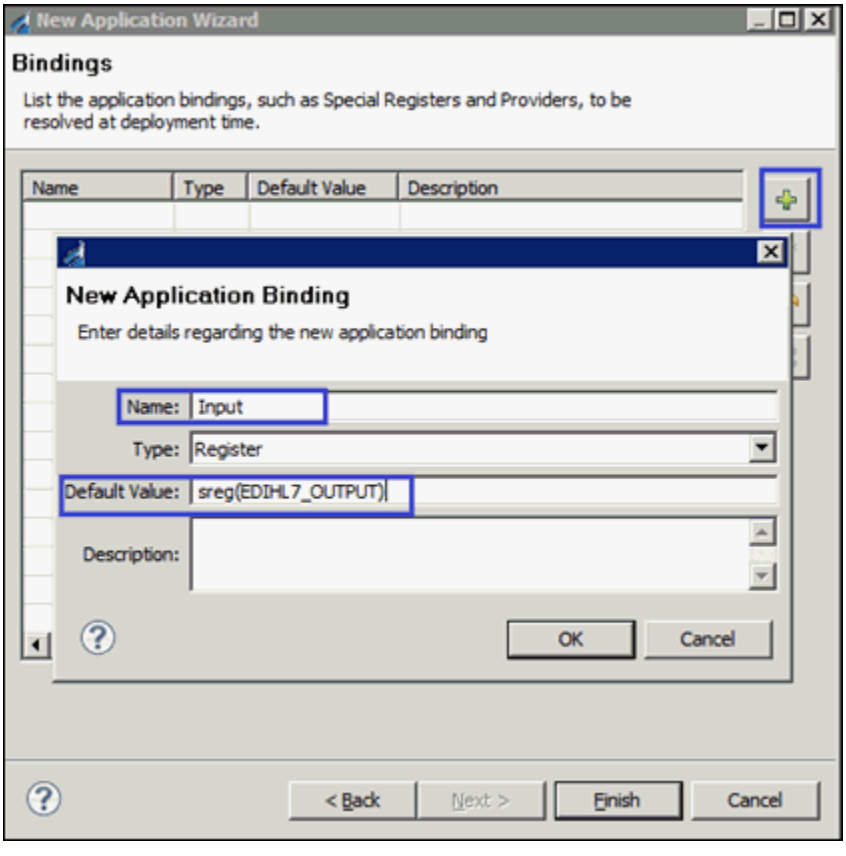
2. Enter a value in Name field (for example, EDIHL7_usr_sample_App), and click *Next*, as shown in the following image.



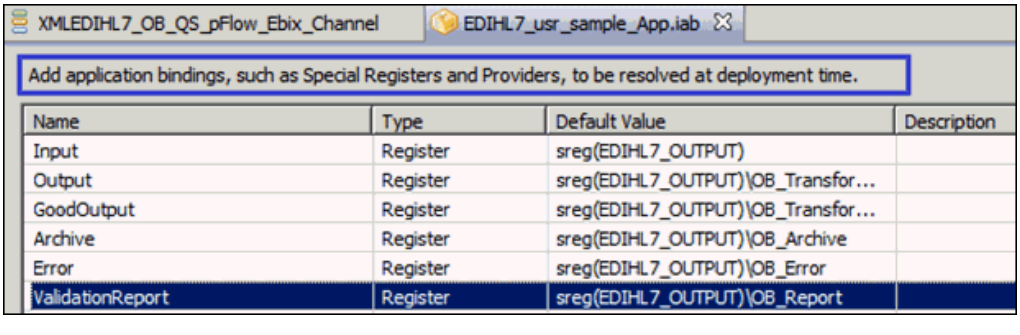
3. Select the *XMLEDIHL7_OB_QS_pFlow_Ebix_Channel* check box from Resource Selection pane and keep clicking *Next* until you get to the Bindings pane, as shown in the following image.



4. In the Bindings pane, add Bindings for Registers you have added in the XMLEDIHL7 Register Set to the iIA EDIHL7_usr_sample_App and click *Finish*, as shown in the following image.

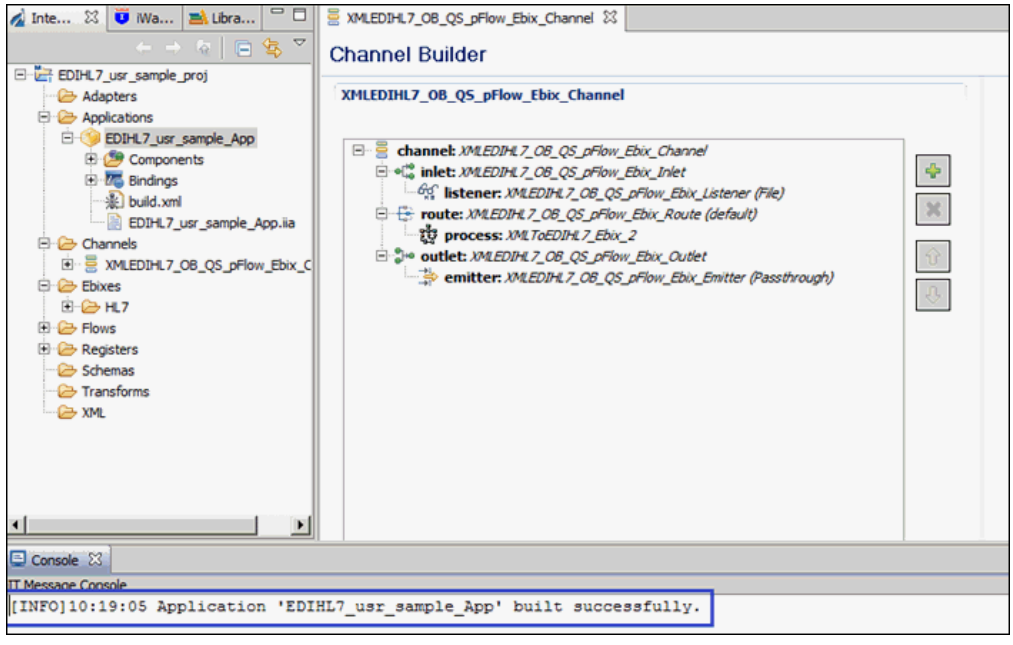


After adding all register bindings, your iIT screen should resemble the following image.



Name	Type	Default Value	Description
Input	Register	sreg(EDIHL7_OUTPUT)	
Output	Register	sreg(EDIHL7_OUTPUT)\OB_Transfor...	
GoodOutput	Register	sreg(EDIHL7_OUTPUT)\OB_Transfor...	
Archive	Register	sreg(EDIHL7_OUTPUT)\OB_Archive	
Error	Register	sreg(EDIHL7_OUTPUT)\OB_Error	
ValidationReport	Register	sreg(EDIHL7_OUTPUT)\OB_Report	

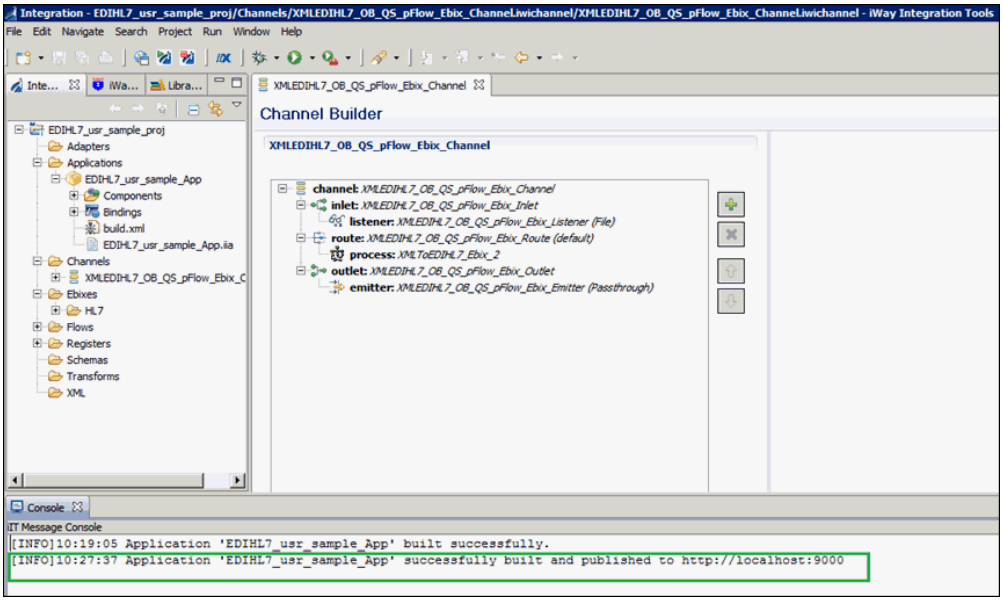
5. To build the application, right-click *EDIHL7_usr_sample_App*, select *Integration Tools*, and then click *Build* from the context menu, as shown in the following image.



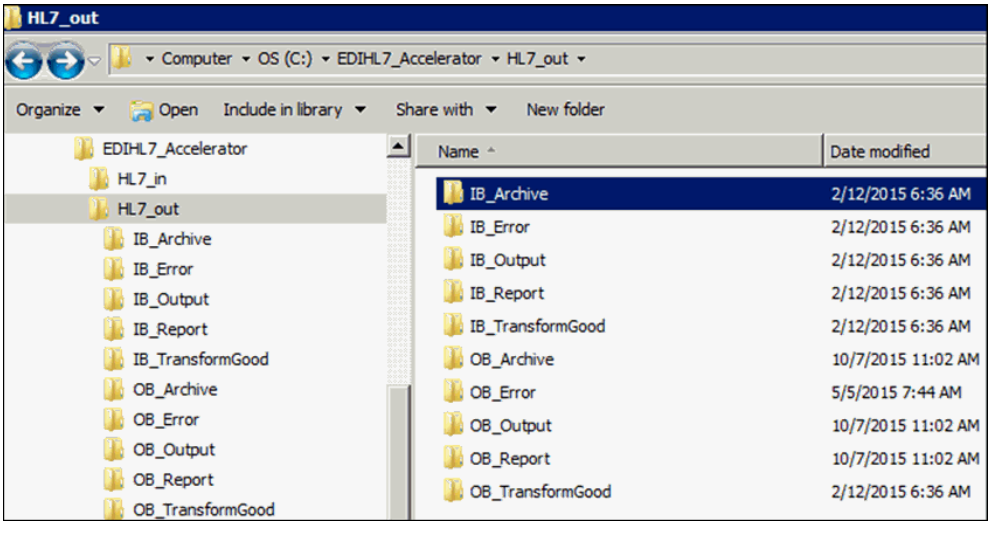
6. To publish the iIA, right-click *EDIHL7_usr_sample_App*, select *Integration Tools*, and then click *Publish to* from the context menu.

7. Provide the iSM server details in the Server URL text box and the other credentials, then click *Finish*.

The iIT page should resemble the following image.



8. Deploy the iIA application on the iSM registry.
9. Have the following folder structure created before starting your application in the iSM console, as shown in the following image.

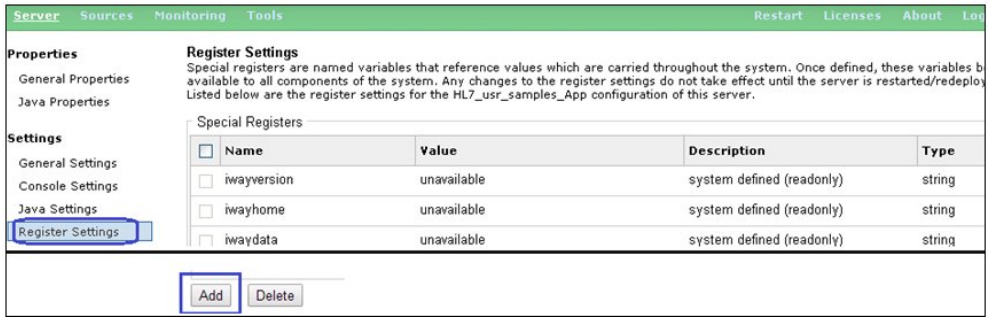


Setting EDIHL7 Outbound System Registers

This section describes how to set system registers using the iWay Service Manager (iSM) Administration Console.

Procedure: How to Set System Registers

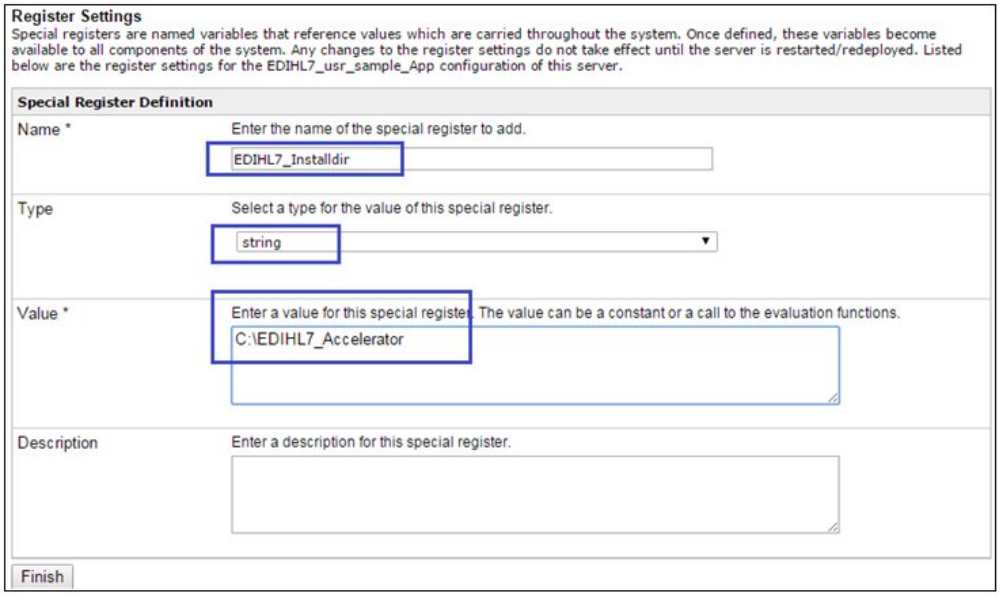
1. Open the iSM Administration Console and select *EDIHL7_usr_sample_App_OB [down]* from the Management drop-down list.
2. From the console bar, click *Server*, *Register Settings*, and then click *Add*, as shown in the following image.



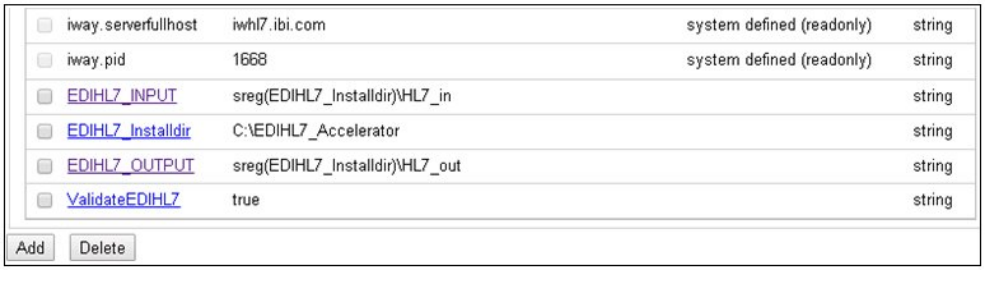
3. For the deployed EDIHL7_usr_sample_App_OB application, define the system registers as listed in the following table:

Register Name	Value
EDIHL7_INPUT	`sreg(EDIHL7_Installdir)\HL7_in`
EDIHL7_installdir	`C:\EDIHL7_Accelerator`
EDIHL7_OUTPUT	`sreg(EDIHL7_Installdir)\HL7_out`

Register Name	Value
ValidateEDIHL7	`true`



After adding the registers, your Register Settings page in the iSM Administration Console should resemble the following image.



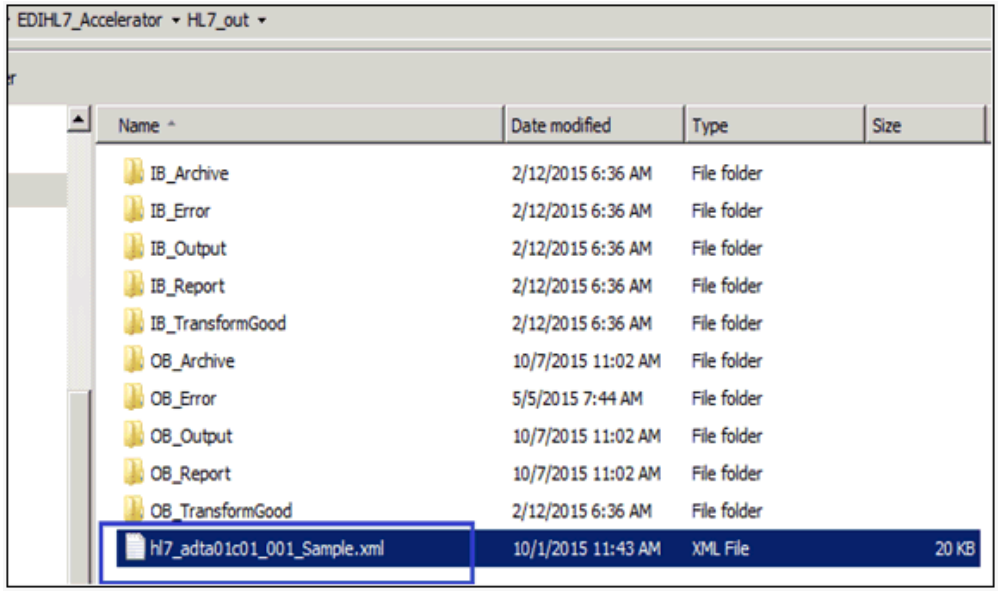
4. Start the deployed application in the iSM Administration Console and ensure that the channel is up and running in the Monitoring section of the console.

Testing the Outbound HL7 Channel Application

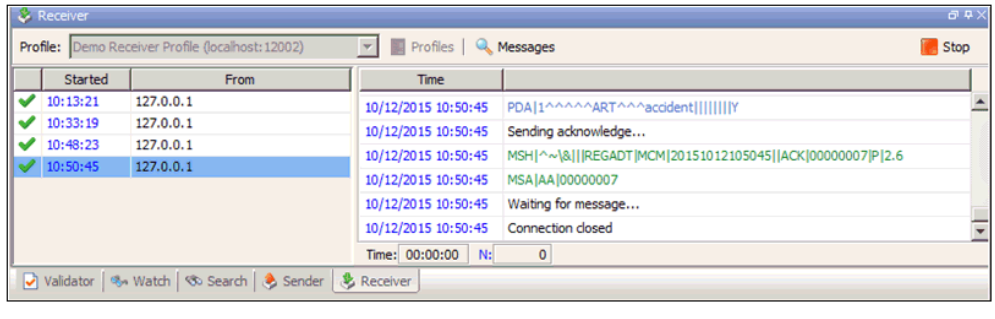
This section describes how to test the outbound channel application.

Procedure: How to Test the Outbound Channel Application

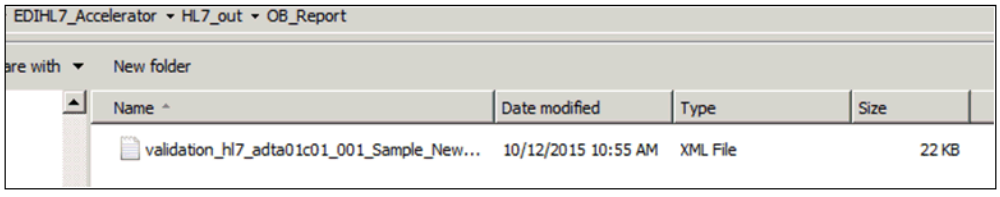
1. Copy and then paste your input XML file to the output directory that you have configured (for example, EDIHL7_Accelerator\HL7_out), as shown in the following image.



2. Look for the acknowledgement message in the EDIHL7_Accelerator\HL7_out\OB_Output directory.



3. Look for the validation report to be received in the output directory that you have configured (for example, EDIHL7_Accelerator\HL7_out\OB_report), as shown in the following image.



   Validation reports contain valid (good) or invalid (bad) HL7-formatted data, copies of the input files, and any error messages that may have occurred during the transformation. Typically the name of the validation report can inform you if the transformation has passed or failed. You can also configure the outbound processing to write the data to two different file locations, one for valid date and the other for invalid data.

4. If the input data contained any errors, you can review this error data in the output directory that you have configured for error handling (for example, EDIHL7_Accelerator\HL7_out \OB_Error).

5. After outbound processing has completed, a copy of the input data that was used during the transformation is stored under the archive directory that you specified (for example, EDIHL7_Accelerator\HL7_out\OB_Archive).

Chapter 8

Outbound Processing: XML to HL7 (Using MLLP)

This section describes how to configure a basic outbound message flow for the iWay Integration Solution for EDIHL7 using the Minimal Lower Layer Protocol (MLLP). The message flow represents the movement and tasks in the conversion of a message from XML format to HL7 format.

In this chapter:

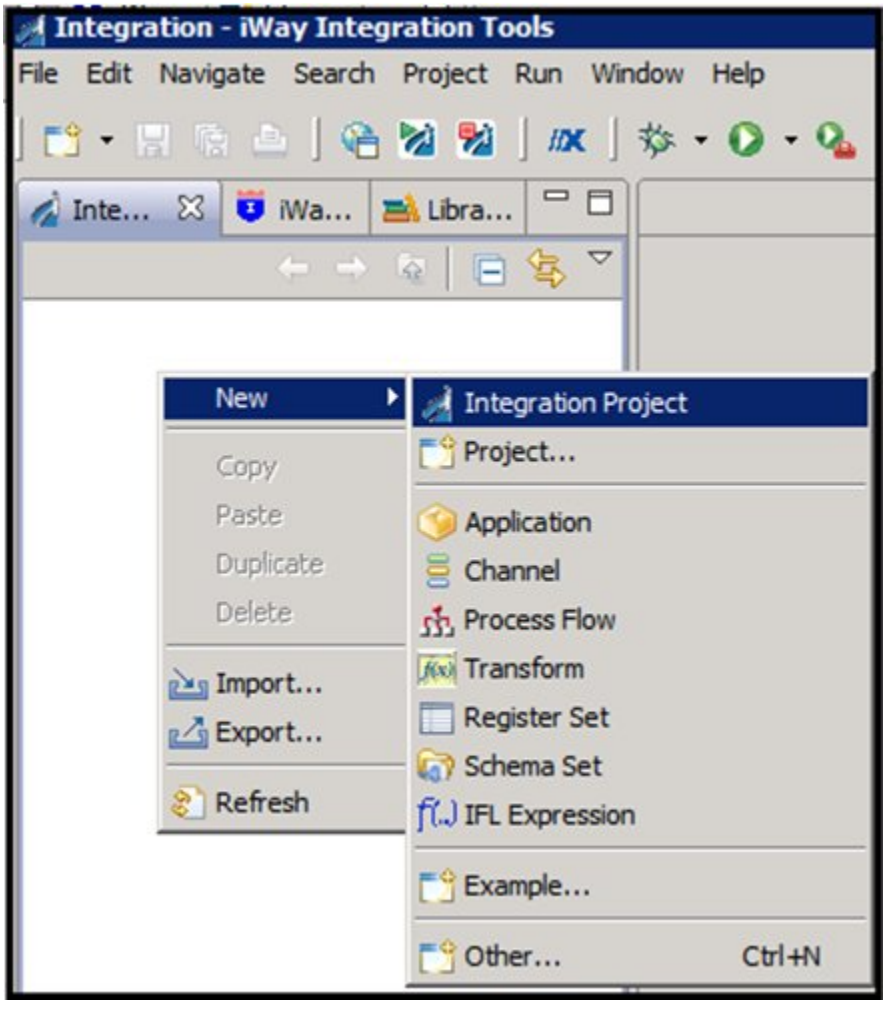
Configuring a Channel for HL7 Outbound Processing Using MLLP

The outbound channel creates an HL7 message from XML and a XML-formatted validation report. The documents are routed to designated folders based on the success or failure results of the transformation.

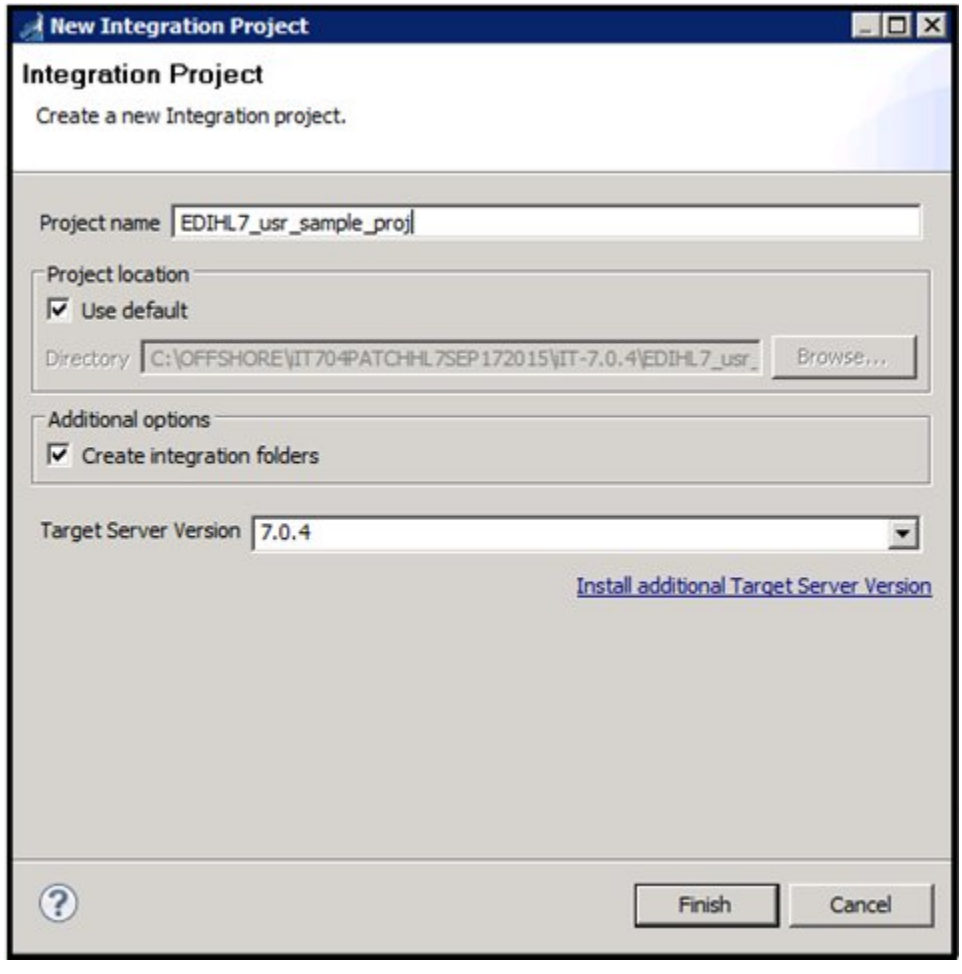
Procedure: How to Create a Channel for Outbound Processing

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.



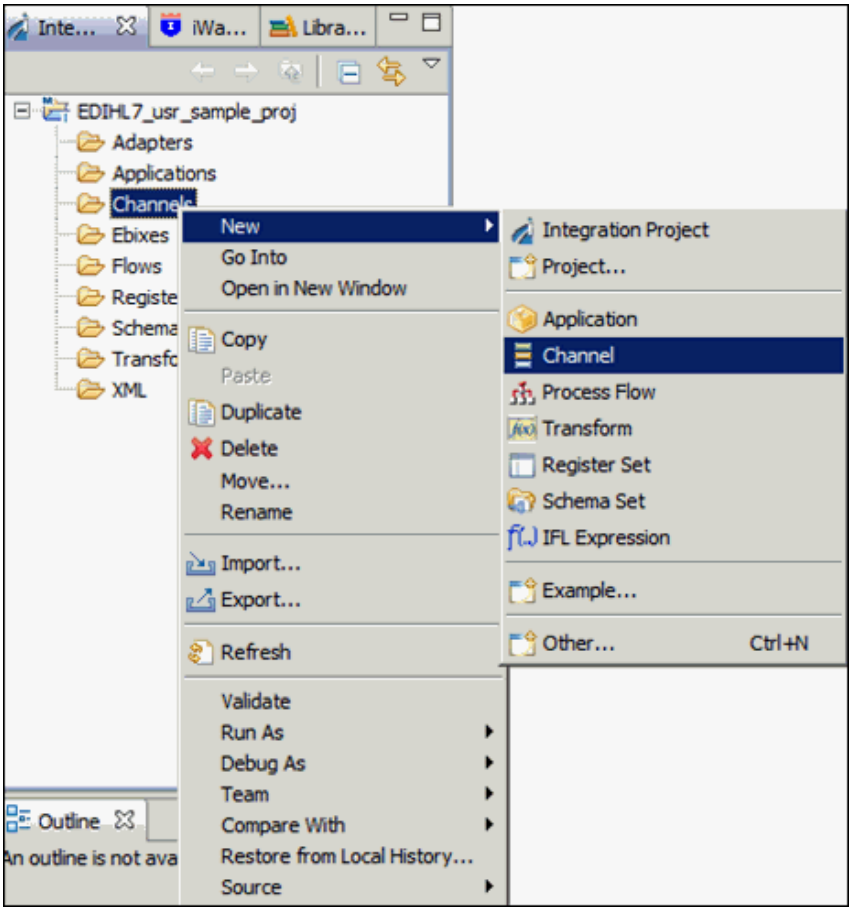
The New Integration Project dialog box opens, as shown in the following image.



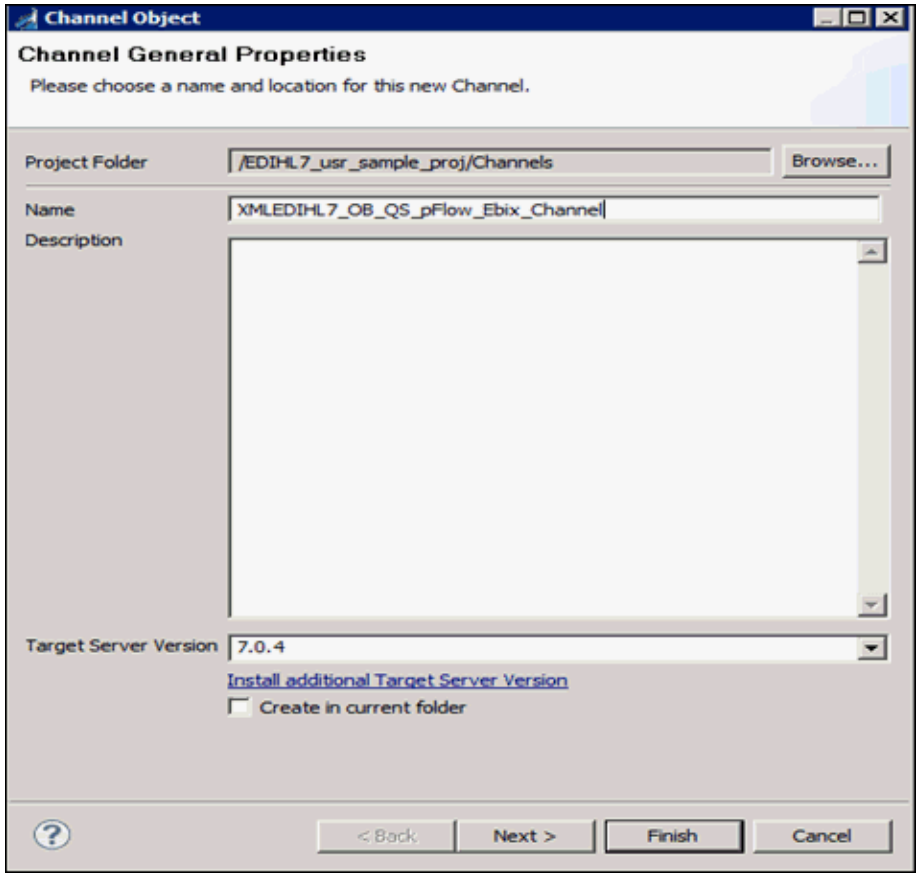
3. Enter a name for the Integration Project (for example, *EDIHL7_usr_sample_proj*) in the Project name field, and then click *Finish*.

   A new Integration Project node called *EDIHL7_usr_sample_proj* is added to the Integration Explorer.

4. Expand the *EDIHL7_usr_sample_proj* Integration Project node, right-click the *Channels* folder, select *New*, and then click *Channel* from the context menu, as shown in the following image.

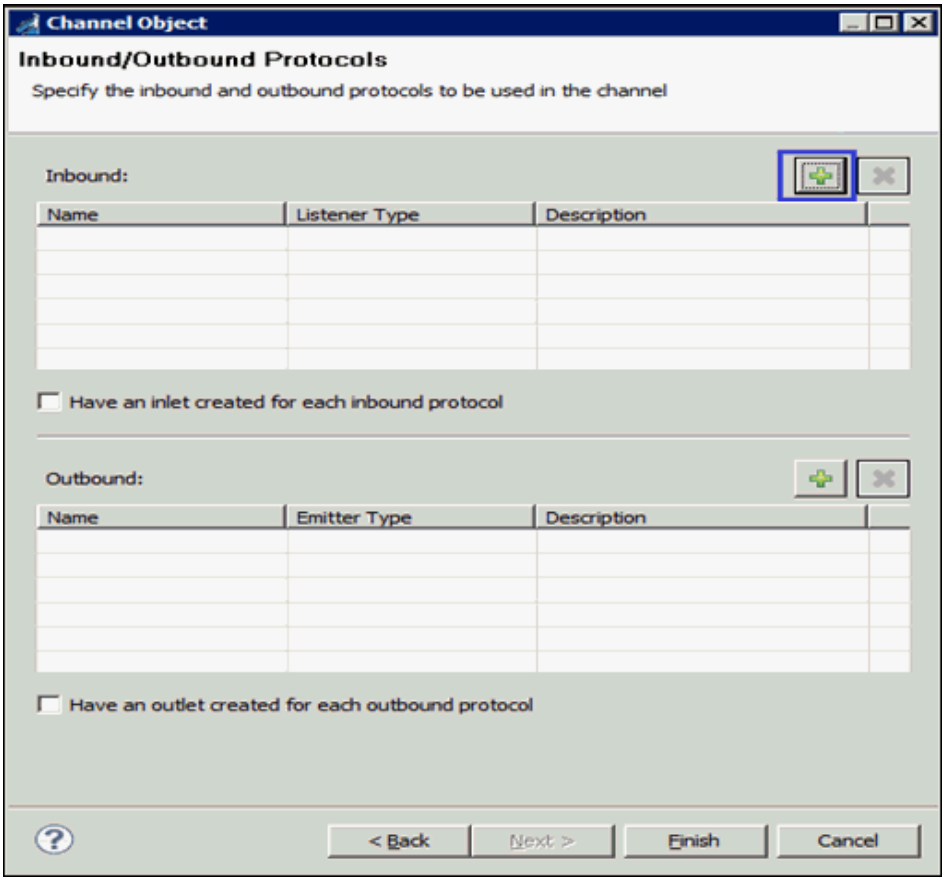


5. Enter a name for the Channel (for example, XMLEDIHL7_OB_QS_pFlow_MLLP_Channel), and then click *Next*, as shown in the following image.

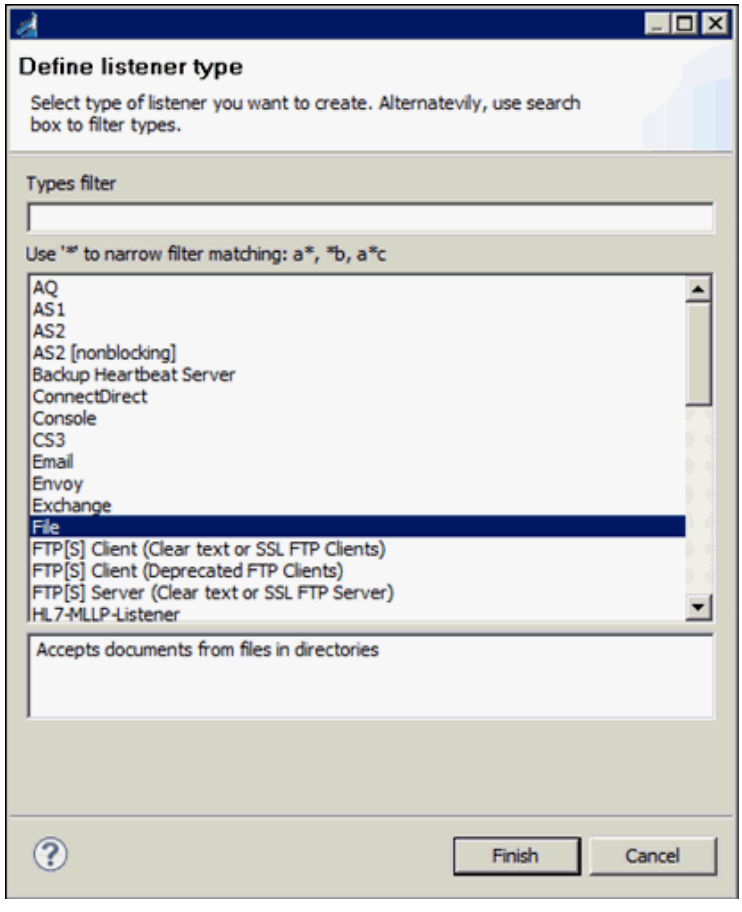


The Inbound/Outbound Protocols pane appears.

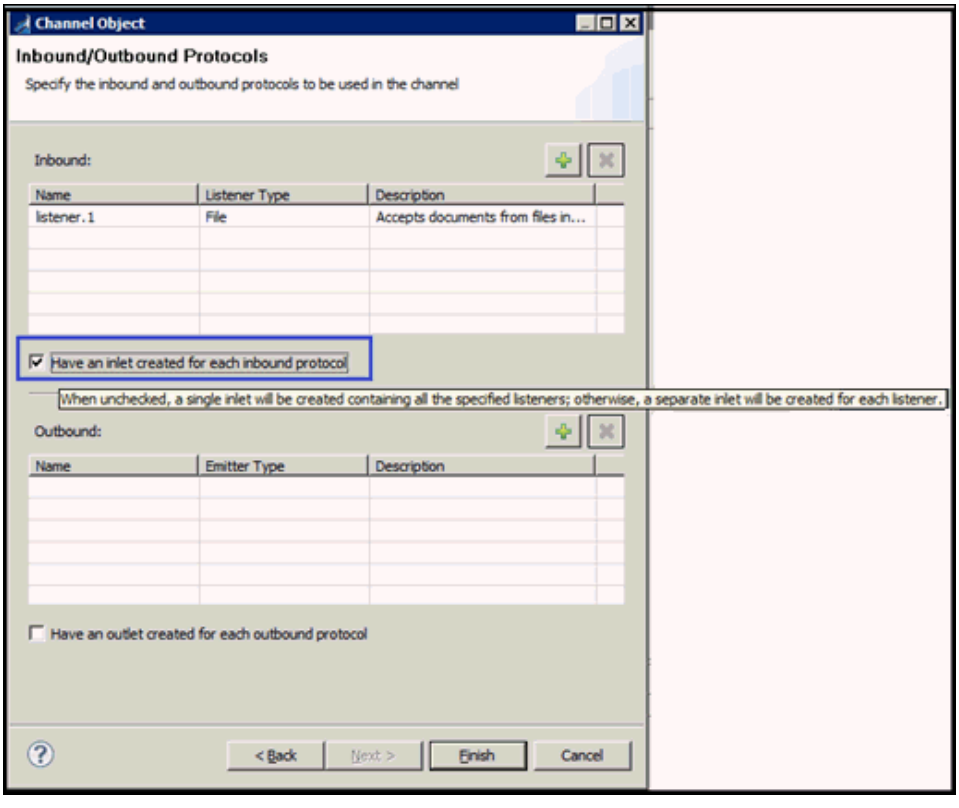
6. Click the *Add* button to add a File Listener in the Inbound section, as shown in the following image.



7. From the Types Filter list, select *File* and then click *Finish*, as shown in the following image.



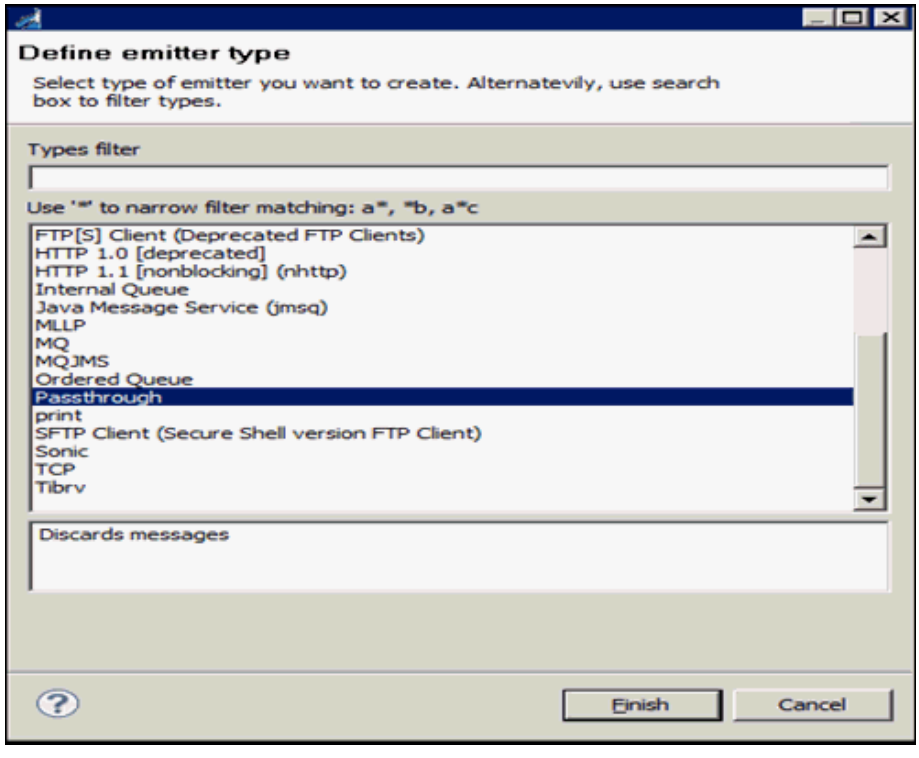
8. Select the *Have an Inlet created for each inbound protocol* check box to create an Inlet for the channel, as shown in the following image.



9. Click the *Add* button in the Outbound section to define an emitter.

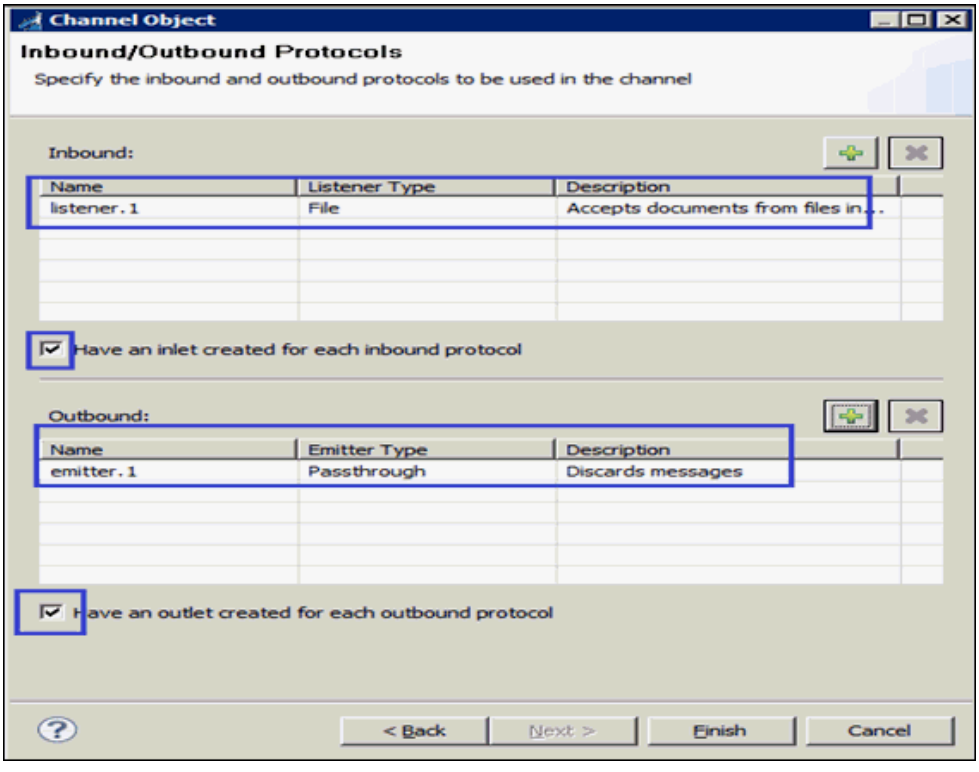
   The Define emitter type pane appears.

10. In the Types filter list, select *Passthrough* to define the passthrough emitter, and then click *Finish*, as shown in the following image.

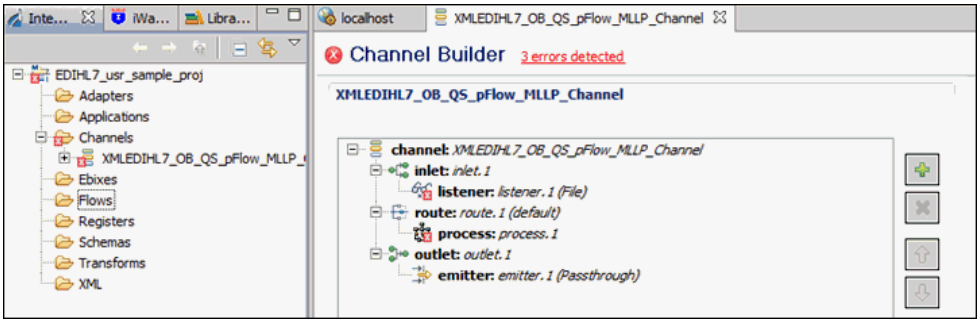


Note: Passthrough does not emit data from the listener. Instead, it just passes the control here and does nothing.

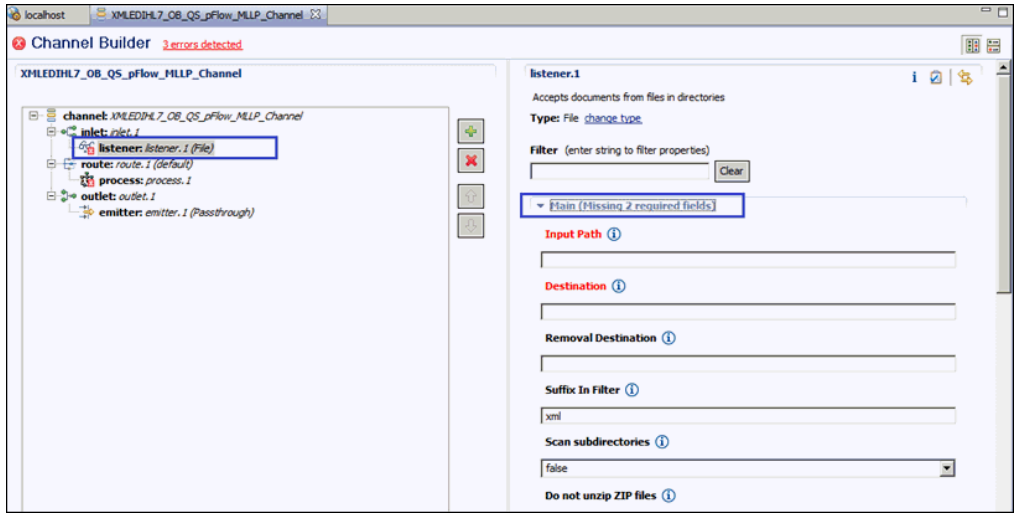
11. Select the *Have an outlet created for each outbound protocol* check box to create an outlet for the channel, and then click *Finish*, as shown in the following image.



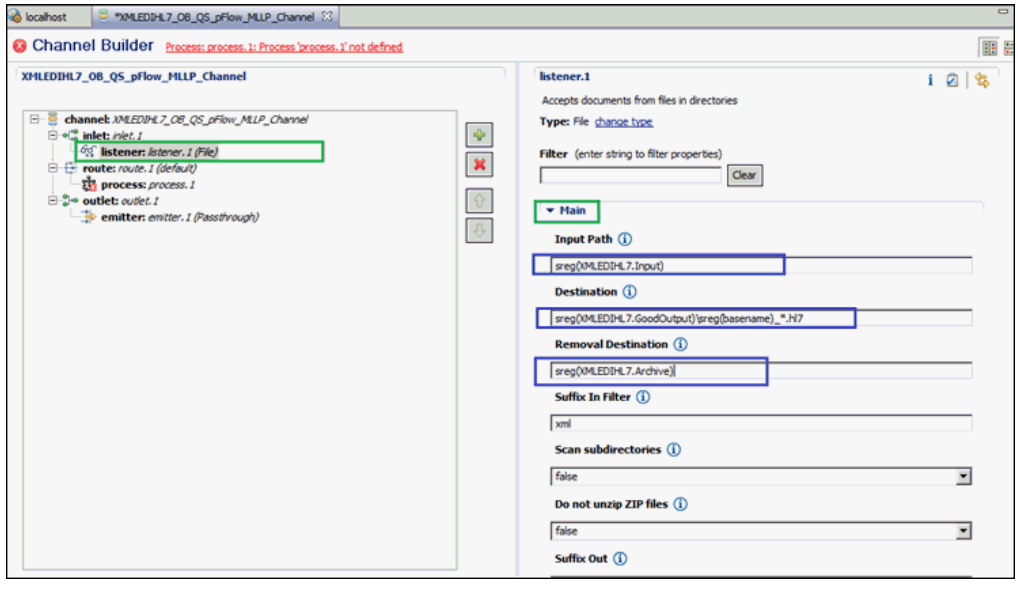
The Channel Builder pane appears.



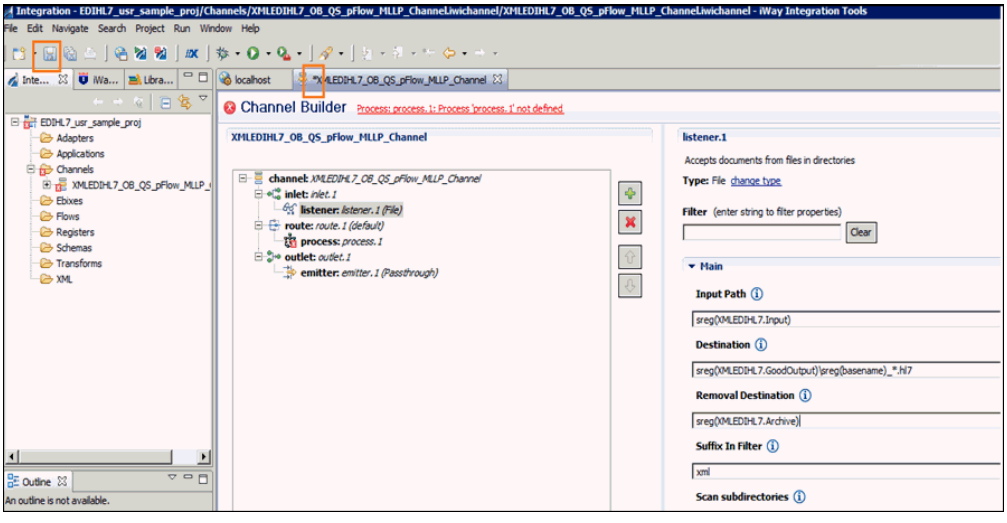
12. Under *inlet: inlet.1*, click *listener: listener.1 (File)* and then expand the *Main (Missing 2 required fields)* configuration parameter on the right pane, as shown in the following image.



13. Enter a valid Input Path, Destination, and Removal Destination (optional) path and then select *hl7* in the Suffix Out drop-down list, as shown in the following image.

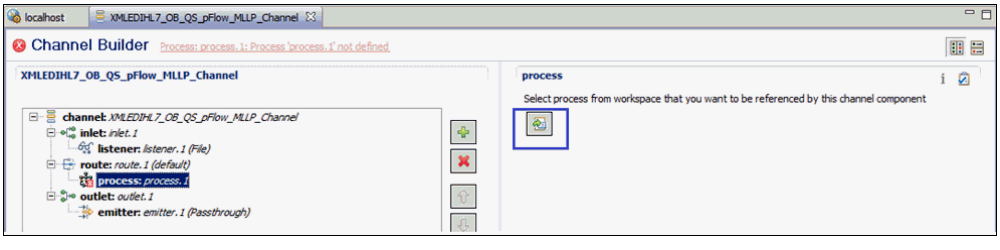


14. Click the *Save* icon near the File menu to save your edited listener configuration. You can also press the shortcut key (Ctrl+S) to save your work if you are using a Windows environment.

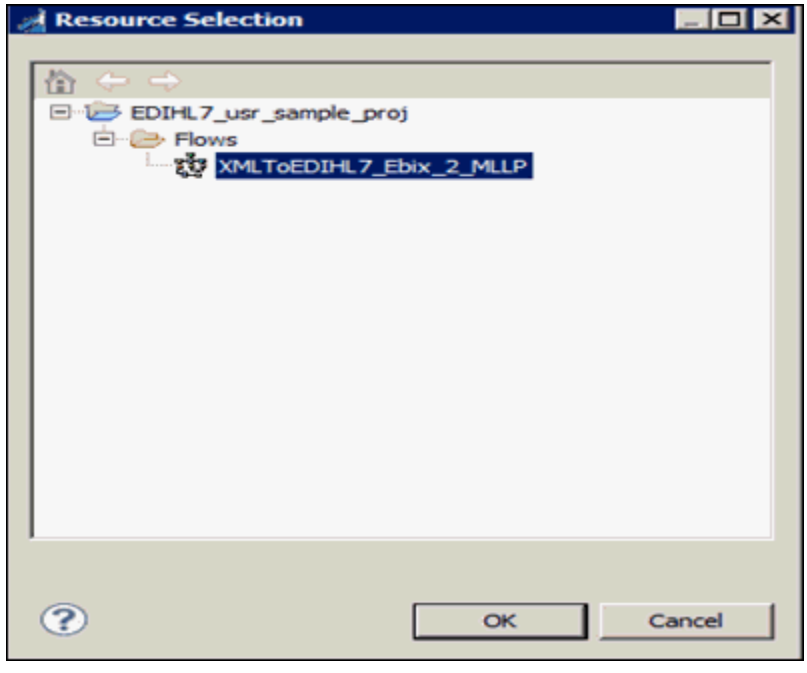


15. Import the *XMLToEDIHL7_Ebix_2_MLLP* process flow from the local drive or create the one in the EDIHL7_usr_sample_proj directory in the Flows folder.

16. Under the *route: route.1(default)* node, select *process: process.1* and click the process icon on the right panel to reference the process flow into your channel, as shown in the following image.



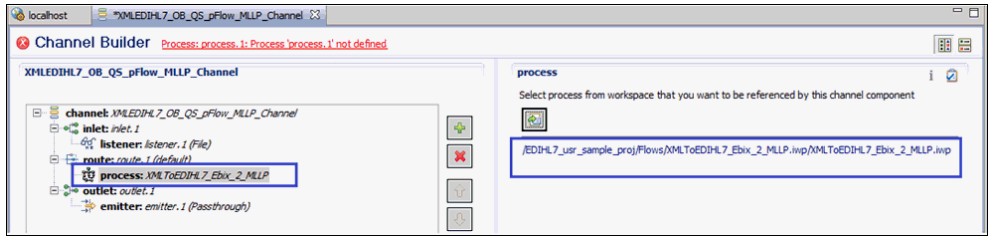
17. Select a process flow from the integration project and then click *OK*.



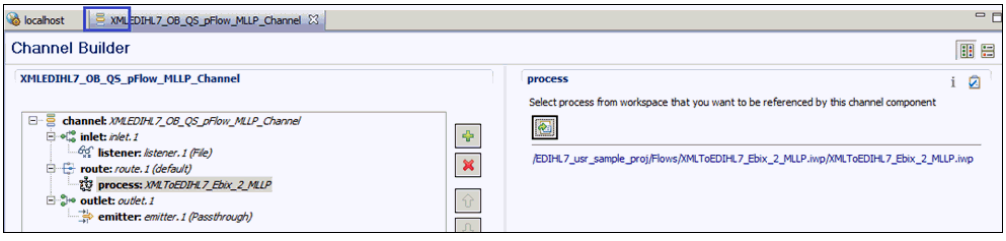
Note: Process flows should already be built and available in the iIT integration project. They can be found in the EDIHL7_usr_sample_proj directory inside the Flows folder.

For more information, see *Process Flow Used for Outbound Processing: XML to HL7 (MLLP)* on page 204.

Your screen should resemble the following image.



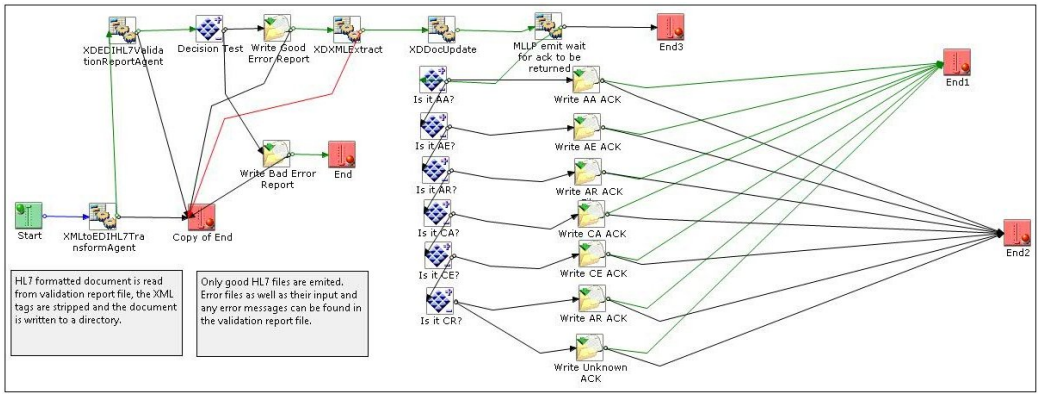
18. Click the *Save* icon near the File menu to save your edited listener configuration. You can also press the shortcut key (Ctrl+S) to save your work if you are using a Windows environment.



Reference: Process Flow Used for Outbound Processing: XML to HL7 (MLLP)

This section provides an overview of the process flow used for outbound processing: XML to HL7 (MLLP). This process flow (XMLToEDIHL7_Ebix_2_MLLP) is already built and available in the iIT integration project. It is located under the *EDIHL7_usr_sample_proj* node inside the Flows subfolder.

The following image shows the entire outbound process flow, including all of the nodes that are used and their connections.



In this process flow, an HL7 formatted document is read from a validation report file. The XML tags are stripped and the document is written to a directory. Only valid HL7 files are emitted. Error files as well as their input and any error messages can be found in the validation report file.

Reference: _hl7ack(): Parse HL7 Acknowledgement Message

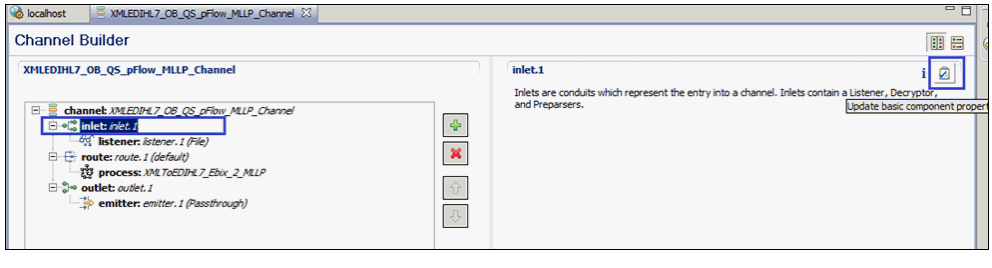
The _hl7ack() iFL function is used in the outbound MLLP process flow. This function parses a HL7 acknowledgement message to return a specific element value. It uses the following format:

```
_hl7ack(query[,defaultValue])
```

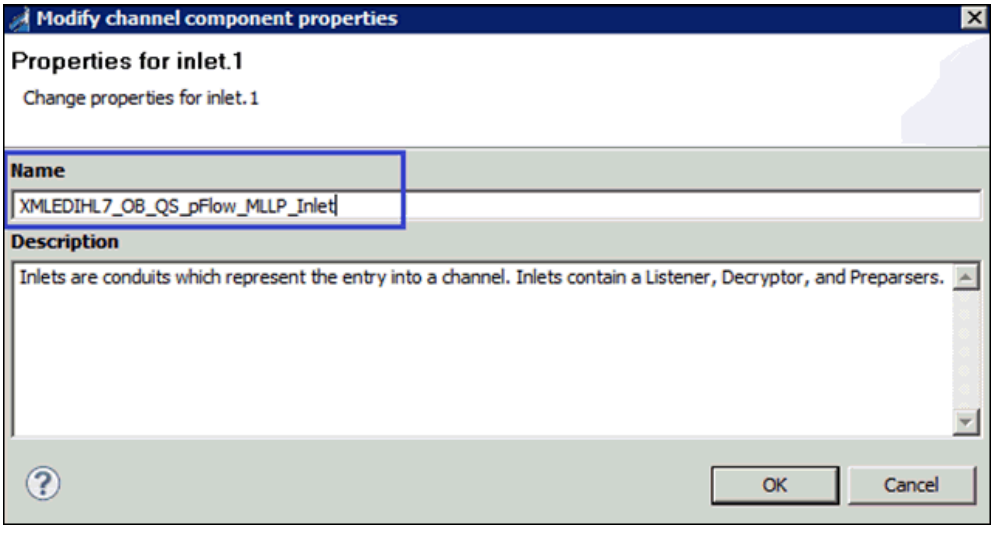
Property	Description
query (Required)	The query property is the name of a specific element in the HL7 acknowledgement message. Specify one of the following allowed values: ❑ **ackcode**. Acknowledgment Code (#01 in segment MSA). ❑ **ctrlid**. Control ID (#02 in segment MSA). ❑ **sndapp**. Sending Application (#03 in segment MSH). ❑ **sndfacil**. Sending Facility (#04 in segment MSH). ❑ **recapp**. Receiving Application (#05 in segment MSH). ❑ **recfacil**. Receiving Facility (#06 in segment MSH).
defaultValue (Optional)	The default value for the queried element.

Procedure: How to Update Basic Details for Channel Components

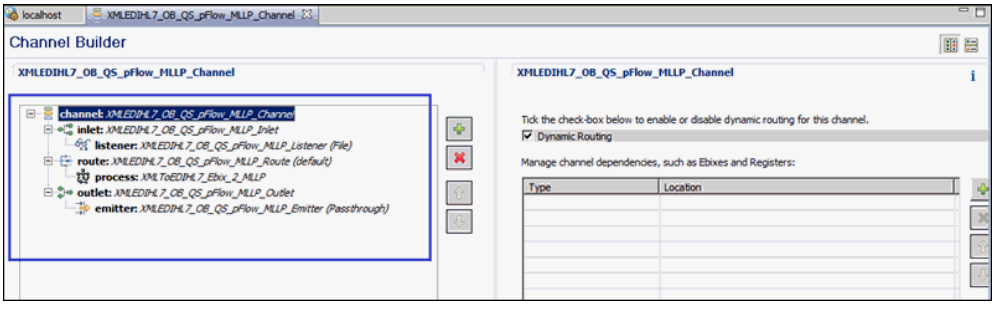
1. Select the channel component *inlet:inlet.1* from the channel and click the update channel component icon on the top right side panel, as shown in the following image.



2. Rename or update the inlet details (for example, XMLEDIHL7_OB_QS_pFlow_MLLP_Inlet), and click *OK*, as shown in the following image.



After renaming or changing the channel components, your iIT Channel Builder should resemble the following image.

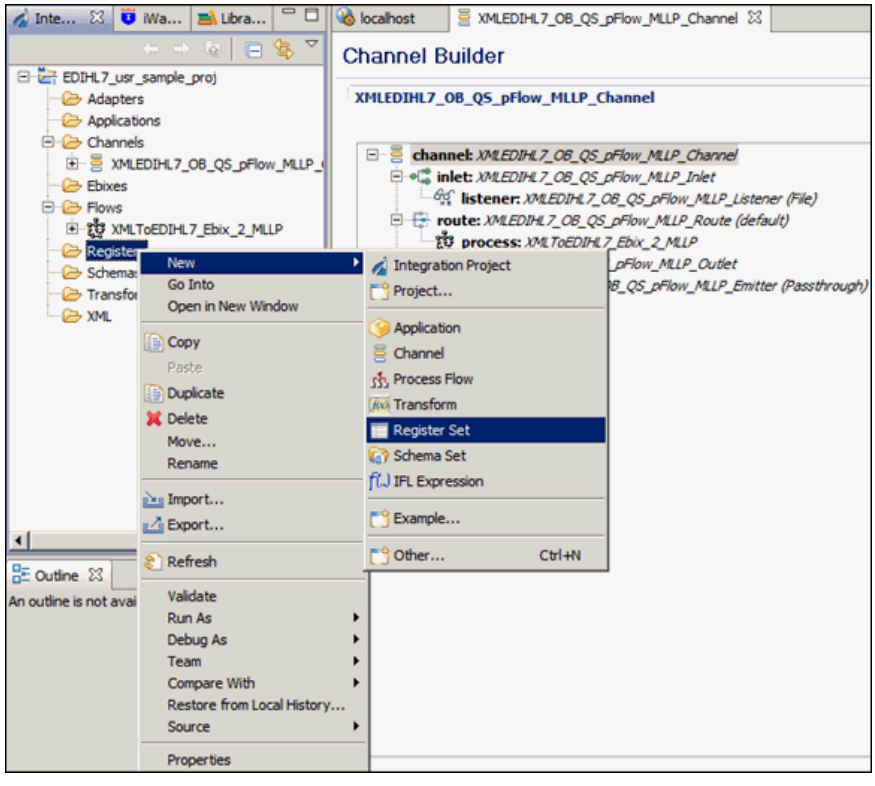


Configuring Register Sets and Registers

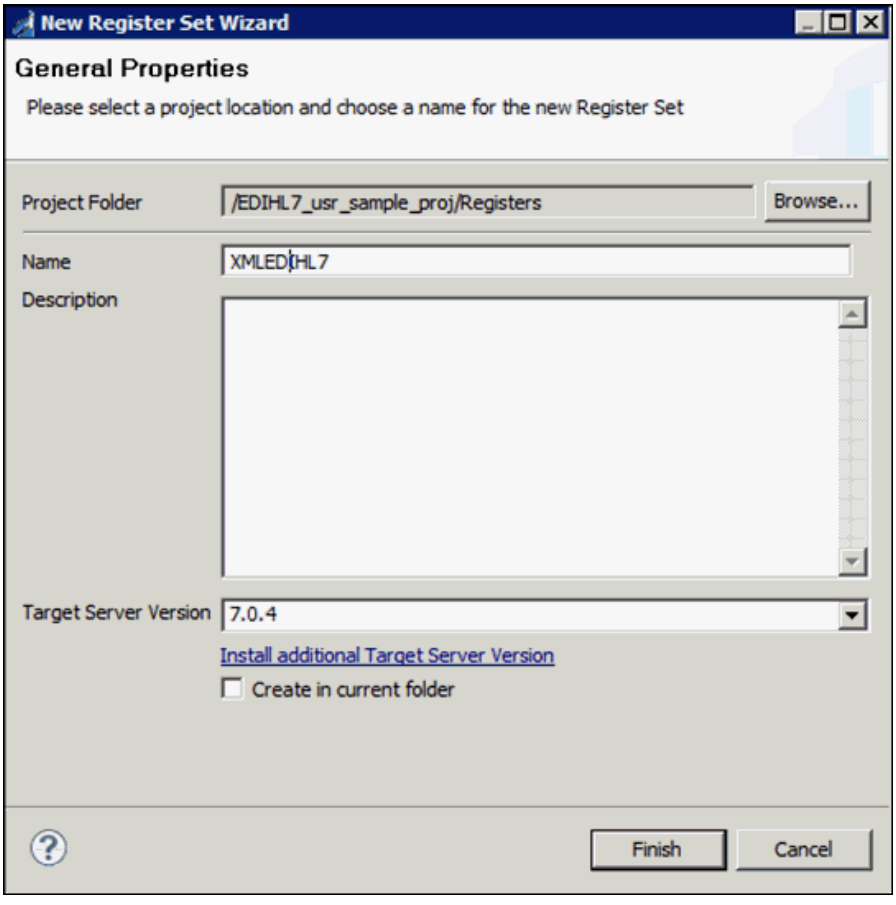
This section describes how to configure register sets and registers using iWay Integration Tools (iIT).

Procedure: How to Configure Register Sets and Registers

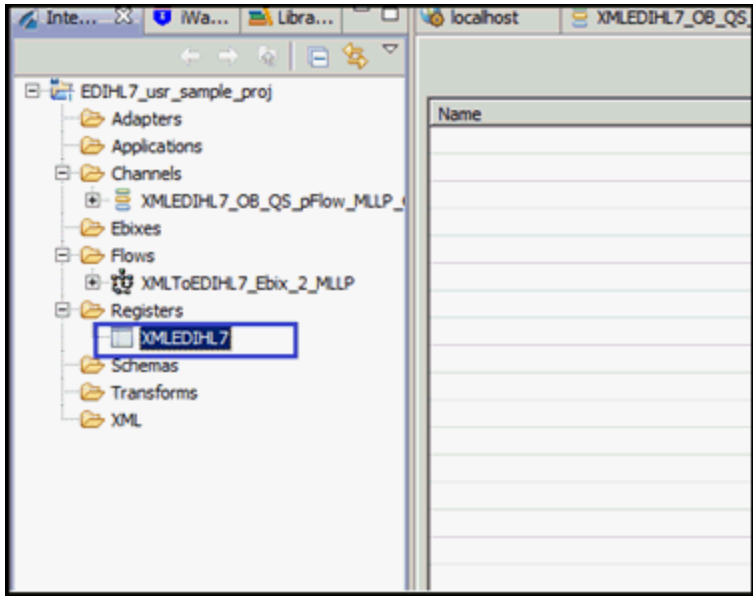
1. In the EDIHL7_usr_sample_proj project tree, right-click the Registers folder, and select *New*, and then click *Register Set* from the context menu, as shown in the following image.



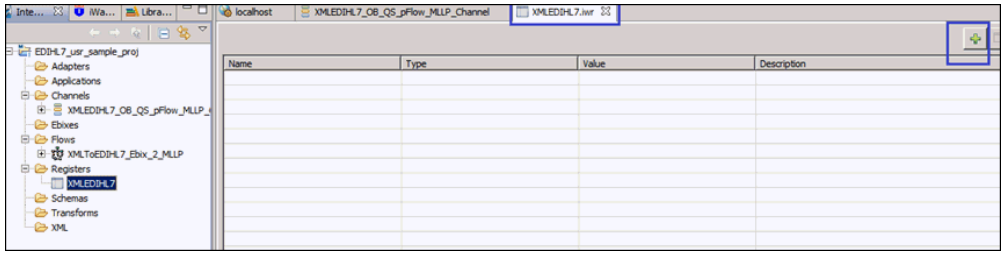
2. In the Name field, enter a name for the register set and click *Finish*, as shown in the following image.



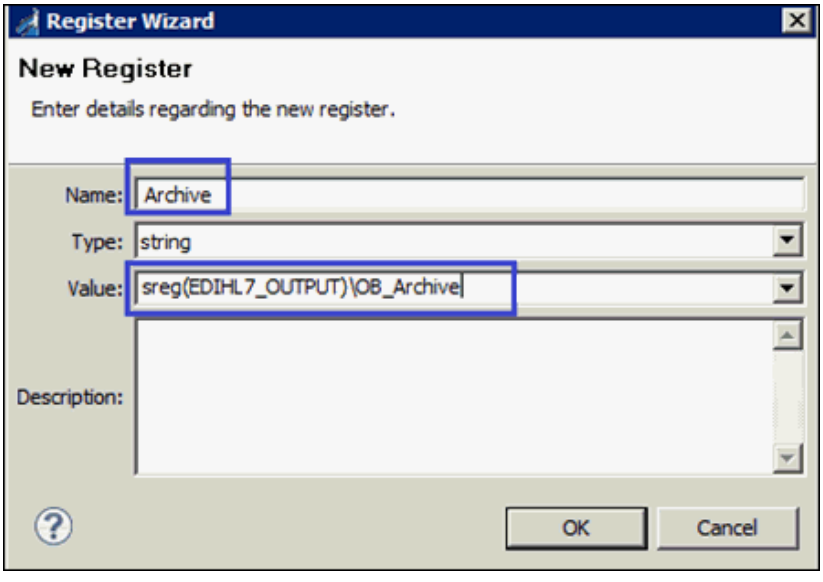
The new register set appears under Registers in the Registers folder, as shown in the following image.



3. Click the *Add a property* icon to add a register to the register set.



4. Enter a name for the new register in the Name field, select a register type from the Type drop-down list (set to string by default), and then enter a value in the Value field, as shown in the following image.



5. Click *OK*.

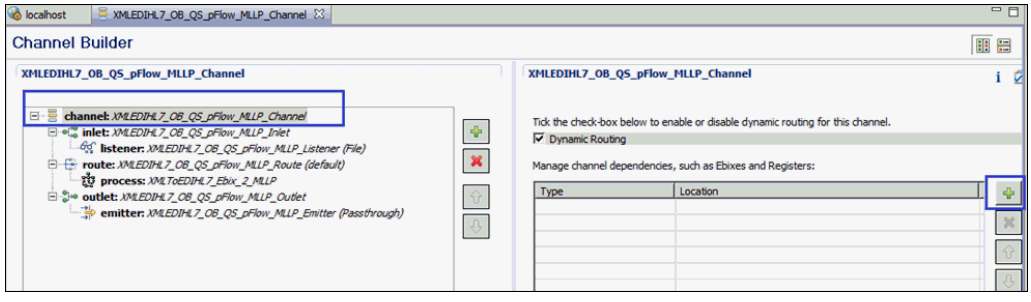
6. Create the following registers under the XMLEDIHL7 registers set you just created, along with the values shown in the table below:

Register Name	Value
Input	Name=Input, Value=sreg(EDIHL7_OUTPUT)
Output	Name=GoodOutput, Value=sreg(EDIHL7_OUTPUT) \OB_TransformGood
Archive	Name=Archive, Value=sreg(EDIHL7_OUTPUT)\OB_Archive
ValidationReport	Name=ValidationReport, Value=sreg(EDIHL7_OUTPUT) \OB_Report
Error	Name=Error, Value=sreg(EDIHL7_OUTPUT)\OB_Error
GoodOutput	Name=Output, Value=sreg(EDIHL7_OUTPUT) \OB_TransformGood

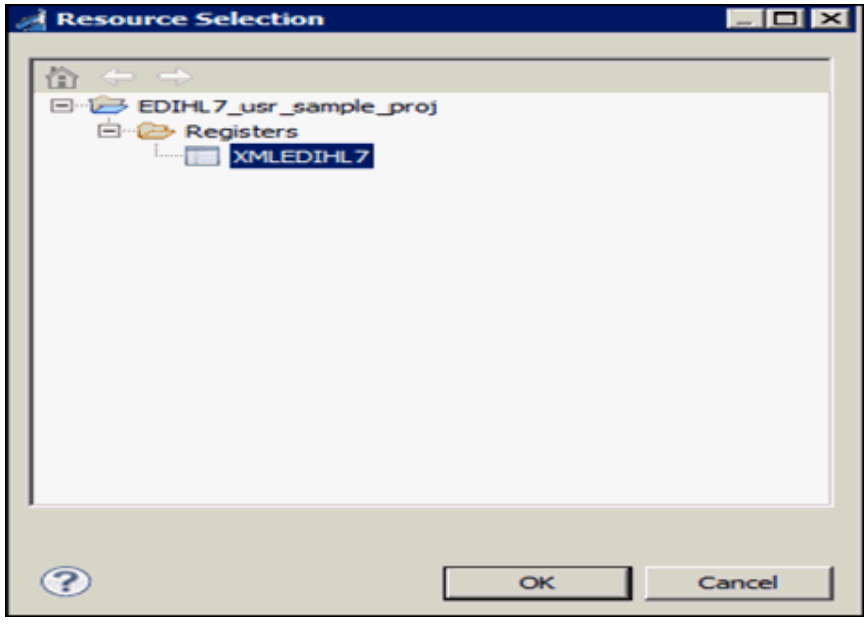
Register Name	Value
Ack	Name=Ack, Value=sreg(EDIHL7_OUTPUT)\OB_OUTPUT

Procedure: How to Add a Register Set to an Outbound Channel as a Dependency

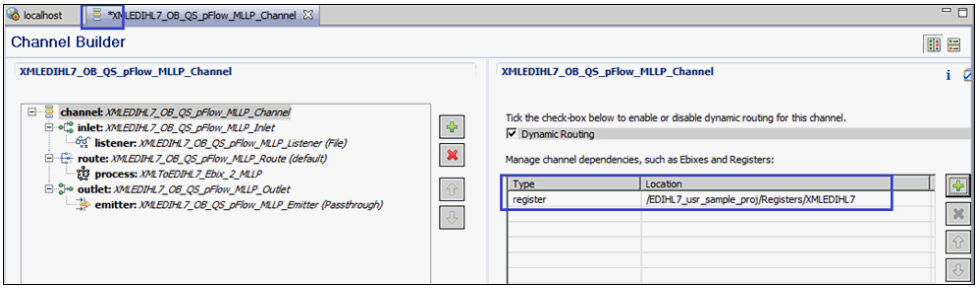
1. Click the *channel: XMLEDIHL7_OB_QS_pFlow_MLLP_Channel* node and then click the Add dependency icon on the far right side panel of the channel properties pane, as shown in the following image.



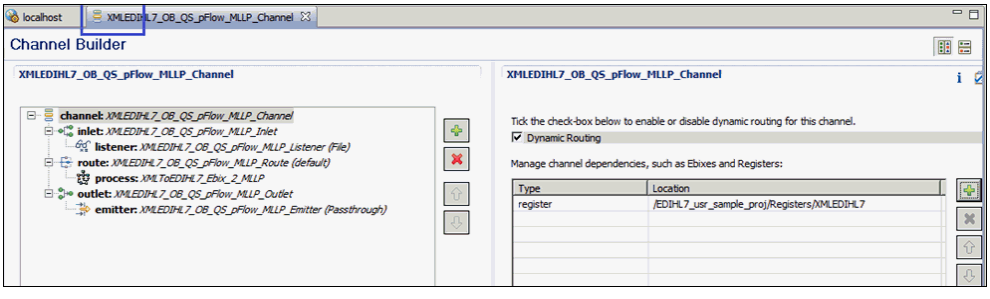
2. From the Registers folder under the integration folder, select *XMLEDIHL7* and click *OK*.



The iIT page should resemble the following image.



3. Click the *Save* icon to save your changes. You can also use the keyboard shortcut (Ctrl+S) if you are using a Windows environment.

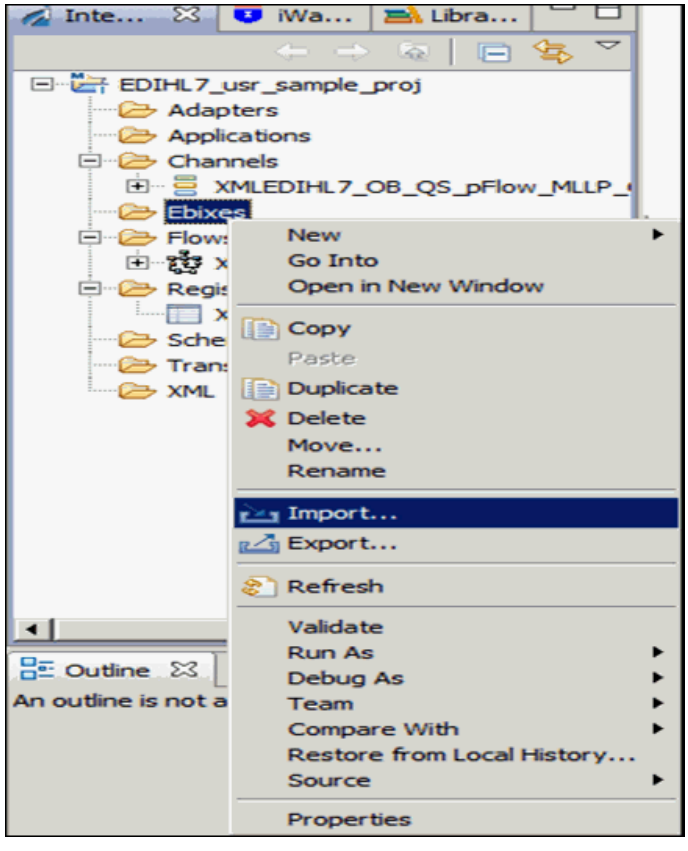


Importing an Ebix Into the Workspace

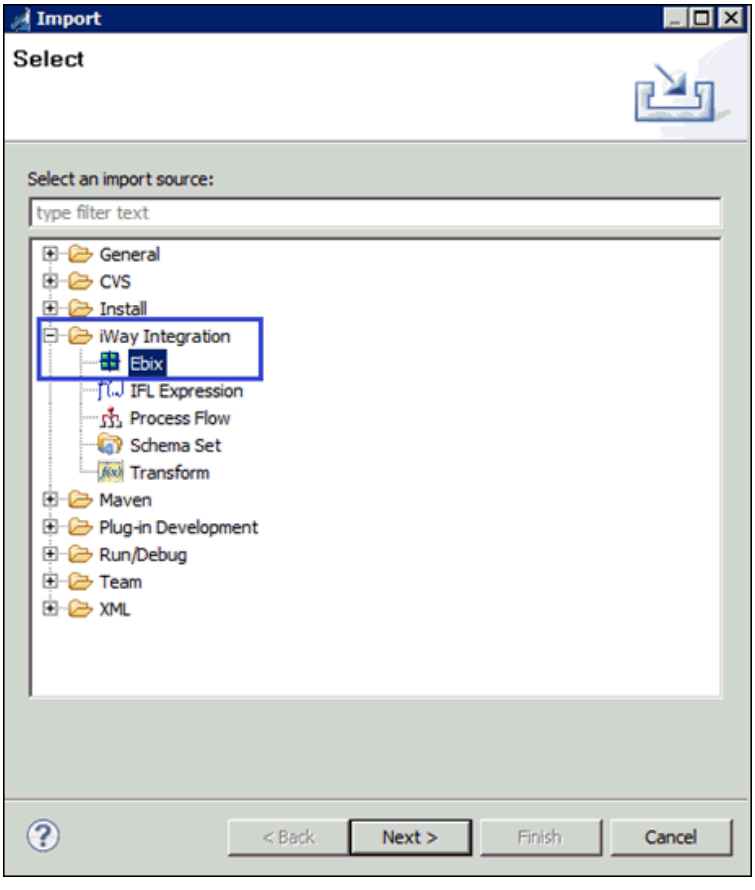
This section describes how to import an Ebix into the workspace using iWay Integration Tools (iIT).

Procedure: How to Import an Ebix Into the Workspace

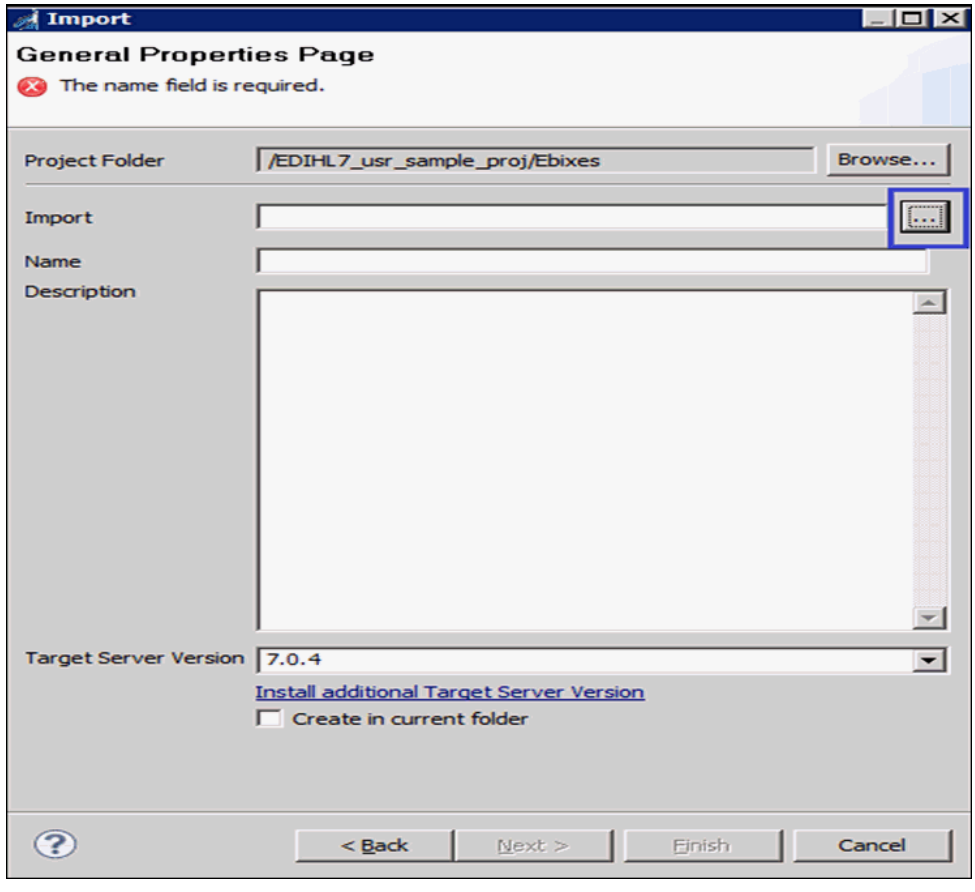
1. In the integration project EDIHL7_usr_sample_proj, right-click the Ebixes folder and then select *Import* from the context menu, as shown in the following image.



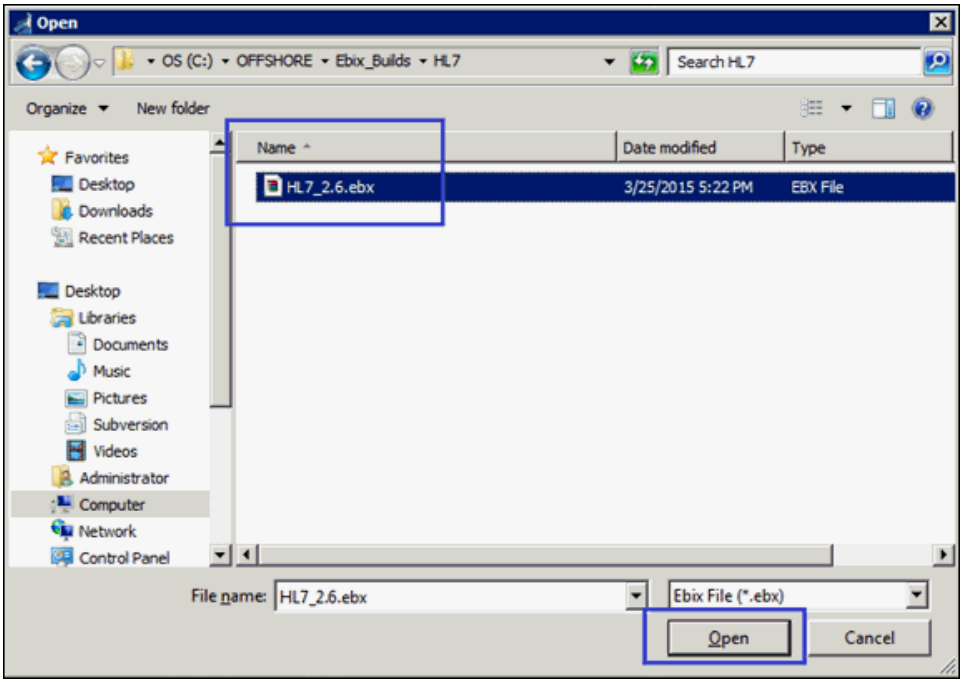
2. Expand the iWay Integration folder, select *Ebix*, and then click *Next*, as shown in the following image.



3. Click the ellipses (...) button to browse and import the ebix from a specific folder location, as shown in the following image.

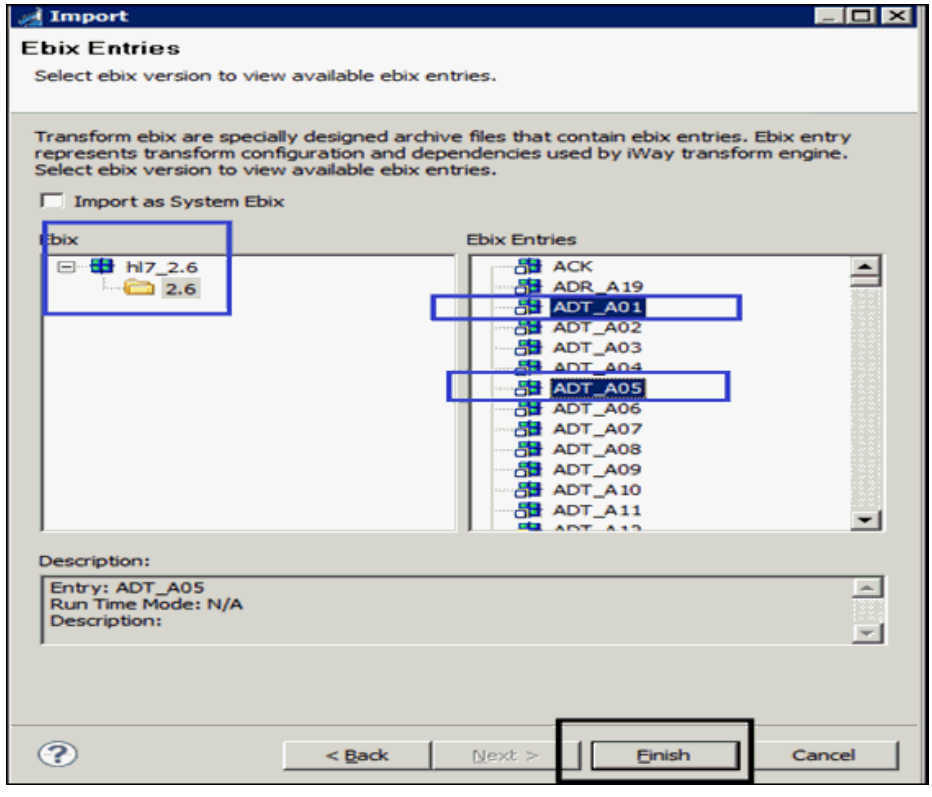


4. Select *HL7_2.6 ebix* from the folder location and click *Open*, as shown in the following image.

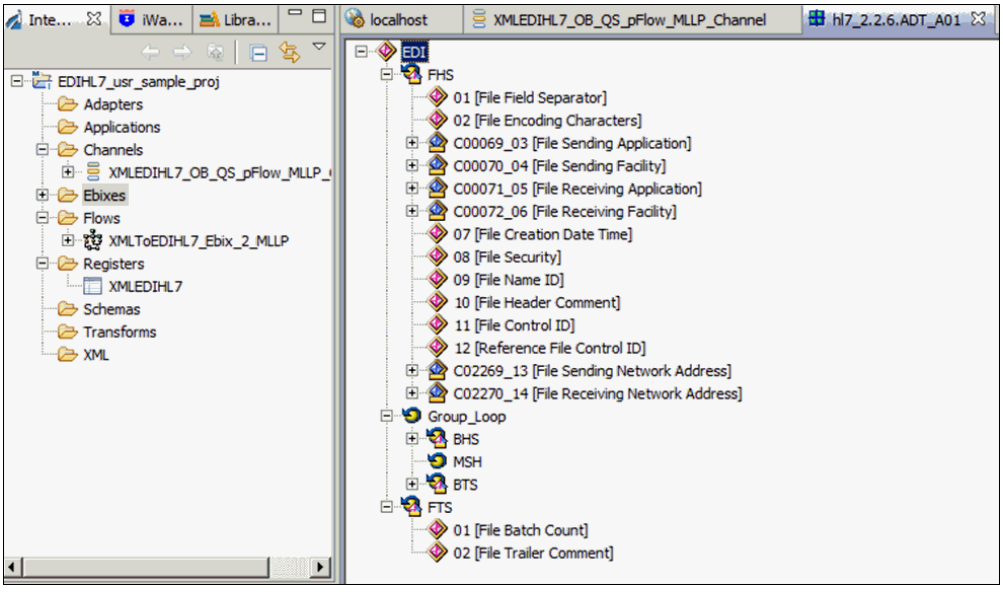


5. In the Import wizard, click *Next*.
6. Expand the hl7_2.6 ebix from the left panel and select the 2.6 folder.

7. Select the messages you wish from the right side panel and click *Finish*, as shown in the following image.

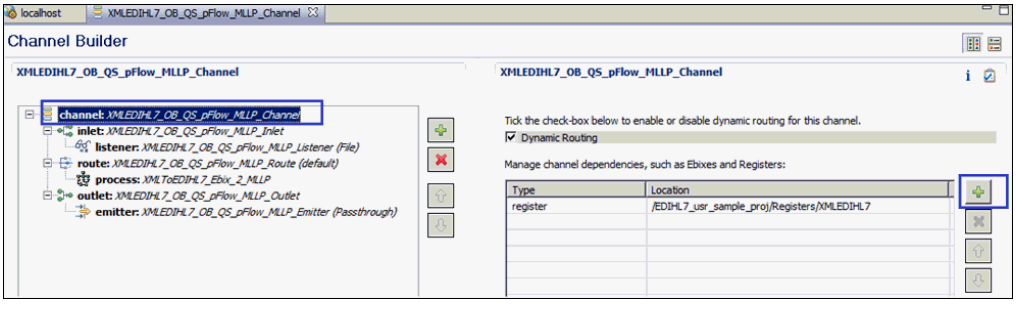


The iIT page should resemble the following image.

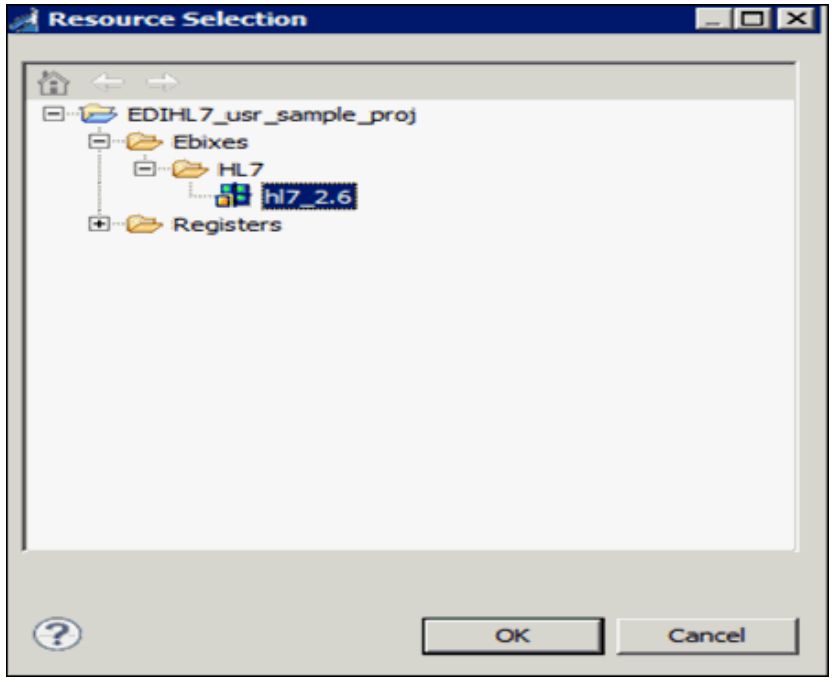


Procedure: How to Add an Ebix to an Outbound Channel as a Dependency

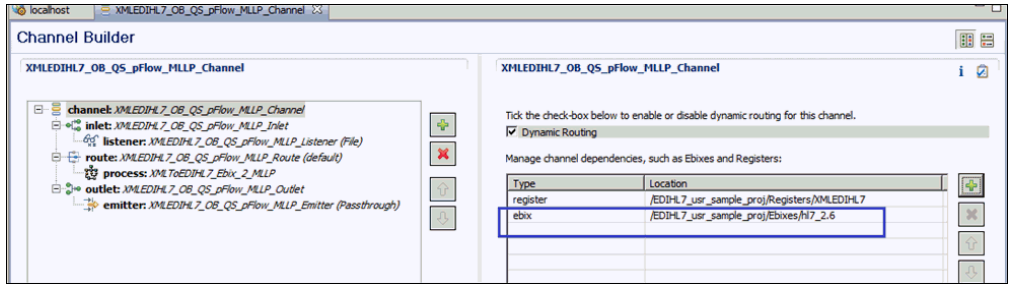
1. Click the *channel: XMLEDIHL7_OB_QS_pFlow_MLLP_Channel* node and then click the *Add dependency* icon on the right side panel, as shown in the following image.



2. Expand *EDIHL7_usr_sample_proj*, *Ebixes*, *HL7*, and then select ebix *hl7_2.6* and click *OK*, as shown in the following image.



3. Click the *Save* button to save your changes or press the keyboard shortcut (Ctrl+S) if you are using a Windows environment.

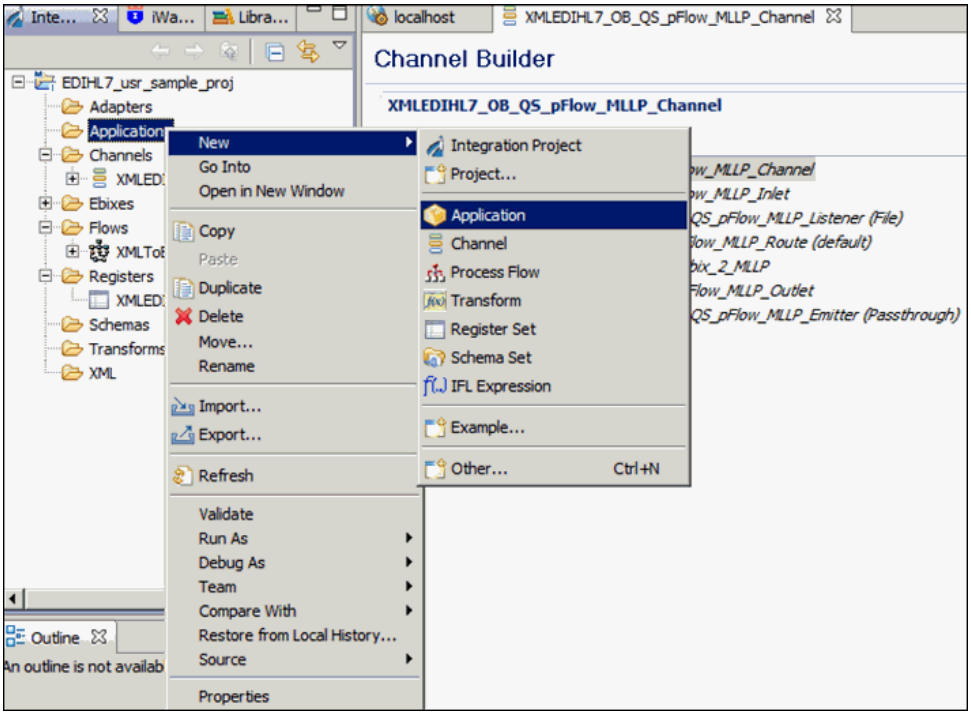


Configuring an iWay Integration Application for Outbound Processing

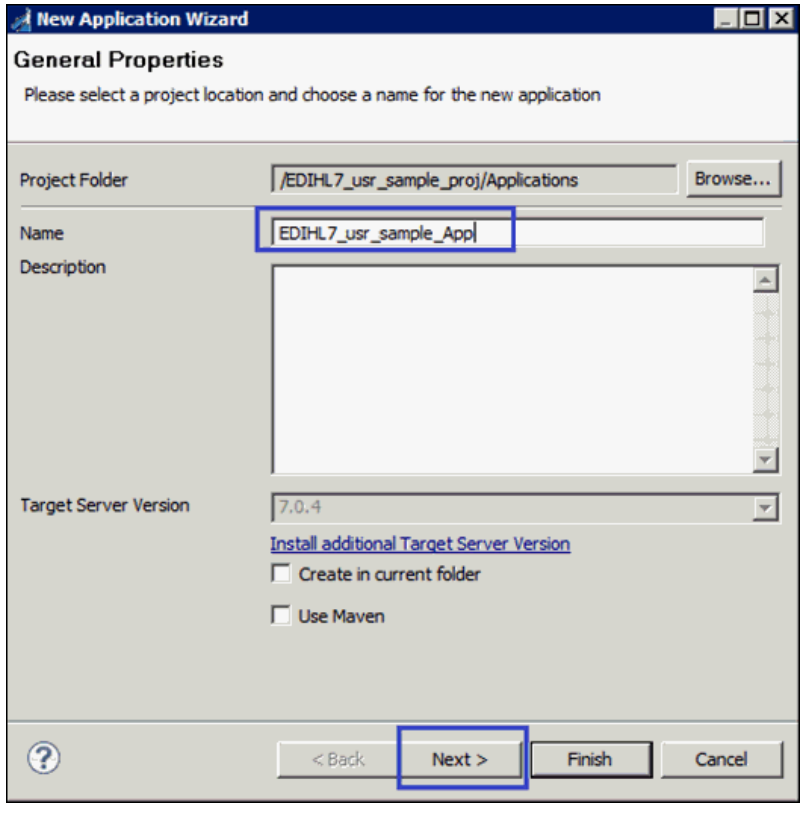
This section describes how to configure an iWay Integration Application (iIA) for outbound processing using iWay Integration Tools (iIT).

Procedure: How to Configure an iWay Integration Application for Outbound Processing

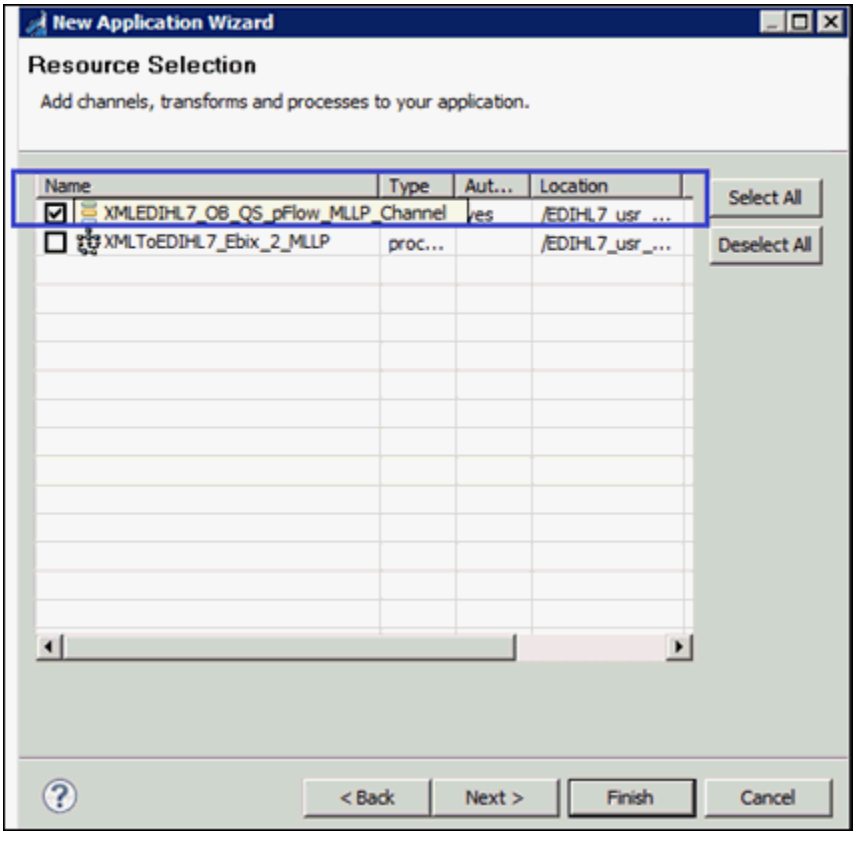
1. Right-click the Applications folder under the EDIHL7_usr_sample_proj integration project, select *New*, and then click *Application* from the context menu, as shown in the following image.



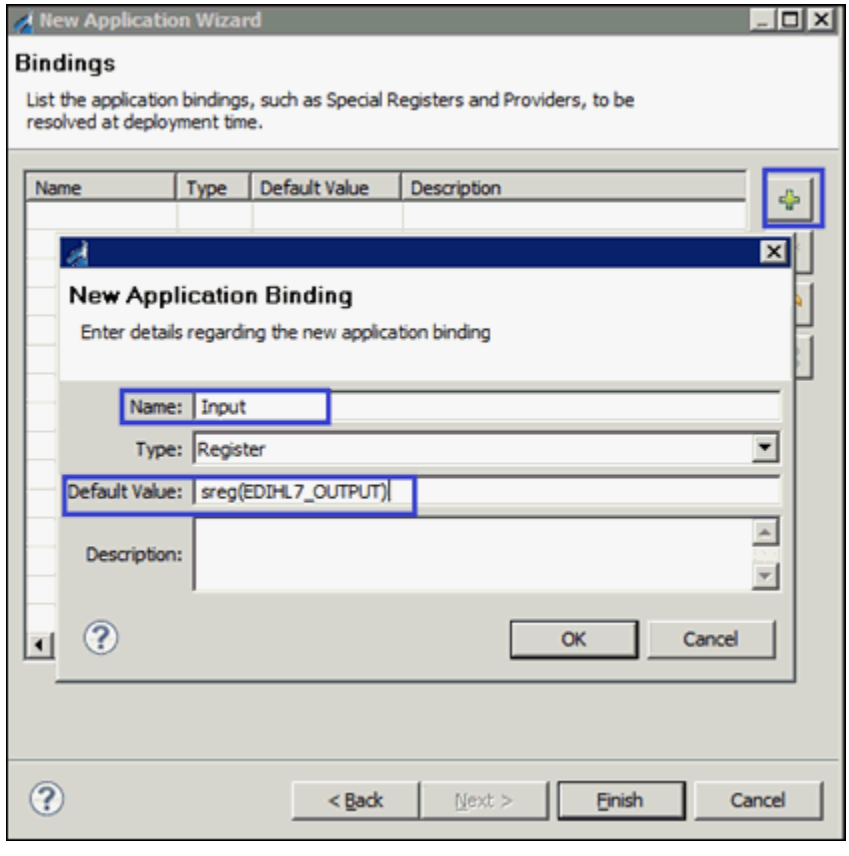
2. Enter a value in Name field (for example, EDIHL7_usr_sample_App), and click *Next*, as shown in the following image.



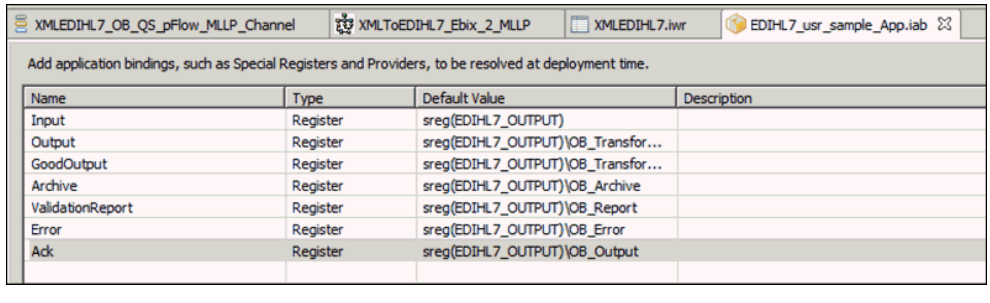
3. Select the *XMLEDIHL7_OB_QS_pFlow_MLLP_Channel* check box from Resource Selection pane and keep clicking *Next* until you get to the Bindings pane, as shown in the following image.



4. In the Bindings pane, add Bindings for Registers you have added in the XMLEDIHL7 Register Set to the iIA EDIHL7_usr_sample_App and click *Finish*, as shown in the following image.

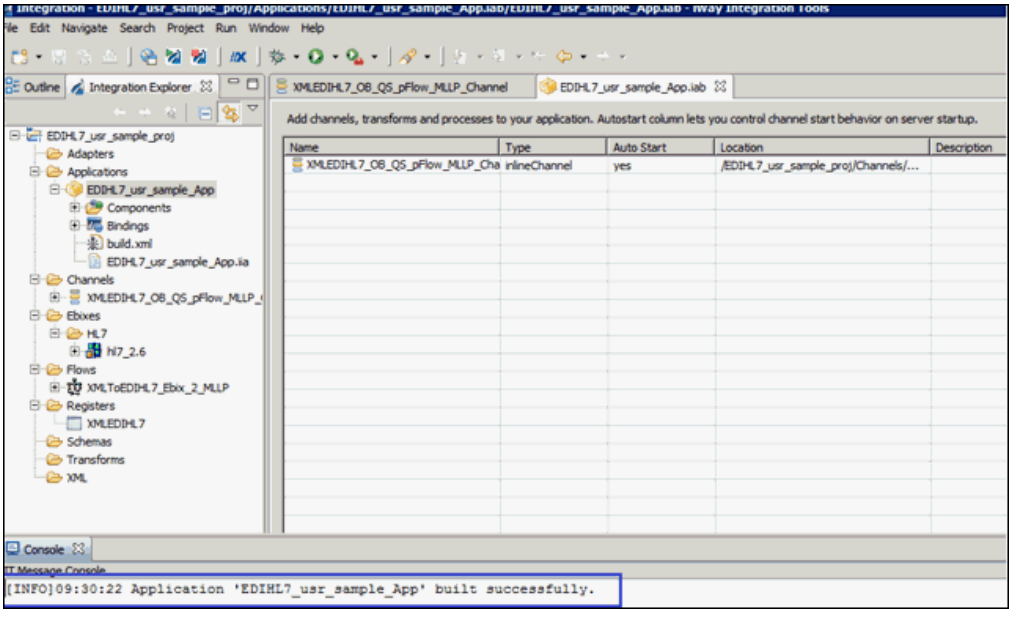


After adding all register bindings, your iIT screen should resemble the following image.



Name	Type	Default Value	Description
Input	Register	sreg(EDIHL7_OUTPUT)	
Output	Register	sreg(EDIHL7_OUTPUT)\OB_Transfor...	
GoodOutput	Register	sreg(EDIHL7_OUTPUT)\OB_Transfor...	
Archive	Register	sreg(EDIHL7_OUTPUT)\OB_Archive	
ValidationReport	Register	sreg(EDIHL7_OUTPUT)\OB_Report	
Error	Register	sreg(EDIHL7_OUTPUT)\OB_Error	
Ack	Register	sreg(EDIHL7_OUTPUT)\OB_Output	

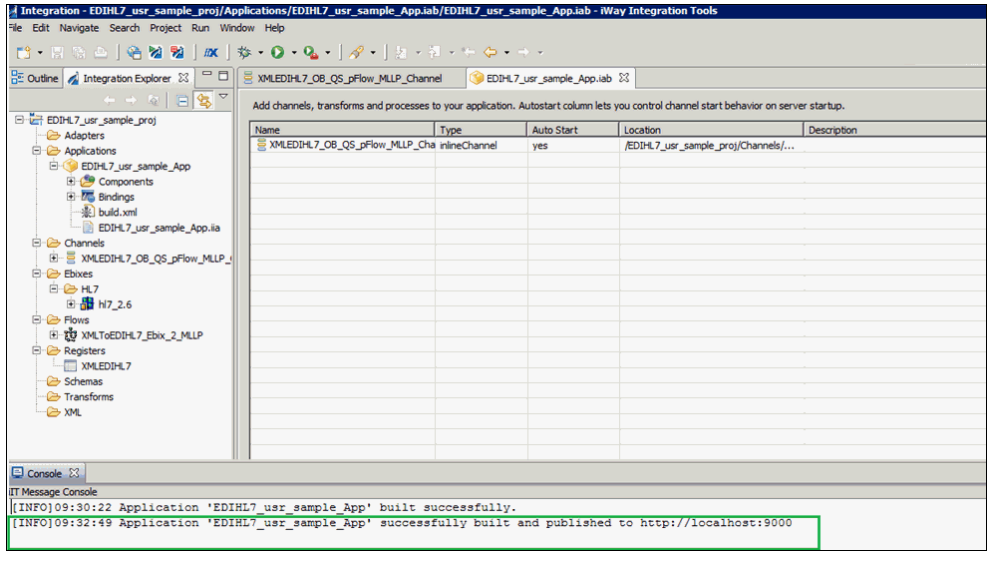
5. To build the application, right-click *EDIHL7_usr_sample_App*, select *Integration Tools*, and then click *Build* from the context menu, as shown in the following image.



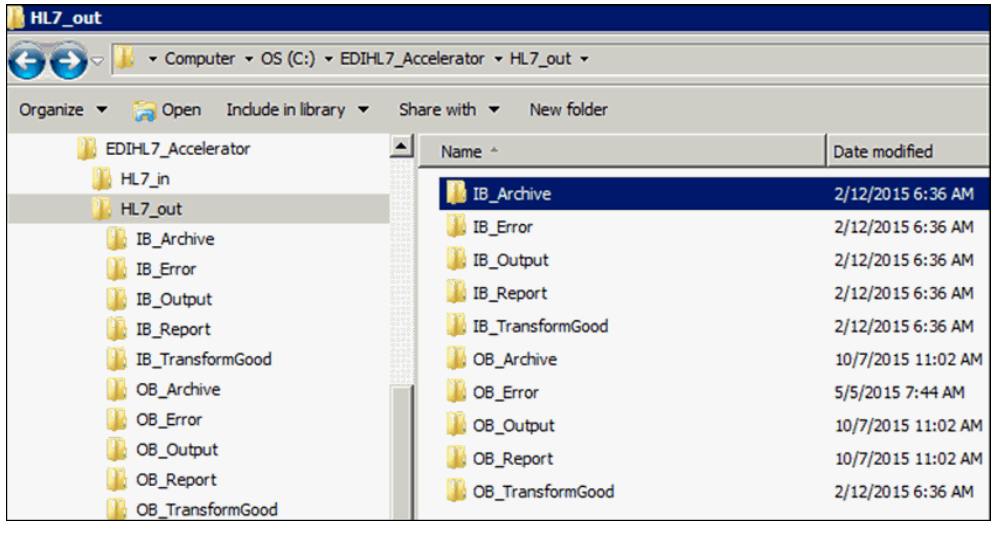
6. To publish the iIA, right-click *EDIHL7_usr_sample_App*, select *Integration Tools*, and then click *Publish to* from the context menu.

7. Provide the iSM server details in the Server URL text box and the other credentials, then click *Finish*.

The iIT page should resemble the following image.



8. Deploy the iIA application on the iSM registry.
9. Have the following folder structure created before starting your application in the iSM console, as shown in the following image.

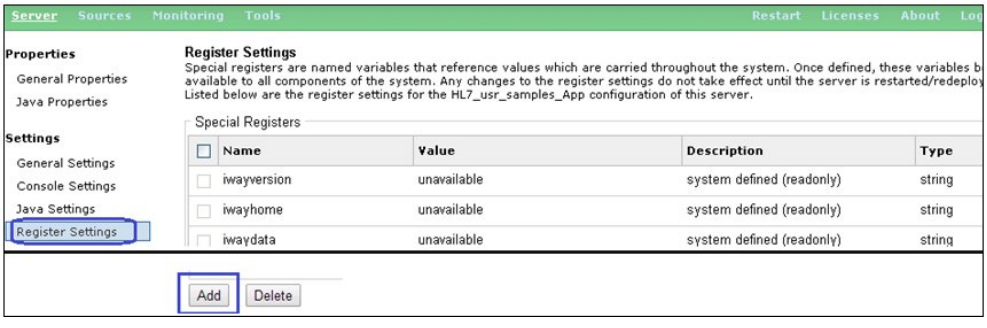


Setting Outbound HL7 System Registers Using MLLP

This section describes how to set system registers using the iWay Service Manager (iSM) Administration Console.

Procedure: How to Set System Registers

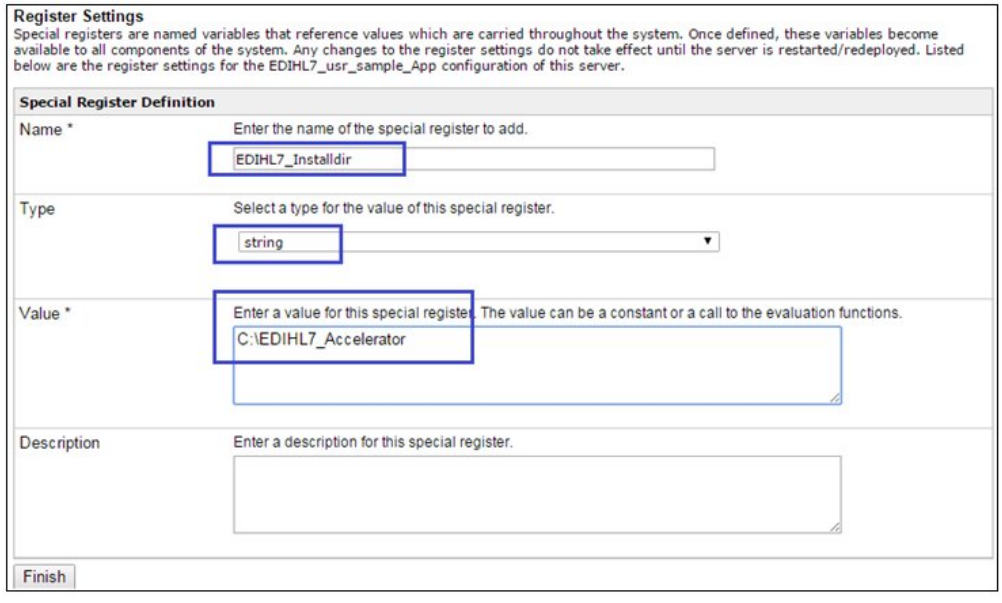
1. Open the iSM Administration Console and select *EDIHL7_usr_sample_App_MLLP_OB [down]* from the Management drop-down list.
2. From the console bar, click *Server*, *Register Settings*, and then click *Add*, as shown in the following image.



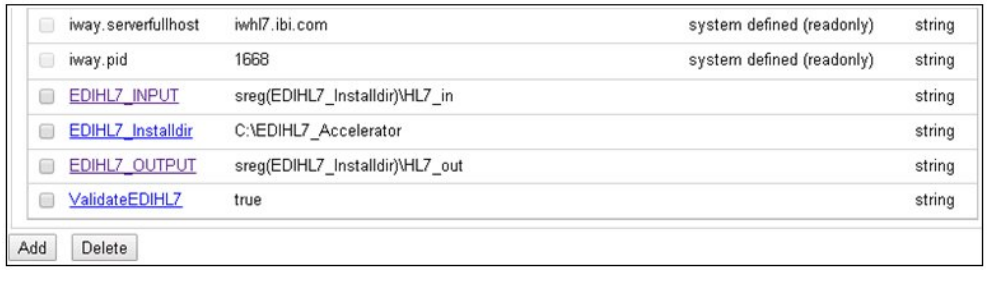
3. For the deployed EDIHL7_usr_sample_App_MLLP_OB application, define the system registers as listed in the following table:

Register Name	Value
EDIHL7_INPUT	`sreg(EDIHL7_Installdir)\HL7_in`
EDIHL7_installdir	`C:\EDIHL7_Accelerator`
EDIHL7_OUTPUT	`sreg(EDIHL7_Installdir)\HL7_out`

Register Name	Value
ValidateEDIHL7	`true`



After adding the registers, your Register Settings page in the iSM Administration Console should resemble the following image.



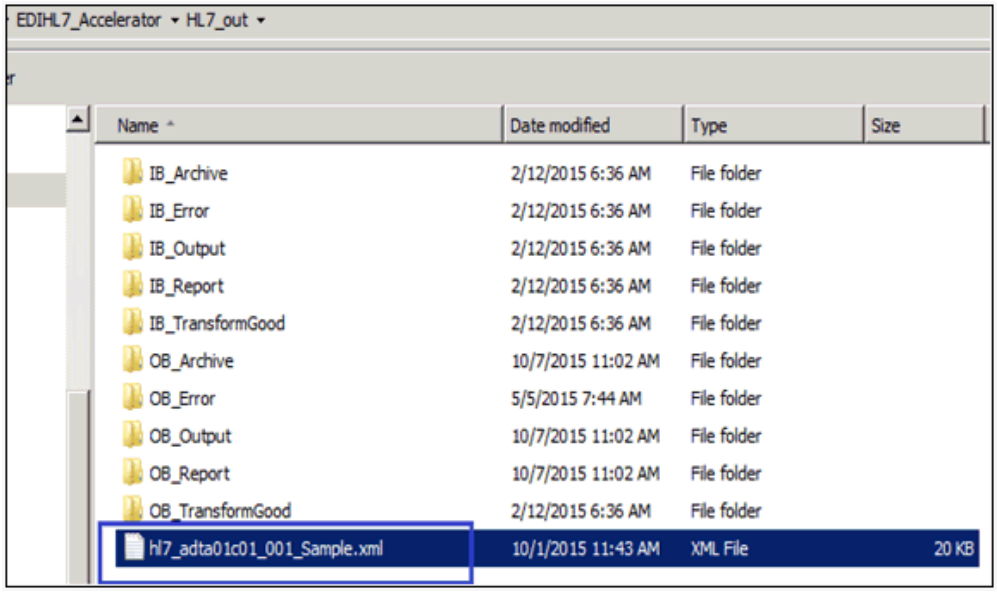
4. Start the deployed application in the iSM Administration Console and ensure that the channel is up and running in the Monitoring section of the console.

Testing the Outbound Channel Application Using MLLP

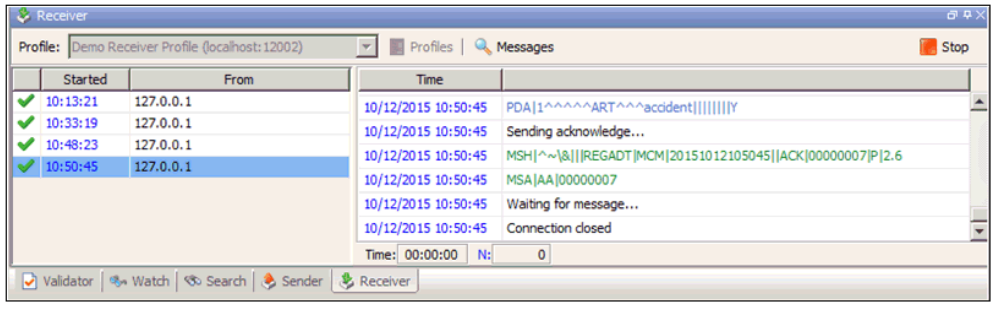
This section describes how to test the outbound channel application using the Minimal Lower Layer Protocol (MLLP).

Procedure: How to Test the Outbound Channel Application Using MLLP

1. Copy and then paste your input XML file to the output directory that you have configured (for example, EDIHL7_Accelerator\HL7_out), as shown in the following image.



2. Look for the acknowledgement message in the EDIHL7_Accelerator\HL7_out\OB_Output directory.



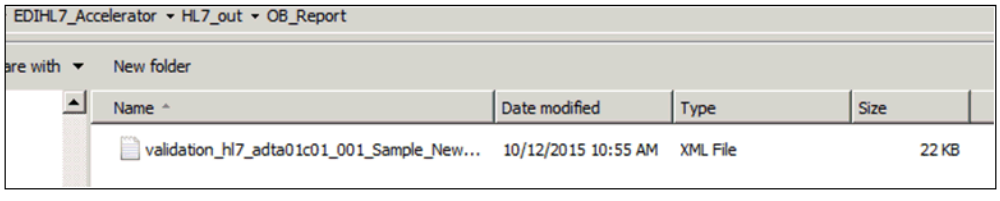
The _hl7ack() iFL function is used in the outbound MLLP process flow. This function parses a HL7 acknowledgement message to return a specific element value. It uses the following format:

```
_hl7ack(query[,defaultValue])
```

For more information on using this function, see *_hl7ack(): Parse HL7 Acknowledgement Message* on page 205.

The _hl7ack() iFL function will return "AA", "AE", "AR", "CA", "CE", "CR" acknowledgement codes. Based on this code the acknowledgement will be stored in different locations.

3. Look for the validation report to be received in the output directory that you have configured (for example, EDIHL7_Accelerator\HL7_out\OB_report), as shown in the following image.



Validation reports contain valid (good) or invalid (bad) HL7-formatted data, copies of the input files, and any error messages that may have occurred during the transformation. Typically the name of the validation report can inform you if the transformation has passed or failed. You can also configure the outbound processing to write the data to two different file locations, one for valid date and the other for invalid data.

4. If the input data contained any errors, you can review this error data in the output directory that you have configured for error handling (for example, EDIHL7_Accelerator\HL7_out \OB_Error).
5. After outbound processing has completed, a copy of the input data that was used during the transformation is stored under the archive directory that you specified (for example, EDIHL7_Accelerator\HL7_out\OB_Archive).

Appendix A

Supported HL7 Versions

This section summarizes the HL7 versions that are currently supported by the iWay Integration Solution for EDIHL7.

In this appendix:

Supported HL7 Versions and Messages

The following table lists the HL7 versions and messages that are supported by the iWay Integration Solution for EDIHL7.

HL7 Version	Supported Messages
2.3	ALL
2.3.1	ALL
2.4	ALL
2.5	ALL
2.5.1	ALL
2.6	ALL
2.7	ALL
2.7.1	ALL

Appendix B

Using HL7 Separators and Terminators

All HL7 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 (iSM) and iWay Integration Tools (iIT).

In this appendix:

HL7 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	@
09	TAB	1E	RS	5B	[

Hex	Char	Hex	Char	Hex	Char
0A	LF	1F	US	5C	\
0B	VT	21	!	5D	]
0C	FF	23	#	5E	^
0D	CR	24	$	5F	_
0E	SO	25	%	60	'
0F	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	-		

Appendix C

Using iWay Integration Tools to Configure an Ebix for HL7

This section describes how to use iWay Integration Tools (iIT) to configure an Ebix for HL7.

In this appendix:

Using iIT to Configure an Ebix File for HL7Overview

You can use iWay Integration Tools (iIT) to import, edit, export, and work with Ebixes for HL7. The topics in this appendix describe how to:

- Import an HL7 version 2.6 Ebix into iIT.
- Add a ZMS segment at the end of an ADT_A01 message to the HL7 version 2.6 Ebix.
- Export the edited Ebix to a physical location.

The edited Ebix can be returned and then tested with the appropriate HL7 version 2.6 ADT_A01 message.

Using iIT to Configure an Ebix File for HL7 Prerequisites

The following list shows the prerequisites for using iWay Integration Tools (iIT) to configure an Ebix for HL7:

- A working knowledge of iWay Service Manager (iSM), iWay Integration Tools (iIT), and HL7.
- iSM Version 8.0.
- iWay Integration Solution for EDIHL7 (Patch).
- iIT Version 8.0.

Loading an Ebix

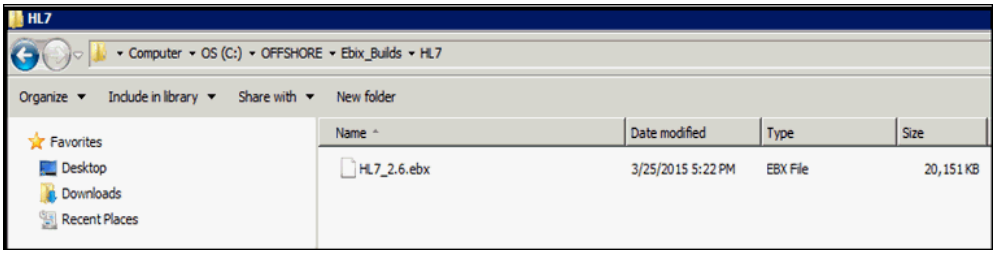
This section describes how to load an Ebix.

Procedure: How to Load an Ebix

To load or import an Ebix:

1. Download the *HL7_ebixes.zip* file from *http://techsupport.informationbuilders.com*.
2. Unzip the downloaded HL7_ebixes.zip and save *HL7_2.6.ebx* into any physical location on your local drive.

   For example, this Ebix contains ADT_A01 in it. Make sure all folders used for HL7_2.6.ebx do not have any blank spaces in the folder name.



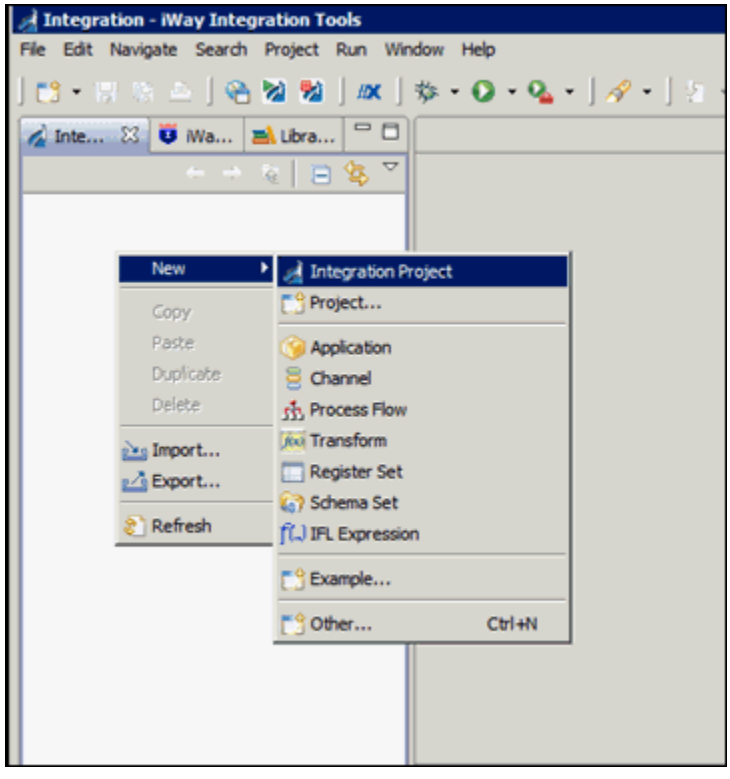
Working With iWay Integration Tools (iIT)

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

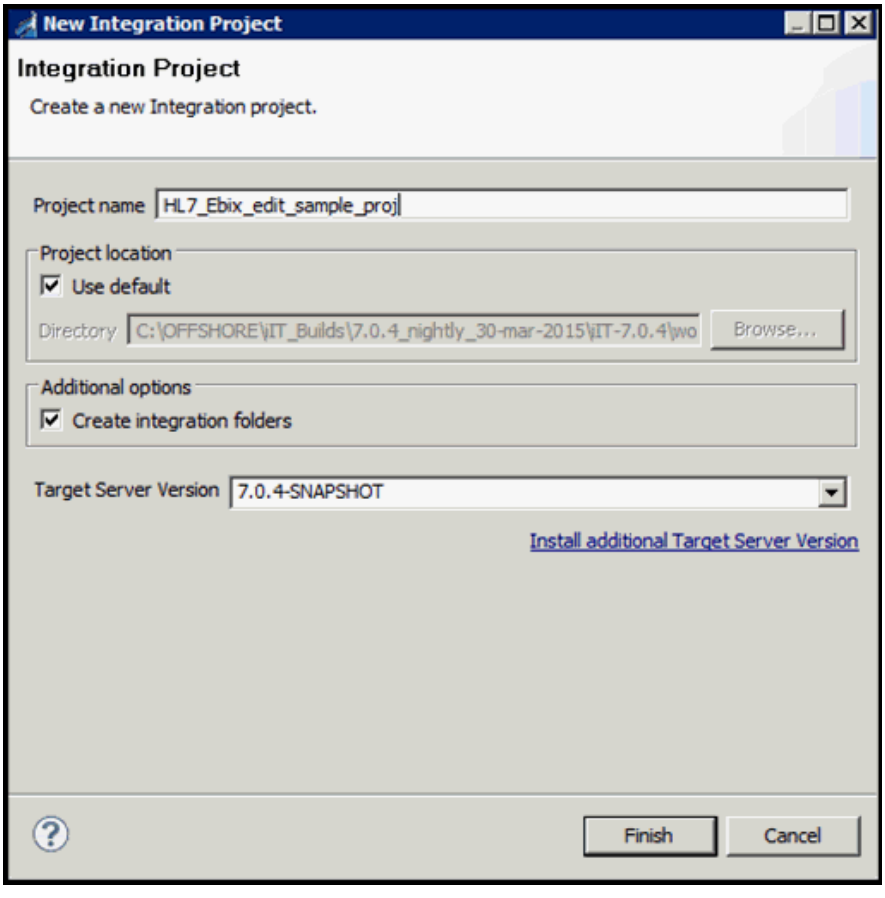
Procedure: How to Import an Ebix

1. Start iWay Integration Tools (iIT).

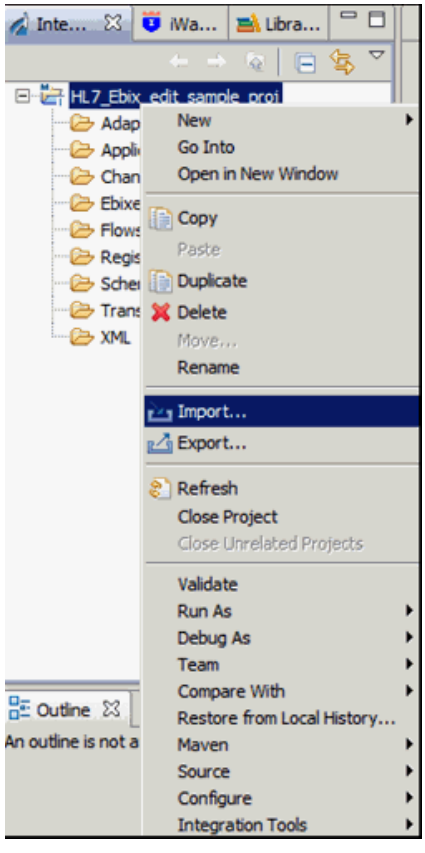
2. Right-click the Integration Explorer pane, click *New*, and select *Integration Project* from the context menu.



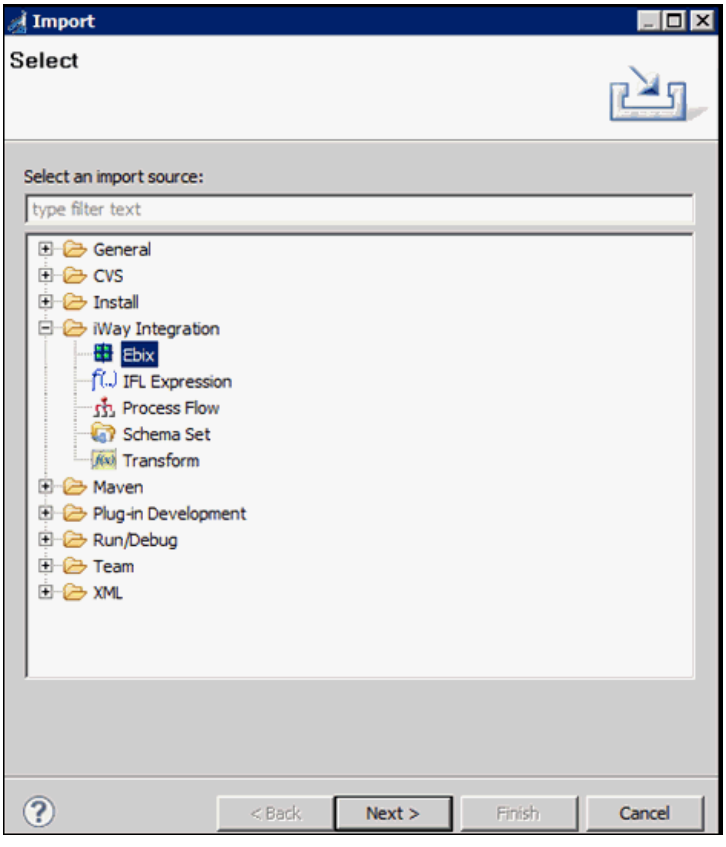
3. Enter a new Integration Project name, for example, *HL7_Ebix_edit_sample_proj*, in the Project name field, and click *Finish*.



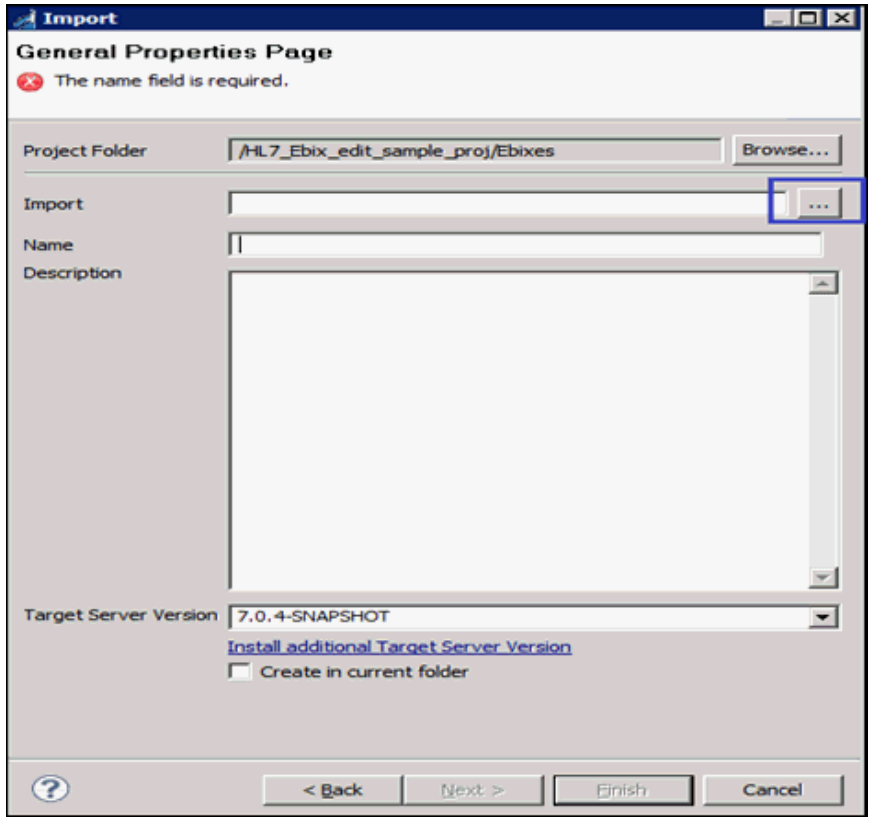
4. Right-click the Integration Explorer pane and select *Import* from the context menu, as shown in the following image.



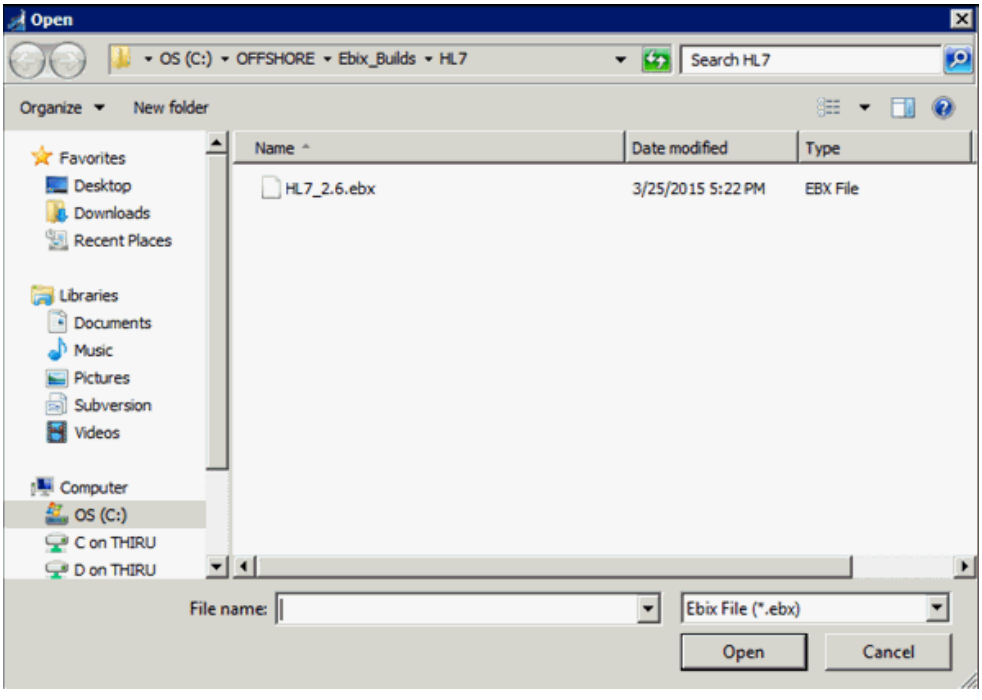
5. In the Import wizard, expand *iWay Integration*, select *Ebix*, and then click *Next*, as shown in the following image.



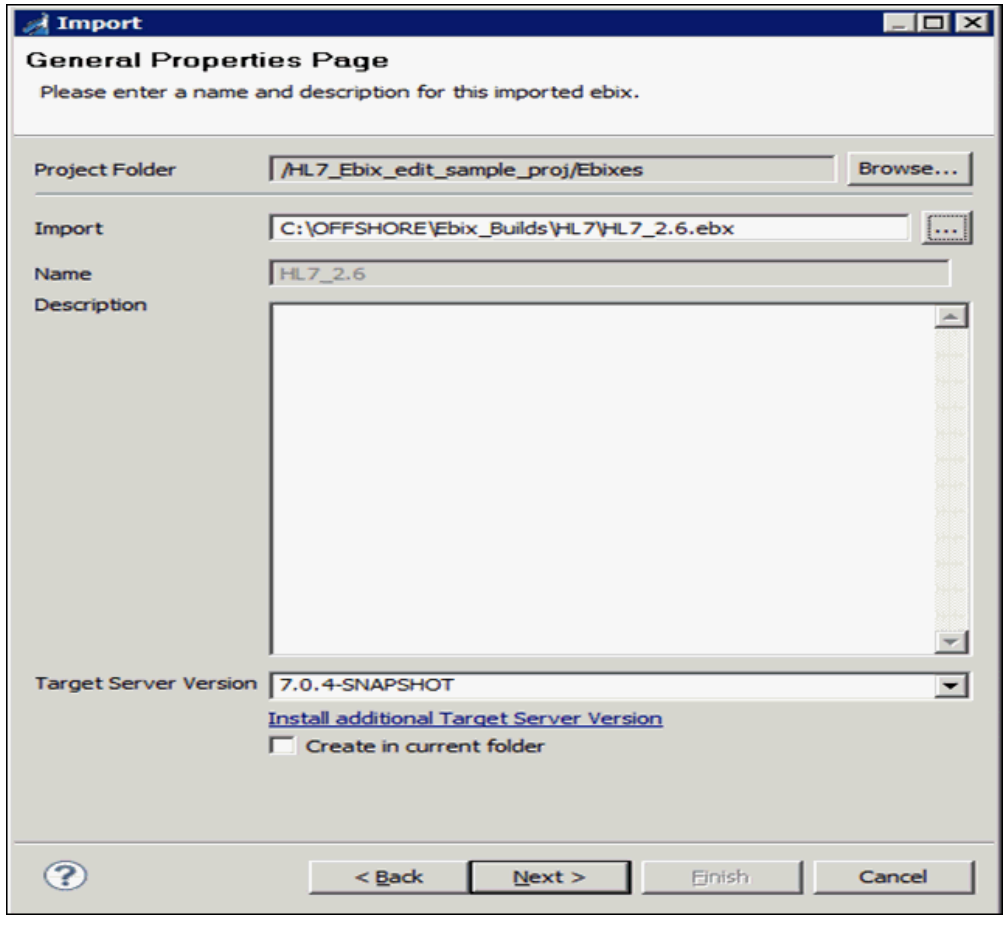
6. Click the *ellipsis button* (...) and select the downloaded *HIPAA_5010X299.ebx* file from the physical drive location.



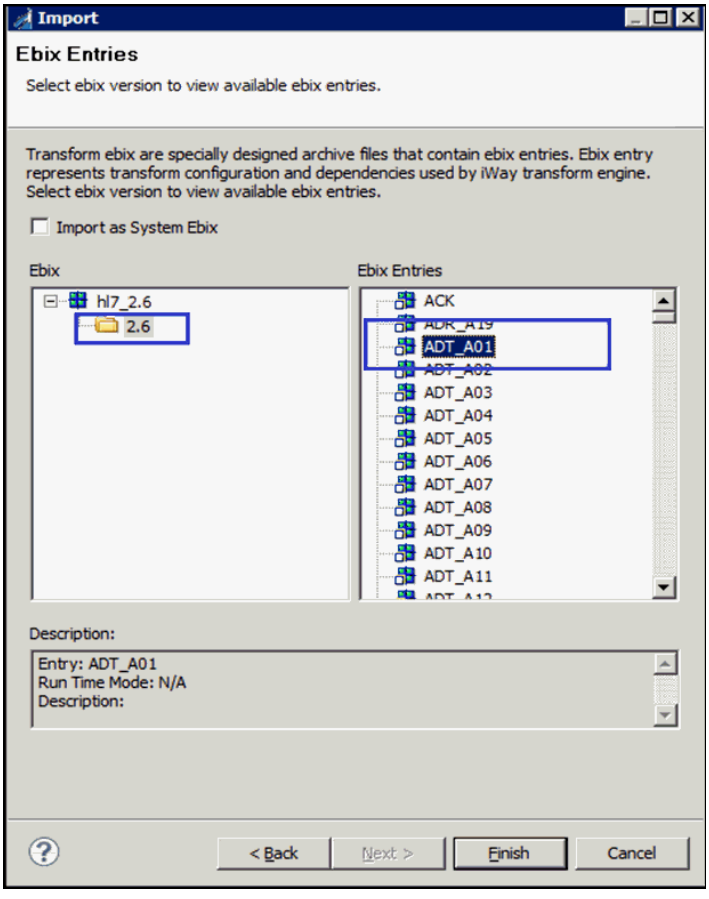
7. Click *Open*, as shown in the following image.



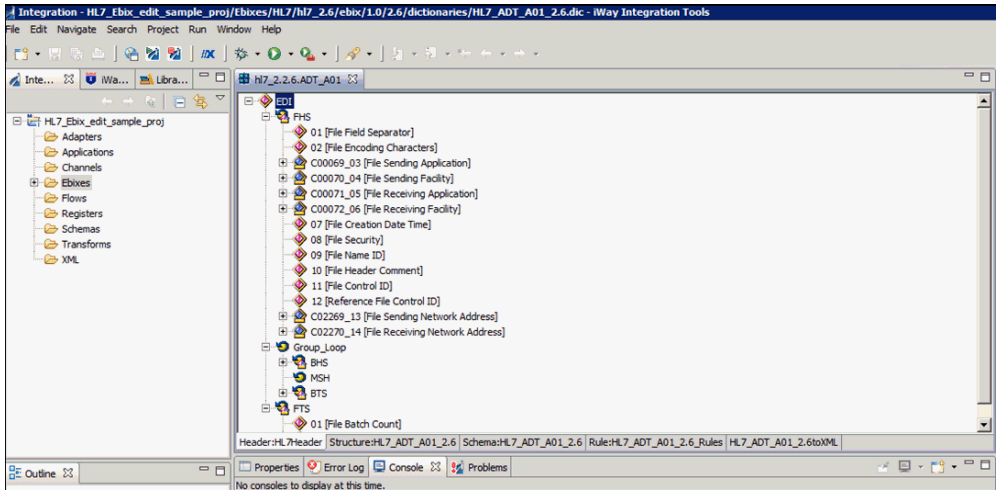
8. Click *Next*, as shown in the following image.



9. Expand the Ebix showed in the Ebix pane, select *ADT_A01* from the Ebix Entries pane, and then click *Finish*.



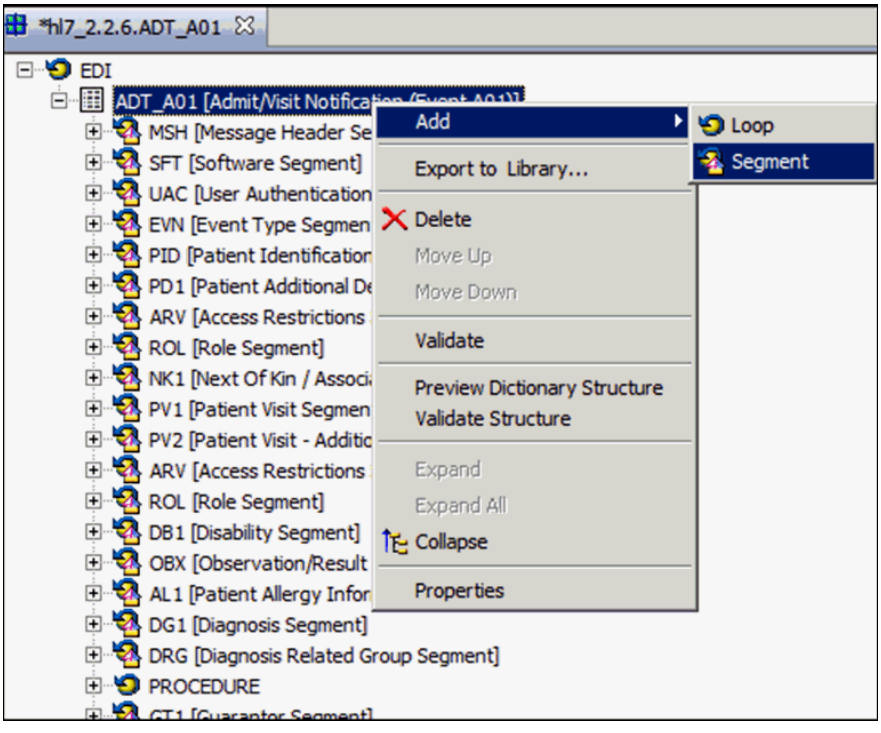
Your iWay Integration Tool interface should resemble the following image:



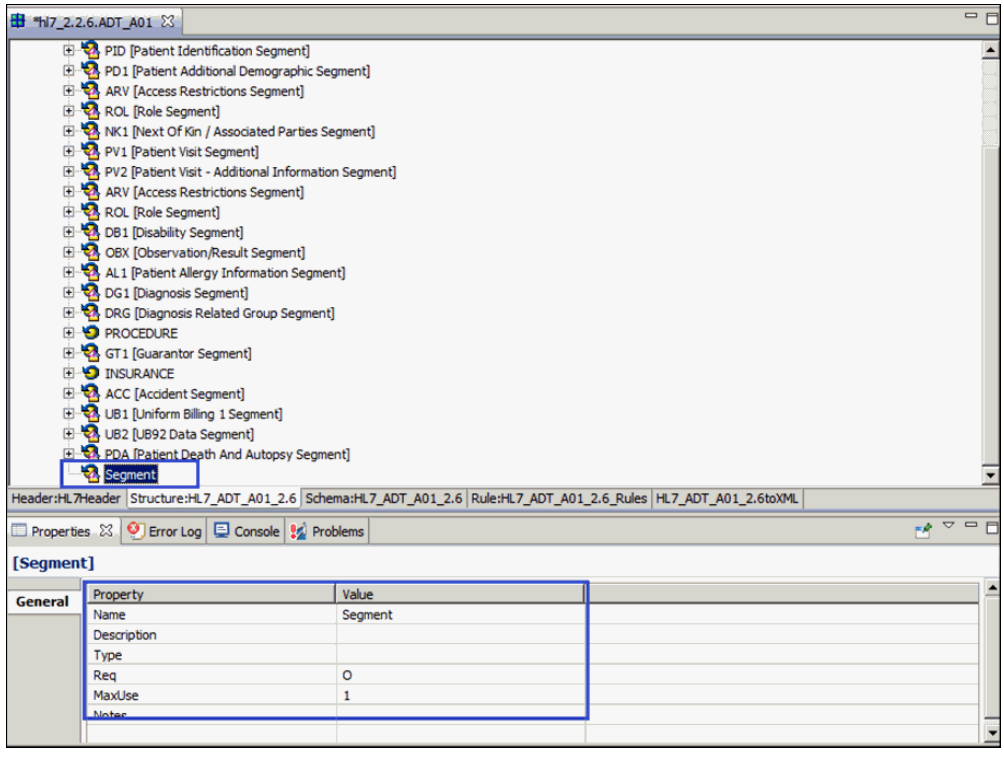
Procedure: How to Edit an Ebix

1. Click the *HL7_ADT_A01_2.6* tab to open the HL7 Ebix structure.

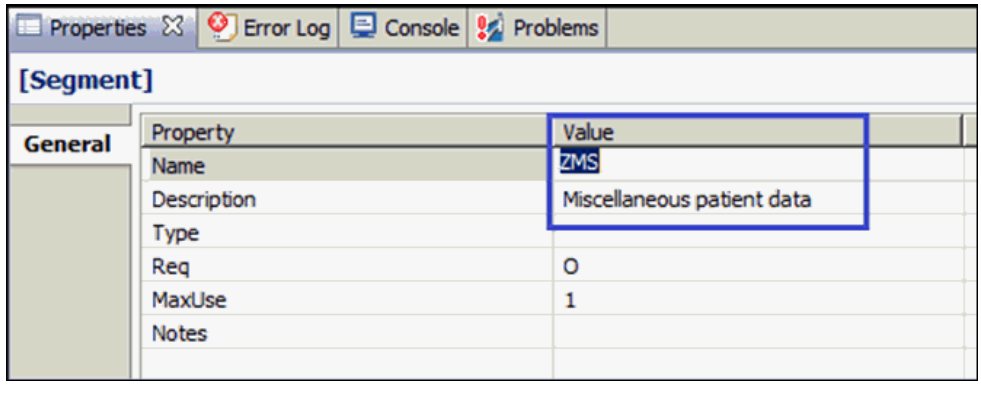
2. Right-click the *ADT_A01 [Admit/Visit Notification (Event A01)]* element, select *Add*, and then click *Segment*, as shown in the following image.



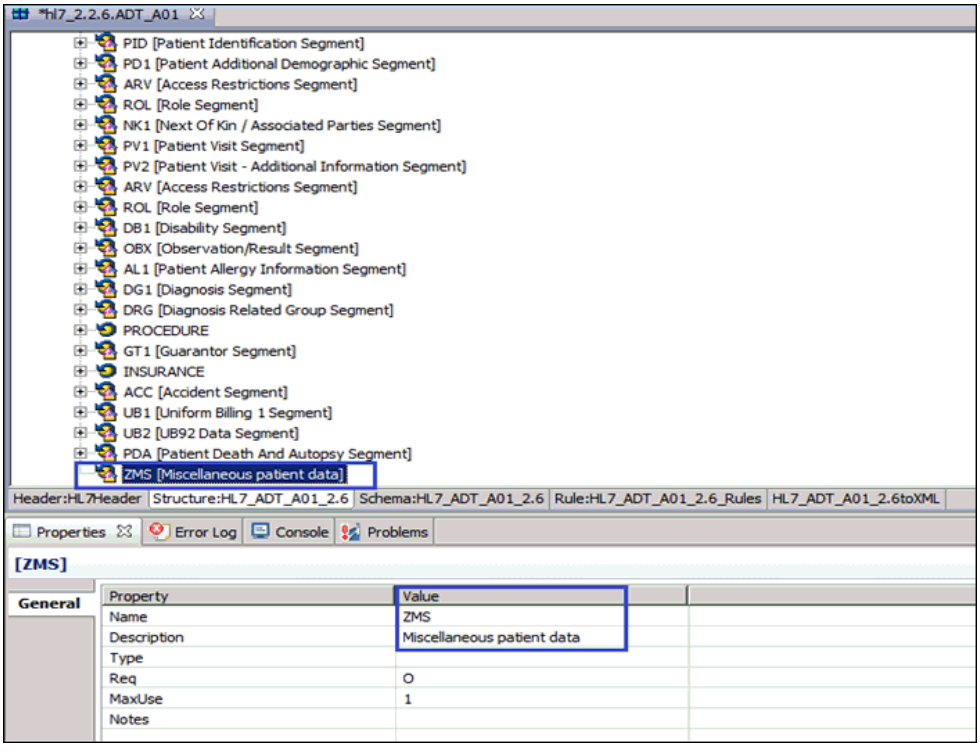
You screen should resemble the following image.



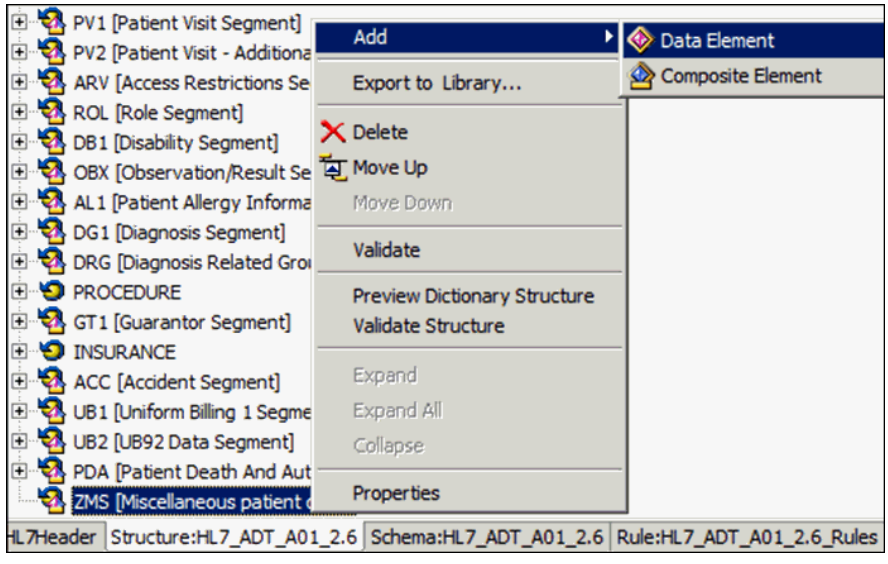
3. Provide valid segment name (for example, *ZMS*) and description (for example, *Miscellaneous patient data*) by double-clicking on the value field of its property, as shown in the following image.



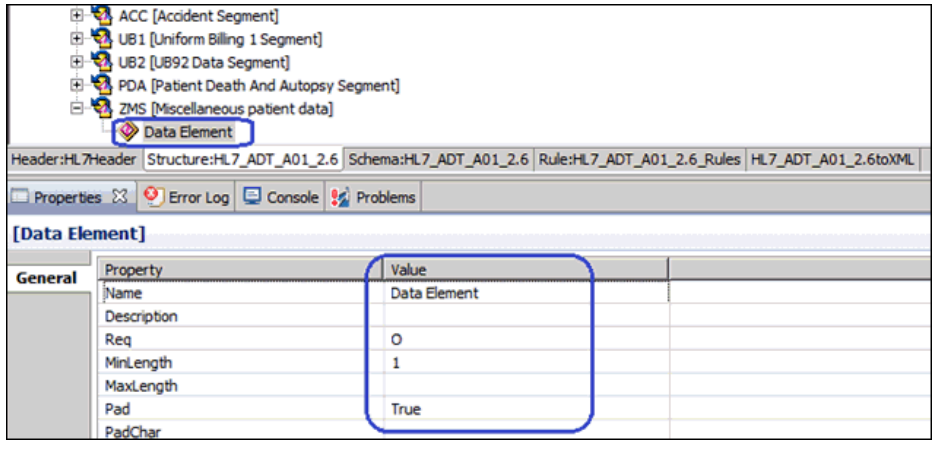
You screen should resemble the following image.



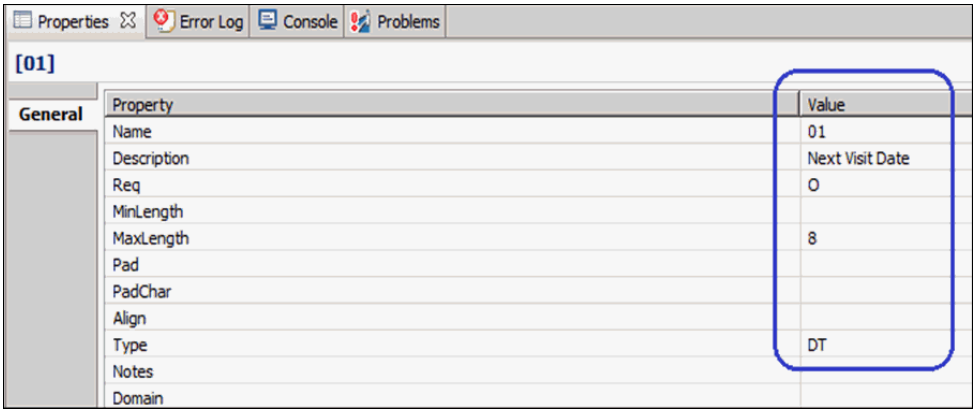
4. Right-click *ZMS [Miscellaneous patient data]* to add Data Elements in it, select *Add*, and then click *Data Element*, as shown in the following image.



5. Right-click the newly created Data Element and select *Properties*.



6. Change the Name field value to *01*, Description value to *Next Visit Date*, MaxLength value to *8*, and the Type value to *DT*, as shown in the following image.



Note: Use Backspace to delete existing values in the value field.

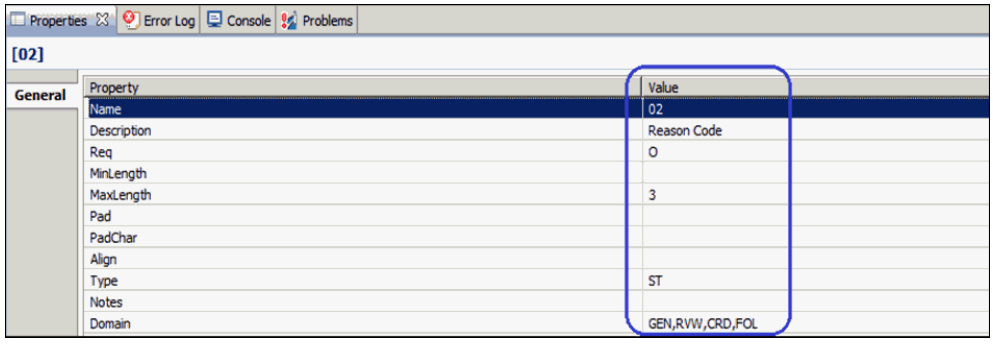
7. Repeat step 4 to create the following two data elements:

   - Reason Code
   - Number Of Days Left

8. Input the following values into their respective parameters in the second data element (Reason Code) under the ZMS segment.

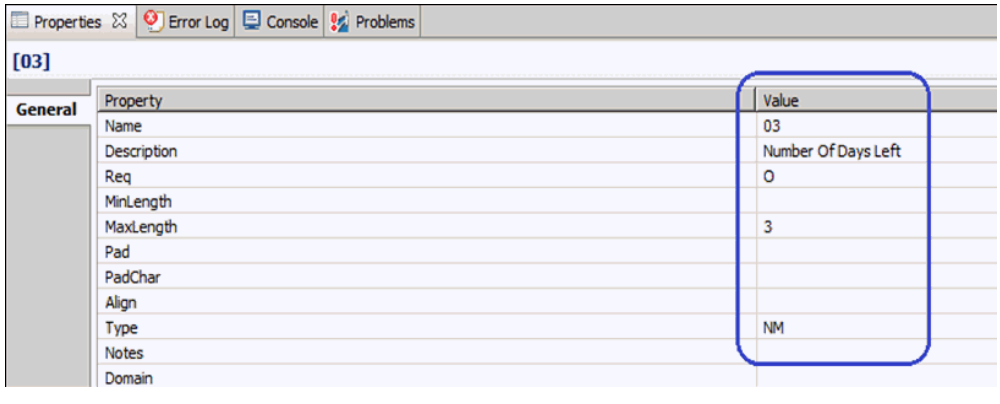
Parameter	Value
Name	02
Description	Reason Code
Req	O
MaxLength	3
Type	ST

Parameter	Value
Domain	GEN, RVW, CRD, FOL

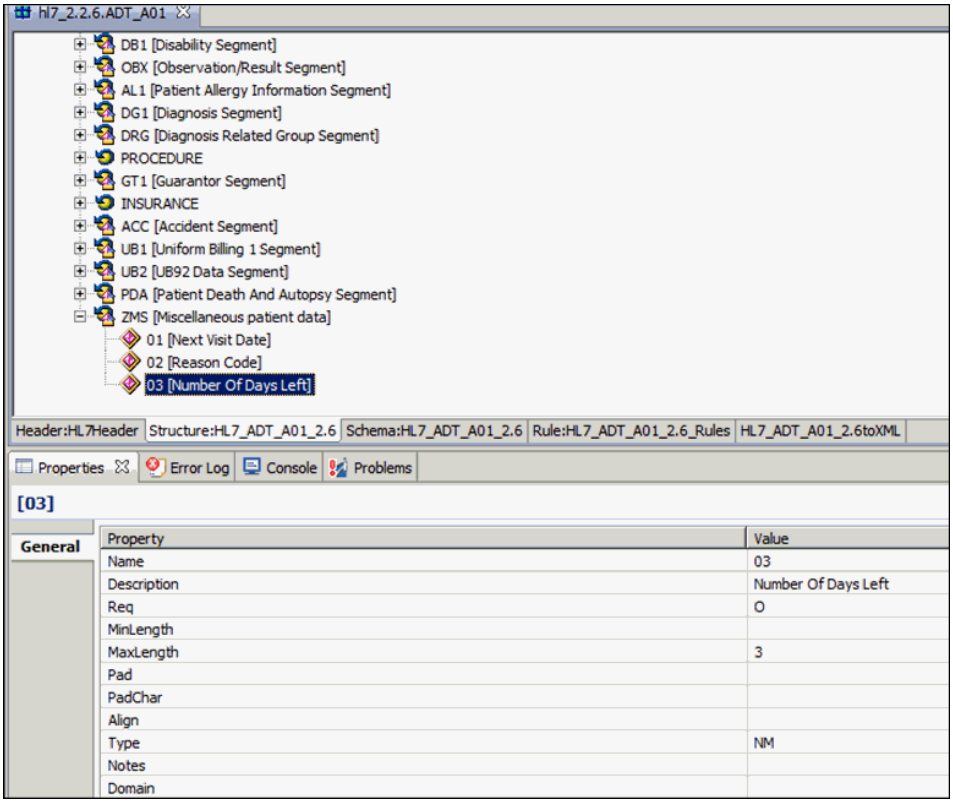


9. Input the following values into their respective parameters in the third data element (Number Of Days Left) under the ZMS segment.

Parameter	Value
Name	03
Description	Number Of Days Left
Req	O
MaxLength	3
Type	NM



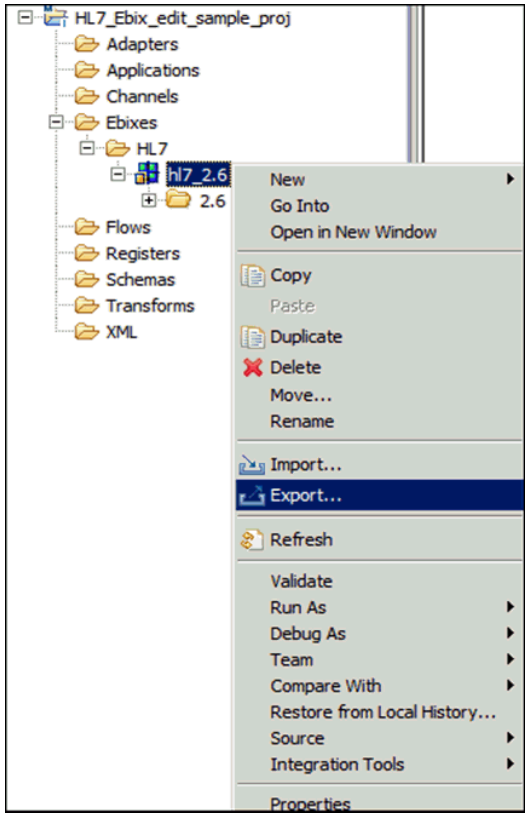
Your iWay Integration Tool screen should resemble the following image.



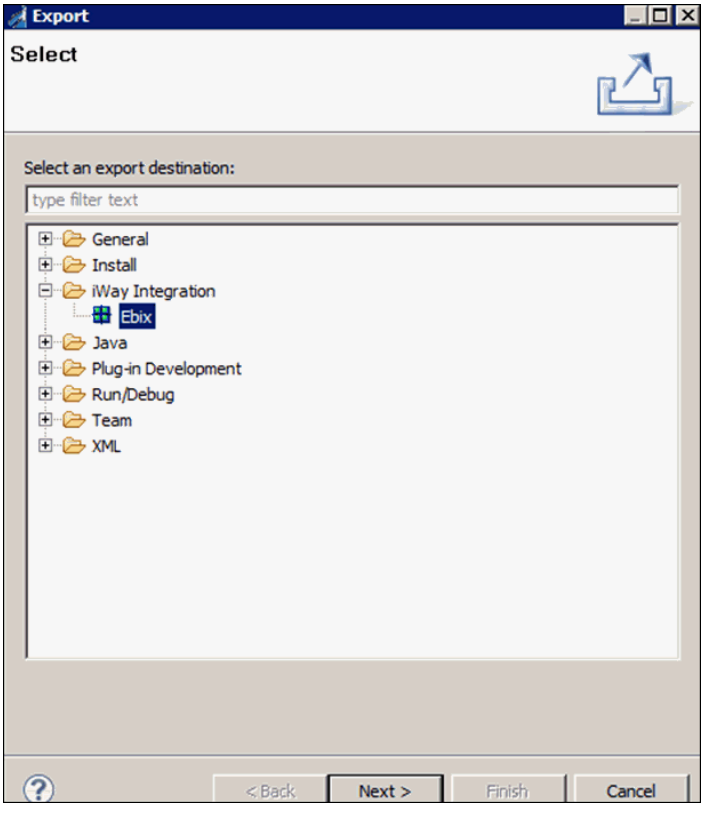
Procedure: How to Export an Ebix

To export an Ebix:

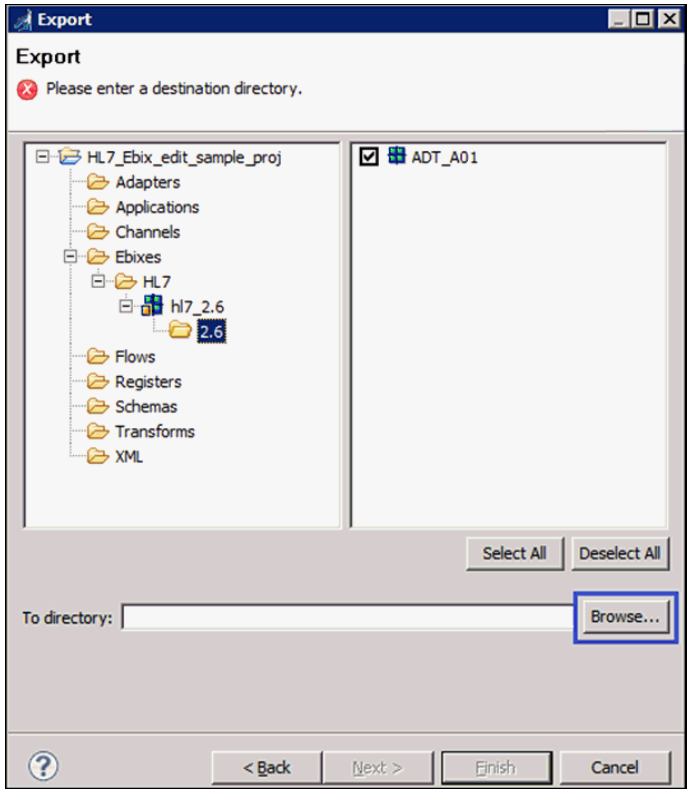
1. Right-click the *hl7_2.6* Ebix from the Integration Explorer window and select the *Export* option from the context menu, as shown in the following image.



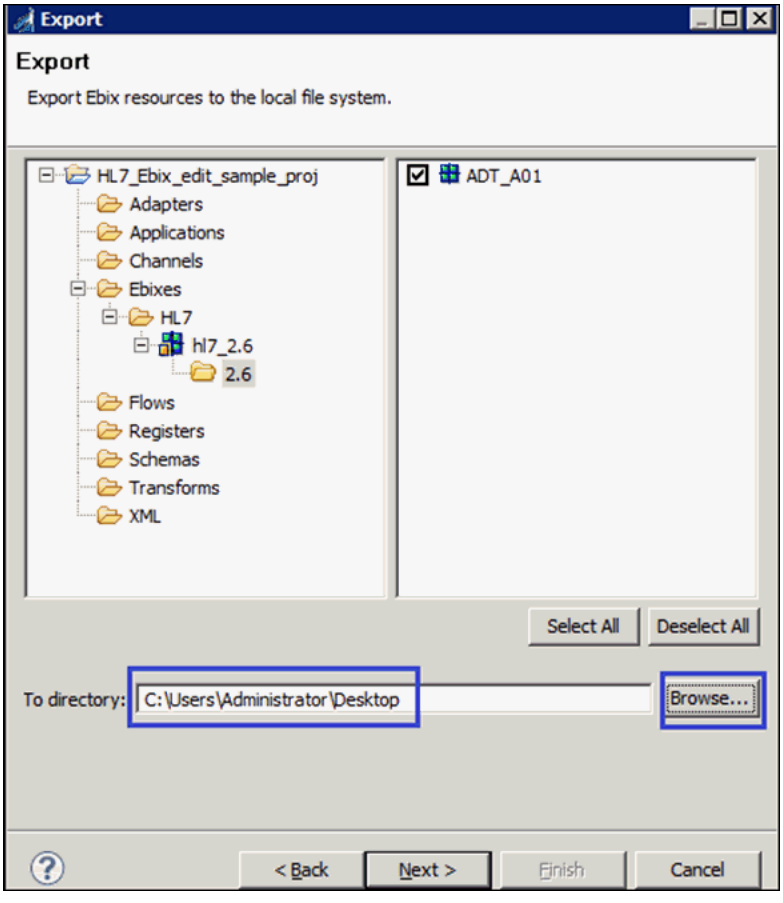
2. Expand the *iWay Integration* folder, select *Ebix*, and then click *Next*, as shown in the following image.



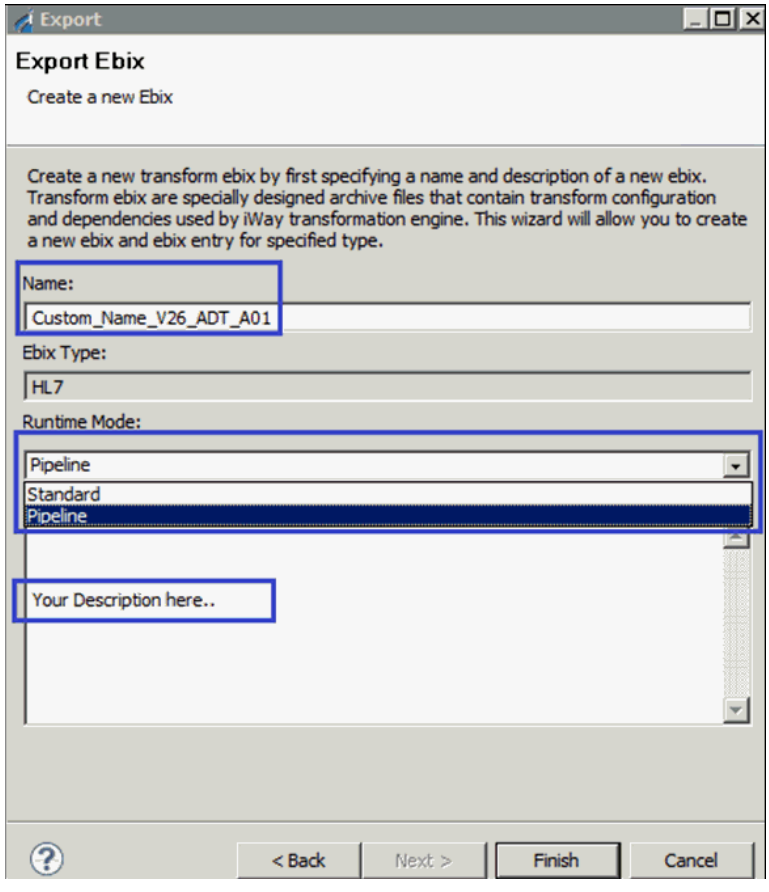
3. Expand the iIT project (for example, *HL7_Ebix_edit_sample_proj*) from the Export wizard and expand *Ebixes*, *HL7*, *HL7_2.6*, select the *2.6* folder from the left pane of Export wizard, and then select the *ADT_A01* check box on the right pane, as shown in the following image.



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 available in the specified location.

Feedback

Customer success is our top priority. Connect with us today!

Information Builders Technical Content Management team is comprised of many talented individuals who work together to design and deliver quality technical documentation products. Your feedback supports our ongoing efforts!

You can also preview new innovations to get an early look at new content products and services. Your participation helps us create great experiences for every customer.

To send us feedback or make a connection, contact Sarah Buccellato, Technical Editor, Technical Content Management at *Sarah_Buccellato@ibi.com.*

To request permission to repurpose copyrighted material, please contact Frances Gambino, Vice President, Technical Content Management at *Frances_Gambino@ibi.com.*

Information Builders

iWay

iWay Integration Solution for EDIHL7 User's Guide

Version 7.0.x and Higher

DN3502093.0418

Information Builders, Inc.
Two Penn Plaza
New York, NY 10121-2898