SeeBeyond ICAN Suite

TCP/IP HL7 eWay Intelligent Adapter User's Guide

Release 5.0

Version 20050505173710.

Contents

Chapter 4

Chapter 6

Chapter 7

Chapter 1

Introducing the TCP/IP HL7 eWay

This document describes how to install, configure, and implement the TCP/IP HL7 eWay Intelligent Adapter, in a typical ICAN environment.

This chapter provides a brief overview of operations and components, general features, and system requirements of the TCP/IP HL7 eWay Intelligent Adapter (also referred to as the TCP/IP HL7 through this document).

What's in This Chapter

1.1 About the TCP/IP HL7 eWay

The TCP/IP HL7 eWay Intelligent Adapter enables the eGate Integrator system to exchange data with an external TCP/IP application, using the HL7 data protocol. eGate with the TCP/IP HL7 eWay utilizes J2EE™ Architecture to provide:

- Macro functionality, providing ease of use and productivity.
- Prebuilt Standards-compliant Inbound and Outbound Template Collaborations that work "as is" or can be modified for your specific needs.
- Customization of functionality, by configuring the eWay properties and/or modifying the Collaborations.
- Journaling and error messaging to JMS queues and topics. This is in addition to eGate's standard alert and debug logging.
- Support for HL7 Standard versions 2.1, 2.2, 2.3, 2.3.1, 2.4, and 2.5.

Note: *Throughout this document the term "JMS queues" is used in the generic sense, actually meaning JMS queue or topic.*

1.1.1 TCP/IP HL7 Features

The TCP/IP HL7 eWay includes the following features:

- Bi-directional, including Client or Server mode in either direction (to or from eGate).
- Handles both HL7 HLLP and MLLP protocols and envelopes.
- Provides a wide variety of recourse action configurations.
- Non-blocking I/O.
- Recovery and retry logic.
- Debug levels.
- Error logging.
- Journaling of HL7 messages and associated ACKs
- HL7 Acknowledgement levels
- Fully supports of the HL7 sequence numbering protocol.
- Full support for HL7 ACK, NAK generation and validation.
- Supports Delayed ACK in both directions.

1.1.2 TCP/IP HL7 eWay Components

The TCP/IP HL7 eWay incorporates three components:

- The HL7 TCP/IP Resource Adapter (commonly referred to as the eWay OTD) that implements the lower layer HL7 protocol over TCP/IP.
- Default inbound and outbound Collaborations that implement the HL7 messaging protocol, sequence numbering and recourse actions.
- Generic HL7 OTDs that provide the structures necessary to parse and create the data messages and ACKs used by the protocol.

The TCP/IP HL7 Object Type Definition (OTD) enables the creation of HL7 interfaces capable of running over TCP/IP, and also utilizes the common eWay services available in eGate. The TCP/IP HL7 eWay works hand in hand with the SeeBeyond HL7 OTD Libraries, versions 2.1 through 2.5.

The TCP/IP HL7 eWay properties allow the user to easily configure the operation of the TCP/IP HL7 eWay. These properties are adopted into the OTD's functions (see **Chapter 4** for more information on the TCP/IP HL7 eWay properties).

The OTD handles all of the lower-layer protocol. The OTD's behavior is customized using the eWay configuration properties. These eWay properties are used by the resource adapter, but are also accessed and used by the prebuilt Collaborations.

1.2 About This Document

This section provides a brief outline of this user's guide.

1.2.1. Organization of Information

The TCP/IP HL7 eWay Intelligent Adapter User's Guide includes the following chapters:

- **Chapter 1 "Introducing the TCP/IP HL7 eWay"** provides a brief description of the TCP/IP HL7 eWay Intelligent Adapter and an overview of the TCP/IP HL7 eWay User's Guide.
- **Chapter 2 "Installing the TCP/IP HL7 eWay"** provides the supported operating systems and system requirements for the TCP/IP HL7 eWay. It also includes directions for installing the Sun AppServer eWay and accessing the accompanying documentation and sample projects.
- **Chapter 3 "TCP/IP HL7 eWay Overview"** explains the basic functionality and operations of TCP/IP HL7 and the TCP/IP HL7 eWay.
- **Chapter 4 "Configuring TCP/IP HL7 eWay Properties"** describes how to configure the CICS eWay to run in your environment and provides a description of the inbound and outbound eWay properties.
- **Chapter 5 "Working with TCP/IP HL7 Collaborations"** provides an overview and description of the structure and functionality of the Inbound and Outbound Collaborations provided as part of the TCP/IP HL7 eWay.
- **Chapter 6 "Working With HL7 OTDs"** describes how to view, test, and modify OTDs using the OTD Editor.
- **Chapter 7 "Working with the TCP/IP HL7 eWay Projects"** describes how to use the TCP/IP HL7 eWay template projects that are installed automatically as part of the TCP/IP HL7 eWay.
- **Chapter 8 "Methods for the TCP/IP HL7 eWay"** describes the TCP/IP HL7 eWay's Java classes and provides directions for accessing the TCP/IP HL7 eWay Javadoc.

1.2.2 Scope of the Document

This user's guide provides a description of the TCP/IP HL7 eWay Intelligent Adapter. It includes directions for installing the eWay, configuring the eWay properties, and implementing the eWay's sample projects. This document is also intended as a reference guide, listing available properties and functions.

1.2.3. Intended Audience

This guide is intended for experienced computer users who have the responsibility of helping to set up and maintain a fully functioning ICAN Suite system. This person must also understand any operating systems on which ICAN Suite will be installed

(Windows or UNIX), and must be thoroughly familiar with Windows-style GUI operations.

1.2.4 Document Conventions

The following conventions are observed throughout this document.

Table 1 Document Conventions

Text	Convention	Example
Names of buttons, files, icons, parameters, variables, methods, menus, and objects	**Bold** text	▪ Click **OK** to save and close. ▪ From the **File** menu, select **Exit**. ▪ Select the **logicalhost.exe** file. ▪ Enter the **timeout** value. ▪ Use the **getClassName()** method. ▪ Configure the **Inbound** File eWay.
Command line arguments, code samples	`Fixed` font. Variables are shown in ***`bold italic`***.	`bootstrap -p` ***`password`***
Hypertext links	**Blue** text	See **Document Conventions** on page 14
Hypertext links for Web addresses (URLs) or email addresses	**Blue underlined** text	**http://www.seebeyond.com** **docfeedback@seebeyond.com**

1.3 SeeBeyond Web Site

The SeeBeyond Web site is your best source for up-to-the-minute product news and technical support information. The site's URL is:

http://www.seebeyond.com

1.4 SeeBeyond Documentation Feedback

We appreciate your feedback. Please send any comments or suggestions regarding this document to:

docfeedback@seebeyond.com

Chapter 2

Installing the TCP/IP HL7 eWay

This Chapter describes how to install the TCP/IP HL7 eWay Intelligent Adapter, as well as the accompanying documentation and sample projects. It also includes theTCP/IP HL7 eWay's supported operating systems and system requirements.

What's in This Chapter

2.1 Supported Operating Systems

The TCP/IP HL7 eWay is available for the following operating systems:

- Windows Server 2000, Windows XP, and Windows Server 2003
- Sun Solaris 8 and 9
- HP NonStop Server G06.22
- HP Tru64 5.1A
- HP-UX 11.0, 11i (PA-RISC), and 11i V2 (11.23)
- IBM AIX 5.1L and 5.2
- Red Hat Linux 8 (Intel Version)
- Red Hat Enterprise Linux AS 2.1

2.2 System Requirements

Although the TCP/IP HL7 eWay, the Repository, and Logical Hosts run on the platforms listed above, the Enterprise Designer requires the Windows operating system. Enterprise Manager can run on any platform that supports Internet Explorer 6.0.

To use the TCP/IP HL7 eWay, you need the following:

- eGate Integrator, version 5.0.3
- File eWay (to support sample projects)
- eGate logical host
- TCP/IP network connection
- To enable Web Services, you must install and configure the SeeBeyond ICAN Suite eInsight Business Process Manager.

Note: *Refer to the TCP/IP HL7 eWay Readme for the eWays most current requirements.*

Upload Requirements

After the eGate or eInsight core products are uploaded to the Repository using the Enterprise Manager, select and upload the **FileeWay.sar**. The File eWay is used to test the HL7 outbound Collaboration. The user must upload the File eWay (**FileeWay.sar**.) prior to uploading the TCP/IP HL7 eWay (**HL7eWay.sar**). This will ensure that all the dependent components, including the File eWay, are installed prior to importing the Collaboration projects. This is one of the tasks executed when the TCP/IP HL7 eWay is uploaded. See **Installing the TCP/IP HL7 eWay on an eGate supported system** on page 17.

Logical Host Requirements

The eWay must be set up and administered using the eGate Enterprise Designer, version 5.0.3 and above. For complete information on eGate Enterprise Designer system requirements, see the *ICAN Suite Installation Guide*.

2.3 External System Requirements

To enable the eWay to communicate properly with the TCP/IP system, you need:

- Host on which the server is running (host name)
- Port on which the server is listening (port number)

2.4 Installing the TCP/IP HL7 eWay

During the ICAN Suite installation process, the Enterprise Manager, a web-based application, is used to select and upload eWay and Add-on files (.sar files) from the ICAN installation CD-ROM to the Repository.

When the Repository is running on a UNIX operating system, the eWays are loaded using the Enterprise Manager on a Windows computer connected to the Repository server using Internet Explorer.

Product Readmes

Open and review the **Readme.txt** file (located in the root directory of the ICAN Installation's Repository CD-ROM) for the latest information prior to installing the eWays. After uploading the TCP/IP HL7 eWay's documentation to the Enterprise Manager, review the **TCP/IP HL7 eWay Readme** for eWay specific information and requirements.

Installing the TCP/IP HL7 eWay on an eGate supported system

The TCP/IP HL7 eWay can be installed during or after the installation of the ICAN Suite. The ICAN Suite installation process includes the following operations:

- Install the eGate Repository
- Upload products to the Repository
- Download components (including the eGate Enterprise Designer and Logical Host)

Follow the directions for installing the ICAN Suite in the *SeeBeyond ICAN Suite Installation Guide.* After you have installed eGate and other purchased core products, do the following:

1. From the Enterprise Manager's **ADMIN** tab, browse to the **Add-ons** directory and select the **ProductsManifest.xml**, and click **Submit**. The available Add-on product list is now displayed.
2. Browse to and select the following files located in the **Add-ons** directory:
   - **FileeWay.sar**: The user must upload the File eWay prior to uploading the TCP/IP HL7 eWay (**HL7eWay.sar**). When the TCP/IP HL7 eWay is uploaded it automatically imports the eWay's component Collaborations along with the sample (template) projects. The sample projects are dependent on the File eWay, and my generate an error if the File eWay is not installed prior to the import.
   - **HL7eWay.sar** After you have uploaded the File eWay, upload **HL7eWay.sar** to install the TCP/IP HL7 eWay.
3. Click on the Manifest File field's **Browse** option, and browse to the Add-ons **Documentation** directory. Select the **ProductsManifest.xml** and click **Submit**. The available Add-on documentation list is now displayed.
4. From the **Documentation** directory, select and upload the following file:
   - **HL7eWayDocs.sar** (to upload the TCP/IP HL7 eWay User's Guide, Javadoc, Readme, and sample projects to the Enterprise Manager)
5. Continue installing the eGate Integrator as instructed in the SeeBeyond *ICAN Suite Installation Guide.*

Adding the eWay to an Existing ICAN Suite Installation

If you are installing the eWay to an existing ICAN installation, do the following:

1. Complete steps 1 through 5 above.

2 Open the Enterprise Designer and select **Update Center** from the Tools menu. The Update Center Wizard appears.

3 For Step 1 of the wizard, simply click **Next**.

4 For Step 2 of the wizard, click the **Add All** button to move all installable files to the **Include in Install** field. Click **Next**.

5 For Step 3 of the wizard, wait for the modules to download, then click **Next**.

6 The wizard's Step 4 window displays the installed modules. Click **Finish.**

7 Restart the Repository and the IDE to complete the installation.

After Installation

Once the eWay is installed and configured it must then be incorporated into a project before it can perform its intended functions. See the *eGate Integrator User's Guide* for more information on incorporating the eWay into an eGate project.

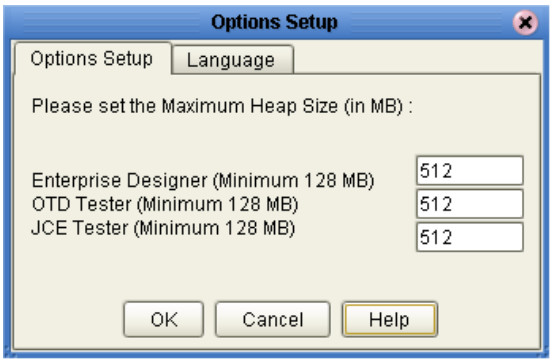
2.4.1. Increasing the ICAN Enterprise Designer Heap Size

Due to the size of the HL7 OTD Libraries, the Enterprise Designer **Heap Size** may need to be increased prior to using eGate with the HL7 OTD Library. If the heap size is not increased it may result in a **OutOfMemoryError** message. To increase the heap size from the Enterprise Designer do the following:

1 From the Enterprise Designer Menu bar click **Tools** and select **Options**. The **Options Setup** dialog box appears.

2 Increase the configured heap size for the Enterprise Designer, OTDTester, and JCE Tester to 512 MB as displayed in Figure 1. Click **OK**.

Figure 1 Options Setup - Heap Size



3 Close and restart the Enterprise Designer to allow your changes to take effect.

If an **OutOfMemoryError** message occurs while trying to open the Enterprise Designer, the heap size settings may be changed prior to starting the Enterprise Designer. In this case change the settings in the **heapSize.bat** file, located in **<*ICAN_Home*>\edesigner\bin** (where <*ICAN_Home*> is the ICAN install directory). Open **heapSize.bat** with a text editor and change the heap size settings from **128** to **512**. Save the file, and restart the Enterprise Designer.

Chapter 3

TCP/IP HL7 eWay Overview

This chapter explains some of the basics of TCP/IP HL7 functionality and how it operates with the TCP/IP HL7 eWay.

Chapter Topics

3.1 The TCP/IP HL7 eWay Architecture

The TCP/IP HL7 eWay's functionality is a result of the TCP/IP HL7 Resource Adapter (RA) combined with the predefined Inbound and Outbound HL7 Collaborations and the generic HL7 OTDs.

TCP/IP HL7 Resource Adapter

The TCP/IP HL7 eWay **Resource Adapter** communicates with external HL7 system, establishes and maintains the TCP/IP socket, manages message enveloping, maintains the sequence numbering file, and provides the HL7 protocol state to the Collaboration. The RA is configured from the eWay Properties Sheet.

HL7 Collaborations

The **Inbound** and **Outbound HL7 Collaborations** provide message validation, sequence numbering, ACK and NAK generation, and recourse actions. The predefined HL7 Collaborations are designed to implement the HL7 standard protocol and inter-operate with similar standard compliant systems by simply changing the eWay property configuration.

If a system does not conform to the HL7 specification, the Collaborations can be modified for that transaction by changing the Java code using the Collaboration Editor. The Collaborations Java code is designed to be available and "transparent" so that the user can easily reference the predefined Java code to see how it currently handles HL7 transactions and make the appropriate modifications.

The Enterprise Designer's Collaboration Editor enables the user to edit or create Java code to modify a Collaboration for their specific needs. In many cases this code can be created graphically (*drag and drop*), using the Collaboration Editor's Business Rules Designer. In situations where user is changing the code of a prebuilt Collaboration, it is recommended that the user duplicate the Collaboration first, and then modify it.

The Collaborations are designed to target "one unit of work" at a time, meaning the resolution of one message at a time. Once the current message transaction is resolved, the Collaboration is free to process the next HL7 message. The general form of the Collaborations is a state machine. Based on the state of the connection, the Collaboration performs the appropriate action.

Additional Collaborations can be added to a project to increase message flow.

Generic HL7 OTDs

The generic HL7 OTDs are version agnostic structures used to send and receive HL7 messages, acknowledgements, and NAKS. They provide the Collaboration with only the essential fields for implementing the HL7 protocol. If a user wishes to do version or message type specific functionality, then they may choose to add the appropriate OTD to the Collaboration.

ICAN Functionality

The TCP/IP HL7 eWay also takes advantage of the SeeBeyond ICAN facilities to provide journaling and error messaging, as well as monitoring, alerting, and logging. Journaling and error messages are sent to JMS queues which allow flexibility for post-processing. For example, invalid messages and their negative acknowledgements (NAKs), are sent to a JMS queue. This JMS queue can then be setup to allow the invalid HL7 messages to be viewed, corrected, and resubmitted automatically to the same eWay, using a SeeBeyond eVision Web application.

Error Queues

Each Collaboration automatically sends invalid messages that either have incorrect data or format, or are deemed unacceptable to a JMS error queue. The error generated by the message and, if appropriate, the NAK associated with it, are written as JMS properties of that message for processing later. The user has the option of sending errors to one common queue, or to a specified queue for each eWay, or some combination of both.

Journaling Queues

When journaling is enabled, the HL7 message and its related acknowledgement (ACK) are written to a JMS journal queue.

The number of JMS error queues or journal queues used by a project is up to the user. From the JMS queue, these messages can be accessed and written to file, sent to a database, sent to a Web application for processing, and so forth.

Monitoring

The Enterprise Manager provides a real-time picture of the eWay's state and status. The monitoring facilities for the eWay display the following status:

- eWay up or down
- Connected to external
- Current sequence number
- Date and time of the last transaction
- eWay properties

Alerting

Alerts are sent from the RA and the Collaborations when conditions are identified that endanger or stop the interface. Users can add their own custom alerts to the Collaborations.

Logging

Log messages are written from both the RA and the Collaborations. Users can also configure there own log messages. The level is set from the Enterprise Manager

For more information on monitoring, alerting and logging, see the ***eGate Integrator System Administration Guide*** and the ***eGate Integrator User's Guide***.

3.2 Modes and Roles

The HL7 eWay and eGate are capable of operating in two modes: **Standard** and **Delayed ACK**. Standard mode being the typical message exchange in HL7. That is, send an HL7 message, receive an HL7 ACK (or the opposite of this flow). Delayed ACK mode being where the exchange of a message requires two acknowledgements: one to confirm the message was received and the other from the external system that actually received the message, to verify that it was received.

In these two modes, the HL7 eWay and eGate may have a number of roles that they play within certain scenarios; that is, certain components can fulfill different responsibilities within a protocol. For example, the outbound Collaboration can fulfill 2 roles in the Delayed ACK mode: one as the Sender of messages that expects 2 delayed ACKS, and another as the Forwarder of ACKS from the external system.

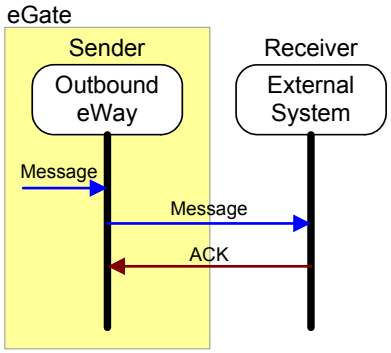
Standard Mode

In Standard mode, the HL7 eWay can assume two roles: Sender and Receiver. These are implemented in the outbound and inbound Collaborations respectively.

HL7 eWay Sender Role

The outbound Collaboration is the implementation of the Sender and the RA is configured for an outbound data flow as displayed in **Figure 2 on page 22**.

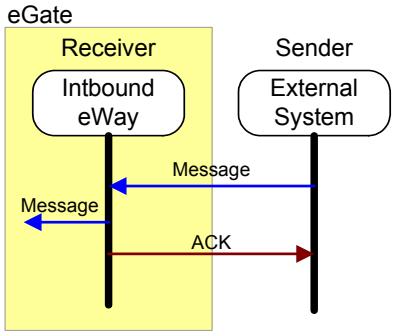
Figure 2 HL7 eWay Sender Role



HL7 eWay Receiver Role

The inbound Collaboration is the implementation of the Receiver role in conjunction with the RA being configured for inbound direction as displayed in Figure 3.

Figure 3 HL7 eWay Sender Role



Delayed ACK Mode

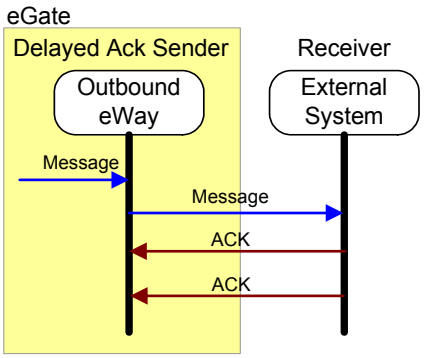
Delayed ACK mode is an extension to the basic or standard HL7 message exchange mode, where there is some middleware component between the **Sender** and the **Receiver**. The sending system expects to receive the ACK from the receiving system in addition to the middleware component. In this mode, eGate can assume two roles in the protocol: as the **Sender** and as the **Receiver**.

eGate Sender Role

In this role, eGate acts as the **Sender** in the exchange. The HL7 RA is configured for the outbound direction, and the HL7 outbound Collaboration is configured so that the **eGate Sends App Ack** property is set to **True**. This parameter is used even though eGate (HL7 eWay) technically does not send an Application ACK. Rather, it receives the Application ACK and provides the compliment behavior to the ACK sent by the inbound configured eWay. The two are related in the protocol.

The purpose of this role is for eGate to act as if it is a system that requires the Delayed ACKs so that it can communicate with a system that operates in the Delayed ACK Receiver role as displayed in **Figure 4 on page 23**.

Figure 4 HL7 eWay Sender Role

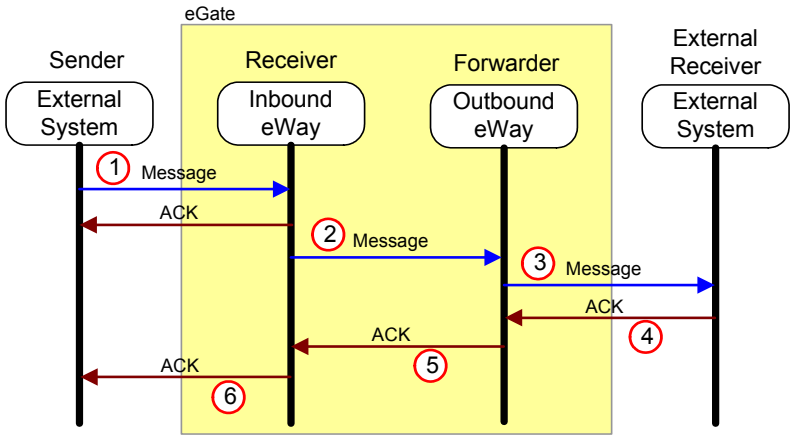


An example implementation of this role is available in the **HL7OutboundDelayedAck sample project** on page 141.

eGate Receiver and Forwarder Role

To perform this role, eGate is configured with two instances of the HL7 eWay, each performing its own role as **Receiver** and **Forwarder**.

Figure 5 HL7 eWay Sender and Forwarder Role



Note: *For Delayed ACK, the Receiver and Forwarder must be on the same integration server.*

The following steps convey the process displayed in Figure 5.

1. The **Receiver** accepts the HL7 message. It then returns the first ACK, with an **MSH - 5**, value "**D**" (for the Delayed ACK), to the **Sender**.
2. The **Receiver** sends the message to the **Forwarder**.
3. The **Forwarder** sends the message to the receiving **External System**.
4. The receiving **External System** sends an ACK to the **Forwarder**.

5 The **Forwarder** receives the ACK from the **External** and forwards it on to the **Receiver**.

6 The **Receiver** then forwards the ACK, with an **MSH - 5**, value "**F**" indicating that it is the receiving External System's ACK, to the **Sender**.

In the **Receiver** role, the HL7 RA is configured for **inbound**, and the inbound Collaboration is used with the parameter, **eGate Sends App Ack,** set to **True**. In the **Forwarder** role, the RA is set to **outbound**, the parameter, **Forward External Acks to eGate**, is set to **True**, and the **outbound Collaboration** is selected.

This configuration table presents the necessary parameters used to configure the eWay to assume the Delayed ACK roles.

Table 2: eWay Delayed Ack Configuration

Role	Direction	eGate Sends App Ack Property	Forward External Acks to eGate
Sender	Out	True	N/A
Receiver	In	True	N/A
Forwarder	Out	N/A	True

An example of the receiver role is provided in the **HL7ForwardMSG Inbound** sample project on page 140. The HL7Forward Outbound project provides a sample implementation of the Forwarder role.

3.3 Inbound Functionality

The Inbound TCP/IP eWay project, **prjHL7Inbound**, provides a sample implementation of an inbound flow using the eWay. It can be configured for "standard inbound" mode or in "forward message" mode.

3.3.1 Inbound eWay Data Flow

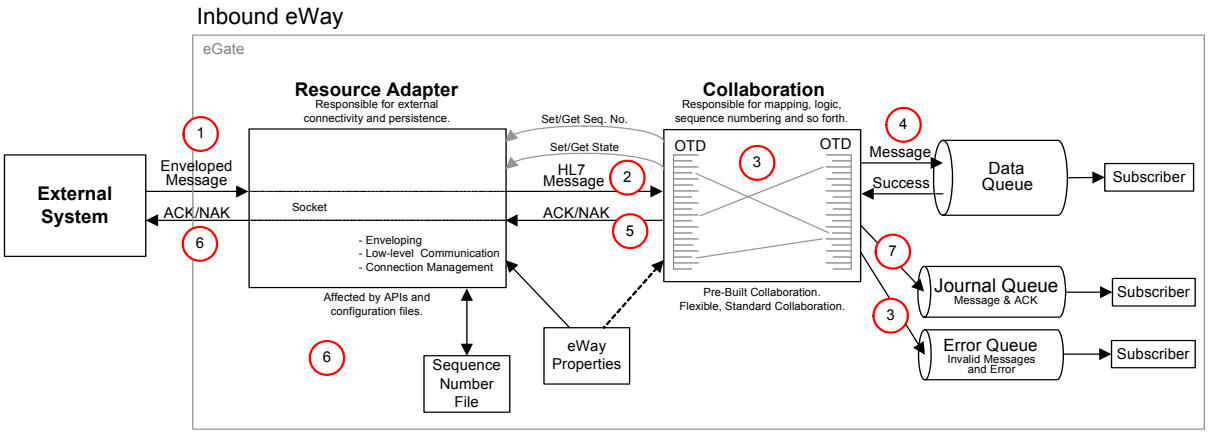
The inbound TCP/IP HL7 eWay project receives HL7 messages from an external system, sends an acknowledgement of the message to the external, provides sequence numbering, writes the HL7 message to a JMS data queue, and also writes the HL7 message and ACK to a JMS journal queue. Any error messages and NAKs are sent to a JMS error queue.

The HL7 data is processed so that all the fields in the MSH segment of the message are stored in an internal structure to generate an HL7 response. Non-HL7 data, including HL7 acknowledgments, automatically generate warnings in the eWay's log file and send an HL7 NAK to the external system.

Standard Inbound Message Mode Data Flow and Architecture

Figure 6 on page 25 illustrates the flow of data and architecture for an inbound eWay. A description of the diagram follows.

Figure 6 Data Flow and Architecture of the Inbound TCP/IP HL7 eWay



1 The external system sends the HL7 message to the eWay.

2 The Collaboration receives the HL7 message.

3 The Collaboration validates the message (if validate is enabled). If it fails, the Collaboration takes the configured recourse action. If the recourse action is stripped and the maximum number of retries has been exceeded, the message and error are written to the Error Queue.

4 The Collaboration writes the message to the Data Queue.

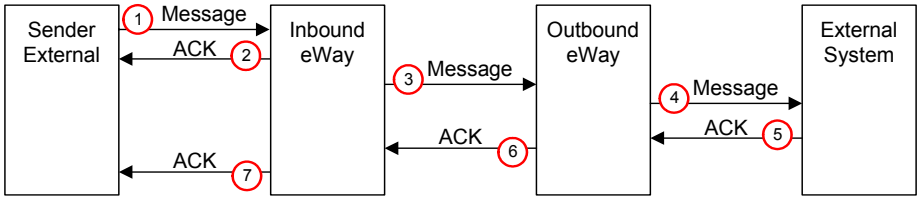
5 The Collaboration then creates the appropriate ACK and sends it to the RA.

6 The RA envelopes the ACK and sends it to the External System.

7 If Journaling is enabled, the message and its ACK are written to the Journal Queue.

Inbound Receiver Message Mode

Figure 7 Inbound Forward Message Role



The Inbound Receiver Message mode is used when the Delayed ACK is configured to fulfill the role of the Receiver in the Delayed ACK scenario. It accepts the message and acknowledges the External and then forwards the message to the component fulfilling the Forwarder role. It then accepts the ACK from the Forwarder and passes it on to the External that sent the message.

The following steps convey the Inbound Forward Message Role displayed in Figure 7.

1. The Sender External, sends an HL7 message to the Inbound eWay, which is configured as a Receiver (the **eGate Sends App Acks** is enabled).
2. The Inbound eWay receives the HL7 message and returns the first Acknowledgement to the External with an **MSA - 5**, value "**D**" for Delayed Acknowledgement. The External receives the ACK, validates the ACK (verifying that it is a Delayed ACK), and waits for another ACK.
3. The Inbound eWay creates a JMS message with the HL7 message as the payload, creates a "reply to" destination, and forwards the HL7 message to the Outbound Forwarder (to a JMS destination).
4. The Outbound Forwarder gets the HL7 message and forwards the message to the External System.
5. The External System receives the HL7 message and returns the HL7 ACK message to the Outbound Forwarder.
6. The Outbound Forwarder gets the HL7 ACK message and sends it to the Inbound Receiver eWay using the "reply to" destination.
7. The Receiver External reads the HL7 ACK message and forwards the second HL7 ACK message with an **MSA - 5**, value "**F**"to the Sender External. The Sender External then takes the appropriate action: for example, journaling the HL7 message and the HL7 ACK.

3.3.2 Message Verification

Message verification begins with reading the message from the external system. The message is expected to match the MLLP envelope, since both HLLP and MLLP envelopes have the Start of Block (SOB), End of Data (EOD), and a Carriage Return (CR) in common.

If a message fails the read verification, it is considered bad data. If read by an inbound eWay, this failure causes the eWay to generate a Canned HL7 NAK. (An outbound eWay ignores the message and logs a warning, reporting the nature of the problem to the log file.)

An HLLP envelope needs further verification as to whether it is data or a NAK, as well as the Block Checksum and Block Size. The eWay behaves as described above if the HLLP envelope verification fails.

After stripping the message envelope, the RA hands the de-enveloped message to the inbound Collaboration where it is parsed into the generic event OTD. This ensures that the general form and MSH segment are valid. If the MSH property is set, the Collaboration verifies that the fields specified in the HL7 segment section are the same as those of the received MSH, otherwise, a NAK is returned.

3.3.3 Acknowledgment Processing

eWay Generates HL7 Acknowledgment

In this scenario, the eWay generates an HL7 ACK after receiving and successfully storing the message in a queue; otherwise, it generates an HL7 NAK. The HL7 ACK or NAK is placed in the proper envelope and sent to the external system.

eGate Sends HL7 Acknowledgement

In this scenario, the eWay acts as a receiver in a Delayed ACK scenario, as described in **Inbound Receiver Message Mode** on page 25.

Canned HL7 NAK

A Canned HL7 NAK is created when a read error occurs, or when an message cannot be identified as an HL7 message. The initial test ensures that the message conforms to the lower-layer protocol. The Resource Adapter uses the MSH section parameters to create an appropriate NAK.

Recourse Actions

Recourse actions can be configured for the inbound eWay for the following conditions:

- **Action on Max Failed Read Retry** in response to the configured maximum number of canned negative acknowledgments that the eWay sends (see **Max Empty Read Retry** on page 55).
- **Action on Max Nak Received** in response to the configured maximum number of times the eWay attempts to read data from the external system after the read/receive operation returns nothing (see **Max NAK Receive Retry** on page 56).
- **Action on Max Nak Sent** in response to the configured maximum number of negative acknowledgments and Canned Naks the eWay sends (see **Max NAK Send Retry** on page 56 and **Max Canned NAK Send Retry**).
- **Action on Max No Response** in response to the configured maximum number of response timeouts the eWay allows while waiting for data from the external system (see **Max No Response** on page 56).
- **Action on Nak Received** in response to receiving an HL7 Application NAK from the external system.
- **Action on No Response** in response to the configured amount of time (in milliseconds) that the eWay waits for a response from the external system (see **Time To Wait For A Response** on page 57).

See **Recourse Actions** on page 33 for more information on the available Recourse Actions.

3.4 Outbound Functionality

The Outbound TCP/IP eWay project, **prjHL7Outbound**, can be implemented in a "Standard Outbound" mode, or in two "Forward message" modes: outbound Delayed ACK or Outbound Forwarder.

3.4.1 Outbound eWay Data Flow

In the outbound direction, the eWay receives HL7 messages from a JMS queue. Each Event is verified to ensure it contains HL7 data only. Legitimate HL7 data coming from eGate is enveloped into its configured format and sent to the external system.

A message in the JMS queue triggers the outbound Collaboration. The outbound Collaboration is provided with an HL7 message to send to the external system

The eWay waits for a configurable number of milliseconds (see **Time To Wait For A Response** on page 94) for an incoming HL7 ACK or NAK from the external system. After receiving an HL7 response from the external system, the eWay strips the message from its envelope and verifies its integrity.

Any non-HL7 acknowledgment received from the external system causes the eWay to resend the same message. If the incoming response is an HL7 ACK or NAK, the eWay may do either of the following, as dictated by its configuration:

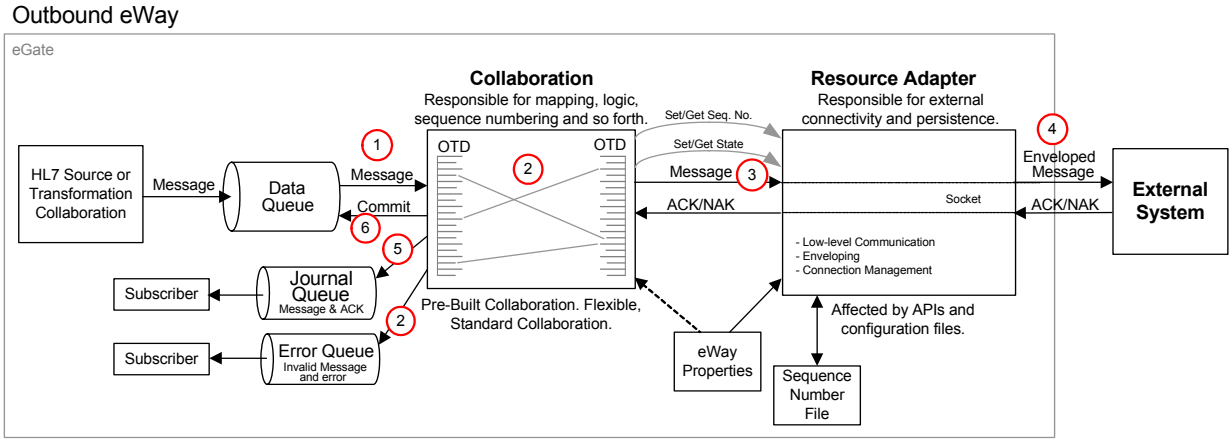
- Recourse Action on NAK received.
- Recourse Action on Max NAK received.

If journaling is set, these messages and their ACKs are placed in the journal file.

Outbound Standard Messaging Mode

Figure 8 illustrates the flow of data for an outbound eWay.

Figure 8 Data Flow for the Outbound TCP/IP HL7 eWay



The following steps convey the process displayed in **Figure 8 on page 28**.

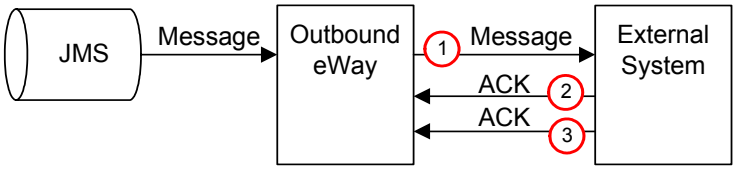
1. An HL7 message triggers the Collaboration. The outbound Collaboration is designed to accept the HL7 Messages.
2. The Collaboration maps the received message into the Generic Event OTD and validates the MSH segment. If validation is enabled, the Collaboration checks the MSH segment of the outbound messages against MSH values configured in the eWays Properties file. If the validation fails or the message cannot be parsed, the message and its error are written to the Error Queue. Note that the HL7 message is always checked for structural correctness.
3. The Collaboration sends the message to the RA.
4. The RA envelopes the message and sends it to the External System and waits for an ACK.
5. The Collaboration receives and validates the ACK, and then journals the ACK and the HL7 message (if journaling is enabled). If the Collaboration receives a NAK, the NAK and the HL7 message are sent to the error queue.
6. Finally, the Collaboration commits the JMS receive.

3.4.2 Outbound eWay Roles for Delayed ACK Scenarios

The outbound eWay can fulfill two roles in a Delayed ACK scenario.

Outbound Delayed ACK Role

Figure 9 Outbound Delayed ACK Role



The outbound Delayed Acknowledgement mode is used to communicate with an external system that is configured to receive messages in a delay ACK way; that is, it sends two ACKs: one to confirm the message was received, and one from the application that accepts the message.

For Delayed ACK mode, the process is similar to that of the standard outbound mode, except that it receives two ACKs. The initial ACK comes from the receiving system.

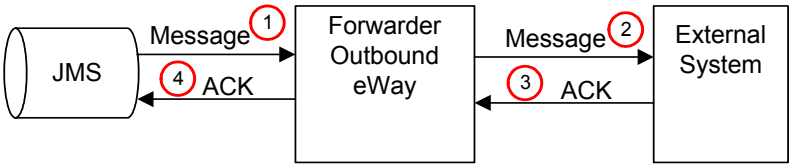
The following steps convey the Outbound Delayed Acknowledgement Role process displayed in Figure 9.

1. The outbound eWay, which is configured as Delayed Acknowledgement role, receives a message from JMS, and sends the message to the External System.
2. The External System receives the message and returns the first Acknowledgement to the outbound eWay with an **MSA - 5**, value "**D**" for Delayed Acknowledgement. The External Sender receives the ACK, validates the ACK (verifying that it is a Delayed ACK), and waits for another ACK.

3 The outbound eWay receives another HL7 ACK message (the second) and validates that the second HL7 ACK message is an **MSA - 5**, with a value of "**F.**" If the second ACK is valid, the eWay commits the message, otherwise it resends the message.

Outbound Forwarder Role

Figure 10 Outbound Forwarder Role



The Outbound Forward Message role is used in conjunction with the with the inbound eWay, which is also configured to handle delayed ACKs. No validation is preformed: the eWay acts as a "pass-through."

The following steps convey the Outbound Forwarder Role displayed in Figure 10.

1 Data is received by the Collaboration, from the JMS queue.

2 The Collaboration extracts the JMS property "reply to" destination from the message, but does no validation, and sends the message to the External System.

3 The eWay receives the ACK from the External System.

4 The Collaboration sends the ACK to the temporary topic that was contained in the "reply to."

Note: *For Delayed ACK, the Receiver and Forwarder must be on the same integration server.*

3.4.3 Message Verification

The only verification that the outbound eWay does is to ensure that the message parses into the generic Event OTD, and that the MSH uses the correct fields. The acknowledge is verified to ensure that the sent message is valid.

3.4.4 Acknowledgment Processing

eWay Generates HL7 Acknowledgment

In this scenario, the eWay generates an HL7 ACK after receiving and successfully storing the message in a queue; otherwise, it generates an HL7 NAK. The HL7 ACK or NAK is placed in the proper envelope and sent to the external system.

eGate Sends HL7 Acknowledgement

In this scenario, the eWay acts as a sender in a Delayed ACK scenario, as described in **Delayed ACK Mode** on page 22.

Canned HL7 NAK

A Canned HL7 NAK is created when a read error occurs, or when an message cannot be identified as an HL7 message. The initial test ensures that the message conforms to the lower-layer protocol. The Resource Adapter uses the MSH section parameters to create an appropriate NAK.

Recourse Action

Recourse action can be configured for the outbound eWay for the following conditions:

- **Action on Max Failed Read Retry** in response to the configured maximum number of times the eWay attempts to read data from the external system after the read/receive operation returns nothing (see **Max Empty Read Retry** on page 92).
- **Action on Max Nak Received** in response to the configured maximum number of negative acknowledgments the eWay receives (see **Max NAK Receive Retry** on page 93). This action is also used in cases where HL7 message validation failed prior to the sending of the HL7 message to the external system.
- **Action on Max No Response** in response to the configured maximum number of response timeouts the eWay allows, while waiting for data from the external system (see **Max No Response** on page 93).
- **Action on Nak Received** in response to eWay receiving an HL7 Application NAK from the external system.
- **Action on No Response** in response to the configured the amount of time (in milliseconds) that the eWay waits for a response from the external system (see **Time To Wait For A Response** on page 94).

See **Recourse Actions** on page 33 for more information on the available Recourse Actions.

Note: *The eWay has internal counters that keep track of all error conditions.*

3.5 General Functionality

This section explains the eWay's general functionality and features.

3.5.1 Non-blocking I/O

The non-blocking I/O feature will prevent the eWay from locking up when an attempt to read or write data blocks. This will enable the eWay to continue its operation in case of any communication errors.

If the read attempt fails for a configurable number of times, the eWay will exit or reset its connection to the external system, based on its configuration.

In the event of a failed write, it is able to resume its write operation to pick up where it previously left off until the entire message is successfully sent.

Without this feature, there is a possibility for the eWay to lock up when a read or write failure occurs, and to be unresponsive to all external messages, including requests from the user or the eGate monitor (for status).

3.5.2 HL7 Sequence Numbering Protocol

The eWay can be configured to use HL7 sequence numbering. The negotiation and incrementation of this number is automatically performed by the eWay. For more details on HL7 sequence numbering, refer to Appendix C (Lower Layer Protocols) of the *HL7 Standard* for the HL7 version you are using.

The sequence number file will open at start-up of the eWay. If the sequence number file does not exist, one is created and populated with a zero sequence number.

The sequence number file is updated on the inbound eWay when the eWay generates the HL7 ACK (this process is transparent to the user), and when the outbound eWay receives the HL7 ACK from the external system.

If you want to change the sequence number at runtime, you must suspend the eWay, edit and save the sequence number file, and reactivate the eWay.

To force the eWay to re-synchronize its sequence number with the external system, you must suspend the eWay, edit the file so that it contains a -1, and re-activate the eWay.

The minimum HL7 sequence number is 1. The maximum HL7 sequence number is 2 billion. A sequence number of "0" is used to start a session.

If the sequence numbers between the eWay and the external cannot be reconciled during start or when exchanging messages, the eWay will shutdown and wait for human intervention as dictated by the HL7 Standard.

3.5.3 Failed Message Handling

The eWay may be configured to send failed or Skipped messages (destined for the external system) to a JMS-based error queue. Messages that fail validation are also written to the error queue. Note that the inbound mode of the eWay will not write messages that fail the MLLP and HLLP validation. These are automatically NAKed and not passed to the Collaboration, but the are logged to the eWay's log file.

The failed or skipped message is written to JMS and the error type and message are written as the JMS properties:

- Error: the actual error message or NAK
- Error Type: the type of error: for example HL7_NAK_error or HL7_Validation_error.

Skipped messages are those which get continuously NAKed by the external system and thus are Skipped (if the eWay is configured as such).

If the eWay is configured for any other recourse action other than skip, the message remains in the queue.

3.5.4 Recourse Actions

The TCP/IP HL7 eWay Recourse Actions include Reset, Resend, Skip Message, and Exit

On **Reset**, the eWay drops its connection and then attempts to reconnect.

On **Resend**, the sequence number file and journal file are opened again (provided the newly loaded configuration parameters are set for sequence numbering and journaling.

On **Skip Message**, the eWay remains connected, but writes the message to an error queue.

On **Exit**, the eWay closes its journal file and sequence number file (provided these were configured for use). The eWay terminates its connection with the external system and shuts down. This allows the user to modify these files and resolve any errors. Once the corrections are made, the eWay can be reactivated from the ICAN Monitor.

Stopping the Collaboration with a Fatal Alert

When the **Exit** Recourse Action is triggered, it logs the error that caused the action, shuts down the Collaboration, which in turn causes the HL7 message to roll back, and sends an alert to the ICAN Monitor.

The **Exit** Recourse Action calls the fatal alerter in the Collaboration.

```
alerter.fatal("error message","HL7");
```

The argument **"*error message*"** is the user-configured alert message. The argument **"HL7"** is the source component (this must be "HL7").

The **Exit** Recourse Action should be applied to any error condition which requires human intervention to correct an error. Once the error condition is resolved, the Collaboration can be restarted from the ICAN Monitor.

3.6 TCP/IP HL7 eWay Operation

This section explains the basic elements of the TCP/IP HL7 eWay's general operation. These elements are:

- **Direction**
- **Connection Type**
- **Lower Layer Protocol**
- **HL7 Acknowledgment Level** on page 36
- **Journaling** on page 37
- **Error Queues** on page 37
- **Alerts and Monitoring** on page 37
- **Support for HL7 Version 2.5 SFT Segments** on page 37
- **Delayed Acknowledgements** on page 37

3.6.1 Direction

The TCP/IP HL7 eWay can be configured as either **HL7 Inbound** or **HL7 Outbound**. This option is selected upon first opening the TCP/IP HL7 eWay properties from the Connectivity Map. Double-click the eWay to open a configuration templates dialog box from which Inbound or Outbound can be selected. Once a choice is made, the Properties Sheet with the appropriate configuration template appears.

3.6.2 Connection Type

The connection type indicates the role the eWay plays in establishing a socket with the external. The role can be as a Client: where the RA connects to the external, or as a Server: where the RA waits for a connection.

Connected as a TCP/IP HL7 Client

As TCP/IP HL7 Client, the eWay obtains the target host name and TCP/IP port number from the properties file, and attempts to establish connection with the specified host. If the connection attempt fails, the eWay continues to try to connect until the connection is made. The eWay attempts to connect immediately at start-up; however, it waits for a configurable number of seconds between connection resets before attempting further connections.

Connected as a TCP/IP HL7 Server

As a TCP/IP HL7 server, the eWay obtains the TCP/IP port number from the properties file. The eWay becomes a TCP/IP HL7 Server on the local machine, at the pre-configured TCP/IP port number.

Once it becomes a server, the eWay listens on its TCP/IP port for any incoming connections. When the external system tries to connect, the eWay accepts the connection and the link is established.

The eWay continues to listen for any other incoming connections and tracks the time since it received the last transaction from the external system. If a second connection is attempted, the eWay checks the elapsed time to determine whether or not the period since the last transaction received from the external is greater than the configurable **SoLinger Timeout**. If so, it abandons the old connection and accepts the new one. However, if the time elapsed since the last transaction from the external is still within the response timeout period, the new incoming connection is dropped.

3.6.3. Lower Layer Protocol

This section describes the two supported envelope types used in the HL7 protocol:

- **HLLP** (Hybrid Lower Layer Protocol)
- **MLLP** (Minimal Lower Layer Protocol)

Note: *According to the HL7 Standard, the End of Block (EOB) Character for the HLLP and MLLP Envelopes must be set to a Carriage Return (CR). Although the End of Block Character is a configurable parameter, it is strongly recommended that this parameter be set to a Carriage Return for the eWay to be fully compliant with the HL7 Standard.*

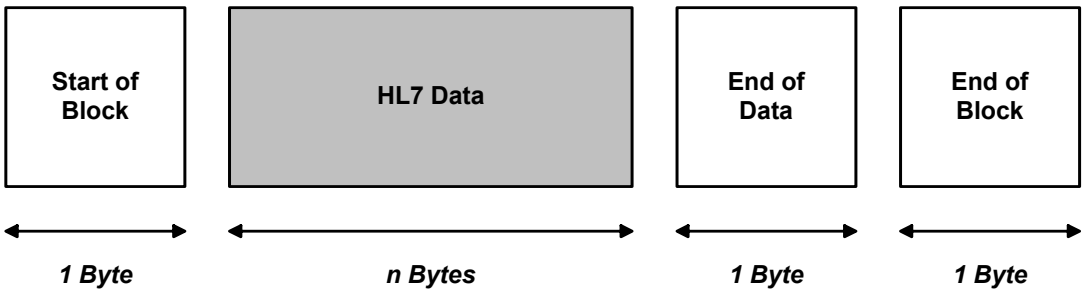
Both envelope types use the following configuration parameters:

- **Start Block Character** on page 43
- **End Data Character** on page 43
- **End Block Character** on page 42

MLLP

The **MLLP** envelope structure is displayed in Figure 11.

Figure 11 MLLP Envelope Structure



This envelope consists of a **Start of Block** component, a **Data** component, an **End of Data** component, and an **End of Block** component. The size of the **HL7 Data** field is

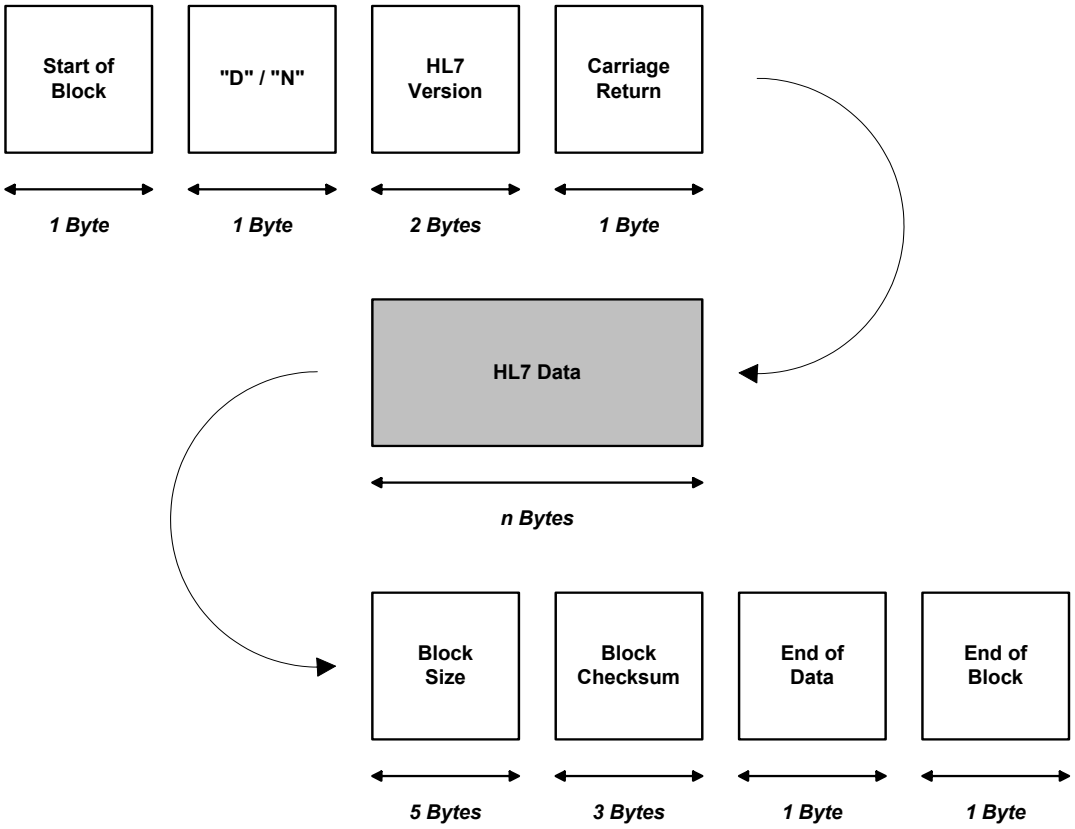
determined by the length of the data (number of bytes between start and end), with a maximum size of 99999 Bytes.

For more detail, refer to Appendix C (Lower Layer Protocols) of the *HL7 Standard* for the HL7 version you are using.

HLLP

The **HLLP** envelope structure is displayed in **Figure 12 on page 36**.

Figure 12 HLLP Envelope Structure



This envelope consists of a **Start of Block** component, a **'D'** (Data) or **'N'** (NAK) indicator, an **HL7 Version** component, a **Carriage Return**, a **Data** component, a **Block Size** component, a **Block Checksum** component, an **End of Data** component, and an **End of Block** component. The size of the **HL7 Data** field is determined by the length of the data (number of bytes between start and end), with a maximum size of 99999 Bytes.

For more detail, refer to Appendix C (Lower Layer Protocols) of the *HL7 Standard* for the HL7 version you are using.

3.6.4 HL7 Acknowledgment Level

The eWay supports the sending and receiving of both HL7 Acknowledgement types:

- (Application acknowledgment) The acknowledgement is sent when the message is successfully received.
- (Commit acknowledgment) The acknowledgement is sent after the message is successfully received and committed to the application database.

3.6.5 Journaling

The eWay provides the option to journal successfully received or sent messages and their corresponding ACKs. The messages are sent to a JMS queue or topic, as configured by the user, and the ACKs are stored as a JMS property, HL7_ACK, of that message.

It is expected that, when enabled, the journal queue has at least one or multiple subscribers that process the contents of the queue so that it remains manageable. For example, the Batch eWay or a database eWay could periodically consume the messages by writing them to a file or a database.

3.6.6 Error Queues

The eWay provides a mechanism to store failed or stripped messages in a JMS queue or topic. The advantage of this is that the messages are then saved in a form readily usable by the other data flows, that can automatically process these messages or make them available to some type of human intervention or message repair, using tools like the JMS queue editor or an eVision application.

3.6.7 Alerts and Monitoring

If the eWay loses the connection to the external system in any direction or connection type, due to a crash, shutdown, or suspension (including recourse actions), an alert is generated. The monitor's status of that eWay is changed to "down" and the eWay's icon is encased in a red warning box. The monitor also displays the number of messages it has processed along with the date and time of the last message sent.

3.6.8 Support for HL7 Version 2.5 SFT Segments

HL7 version 2.5 adds a new SFT segment to every message. The eWay not only sends and receives messages with the new segment, it can automatically create and populate them, using information from the eWay properties, for the outbound message and the ACK sent from the inbound mode. This feature is only available when the Version ID property is set as 2.5.

3.6.9 Delayed Acknowledgements

The eWay supports Delayed Acknowledgements in either direction and in a number of roles. This functionality is described in detail in **Outbound eWay Roles for Delayed ACK Scenarios** on page 29

Chapter 4

Configuring TCP/IP HL7 eWay Properties

This chapter explains how to configure the TCP/IP HL7 eWay properties.

What's in This Chapter

4.1 Creating and Configuring the TCP/IP HL7 eWay

All eWays contain a set of parameters with properties unique to that eWay type. After the eWays are established and a TCP/IP HL7 External System is created in the Project's Environment, the eWay parameters can be modified for your specific system. The TCP/IP HL7 eWay parameters are modified from two locations:

- From the **Connectivity Map**. These parameters most commonly apply to a specific component eWay, and may vary from other eWays (of the same type) in the project.
- From the **Environment Explorer tree**. These parameters are commonly global, applying to all eWays (of the same type) in the project. The saved properties are shared by all eWays in the TCP/IP HL7 External System window.

4.1.1 Selecting TCP/IP HL7 as the External Application

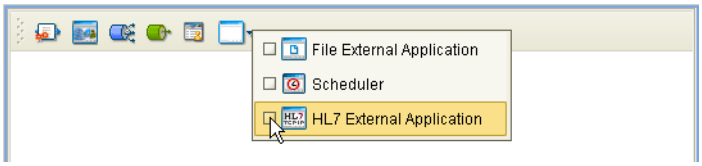
To create a TCP/IP HL7 eWay you must first create a TCP/IP HL7 External Application in your Connectivity Map. TCP/IP HL7 eWays are located in the link between a TCP/IP HL7 External Application and a Service. Services are containers for Java Collaborations, Business Processes, eTL processes, and so forth.

To create the TCP/IP HL7 External Application

1. From the Connectivity Map toolbar, click the **External Applications** icon.

2 Select the **TCP/IP HL7 External Application** from the menu (see **Figure 13 on page 39**). The selected TCP/IP HL7 External Application icon appears on the Connectivity Map toolbar.

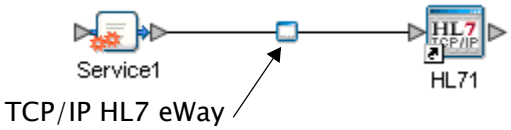
Figure 13 External Applications Selection Menu



3 Drag the new **TCP/IP HL7 External Application** from the toolbar onto the Connectivity Map canvas. This represents an external TCP/IP HL7 system.

From the Connectivity Map, you can associate (bind) the External Application with the Service to establish an eWay (see Figure 14).

Figure 14 eWay Location



When TCP/IP HL7 is selected as the External Application, it automatically applies the default TCP/IP HL7 eWay properties provided by the OTD to the eWay that connects it to the Service. These properties can then be modified for your specific system using the **Properties Sheet**.

4.1.2 Modifying the TCP/IP HL7 eWay Properties

A Project's eWay properties can be modified after the eWays have been established in the Connectivity Map and the Environment has been created.

Modifying the TCP/IP HL7 eWay (Connectivity Map) Properties

1 From the Connectivity Map, double click the eWay icon located in the link between the associated External Application and the Service. The **Templates** dialog box appears.

2 From the Templates dialog box, select **HL7 Inbound eWay** or **HL7 Outbound eWay** (for this example select **HL7 Outbound**). Click **OK**.

3 The eWay **Properties Sheet** opens to the **TCP/IP HL7 eWay Outbound Connectivity Map Properties**. Make any necessary modifications and click **OK** to save the settings.

Modifying the TCP/IP HL7 eWay (Environment Explorer) Properties

1 From the Environment Explorer tree, right-click the **TCP/IP HL7 Outbound** External System or **TCP/IP HL7 Inbound** External System. Select **Properties** from the shortcut menu. The **Properties Sheet** opens with the TCP/IP HL7 eWay Environment properties.

2 Make any necessary modifications to the Environment properties of the TCP/IP HL7 eWays, and click **OK** to save the settings.

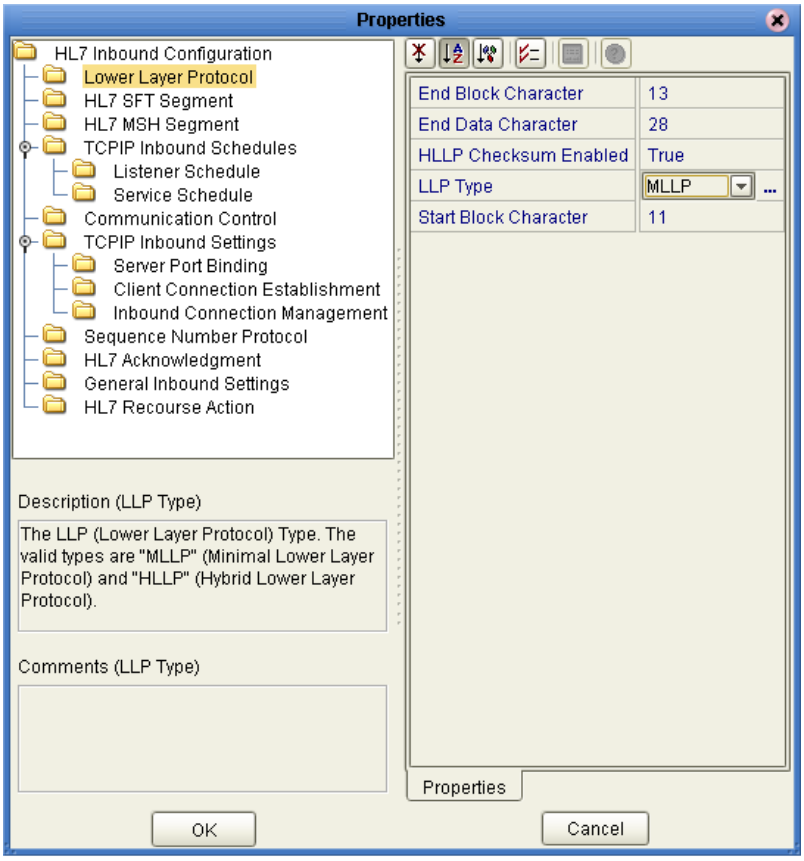
4.1.3 Using the Properties Sheet

Modifications to the eWay configuration properties are made from the TCP/IP HL7 eWay Properties Sheet.

To modify the default eWay configuration properties

1. Open the Properties Sheet to the TCP/IP HL7 eWay default properties. An eWay has two different sets of parameters: those specific to that particular eWay (accessed from the Connectivity Map), and those that are common to all eWays of this type (accessed from the Environment Explorer tree). In addition, the TCP/IP HL7 eWay properties will differ depending on whether the eWay is a Inbound or Outbound eWay. Open the Properties Sheet to the TCP/IP HL7 Inbound eWay's Connectivity Map default properties.
2. From the upper-right pane of the Properties Sheet, select a directory under root Configuration directory. The parameters contained in that directory are now displayed in the Properties pane of the Properties Sheet. For example, clicking on the **Lower Layer Protocol** directory displays the editable parameters in the right pane, as shown in **Figure 15 on page 40**.

Figure 15 Properties Sheet: TCP/IP HL7 Properties



3. Click on any property field to make it editable. For example, click on the **LLP Type** parameter to edit its value. If a parameter's value is true/false or multiple choice, the field reveals a submenu of property options.

Click on the ellipsis (...) in the properties field to open a separate configuration dialog box. This is helpful for large values that cannot be fully displayed in the parameter's property field. Enter the property value in the dialog box and click **OK**. The value is now displayed in the parameter's property field.

4 A description of each parameter is displayed in the **Description** pane when that parameter is selected, providing an explanation of any required settings or options.

5 The **Comments** pane provides an area for recording notes and information regarding the currently selected parameter. This is saved for future referral.

6 After modifying the configuration properties, click **OK** to save the changes and close the Properties Sheet.

4.2 TCP/IP HL7 eWay Configuration Properties

The TCP/IP HL7 eWay includes four following sets of properties:

- **TCP/IP HL7 Inbound eWay Connectivity Map Properties** on page 42
- **TCP/IP HL7 Inbound eWay Environment Properties** on page 71
- **TCP/IP HL7 Outbound eWay Connectivity Map Properties** on page 73
- **TCP/IP HL7 Outbound eWay Environment Explorer Properties** on page 97

4.3 TCP/IP HL7 Inbound eWay Connectivity Map Properties

The TCP/IP HL7 Server (inbound) eWay configuration parameters, accessed from the Connectivity Map, are organized into the following sections:

4.3.1 Lower Layer Protocol

Provides Lower Layer Protocol (LLP) configuration settings. This section contains the following top level parameters:

End Block Character

Description

Specifies the End Block Character (the last envelope marker character in the HL7 envelope) as a decimal ascii number.

Required Values

A decimal within the range of 1 to 127. To be strictly compliant with the HL7 Standard, this parameter MUST be set to a Carriage Return (decimal 13). The default value is 13.

End Data Character

Description

Specifies the End Data Character (The second to the last envelope marker character in the HL7 envelope) as a decimal ASCII number. The allowed range is 1 to 127.

Required Values

A decimal within the range of 1 to 127. Unless there is a conflict, the value should be ASCII FS (decimal **28**). The default value is 28.

HLLP Checksum Enabled

Description

Specifies whether the HLLP (Hybrid Lower Level Protocol) Checksum is enabled or disabled.

Required Values

True or **False**. **True** indicates that the HLLP Checksum is enabled.

LLP Type

Description

Specifies the LLP (Lower Layer Protocol) type. The valid types are:

- **MLLP** (Minimal Lower Layer Protocol)
- **HLLP** (Hybrid Lower Layer Protocol)

For more information on the available envelope types see **Lower Layer Protocol** on page 35

Required Values

MLLP or **HLLP**. MLLP is the configured default value.

Start Block Character

Description

Specifies the Start Block Character (the first envelope marker character in the HL7 envelope) as a decimal ascii number.

Required Values

A decimal within the range of 1 to 127. Unless there is a conflict, the value should be ASCII VT (decimal 11). The default value is 11.

4.3.2 HL7 SFT Segment

Provides HL7 SFT Segment configuration settings. The SFT segment is available starting with HL7 version 2.5. This segment provides additional information about one or more software products used as sending applications. The primary purpose of this segment is for diagnostic use. There may be additional uses per site-specific agreements.

This section contains the following top level parameters:

Note: *This section only applies to HL7 version 2.5.*

Enable

Description

Specifies whether the **SFT** optional segment is enabled in the ACK.

Note: *If enable is set to true, and the HL7 version is not configured as 2.5, the eWay will error upon startup.*

Required Values

True or **False**. **True** indicates that the SFT segment is enabled in the ACK.

Software Binary ID

Description

Specifies HL7 segment **SFT-04**, the Software Binary ID. This property is available starting with HL7 version 2.5. Software Binary IDs are issued by a vendor for each unique software version instance. These IDs are used to differentiate between differing versions of the same software. Identical Primary IDs indicate that the software is identical at the binary level, but configuration settings may differ.

Required Values

The unique Software Binary ID. The configured default is **5.0.0**.

Note: *This property only applies to HL7 version 2.5.*

Software Certified Version or Release Number

Description

Specifies HL7 segment **SFT-02**, the **Software Certified Version or Release Number**. The latest software version number or release number for the sending system, helps to provide a more complete profile of the application that is sending or receiving HL7 messages. Version numbers are important in identifying the specific release of an application. In some situations, the receiving application validates the software certified version or release number against a list of "certified" versions or releases of the particular software. This helps determine whether the sending application adheres to specific Business Rules required by the receiving application. Alternatively, the software may perform different processing, depending on the version of the sending software.

Required Values

The software certified version or release number. The configured default is **ICAN 5**.

Note: *This property only applies to HL7 version 2.5.*

Software Install Date

Description

Specifies HL7 segment **SFT-06**, the Software Install Date. This is the date on which the submitting software was installed at the sending site. the software install date on its own can often provide key information about the behavior of the application. This is necessary for providing a more complete profile of the sending application.

Required Values

The date of installation for the sending application software.

Note: *This property only applies to HL7 version 2.5.*

Software Product Information

Description

Specifies HL7 segment **SFT-05**, software product identification information. This may include a description of the software application, configuration settings, modifications made to the software, and so forth. This field can contain any additional information about the sending application, with the transaction it has submitted. The information is used for diagnostic purposes and may provide greater flexibility for identifying the application software.

Required Values

Information that may help to identify the specific sending software. This field should only be used when performing diagnostics.

Note: *This property only applies to HL7 version 2.5.*

Software Product Name

Description

Specifies HL7 segment **SFT-03**, the name of the software product that submitted the transaction. The software product name is a key component for identifying the sending application.

Required Values

The sending software product name. The default value is **HL7 eWay**.

Note: *This property only applies to HL7 version 2.5.*

Software Vendor Organization

Description

Specifies HL7 segment **SFT-01**, the name of the company that publishes and/or distributes the sending software that created the transaction. The purpose of this field, along with the remaining fields in this segment, is to provide a more complete profile of the sending applications. The Software Vendor Organization field identifies the vendor who is responsible for maintaining the application.

Required Values

The name of the sending software publisher or vendor. The default value is **SeeBeyond Technology Corporation**.

Note: *This property only applies to HL7 version 2.5.*

4.3.3 HL7 MSH Segment

Provides HL7 MSH Header segment configuration settings. This section contains the following top level parameters:

Alternate Character Set Handling Scheme

Description

Specifies the value for the alternate character set handling scheme to be used when any alternative character sets are used and a special handling scheme is necessary (see HL7 Table 0356). This is the 20th field in the HL7 MSH segment **(MSH-20)**.

Required Values

Available values include **ISO 2022-1994**, **2.3**, or **<null>** (blank). Leaving the field blank indicates that no character set switching will occur.

Character Set

Description

Specifies the character set(s) in use by the messages (see HL7 Table 0211). If the field is left blank, the character set in use is understood to be the 7-bit ASCII set. This is the 18th field in the HL7 MSH segment (**MSH-18**).

Required Values

The configured default is **8859/1** (printable 7-bit ASCII character set). See HL7 Table 0211 for available values and descriptions.

Conformance Statement ID

Description

The Conformance Statement ID (Message Profile Identifier in V2.5) is a unique identifier that applies to a query's Conformance Statement, or as a Message Profile Identifier, asserts constancy with a message profile (grammar, syntax, usage, and so forth). This is the 21st field in the HL7 MSH segment (**MSH-21**).

Required Values

An HL7 Conformance Statement ID value or leave blank.

Country Code

Description

Specifies a code that indicates the country of origin for the message (see HL7 Table 0399). Used to specify default elements in a message, such as currency. This is the 17th field in the HL7 MSH segment (**MSH-17**).

Required Values

The Country Code value uses the 3-character (alphabetic) form of ISO 3166. The default value is **USA**.

Encoding Characters

Description

Specifies four encoding characters in the following order:

- Component separator
- Repetition separator
- Escape character
- Subcomponent separator

This is the second field in the HL7 MSH segment (**MSH-02**).

Required Values

HL7 encoding characters in the respective order. The configured default is: **^~\&** (ASCII 94, 126, 92, and 38, respectively).

Field Separator

Description

Specifies the separator between the segment ID and the first real field. This value defines the character that is used as a separator for the rest of the message. This is the first field in the HL7 MSH segment (**MSH-01**).

Required Values

Field Separator character value as a decimal ascii number. The allowed range is 1 to 127. The default setting is 124 which is character '|'.

Principal Language of Message

Description

Specifies the principal language of the message. Codes come from ISO 639. This is the 19th field in the HL7 MSH segment (**MSH-19**).

Required Values

The 2-character ISO 639 alphabetic code.

Processing ID

Description

Specifies the sub-component Processing ID of MSH-11. MSH-11 is used to indicate whether a message is processed as defined in the HL7 Application (level 7) Processing rules.

Required Values

Requires one of the following:

- **D** - for Debugging
- **P** - for Production
- **T** - for Training

In some cases there may be an additional sub-component "Processing Mode" following the initial value.

Receiving Application

Description

Specifies the receiving application among other applications within the network enterprise. The network enterprise consists of the applications that participate in the exchange of HL7 messages within the enterprise. This is the fifth field in the HL7 MSH segment (MSH-05).

Required Values

User defined value for the HL7 receiving application. The default value is **SeeBeyond HL7 eWay**.

Receiving Facility

Description

Specifies (further identifies) the receiving application among multiple identical instances of the application running on behalf of different organizations. This is the sixth field in the HL7 MSH segment (MSH-06).

Required Values

User defined value for the HL7 receiving facility. The default value is **SeeBeyond HL7 eWay**.

Security

Description

Specifies the implemented application level security features. This is the eighth field in the HL7 MSH segment (MSH-08).

Required Values

Under development by HL7.

Sending Application

Description

Specifies the sending application among other applications within the network enterprise. The network enterprise consists of the applications that participate in the exchange of HL7 messages within the enterprise. This is the third field in the HL7 MSH segment (MSH-03).

Required Values

User defined value for the HL7 sending application. The default value is **SeeBeyond HL7 eWay**.

Sending Facility

Description

Specifies (further identifies) the sending application among multiple identical instances of the application running on behalf of different organizations. This is the forth field in the HL7 MSH segment (MSH-04).

Required Values

User defined value for the HL7 sending facility. The default value is **SeeBeyond HL7 eWay**.

Validate MSH

Description

Specifies whether to validate the MSH segment of the data message (for inbound) and the MSH segment of the ACK (for outbound).

This parameter is used for inbound Collaboration code.

Required Values

True or **False**. True indicates that the Collaboration validates the MSH segment.

Note: *This property does not affect structural validation of the whole HL7 message itself. Structural validation is always performed.*

Version ID

Description

Specifies the particular HL7 version to be matched by the receiving system to its own version. This is the 12th field in the HL7 MSH segment (MSH-12).

Required Values

The HL7 Standard version value as displayed in HL7 Table 0104 - Version ID.

4.3.4 TCPIP Inbound Schedules - Listener Schedule

This section configures the scheduler used by the inbound TCP/IP Server. The server waits for a new client connection establishment request. These parameters are used to configure the listener/monitor that listens on the specified port.

Two J2EE schedulers are available (see **Scheduler** on page 52):

- **Timer Service**: available for **J2EE (prior to JCA 1.5)** and above. This scheduler is configured using the **At Fixed Rate**, **Delay**, and **Period** properties.
- **Work Manager**: available for **JCA 1.5** and above. This scheduler is configured using the **Delay** and **Period** properties.

Both schedulers provide the functionality required by the inbound TCP/IP Server.

This section contains the following top level parameters:

At Fixed Rate

Description

Specific to the **Timer Service** configuration only. Specifies whether a **Fixed-Rate** execution or **Fixed-Delay** execution is used.

- **Fixed-Rate:** A fixed-rate execution means that each execution is scheduled relative to the scheduled time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying Object.wait(long) is accurate).
- **Fixed-Delay:** A fixed-delay execution means that each execution is scheduled relative to the actual time of the previous execution. If an execution is delayed for any reason (such as garbage collection or other background activity), subsequent executions will be delayed as well. As a result, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the

system clock underlying Object.wait(long) is accurate).

Required Values

True or **False**. **True** indicates that a fixed-rate execution is used. **False** indicates that a fixed-delay execution is used.

Delay

Description

Applies to both the **Timer Service** or the **Work Manager**. Specifies, in milliseconds, the length of delay time before the task is executed.

Required Values

The length of time before the task is executed, in milliseconds (1,000 milliseconds is equal to 1 second).

Period

Description

Applies to both the **Timer Service** or the **Work Manager**. Specifies the regular interval, in milliseconds, between successive task executions.

Required Values

The appropriate length of time between successive task executions, in milliseconds (1,000 milliseconds is equal to 1 second).

Schedule Type

Description

This property configuration, though visible from the Properties Editor, is disabled. The only available schedule type is Repeated, indicating that the task is scheduled for repeated execution at regular intervals defined by **Period** property in this section (see **Period** on page 52).

Scheduler

Description

Specifies the scheduler type for this inbound communication. There are two options:

- **Timer Service**: The task is scheduled through the J2EE Timer Service. Timer Service is supported by **J2EE (prior to JCA 1.5)** and above.
- **Work Manager**: The task is scheduled through the J2EE Work Manager. Work Manager is supported by **JCA 1.5** and above.

Required Values

Select **Timer Service** or **Work Manager**. If your container doesn't support JCA Work Manager (prior to JCA1.5), select Timer Service.

4.3.5 TCPIP Inbound Schedules - Service Schedule

This section configures the scheduler used by the TCP/IP Server that executes the business tasks (Collaboration Rules) over the existing connection. This scheduler affects the actual Business Rules defined by the user.

Two J2EE schedulers are available (see **Scheduler** on page 54):

- **Timer Service**: available for **J2EE (prior to JCA 1.5)** and above. This scheduler is configured using the **At Fixed Rate**, **Delay**, **Period**, and **Schedule Type**, properties.
- **Work Manager**: available for **JCA 1.5** and above. This scheduler is configured using the **Delay**, **Period**, and **Schedule Type**, properties.

Both schedulers provide the functionality required by the inbound TCP/IP Server.

This section contains the following top level parameters:

- **At Fixed Rate** on page 53
- **Delay** on page 54
- **Period** on page 54
- **Schedule Type** on page 54
- **Scheduler** on page 54

At Fixed Rate

Description

Specific to the **Timer Service** configuration only. Specifies whether a **Fixed-Rate** execution or **Fixed-Delay** execution is used. This is used for the "Repeated" schedule type by the "Timer Service" scheduler.

- **Fixed-Rate:** A fixed-rate execution means that each execution is scheduled relative to the scheduled time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying Object.wait(long) is accurate).
- **Fixed-Delay:** A fixed-delay execution means that each execution is scheduled relative to the actual time of the previous execution. If an execution is delayed for any reason (such as garbage collection or other background activity), subsequent executions will be delayed as well. As a result, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying Object.wait(long) is accurate).

Required Values

True or **False**. **True** indicates that a fixed-rate execution is used. **False** indicates that a fixed-delay execution is used.

Delay

Description

Applies to both the **Timer Service** or the **Work Manager**. Specifies in milliseconds, the length of delay time before the task is executed.

Required Values

The appropriate length of time before a task is executed, in milliseconds (1,000 milliseconds is equal to 1 second).

Period

Description

Applies to both the **Timer Service** or the **Work Manager**. Specifies the regular interval in milliseconds between successive task executions. This is used for the **Repeated** schedule type (see **Schedule Type** on page 54).

Required Values

The appropriate length of time between successive task executions, in milliseconds (1,000 milliseconds is equal to 1 second).

Schedule Type

Description

Applies to both the **Timer Service** or the **Work Manager**. Specifies whether the task is scheduled to occur once or be repeated.

- **OneTime**: The task will be scheduled for one-time execution.
- **Repeated**: The task will be scheduled for repeated execution at regular intervals defined by **Period** property in this section (see **Period** on page 54).

Required Values

Select **OneTime** or **Repeated**.

Scheduler

Description

Specifies the scheduler type for this inbound communication. There are two options:

- **Timer Service**: The task is scheduled through the J2EE Timer Service. Timer Service is supported by **J2EE (prior to JCA 1.5)** and above.
- **Work Manager**: The task is scheduled through the J2EE Work Manager. Work Manager is supported by **JCA 1.5** and above.

Required Values

Select **Timer Service** or **Work Manager**. If your container doesn't support JCA Work Manager (prior to JCA1.5), select Timer Service.

4.3.6 Communication Control

Controls data transferring (sending/receiving) over TCP/IP connection. This section contains the following top level parameters:

Enable Journaling

Description

Specifies whether message journaling is enabled.

This parameter is used for inbound Collaboration code.

Required Values

True or **False**. True indicates that journaling is enabled.

Max Canned NAK Send Retry

Description

Specifies the maximum number of canned negative acknowledgments that the eWay sends before taking recourse action (see **Action on Max Nak Sent** on page 69).

Required Values

The appropriate maximum number of canned NAK to send before taking recourse action. **0** indicates that the eWay will not attempt to create or send a canned NAK.

Max Empty Read Retry

Description

Specifies the maximum number of times the eWay attempts to read data from the external system after the read/receive operation returns nothing. This applies to the read or receive operation after a response starts to arrive. **Empty Read** means that a timeout occurs on the read/receive operation, which takes the **SoTimeout** parameter in the **TCPIP Server Base Settings** section as the applied timeout setting (see **SoTimeout** on page 61). The corresponding recourse action is specified by the **Action on Max Failed Read Retry** on page 68.

Required Values

A number indicating the appropriate maximum number or retries.

Max NAK Receive Retry

Description

Specifies the maximum number of negative acknowledgments the eWay receives before taking recourse action (see **Action on Max Nak Received** on page 69).

This parameter is used for the inbound Collaboration code.

Required Values

A number indicating the appropriate maximum number of NAKs received before taking recourse action. The default value is **30**.

Max NAK Send Retry

Description

Specifies the maximum number of negative acknowledgments the eWay sends before taking recourse action (see **Action on Max Nak Sent** on page 69).

This parameter is used for the inbound Collaboration code.

Required Values

A number that indicates the appropriate maximum number of NAKs sent by the eWay before recourse action is taken. The default value is **30**.

Max No Response

Description

Specifies the maximum number of response timeouts the eWay allows, while waiting for data from the external system, before taking recourse action (see **Action on Max No Response** on page 69).

This parameter is used for the inbound Collaboration code.

Required Values

A number indicating the appropriate number of timeouts that may occur before taking recourse action. The configured default is **30**.

Additional Information

This parameter is only used by outbound eWays and works in conjunction with the Resend option of the Recourse Action parameter **Action on No Response** (see **Action on No Response** on page 70). It configures the eWay to resend the last message for the specified maximum number of times before the subsequent recourse action is taken.

Time To Wait For A Response

Description

Specifies the amount of time (in milliseconds) that the eWay waits for a response from the external system before taking recourse action (see **Action on No Response** on page 70). Any data from the external system is considered a response. This property corresponds to the initial read/receive operation timeout. Once a response is received, the following read/receive operation uses the **SoTimeout** specified timeout (see **SoTimeout** on page 61). Value **0** is interpreted as an infinite timeout.

Required Values

A length of time in milliseconds (1000 milliseconds equals 1 second) that the eWay waits for a response before The valid range is from **1** to **2147483647** (bytes). The configured default is **30000**.

4.3.7 TCPIP Inbound Settings

Presents the java Socket and ServerSocket options. For more information see the JDK Javadoc. This section contains the following top level parameters:

Connection Type

Description

Specifies how the eWay establishes the TCP/IP connection:

- **Client**: The eWay connects to an external server (host/port) to establish the connection. The eWay is in active mode.
- **Server**: The eWay waits/listens on a certain port for an incoming connection request from an external client. Once the request is received, the eWay accepts the request and establishes the connection. The eWay is in passive mode.

Required Values

Client or **Server**. **Server** is the default setting. Unless you specifically require Client mode, leave this value as the default, Server.

Keep Alive

Description

Specifies whether the client's SO_KEEPALIVE option is enabled or disabled. When the option is set for a TCP socket and no data has been exchanged across the socket in either direction for 2 hours, TCP automatically sends a KEEPALIVE probe to the peer (the actual value is implementation dependent). This probe is a TCP segment to which the peer must respond. One of three responses is expected:

1. The peer responds with the expected ACK. The application is not notified (since everything is OK). TCP will send another probe following another 2 hours of inactivity.
2. The peer responds with an RST, which tells the local TCP that the peer host has crashed and rebooted. The socket is closed.

3 There is no response from the peer. The socket is closed. The purpose of this option is to detect if the peer host has crashed. This is used for the accepted client Socket.

Required Values

True or **False**. True indicates that server **SO_KEEPALIVE** option is enabled.

Note: *For some properties, the server socket itself does not have direct properties settings associated with it. Instead, the properties map to the accepted client socket.*

Receive Buffer Size

Description

Sets or gets the value of the **SO_RCVBUF** option for the current socket, that is the buffer size used by the operating system for input on this socket. It provides an estimate of the size of the underlying buffers used by the platform for incoming network I/O. When used in set, this is a suggestion for the kernel from the application regarding the size of buffers to use for the data to be received over the socket. When used in get, this must return the actual size of the buffer used by the platform when receiving data on this socket.

Required Values

A number indicating the receive buffer size. The configured default is **8192**.

Send Buffer Size

Description

Sets or gets the value of the **SO_SNDBUF** option for the current socket, that is the buffer size used by the operating system for output on this socket. It provides an estimate of the size of the underlying buffers used by the platform for outgoing network I/O. When used in set, this is a suggestion for the kernel from the application regarding the size of buffers to use for the data to be sent over the socket. When used in get, this must return the actual size of the buffer used by the platform when sending out data on this socket.

Required Values

A number indicating the send buffer size. The configured default is **8192**.

Server Socket Factory Implementation Class Name

Description

Specifies the name of the Java class that implements the server socket factory. This class is used to create the server socket. If you have provided your own server socket implementation, enter the name of the Java class that contains this implementation here. The factory implementation class must implement the following interface:

```
com.stc.connector.tcpip.model.factory.TCPIPSocketFactory
```

Required Values

A valid Java class name; the default is:

```
com.stc.connector.tcpip.model.factory.TCPIPSocketFactoryImpl
```

ServerSO Timeout

Description

Sets or gets the value of the **SO_TIMEOUT** for **ServerSocket**, in milliseconds. Used for **ServerSocket.accept()**. When this option is set to a non-zero timeout, calling **accept()** for this ServerSocket will block for the configured length of time. If the timeout expires, a **java.net.SocketTimeoutException** (or **java.net.InterruptedIOException**) is thrown, but the ServerSocket remains valid. Enable this option prior to entering the blocking operation. This parameter is only used when the **Connection Type** is set as **Server**

Required Values

The a number indicating the **Server SO Timeout** in milliseconds. The configured default is **10000** (10 seconds). The timeout must be greater than zero (**0**). A timeout of zero is interpreted as an infinite timeout.

SoLinger

Description

Specifies whether the eWay performs a linger-on-close timeout. This option disables/enables immediate return from a **close()** of a TCP Socket. This parameter is used in conjunction with **SoLinger Timeout**.

- **True with SoLinger Timeout set to a non-zero integer timeout**: This means that a **close()** will block, pending the transmission and acknowledgement of all data written to the peer, at which point the socket is closed gracefully. Upon reaching the linger timeout, the socket is closed forcefully with a TCP RST.
- **True with SoLinger Timeout set to a timeout of zero**: Indicates that a forceful close is done immediately. See **SoLinger Timeout** on page 60.

Required Values

True or **False**. True enables the SO_Linger option.

SoLinger Timeout

Description

Specifies the server's **SoLinger** time-out in seconds. **SoLinger Timeout** is used in conjunction with **SoLinger** (see **SoLinger** on page 60) to configure the "linger-on-close" timeout.

When **SoLinger** is set to **true** (enabled), the **SoLinger Timeout** value indicates the following:

- **A non-zero integer** means that calling **close()** will block, pending the transmission and acknowledgement of all data written to the peer, at which point the socket is closed gracefully. Upon reaching the linger timeout, the socket is closed forcefully

with a TCP RST. If the specified timeout value exceeds 65,535 it will be reduced to 65,535.

- **A zero integer** indicates that a forceful close is done immediately.

Required Values

An integer between **-1** and **65535**. The default is **-1** seconds, which indicates that the **SoLinger** option is disabled (set as false). Zero (**0**) indicates that SoLinger immediately performs a forceful close. An integer of **1** to **65535** indicates the number of seconds for the time-out.

SoTimeout

Description

Sets or gets the value of the SoTimeout in milliseconds. Used for the accepted client socket. With this option set to a non-zero timeout, calling **read()** on the **InputStream** associated with this socket will block for only the configured length of time. If the timeout expires, **a java.io.InterruptedIOException** (or **java.net.SocketTimeoutException**) is thrown, but the Socket remains valid.

Enable this option prior to entering the blocking operation.

Required Values

The **SoTimeout** value in milliseconds. The configured default is **10000** (10 seconds). The timeout must be greater than zero (0). A timeout of zero is interpreted as an infinite timeout.

TcpNoDelay

Description

Specifies whether the server's **TcpNoDelay** option (that is, Nagle's algorithm) is enabled or disabled.

- **True**: Indicates that the server allows data packets that are less than the maximum transfer unit (MTU) size to be sent out immediately over the network. A setting of True may improve performance for higher-speed networks.
- **False**: Indicates that the server does not allow data packets that are less than the MTU size be sent out immediately over the network.

This is used for the accepted client socket.

Required Values

True or **False**. The configured default is **False**.

4.3.8 TCPIP Inbound Settings - Server Port Binding

Defines configuration parameters used for controlling the server port binding. This section contains the following top level parameters:

- **Max Binding Retry** on page 62
- **Retry Binding Interval** on page 62

Note: *This section is only used when the Connection Type is set as Server.*

Max Binding Retry

Description

Specifies the maximum number of times the eWay will attempt to bind to the specified TCP/IP port on the localhost.

Required Values

A number indicating the number of bind attempts. The configured default is 3.

Retry Binding Interval

Description

Specifies the length of time (in milliseconds) the eWay waits between attempts to bind to the specified TCP/IP port on the localhost.

Required Values

A number indicating the length of time in milliseconds that the eWay waits between attempts. The configured default is **30000** (30 seconds).

4.3.9 TCPIP Inbound Settings - Client Connection Establishment

Defines some configuration parameters used for controlling the connection establishment. This section is only used when the **Connection Type** is set as **Client**. This section contains the following top level parameters:

- **Time to Wait Before Attempting Connection** on page 63

Time to Wait Before Attempting Connection

Description

Specifies the length of time (in milliseconds) that the eWay waits before attempting to connect to the external system.

Required Values

A number indicating the amount of time (in milliseconds) to wait before attempting to connect to an external system. The configured default is **30000** (30 seconds).

4.3.10 TCPIP Inbound Settings - Connection Management

Defines configuration parameters used for inbound Server Connection Management. For example, the connection pool and the life cycle of the accepted connection. This section contains the following top level parameters:

Close Notification

Description

Specifies the close notification value. When the server receives a notification with content that matches this parameter's value, the server safely closes the connection and cancels any corresponding schedules.

Required Values

A String indicating the trigger value that notifies the server to close the connection. The configured default is **QUIT**.

Connection Pool Size

Description

Specifies the maximum number of concurrent connections allowed for the specific listener/monitor which is listening on, or monitoring a specified TCP/IP port. This represents the capability or availability of this server's services. Each connect-request from a client gains one concurrent connection. This parameter also represents the maximum number of clients who can concurrently connect to this server's services, and get served by the specific listener/monitor at the same time.

Required Values

A number indicating the maximum number of concurrent connections available from a listener/monitor for a specific TCP/IP port. **0** indicates that there is no limit. The configured default is **50**.

Idle Timeout

Description

Specifies the length of time (in milliseconds) for inactivity of the requestor (client). The eWay attempts to detect activity on client side (the other side of the connection). If no client activity (no i/o request comes over the connection from the client) for a specified time period, then the connection is closed from the server side to release the resource. The value is in milliseconds. If you want to disable this IdleTimeout checking, just specify 0 for this parameter.

Required Values

A String indicating the trigger value that notifies the server to close the connection. The configured default is **QUIT**.

Scope Of Connection

Description

Specifies the scope of the accepted connection which is used by the eWay. The two options are:

- **Resource Adapter Level**: The resource adapter will close the connection upon receiving a closure request, so the connection may "keep alive" during multiple executions of the Collaboration.
- **Collaboration Level**: The connection is closed once the Collaboration has been executed, so the connection has the same life cycle as the Collaboration.

Required Values

Resource Adapter Level or Collaboration Level. The configured default value is Collaboration Level.

4.3.11 Sequence Number Protocol

Provides sequence number protocol configuration settings. HL7 sequence numbering is used to help prevent duplication of data. This section contains the following top level parameters:

- **Sequence Number Enabled** on page 65

Sequence Number Enabled

Description

Specifies whether Sequence Number Protocol is enabled or disabled. HL7 sequence numbering is used to help prevent duplication of data. **True** indicates that sequence numbering is enabled.

Required Values

True or **False**. The configured default is **True**.

4.3.12 HL7 Acknowledgment

Specifies how the application acknowledgment Events are handled. This section contains the following top level parameters:

- **Acknowledgment Level** on page 66
- **eGate Sends App Acks** on page 66
- **Forward External Acks to eGate** on page 66
- **Timeout For Delayed Ack** on page 67

Acknowledgment Level

Description

Specifies whether the external application sends an HL7 application Acknowledgement after successfully receiving a message, or after the message has been successfully committed to the application database. The options are:

- **A:** (Application acknowledgment) The acknowledgement is sent when the message is successfully received.
- **C**: (Commit acknowledgment) The acknowledgement is sent after the message is successfully received and committed to the application database.

Required Values

A or **C**. The configured default is **A**.

eGate Sends App Acks

Description

Specifies whether the HL7 application acknowledgment sent to the external system is generated by the eWay or forwarded from eGate.

- **True** indicates that eGate receives or creates the HL7 application acknowledgment and sends it to the eWay, which in turn forwards it to the external system.
- **False** indicates that the eWay creates and sends the HL7 application acknowledgment directly to the external system.

This parameter is used for inbound Collaboration code.

Required Values

True or **False**. The configured default is **False**.

Forward External Acks to eGate

Description

Specifies whether the HL7 application acknowledgment is forwarded to eGate. When an HL7 application acknowledgment is received, it is sometimes necessary to forward the contents of the HL7 application acknowledgment to eGate (as data).

- **True** indicates that eWay forwards HL7 application acknowledgments from the external system to eGate for processing.
- **False** indicates that HL7 application acknowledgments from the external system are not forwarded to eGate by the eWay.

This parameter is used for inbound Collaboration code.

Required Values

True or **False**. The configured default is **False**.

Timeout For Delayed Ack

Description

Specifies the timeout value for delayed ACK in milliseconds.

This parameter is used for the inbound Collaboration code.

Required Values

A number indicating the timeout in milliseconds. The configured default is **30000** (30 seconds).

4.3.13 General Inbound Settings

Provides the general HL7 server configuration settings. This section contains the following top level parameters:

- **Dedicated Session Mode** on page 67
- **Max Data Size** on page 67
- **Scope Of State** on page 68

Dedicated Session Mode

Description

Specifies whether the server Dedicated Session Mode is enabled or disabled. When the server Dedicated Session Mode is enabled, the current client's request exclusively holds the server port to which it connects. The next client's request to the same port is blocked or rejected until the previous request concludes and releases the connection.

Required Values

True or **False**. **True** indicates that server Dedicated Session Mode is enabled. The configured default is **False**.

Max Data Size

Description

Specifies the maximum amount of data that the programs can hold internally. The valid range is from 1 to 2GB (which is the max value of a Java integer).

Required Values

A number indicating the maximum amount of data. The valid range is from **1** to **2147483647** (bytes). The configured default is **2147483647**.

Scope Of State

Description

Specifies the maximum amount of data that the programs can hold internally. The valid range is from 1 to 2GB (which is the max value of java integer).

It is used to define the scope of State object, which is an OTD node. The valid options for this parameter are:

- **Resource Adapter Level**: The State has the same life cycle as the resource adapter.
- **Connection Level**: The State has the same life cycle as the connection.
- **OTD Level**: The State has the same life cycle as the OTD object.

This scope represents the life cycle of the State.

Required Values

Resource Adapter Level, **Connection Level**, or **OTD Level**. The configured default is **Resource Adapter Level**.

4.3.14 HL7 Recourse Action

Determines the actions the eWay takes when operations occur outside the configured constraints. This section contains the following top level parameters:

- **Action on Max Failed Read Retry** on page 68
- **Action on Max Nak Received** on page 69
- **Action on Max Nak Sent** on page 69
- **Action on Max No Response** on page 69
- **Action on Nak Received** on page 70
- **Action on No Response** on page 70

Action on Max Failed Read Retry

Description

Specifies the action the eWay takes after it has reached the empty read limit set by the **Max Empty Read Retry** parameter. This parameter is used by inbound eWays only. The recourse options are:

- **Exit**: The eWay terminates its connection with the external system and shuts down.
- **Reset**: The eWay closes its connection with the external system and goes through the connection scenario.

This parameter is used for inbound Collaboration code.

Required Values

Exit or **Reset**. The configured default is **Reset**.

Action on Max Nak Received

Description

Specifies the action the eWay takes when the maximum number of HL7 Application NAKs have been received from the external system, as set by the **Max NAK Receive Retry** parameter (see **Max NAK Receive Retry** on page 56). The options are:

- **Exit**: The eWay terminates its connection with the external system and shuts down.
- **Reset**: The eWay closes its connection with the external system and goes through the connection scenario.
- **Skip Message**: The eWay remains connected, but writes the message to an error queue.

This parameter is used for inbound Collaboration code.

Note: *Do not set both "Action On NAK Received" and "Action On Max NAK Received" parameters to "Skip Message."*

Required Values

Exit, **Reset**, or **Skip Message**. The configured default is **Skip Message**.

Action on Max Nak Sent

Description

Specifies the action taken by the eWay when it has sent the maximum allowed number of NAKs to the external system, as set by the **Max NAK Send Retry** parameter (see **Max NAK Send Retry** on page 56). The options are:

- **Exit**: The eWay terminates its connection with the external system and shuts down.
- **Reset**: The eWay closes its connection with the external system and goes through the connection scenario.

This parameter is used for inbound Collaboration code.

Required Values

Exit or **Reset**. The default value is Exit.

Action on Max No Response

Description

Specifies the action the eWay takes when it attempts to send a message to the external system the maximum allowed number of times, and does not receive any response (HL7 Application Acknowledgement) from the external system. The maximum number times the eWay sends a message without receiving a response is determined by the **Max No Response** parameter (see **Max No Response** on page 56). The options are:

- **Exit**: The eWay terminates its connection with the external system and shuts down.
- **Reset**: The eWay closes its connection with the external system and goes through the connection scenario.

This parameter is used for inbound Collaboration code.

Required Values

Exit or **Reset**. The default value is Reset.

Action on Nak Received

Description

Specifies the action taken by the eWay when it receives an HL7 Application NAK from the external system. The options are:

- **Resend**: The eWay attempts to resend the message to the external system.
- **Reset**: The eWay closes its connection with the external system and goes through the connection scenario.
- **Skip Message**: The eWay remains connected, but writes the message to an error queue.

Note: *Do not set both the "Action On NAK Received" and "Action On Max NAK Received" parameters to "Skip Message."*

This parameter is used for inbound Collaboration code.

Required Values

Resend, **Reset**, or **Skip Message**. The configured default is **Resend**.

Action on No Response

Description

Specifies the action taken by the eWay when no ACK is received from the external system in the allotted time. The amount of time is determined by the **Time To Wait For A Response** parameter (see **Time To Wait For A Response** on page 57). The options are:

- **Exit**: The eWay terminates its connection with the external system and shuts down.
- **Resend**: The eWay attempts to resend the message to the external system. The Resend option is only allowed when sequence numbering is in effect.
- **Reset**: The eWay closes its connection with the external system and goes through the connection scenario.

This parameter is used for inbound Collaboration code.

Required Values

Exit, **Resend**, or **Reset**. The configured default is **Reset**.

4.4 TCP/IP HL7 Inbound eWay Environment Properties

The TCP/IP HL7 Inbound eWay configuration parameters, accessed from the Environment Explorer tree, are organized into the following sections:

- **Sequence Number Protocol** on page 71
- **TCPIP Inbound Settings** on page 71

4.4.1 Sequence Number Protocol

Provides Sequence Number Protocol configuration settings. HL7 sequence numbering is used to help prevent duplication of data. This section contains the following top level parameters:

- **Sequence Number File Location** on page 71

Sequence Number File Location

Description

Specifies the Sequence Number File Location (a local directory). This is required when the **Sequence Number Protocol** is Enabled. The Sequence Number File Location is a non-volatile directory that stores the sequence number files that are used to persist the HL7 Sequence Number. This unique base file name is automatically generated according to project/Collaboration information.

For the Inbound eWay the file names are created as follows:

```
<project name> + <deployment name> + <external application node name>
+  <Collaboration name> + .seqno
```

For example: prjHL7Inbound_dpIn_eaHL7Inbound_jcdHL7inbound1.seqno

Required Values

The path and directory where the sequence number file is located. The default setting is:

```
C:/temp/hl7inbound/seq
```

4.4.2. TCPIP Inbound Settings

Presents the java Socket and ServerSocket options. For more information, refer to the JDK Javadoc. This section contains the following top level parameters:

- **Host** on page 72
- **ServerPort** on page 72

Host

Description

Specifies the host name or IP address to use for establishing a TCPIP connection. This parameter is only used when the **Connection Type** is set as **Client**.

Required Values

- The host name, or IP address of the eGate host.

ServerPort

Description

Specifies the server port number of the local host. The port number that eGate listens to for connections from eWays. A value of **0** creates a socket on any free port.

Required Values

A number indicating the port number of the eGate server. The configured default is **8888**.

4.5 TCP/IP HL7 Outbound eWay Connectivity Map Properties

The TCP/IP HL7 Server (outbound) eWay configuration parameters, accessed from the Connectivity Map, are organized into the following sections:

4.5.1 General Outbound Settings

Provides the general HL7 outbound configuration settings. This section contains the following top level parameter:

Max Data Size

Description

Specifies the maximum size of data that the programs can hold internally. The valid range is from 1 to 2GB (which is the max value of java integer).

Required Values

A number indication the maximum size, between **1** to **2147483647** (bytes). The configured default is **2147483647**.

Scope Of State

Description

Specifies It is used to define the scope of State object, which is an OTD node. The valid options for this parameter are:

- **Resource Adapter Level**: The State has the same life cycle as the resource adapter.
- **Connection Level**: The State has the same life cycle as the connection.
- **OTD Level**: The State has the same life cycle as the OTD object.

This scope represents the life cycle of the State.

Required Values

Resource Adapter Level, **Connection Level**, or **OTD Level**. The configured default is **Resource Adapter Level**.

4.5.2. Sequence Number Protocol

Provides sequence number protocol configuration settings. This section contains the following top level parameters:

- **Sequence Number Enabled** on page 74

Note: *Many of the parameters for the eWay are specific to the direction the data is travelling, that is whether the eWay is Inbound or Outbound to eGate.*

Sequence Number Enabled

Description

Specifies whether Sequence Number Protocol is enabled or disabled. HL7 sequence numbering is used to help prevent duplication of data. **True** indicates that sequence numbering is enabled.

Required Values

True or **False**. The configured default is **True**.

4.5.3 TCPIP Outbound Settings

Presents the java Socket options. For more information see the JDK Javadoc. This section contains the following top level parameters:

Connection Type

Description

Specifies how the eWay establishes the TCP/IP connection:

- **Client**: The eWay connects to an external server (host/port) to establish the connection. The eWay is in active mode.
- **Server**: The eWay waits/listens on a particular port for an incoming connection request from an external client. Once the request is received, the eWay accepts the request and establishes the connection. The eWay is in passive mode.

Required Values

Client or **Server**. **Server** is the default setting. Unless you specifically require Server mode, leave this value as the default, Client.

Keep Alive

Description

Specifies whether the client's SO_KEEPALIVE option is enabled or disabled. When the option is set for a TCP socket and no data has been exchanged across the socket in either direction for 2 hours, TCP automatically sends a KEEPALIVE probe to the peer (the actual value is implementation dependent). This probe is a TCP segment to which the peer must respond. One of three responses is expected:

A. The peer responds with the expected ACK. The application is not notified (since everything is OK). TCP will send another probe following another 2 hours of inactivity.

B. The peer responds with an RST, which tells the local TCP that the peer host has

crashed and rebooted. The socket is closed.

C There is no response from the peer. The socket is closed. The purpose of this option is to detect if the peer host has crashed. This is used for the accepted client Socket.

Required Values

True or **False**. True indicates that **SO_KEEPALIVE** option is enabled.

Receive Buffer Size

Description

Sets or gets the value of the **SO_RCVBUF** option for the current socket, that is the buffer size used by the operating system for input on this socket. It provides an estimate of the size of the underlying buffers used by the platform for incoming network I/O. When used in set, this is a suggestion for the kernel from the application regarding the size of buffers to use for the data to be received over the socket. When used in get, this must return the actual size of the buffer used by the platform when receiving data on this socket.

Required Values

A number indicating the receive buffer size. The configured default is **8192**.

Send Buffer Size

Description

Sets or gets the value of the **SO_SNDBUF** option for the current socket, that is the buffer size used by the operating system for output on this socket. It provides an estimate of the size of the underlying buffers used by the platform for outgoing network I/O. When used in set, this is a suggestion for the kernel from the application regarding the size of buffers to use for the data to be sent over the socket. When used in get, this must return the actual size of the buffer used by the platform when sending out data on this socket.

Required Values

A number indicating the send buffer size. The configured default is **8192**.

ServerSoTimeout

Description

Sets or gets the value of the SoTimeout for the ServerSocket, in milliseconds. Used for **ServerSocket.accept()**. With this option set to a non-zero timeout, calling **accept()** for **ServerSocket** will block for only this period of time. If the timeout expires, a **java.net.SocketTimeoutException** (or **java.net.InterruptedIOException**) is thrown, though the ServerSocket remains valid. Enable this option prior to entering the blocking operation. This parameter is only used when the **Connection Type** is set as **Server**.

Required Values

The **SoTimeout** value in milliseconds. The configured default is **60000** (60 seconds). The timeout must be greater than zero (0). A timeout of zero is interpreted as an infinite timeout.

Socket Factory Implementation Class Name

Description

Specifies the name of the Java class that implements the socket factory. This class is used to create the socket. If you have provided your own socket implementation, enter the name of the Java class that contains this implementation here. The factory implementation class must implement the following interface:

```
com.stc.connector.tcpip.model.factory.TCPIPSocketFactory
```

Required Values

A valid Java class name; the default is:

```
com.stc.connector.tcpip.model.factory.TCPIPSocketFactoryImpl
```

SoLinger

Description

Specifies whether the eWay performs a linger-on-close timeout. This option disables/enables immediate return from a **close()** of a TCP Socket. This parameter is used in conjunction with **SoLinger Timeout**.

- **True with SoLinger Timeout set to a non-zero integer timeout**: This means that a **close()** will block, pending the transmission and acknowledgement of all data written to the peer, at which point the socket is closed gracefully. Upon reaching the linger timeout, the socket is closed forcefully with a TCP RST.
- **True with SoLinger Timeout set to a timeout of zero**: Indicates that a forceful close is done immediately. See **SoLinger Timeout** on page 77.

Required Values

True or **False**. True enables the SO_Linger option.

SoLinger Timeout

Description

Specifies the server's **SoLinger** time-out in seconds. **SoLinger Timeout** is used in conjunction with **SoLinger** (see **SoLinger** on page 77) to configure the "linger-on-close" timeout.

When **SoLinger** is set to **true** (enabled), the **SoLinger Timeout** value indicates the following:

- **A non-zero integer** means that calling **close()** will block, pending the transmission and acknowledgement of all data written to the peer, at which point the socket is closed gracefully. Upon reaching the linger timeout, the socket is closed forcefully with a TCP RST. If the specified timeout value exceeds 65,535 it will be reduced to 65,535.
- **A zero integer** indicates that a forceful close is done immediately.

Required Values

An integer between **-1** and **65535**. The default is **-1** seconds, which indicates that the **SoLinger** option is disabled (set as false). Zero (**0**) indicates that SoLinger immediately performs a forceful close. An integer of **1** to **65535** indicates the number of seconds for the time-out.

SoTimeout

Description

Sets or gets the value of the SoTimeout in milliseconds. With this option set to a non-zero timeout, calling **read()** on the **InputStream** associated with this socket will block for only this configured length of time. If the timeout expires, **a java.io.InterruptedIOException** (or **java.net.SocketTimeoutException**) is thrown, but the Socket remains valid. Enable this option prior to entering the blocking operation.

Required Values

The **SoTimeout** value in milliseconds. The configured default is **10000** (10 seconds). The timeout must be greater than zero (0). A timeout of zero is interpreted as an infinite timeout.

TcpNoDelay

Description

Specifies whether the server's **TcpNoDelay** option (that is, Nagle's algorithm) is enabled or disabled.

- **True**: Indicates that the server allows data packets that are less than the maximum transfer unit (MTU) size to be sent out immediately over the network. A setting of True may improve performance for higher-speed networks.
- **False**: Indicates that the server does not allow data packets that are less than the MTU size be sent out immediately over the network.

This is used for the accepted client socket.

Required Values

True or **False**. The configured default is **False**.

4.5.4 TCPIP Outbound Settings - Server Port Binding

Specifies configuration parameters used for controlling server port binding. This parameter is only used when the **Connection Type** is set as **Server**. This section contains the following top level parameters:

- **Max Binding Retry** on page 79
- **Retry Binding Interval** on page 79

Max Binding Retry

Description

Specifies the maximum number of times the eWay attempts to bind to the specified TCP/IP port on the localhost before giving up.

Required Values

A number indicating the number of times the eWay attempts to bind to the specified TCP/IP port on the localhost.

Retry Binding Interval

Description

Specifies the number of milliseconds the eWay waits between attempts to bind to the specified TCP/IP port on the localhost.

Required Values

A number indicating the length of times in milliseconds, between attempts to bind to the specified TCP/IP port. The configured default is **30000** (30 seconds).

4.5.5 TCPIP Outbound Settings - Connection Establishment

Specifies configuration parameters that are used to control the connection establishment. This section contains the following top level parameters:

- **Always Create New Connection** on page 80
- **Auto Reconnect Upon Matching Failure** on page 80
- **Max Connection Retry** on page 80
- **Retry Connection Interval** on page 80
- **Time To Wait Before Attempting Connection** on page 81

Note: *This section is only used when the Connection Type is Client.*

Always Create New Connection

Description

Specifies whether the eWay always attempts to create a new connection when a connection establishment request is received.

- **True** indicates that the eWay always attempts to create a new connection without attempting to match an existing connection.
- **False** indicates that the eWay attempts to match an existing connection (managed by the container).

Required Values

True or **False**. The configured default is **False**.

Auto Reconnect Upon Matching Failure

Description

Specifies whether to attempt to re-connect automatically when the eWay gets a matching connection from a container, even though this connection is not valid due to various reasons: for example, the external side of the connection is closed/reset due to the external application's logic.

- **True** indicates that the eWay discards the invalid matching connection and automatically attempts to reconnect using a new connection.
- **False** indicates that the eWay does not automatically attempt to reconnect using a new connection: instead, the eWay defers the reconnect control to the user Business Rules. The user must detect this type of failure and act appropriately.

Required Values

True or **False**. The configured default is **True**.

Max Connection Retry

Description

Specifies the maximum number of times the eWay attempts to connect to a specific external TCP/IP destination (host/port) before giving up.

Required Values

A number indicating the number of times the eWay will attempt to connect.

Retry Connection Interval

Description

Specifies the length of time (in milliseconds) the eWay waits between attempts to connect to a specific external TCP/IP destination (host/port).

Required Values

A number indicating the length of time (in milliseconds) the eWay waits between attempts to connect. The configured default is **30000** (or 30 seconds).

Time To Wait Before Attempting Connection

Description

Specifies the length of time (in milliseconds) the eWay waits before attempting to connect to the external system.

Required Values

A number indicating the length of time (in milliseconds) the eWay waits before attempts to connect. The configured default is **0**.

4.5.6 Lower Layer Protocol

Provides Lower Layer Protocol (LLP) configuration settings. This section contains the following top level parameters:

End Block Character

Description

Specifies the End Block Character (the last envelope marker character in the HL7 envelope) as a decimal ascii number.

Required Values

A decimal within the range of 1 to 127. To be strictly compliant with the HL7 Standard, this parameter MUST be set to a Carriage Return (decimal 13). The default value is 13.

End Data Character

Description

Specifies the End Data Character (the second to the last envelope marker character in the HL7 envelope) as a decimal ascii number. The allowed range is 1 to 127.

Required Values

A decimal within the range of 1 to 127. Unless there is a conflict, the value should be ASCII FS (decimal **28**). The default value is 28.

HLLP Checksum Enabled

Description

Specifies whether the HLLP (Hybrid Lower Level Protocol) checksum is enabled or disabled.

Required Values

True or **False**. **True** indicates that the HLLP Checksum is enabled.

LLP Type

Description

Specifies the LLP (Lower Layer Protocol) type. The valid types are:

- **MLLP** (Minimal Lower Layer Protocol)
- **HLLP** (Hybrid Lower Layer Protocol)

For more information on the available envelope types see

Required Values

MLLP or **HLLP**. MLLP is the configured default value.

Start Block Character

Description

Specifies the Start Block Character (the first envelope marker character in the HL7 envelope) as a decimal ascii number.

Required Values

A decimal within the range of 1 to 127. Unless there is a conflict, the value should be ASCII VT (decimal 11). The default value is 11.

4.5.7 HL7 SFT Segment

Provides HL7 SFT Segment configuration settings. The SFT segment is available starting with HL7 version 2.5. This segment provides additional information about one or more software product used as sending applications. The primary purpose of this segment is for diagnostic use. There may be additional uses per site-specific agreements.

This section contains the following top level parameters:

Enable

Description

Specifies whether the optional SFT segment is enabled in the ACK message.

Required Values

True or **False**. True indicates that the SFT segment is enabled.

Note: *If enable is set to true, and the HL7 version is not configured as 2.5, the eWay will error upon startup.*

Software Binary ID

Description

Specifies HL7 segment **SFT-04**, the Software Binary ID. This property is available starting with HL7 version 2.5. Software Binary IDs are issued by a vendor for each unique software version instance. These IDs are used to differentiate between differing versions of the same software. Software IDs are issued for each unique software version instance. Identical Primary IDs indicate that the software is identical at the binary level, but configuration settings may differ.

Required Values

The unique Software Binary ID. The configured default is **5.0.0**.

Note: *This property only applies to HL7 version 2.5.*

Software Certified Version or Release Number

Description

Specifies HL7 segment **SFT-02**, the **Software Certified Version or Release Number**: the latest software version number or release number for the sending system. This helps to provide a more complete profile of the application that is sending or receiving HL7 messages. Version numbers are important in identifying the specific _release_ of an application. In some situations, the receiving application validates the software certified version or release number against a list of "certified" versions or releases of the particular software. This helps determine whether the sending application adheres to specific Business Rules required by the receiving application. Alternatively, the software may perform different processing, depending on the version of the sending software.

Required Values

The software certified version or release number. The configured default is **ICAN 5**.

Note: *This property only applies to HL7 version 2.5.*

Software Install Date

Description

Specifies HL7 segment **SFT-06**, the Software Install Date. This is the date on which the submitting software was installed at the sending site. the software install date on its own can often provide key information about the behavior of the application. This is necessary for providing a more complete profile of the sending application.

Required Values

The date of installation for the sending application software.

Note: *This property only applies to HL7 version 2.5.*

Software Product Information

Description

Specifies HL7 segment **SFT-05**, software product identification information. This may include a description of the software application, configuration settings, modifications made to the software, and so forth. This field can contain any additional information about the sending application, with the transaction it has submitted. The information is used for diagnostic purposes and may provide greater flexibility for identifying the application software.

Required Values

Information that may help to identify the specific sending software. This field should only be used when performing diagnostics.

Note: *This property only applies to HL7 version 2.5.*

Software Product Name

Description

Specifies HL7 segment **SFT-03**, the name of the software product that submitted the transaction. The software product name is a key component for identifying the sending application.

Required Values

The sending software product name. The default value is **HL7 eWay**.

Note: *This property only applies to HL7 version 2.5.*

Software Vendor Organization

Description

Specifies HL7 segment **SFT-01**, the name of the company that publishes and/or distributes the sending software that created the transaction. The purpose of this field, along with the remaining fields in this segment, is to provide a more complete profile of the sending applications. The Software Vendor Organization field identifies the vendor who is responsible for maintaining the application.

Required Values

The name of the sending software publisher or vendor. The default value is **SeeBeyond Technology Corporation**.

Note: *This property only applies to HL7 version 2.5.*

4.5.8 HL7 MSH Segment

Provides HL7 MSH Header segment configuration settings. This section contains the following top level parameters:

Alternate Character Set Handling Scheme

Description

Specifies the value for the Alternate character set handling scheme to be used when any alternative character sets are used and a special handling scheme is necessary (see HL7 Table 0356). This is the 20th field in the HL7 MSH segment (MSH-20).

Required Values

Available values include **ISO 2022-1994**, **2.3**, or **<null>** (blank). Leaving the field blank indicates that no character set switching will occur.

Character Set

Description

Specifies the character set(s) in use by the messages (see HL7 Table 0211). If the field is left blank, the character set in use is understood to be the 7-bit ASCII set. This is the 18th field in the HL7 MSH segment (MSH-18).

Required Values

The configured default is **8859/1** (printable 7-bit ASCII character set). See HL7 Table 0211 for available values and descriptions.

Conformance Statement ID

Description

The Conformance Statement ID (Message Profile Identifier in V2.5) is a unique identifier that applies to a query's Conformance Statement, or as a Message Profile Identifier, asserts constancy with a message profile (grammar, syntax, usage, and so forth). This is the 21st field in the HL7 MSH segment (MSH-21).

Required Values

An HL7 Conformance Statement ID value or leave blank.

Country Code

Description

Specifies a code that indicates the country of origin for the message (see HL7 Table 0399). Used to specify default elements in a message, such as currency. This is the 17th field in the HL7 MSH segment (MSH-17).

Required Values

The Country Code value uses the 3-character (alphabetic) form of ISO 3166. The default value is **USA**.

Encoding Characters

Description

Specifies four encoding characters in the following order:

- Component separator
- Repetition separator
- Escape character
- Subcomponent separator

This is the second field in the HL7 MSH segment (MSH-02).

Required Values

HL7 encoding characters in the respective order. The configured default is: **^~\&** (ASCII 94, 126, 92, and 38, respectively).

Field Separator

Description

Specifies the separator between the segment ID and the first real field. This value defines the character that is used as a aspirator for the rest of the message. This is the first field in the HL7 MSH segment (MSH-01).

Required Values

Field Separator character value as a decimal ascii number. The allowed range is 1 to 127. The default setting is 124 which is character '|'.

Principal Language of Message

Description

Specifies the principal language of the message. Codes come from ISO 639. This is the 19th field in the HL7 MSH segment (MSH-19).

Required Values

The 2-character ISO 639 alphabetic code.

Processing ID

Description

Specifies the sub-component Processing ID of MSH-11. MSH-11 is used to indicate whether a message is processed as defined in the HL7 Application (level 7) Processing rules.

Required Values

D for Debugging, **P** for Production, **T** for Training. In some cases there may be an additional sub-component "Processing Mode" following the initial value.

Receiving Application

Description

Specifies the receiving application among other applications within the network enterprise. The network enterprise consists of the applications that participate in the exchange of HL7 messages within the enterprise. This is the fifth field in the HL7 MSH segment (MSH-05).

Required Values

User defined value for the HL7 receiving application. The default value is **SeeBeyond HL7 eWay**.

Receiving Facility

Description

Specifies (further identifies) the receiving application among multiple identical instances of the application running on behalf of different organizations. This is the sixth field in the HL7 MSH segment (MSH-06).

Required Values

User defined value for the HL7 receiving facility. The default value is **SeeBeyond HL7 eWay**.

Security

Description

Specifies the implemented application level security features. This is the eighth field in the HL7 MSH segment (MSH-08).

Required Values

Under development by HL7.

Sending Application

Description

Specifies the sending application among other applications within the network enterprise. The network enterprise consists of the applications that participate in the exchange of HL7 messages within the enterprise. This is the third field in the HL7 MSH segment (MSH-03).

Required Values

User defined value for the HL7 sending application. The default value is **SeeBeyond HL7 eWay**.

Sending Facility

Description

Specifies (further identifies) the sending application among multiple identical instances of the application running on behalf of different organizations. This is the forth field in the HL7 MSH segment (MSH-04).

Required Values

User defined value for the HL7 sending facility. The default value is **SeeBeyond HL7 eWay**.

Validate MSH

Description

Specifies whether to validate the MSH segment of the data message (for inbound) and the MSH segment of the ACK (for outbound).

This parameter is used for outbound Collaboration code.

Required Values

True or **False**. True indicates that the Collaboration validates the MSH segment.

Note: *This property does not affect structural validation of the whole HL7 message itself. Structural validation is always performed.*

Version ID

Description

Specifies the particular HL7 version to be matched by the receiving system to its own version. This is the 12th field in the HL7 MSH segment (MSH-12).

Required Values

The HL7 Standard version value as displayed in HL7 Table 0104 - Version ID.

4.5.9 HL7 Acknowledgment

Provides HL7 acknowledgment configuration settings that control how the application acknowledgment Events are handled. This section contains the following top level parameters:

- **Acknowledgment Level** on page 90
- **eGate Sends App Acks** on page 91
- **Forward External Acks to eGate** on page 91
- **Timeout For Delayed Ack** on page 91

Acknowledgment Level

Description

Specifies whether the external application is configured to send an HL7 application acknowledgement after successfully receiving the message or after the message has been successfully committed to the application database. The valid levels are:

- **A:** (Application acknowledgment) The acknowledgement is sent when the message is successfully received.
- **C**: (Commit acknowledgment) The acknowledgement is sent after the message is successfully received and committed to the application database.

Required Values

A or **C**. The configured default is **A**.

eGate Sends App Acks

Description

Used by both the inbound and outbound Collaboration.

- **Inbound**: Specifies whether the HL7 application acknowledgment sent to the external system is generated by the eWay or forwarded from eGate.
  - **True** indicates that the eWay receives the external receiving HL7 application acknowledgment from eGate and sends it to the external system.
  - **False** indicates that the eWay creates and sends the HL7 application acknowledgment directly to the external system.
- **Outbound**: Specifies whether the outbound Collaboration is in outbound Delayed ACK role; that is, the outbound eWay is connecting to an external system that communicates as a Delayed ACK receiver and is sending two ACKs to the eWay.
  - **True** indicates that the eWay is expecting a Delayed ACK (2 ACKS).
  - **False** indicates that the eWay does not expect a Delayed ACK.

Required Values

True or **False**. The configured default is **False**.

Forward External Acks to eGate

Description

Specifies whether the HL7 application acknowledgment is forwarded to eGate. When an HL7 application acknowledgment is received, it is sometimes necessary to forward the contents of the HL7 application acknowledgment to eGate (as data).

- **True** indicates that eWay forwards HL7 application acknowledgments from the external system to eGate for processing.
- **False** indicates that HL7 application acknowledgments from the external system are not forwarded to eGate by the eWay.

This parameter is used for the outbound Collaboration code.

Required Values

True or **False**. The configured default is **False**.

Timeout For Delayed Ack

Description

Specifies the timeout value for delayed ACK in milliseconds.

This parameter is used for outbound Collaboration code.

Required Values

A number indicating the timeout in milliseconds. The configured default is **30000** (30 seconds).

4.5.10 Communication Control

Controls data transferring (sending/receiving) over TCP/IP connection. This section contains the following top level parameters:

Enable Journaling

Description

Specifies whether message journaling is enabled.

This parameter is used for outbound Collaboration code.

Required Values

- **True** or **False**. True indicates that journaling is enabled.

Max Canned NAK Send Retry

Description

Specifies the maximum number of canned negative acknowledgments that the eWay sends before taking recourse action (see **Action on Max Nak Sent** on page 95).

Required Values

The appropriate maximum number of canned NAK to send before taking recourse action. **0** indicates that the eWay will not attempt to create or send a canned NAK.

Max Empty Read Retry

Description

Specifies the maximum number of times the eWay attempts to read data from the external system after the read/receive operation returns nothing. This applies to the read or receive operation after a response starts to arrive. **Empty Read** means that a timeout occurs on the read/receive operation, which takes the **SoTimeout** parameter in the **TCPIP Server Base Settings** section as the applied timeout setting (see **SoTimeout** on page 78). The corresponding recourse action is specified by the **Action on Max Failed Read Retry** on page 94.

Required Values

A number indicating the appropriate maximum number or retries.

Max NAK Receive Retry

Description

Specifies the maximum number of negative acknowledgments the eWay receives before taking recourse action (see **Action on Max Nak Received** on page 95).

This parameter is used for outbound Collaboration code.

Required Values

A number indicating the appropriate maximum number of NAKs received before taking recourse action. The default value is 30.

Max NAK Send Retry

Description

Specifies the maximum number of negative acknowledgments the eWay sends before taking recourse action (see **Action on Max Nak Sent** on page 95). Recourse is specified by the

This parameter is used for outbound Collaboration code.

Required Values

A number that indicates the appropriate maximum number of NAKs sent by the eWay before recourse action is taken. The default value is 30.

Max No Response

Description

Specifies the maximum number of response timeouts the eWay allows, while waiting for data from the external system, before taking recourse action (see **Action on Max No Response** on page 95).

This parameter is used for outbound Collaboration code.

Required Values

A number indicating the appropriate number of timeouts that may occur before taking recourse action. The configured default is **30**.

Additional Information

This parameter is only used by outbound eWays and works in conjunction with the Resend option of the Recourse Action parameter **Action on Max No Response** on page 95. It configures the eWay to resend the last message for the specified maximum number of times before the subsequent recourse action is taken.

Time To Wait For A Response

Description

Specifies the amount of time (in milliseconds) that the eWay waits for a response from the external system before taking recourse action (see **Action on No Response** on page 96). Any data from the external system is considered a response. This property corresponds to the initial read/receive operation timeout. Once a response is received, the following read/receive operation uses the **SoTimeout** specified timeout (see **SoTimeout** on page 78). Value 0 is interpreted as an infinite timeout.

Required Values

A length of time in milliseconds (1000 milliseconds equals 1 second) that the eWay waits for a response before The valid range is from **1** to **2147483647** (bytes). The configured default is **30000**.

4.5.11 HL7 Recourse Action

Determines the actions the eWay takes when operations occur outside the configured constraints. This section contains the following top level parameters:

- **Action on Max Failed Read Retry** on page 94
- **Action on Max Nak Received** on page 95
- **Action on Max Nak Sent** on page 95
- **Action on Max No Response** on page 95
- **Action on Nak Received** on page 96
- **Action on No Response** on page 96

Action on Max Failed Read Retry

Description

Specifies the action the eWay takes after it has reached the empty read limit set by the **Max Empty Read Retry** parameter. The recourse options are:

- **Exit**: The eWay terminates its connection with the external system and shuts down.
- **Reset**: The eWay closes its connection with the external system and goes through the connection scenario.

This parameter is used for outbound Collaboration code.

Required Values

Exit or **Reset**. The configured default is **Reset**.

Action on Max Nak Received

Description

Specifies the action the eWay takes when the maximum number of HL7 Application NAKs have been received from the external system, as set by the **Max NAK Receive Retry** parameter (see **Max NAK Receive Retry** on page 93). The options are:

- **Exit**: The eWay terminates its connection with the external system and shuts down.
- **Reset**: The eWay closes its connection with the external system and goes through the connection scenario.
- **Skip Message**: The eWay remains connected, but writes the message to an error queue.

This parameter is used for outbound Collaboration code.

Note: *Do not set both "Action On NAK Received" and "Action On Max NAK Received" parameters to "Skip Message."*

Required Values

Exit, **Reset**, or **Skip Message**. The configured default is **Skip Message**.

Action on Max Nak Sent

Description

Specifies the action taken by the eWay when it has sent the maximum allowed number of NAKs to the external system, as set by the **Max NAK Send Retry** parameter (see **Max NAK Receive Retry** on page 93). The options are:

- **Exit**: The eWay terminates its connection with the external system and shuts down.
- **Reset**: The eWay closes its connection with the external system and goes through the connection scenario.

This parameter is used for outbound Collaboration code.

Required Values

Exit or **Reset**. The default value is Exit.

Action on Max No Response

Description

Specifies the action the eWay takes when it attempts to send a message to the external system the maximum allowed number of times, and does not receive any response (HL7 Application Acknowledgement) from the external system. The maximum number times the eWay sends a message without receiving a response is determined by the **Max No Response** parameter (see **Max No Response** on page 93). The options are:

- **Exit**: The eWay terminates its connection with the external system and shuts down.
- **Reset**: The eWay closes its connection with the external system and goes through the connection scenario.

This parameter is used for outbound Collaboration code.

Required Values

Exit or **Reset**. The default value is Reset.

Action on Nak Received

Description

Specifies the action taken by the eWay when it receives an HL7 Application NAK from the external system. The options are:

- **Resend**: The eWay attempts to resend the message to the external system.
- **Reset**: The eWay closes its connection with the external system and goes through the connection scenario.
- **Skip Message**: The eWay remains connected, but writes the message to an error queue.

Note: *Do not set both the "Action On NAK Received" and "Action On Max NAK Received" parameters to "Skip Message."*

This parameter is used for outbound Collaboration code.

Required Values

Resend, **Reset**, or **Skip Message**. The configured default is **Resend**.

Action on No Response

Description

Specifies the action taken by the eWay when no ACK is received from the external system in the allotted time. The amount of time is determined by the **Time To Wait For A Response** parameter (see **Time To Wait For A Response** on page 94). The options are:

- **Exit**: The eWay terminates its connection with the external system and shuts down.
- **Resend**: The eWay attempts to resend the message to the external system. The Resend option is only allowed when sequence numbering is in effect.
- **Reset**: The eWay closes its connection with the external system and goes through the connection scenario.

This parameter is used for outbound Collaboration code.

Required Values

Exit, **Resend**, or **Reset**. The configured default is **Reset**.

4.6 TCP/IP HL7 Outbound eWay Environment Explorer Properties

The TCP/IP HL7 client (inbound) eWay configuration parameters, accessed from the Environment Explorer tree, are organized into the following section:

4.6.1 Sequence Number Protocol

Provides Sequence Number Protocol configuration settings. HL7 sequence numbering is used to help prevent duplication of data. This section contains the following top level parameters:

Sequence Number File Location

Description

Specifies the Sequence Number File Location (a local directory). This is required when the **Sequence Number Protocol** is Enabled. The Sequence Number File Location is a non-volatile directory that stores the sequence number files that are used to persist the HL7 sequence number. This unique base file name is automatically generated according to project/Collaboration information.

For the Outbound eWay the file names are created as follows:

<project name> + <deployment name> + <collab name> + <external application node name> + .seqno

For example: prjHL7Outbound_dpOut_jcolHL7Outbound_eaHL7Outbound.seqno

Required Values

The path and directory where the sequence number file is located. The default setting is:

```
C:/temp/hl7outbound/seq
```

4.6.2. TCPIP Outbound Settings

Presents the java Socket options. For more information, please refer JDK javadoc. This section contains the following top level parameters:

Host

Description

Specifies the host name or IP address used to establish a TCPIP connection.

Required Values

The host name or IP address. The default value is **localhost**.

Port

Description

Specifies the port number of the host.

Required Values

The number indicating the port number of the host. The default value is **7777**.

Chapter 5

Working with TCP/IP HL7 Collaborations

This chapter provides an overview and description of the structure and functionality of the Inbound and Outbound Collaborations provided as part of the TCP/IP HL7 eWay. It also provides information on how to incorporate them into a project

Chapter Topics

5.1 The TCP/IP HL7 eWay Collaborations

The TCP/IP HL7 eWay includes one inbound and one outbound Collaboration, presented within the template projects for inbound and outbound HL7 messaging. These Collaborations are designed to work "as is" for HL7 compliant interfaces, and can be configured for your specific needs using only the property configuration files. If an interface requires special functionality, the Collaboration's Java code is easily accessible for modification, much of which can be created graphically (*drag and drop*), using the Collaboration Editor's Business Rules Designer.

The Collaborations contain a number of OTDs that extend functionality for HL7 message handling, logging, error messaging, journaling, and sequence numbering. These include both generic HL7 OTDs for HL7 ACK/NAK generation or verification, and the Resource Adapter that communicates to the external system and offers services to the integration server. The Collaboration controls messaging protocol and all business logic.

The Collaborations are designed to target "one unit of work" at a time, meaning the resolution of one message at a time. The basic structure of both Collaborations is a state machine implemented in a Java switch statement. The state machine keeps track of the messaging protocol so that when a Collaboration is invoked, it can retrieve the state of the connection just handed to it by the RA, and then execute the proper actions based on the state machine.

At the end of each action, the state is set for the next execution of the Collaboration. There are three main states:

- **To Establish**: This means a new or reset connection needs to have an HL7 session established. If sequence numbering is used, the sequence numbers need to be negotiated.
- **Messaging**: This is where the exchange of messages and ACKs is takes place.
- **Shutdown**: This is where any cleanup can happen before the connection is closed, or to close the connection.

Additional Collaborations can be added to a project to increase message flow.

Note: *The TCP/IP HL7 Inbound Collaboration publishes received data as a Byte message in JMS using the sendBytes() method. However, the HL7 Outbound Collaboration expects a Text message from JMS. The eWay is not designed for the HL7 Outbound Collaboration to subscribe to a JMS data queue created by the HL7 Inbound Collaboration directly. HL7 Inbound and Outbound Collaborations are designed to communicate through an HL7 eWay TCP/IP connection.*

5.1.1 Inbound Collaboration Overview

The Inbound HL7 Collaboration, **jcdHL7inbound**, contains OTDs for the HL7 Resource Adapter, JMS Data, HL7 ACK, JMS Journal, and JMS Error, as well as the Generic HL7 Event. The Collaboration works with its own internal code and the Properties Configuration files.

Inbound Collaboration - Part 1

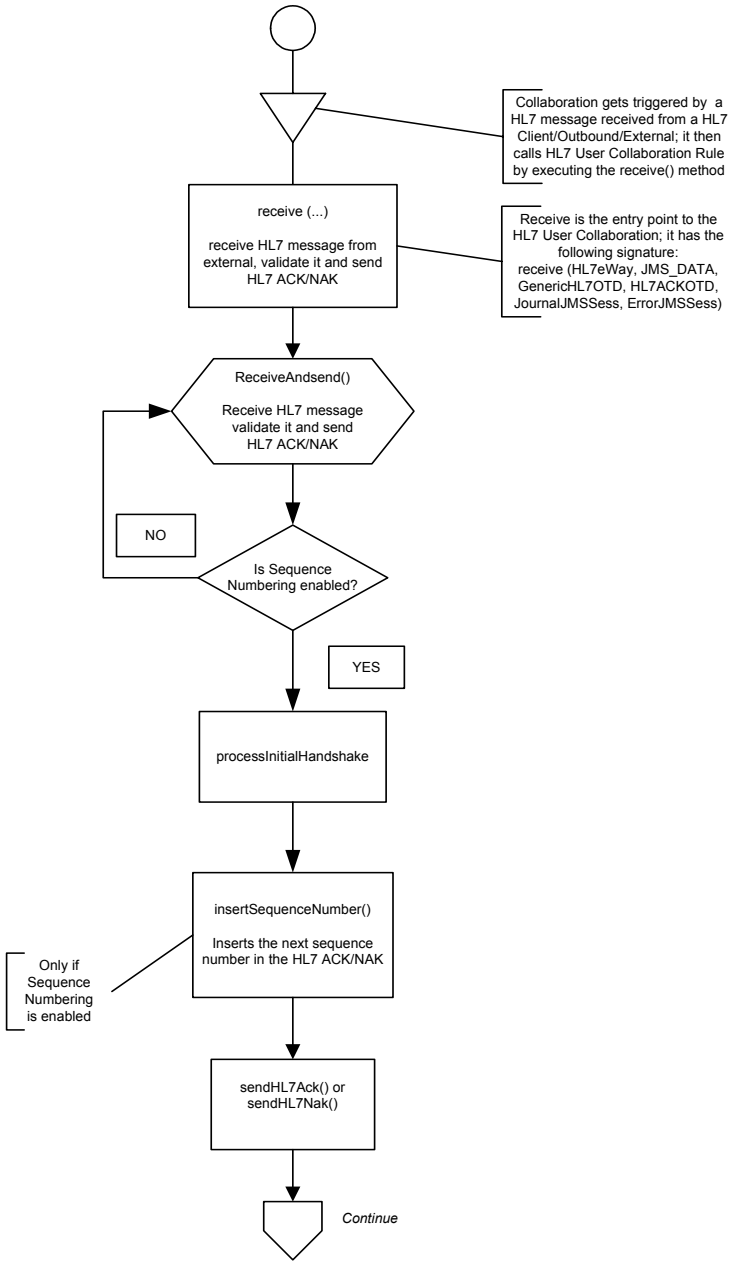
The inbound Collaboration is triggered by an HL7 message received from an external system or an outbound HL7 Client. The Collaboration calls the HL7 User Collaboration Rule by executing **receive()**. The receive method is the entry point to the HL7 User Collaboration, with the following signature: public void receive(com.stc.connector.appconn.tcpip.hl7.HL7ServerApplication input, com.stc.connectors.jms.JMS otdJMS_DATA, com.stc.SeeBeyond.OTD_Library.HL7.Generic.HL7_GENERIC_EVT.GENERIC_EVT otdHL7_GENERIC_EVT_1, com.stc.SeeBeyond.OTD_Library.HL7.Generic.HL7_ACK.ACK otdHL7_ACK_1,com.stc.connectors.jms.JMS otdJMS_JOURNAL, com.stc.connectors.jms.JMS otdJMS_ERROR) throws Throwable.

Once the message is received, the Collaboration determines whether the message needs to be validated. The HL7 message is then validated making sure that the message structure is correct. Various fields in the MSH segment of the message are also validated, such as Version ID and Sending Facility. If these fields do not match the configuration, a NAK is returned.

If sequence numbering is enabled the Collaboration checks to see if the messages sequence number is valid. If the sequence number is not valid, the eWay sends a NAK.

The validated HL7 message moves on to **processInitialHandshake()** and the sequence numbers are synchronized. The sequence number within the message is checked against the expected sequence number. If the numbers match, the Collaboration sends an ACK, if not it sends a NAK. The ACK or NAK includes information from various fields of the incoming MSA segment. The ACKs level of acknowledgement is set to **A** (acknowledgement is sent when the message is successfully received), or **C** (acknowledgement is sent after the message is successfully received and committed to the application database). See **Figure 16 on page 101**.

Figure 16 The Inbound Collaboration - Part 1



Inbound Collaboration - Part 2

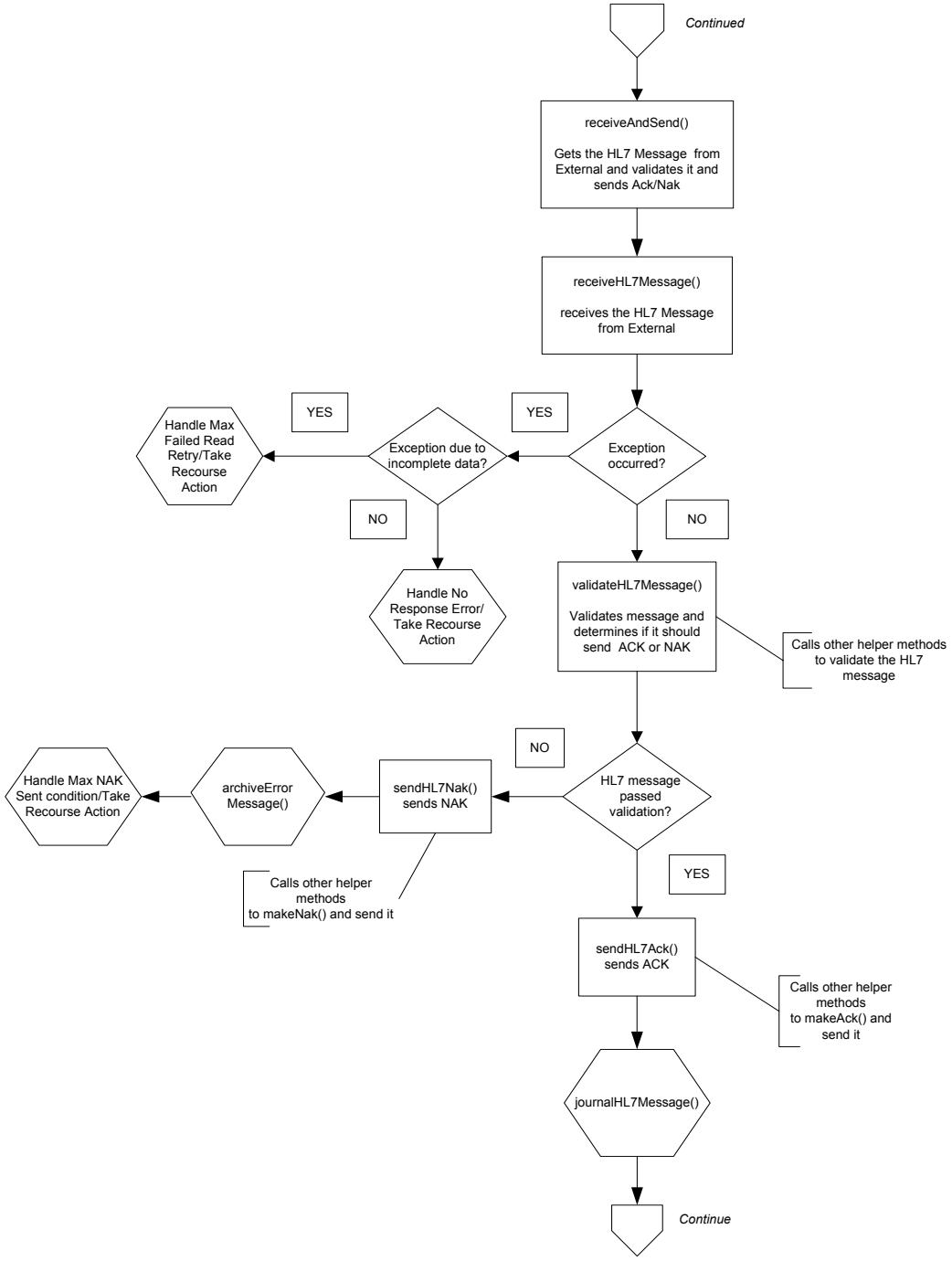
The Collaboration receives the HL7 message from the external using **receiveHL7message()**. If an exception occurs due to incomplete data, and the eWay fails to read the data within the configured number of retries, the associated recourse action is taken (see **Action on Max Failed Read Retry** on page 68). If the exception is due to no response, the associated recourse action is taken (see **Action on No Response** on page 70).

If no exception occurs, **validateHL7Message()** is called, which validates the message to determine whether to ACK or NAK the message. Other helper methods are also called to validate the HL7 message.

If the HL7 message does not pass validation, the Collaboration calls **makeNak()** and **sendHL7Nak()** to create and send the NAK to the external. The HL7 message, with the NAK, is archived to the Error Queue. If the number of consecutive NAKs sent surpasses the maximum number of retries, the associated recourse action is taken (see **Action on Max Nak Sent** on page 69).

If the HL7 message passes validation, the Collaboration calls **makeAck()** and **sendHL7Ack()** to create and send the ACK to the external. See **Figure 17 on page 103**.

Figure 17 The Inbound Collaboration - Part 2

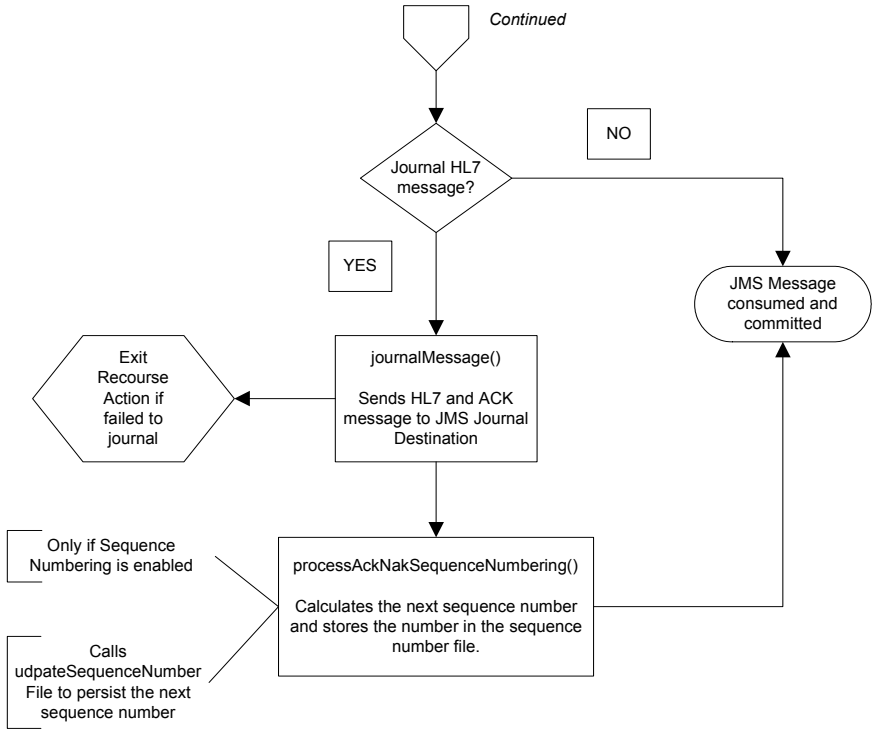


Inbound Collaboration - Part 3

After the ACK is sent, the HL7 message and the ACK are journaled to the JMS Queue Journal destination. If the message fails to journal the associated recourse action is taken.

If Sequence Numbering is enabled, the **processAckNakSequenceNumbering** method calculates the next sequence number and stores the number in the sequence number file by calling the **updateSequenceNumberFile** method to persist the next sequence number (see Figure 18.

Figure 18 The Inbound Collaboration - Part 3



5.1.2 Outbound Collaboration Overview

The Outbound HL7 Collaboration, **jcdHL7Outbound**, contains OTDs for the HL7 Resource Adapter, JMS Data, HL7 ACK, JMS Journal, and JMS Error, as well as the Generic HL7 Event. The Collaboration works with its own internal code and the Properties Configuration files. The outbound Collaboration assumes that it is reading valid HL7 messages, so the data flow that feeds this Collaboration must ensure this.

Outbound Collaboration - Part 1

The Collaboration is triggered by a JMS HL7 message. The Collaboration then calls the HL7 User Collaboration Rule by executing the **receive** method. Receive is the entry point to the HL7 User Collaboration, with the following signature: receive (input, otdHL7eWay_1, otdJMS_JOURNAL, otdJMS_ERROR, otdHL7_ACK_1, otdHL7_GENERIC_EVT_1).

The incoming HL7 message is then validated, making sure that the message structure is correct. Various fields of the message are also validated, such as Sending Facility, version ID, and MSH.

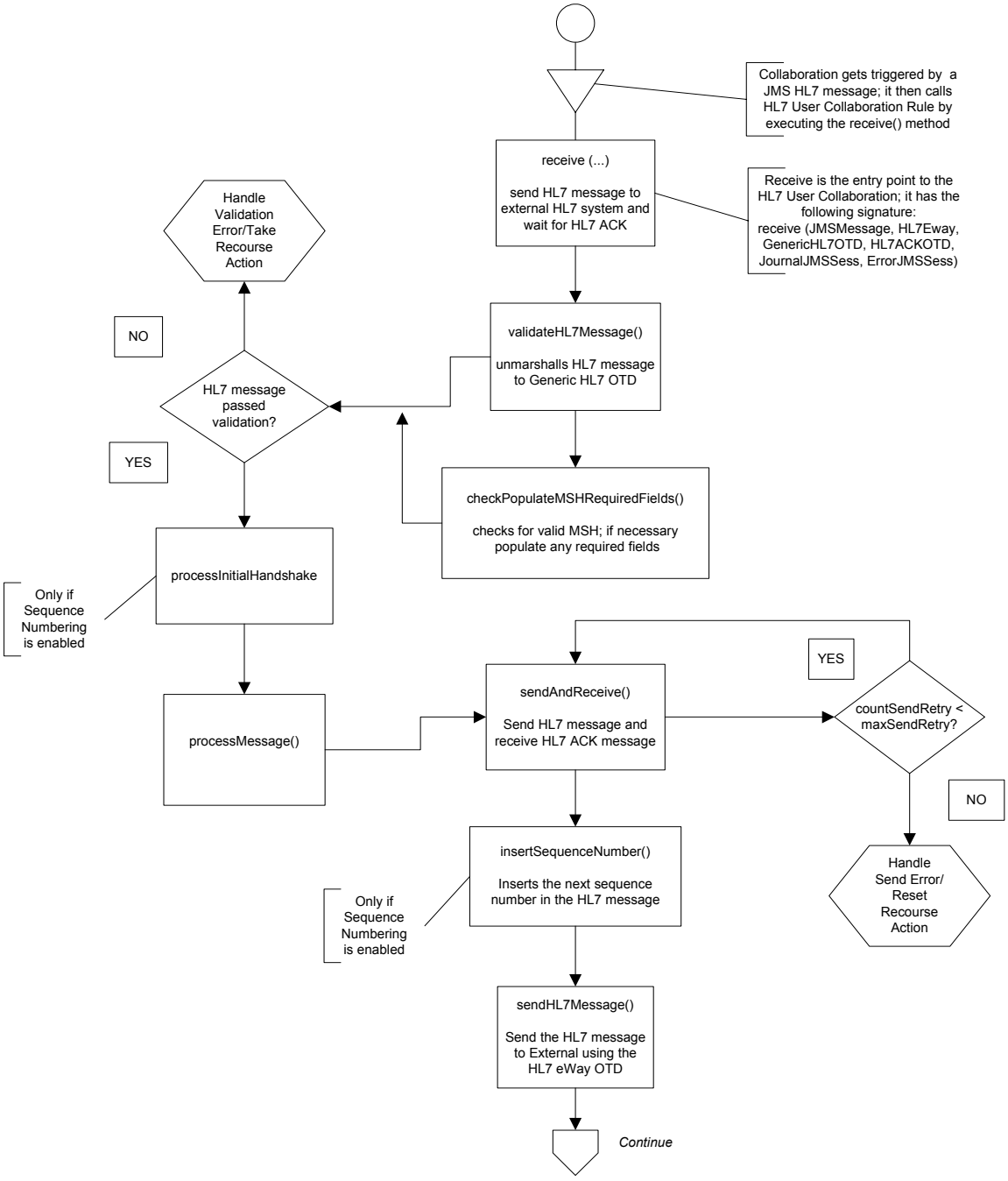
If the message does not pass validation, an error occurs and the associated recourse action is applied. If the HL7 message passes validation, the message moves on to **processInitialHandshake()** to receive a sequence number (if sequences numbering is enabled). The Collaboration takes the sequence number from the Sequence Numbering file and determines the next number to use. This number is then inserted into the HL7 message.

Next, the message moves on to **processMessage()**, which calls the helper method, **sendAndReceive()**. The **sendAndReceive** method sends the HL7 message, waits for an HL7 ACK message, and processes the ACK or NAK. The validation also checks the message structure to see if the message is unmarshaled. If a valid ACK is not received, it continues to send the HL7 message up to the configured number of retries, at which time an error occurs and the associated recourse action is taken. If a valid ACK is received, the message moves on to **insertSequenceNumber()**.

If sequence numbering is enabled, the **insertSequenceNumber** method inserts the sequence number and call **sendHL7Message()**.

The **sendHL7Message** method sends the HL7 message to the external using the HL7 eWay OTD (see **Figure 19 on page 106**).

Figure 19 Outbound Collaboration - Part 1



Outbound Collaboration - Part 2

The Collaboration receives the HL7 ACK or NAK from the external using **receiveHL7AckNak()**. If an exception occurs due to incomplete data, and the eWay fails to read the data within the configured number of retries, the associated recourse action is taken. If the exception is due to no response, the associated recourse action is taken.

If no exception occurs, the ACK or NAK message moves on to **isAckMessage()**, which validates the message to determine whether the message is an ACK or a NAK.

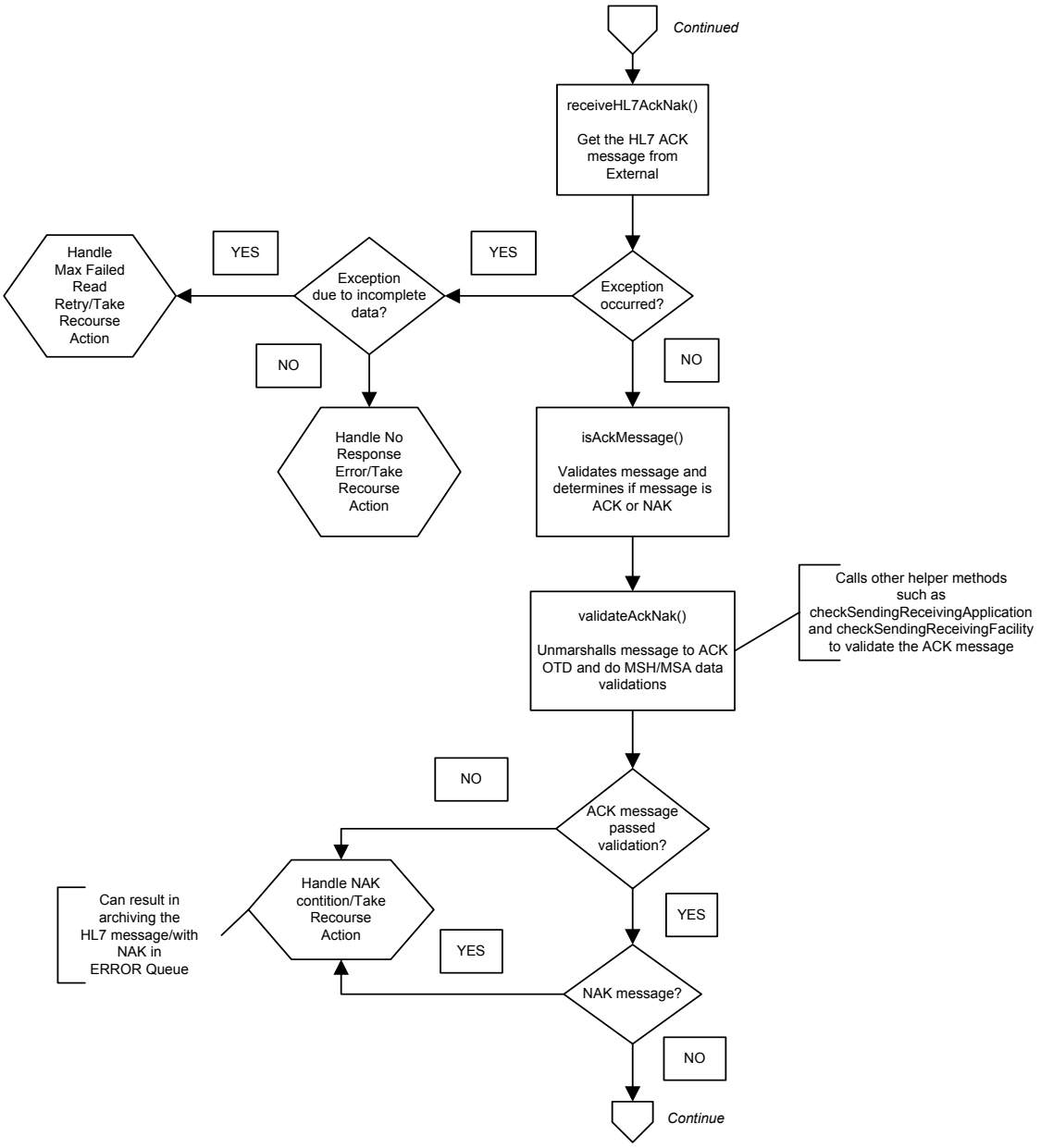
Next the validateAckNak method unmarshalls the message to the ACK OTD and does validation on the MSH/MSA data. In addition, it also calls other helper methods such as **checkSendingReceivingApplication()** and **checkSendingReceivingFacility()** to validate the ACK message.

If the message does not pass validation it is handled as a NAK, the associated recourse action is taken, and the message is archived in the Error Queue.

If the message is a NAK, the associated recourse action is taken, and the message is archived in the Error Queue, along with the NAK message, as a JMS property.

If the message is an ACK and passes validation, the message is sent on to the **journalMessage** method (see Figure 20).

Figure 20 Outbound Collaboration - Part 2

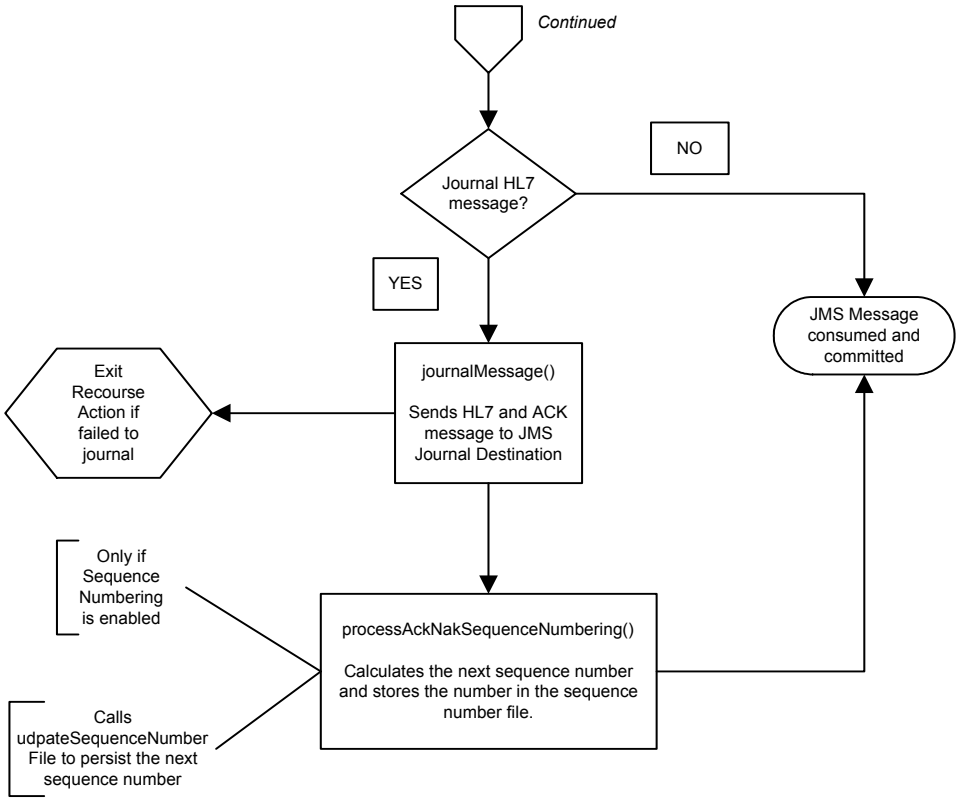


Outbound Collaboration - Part 3

If the ACK message validates, the HL7 message and ACK message are sent to the JMS Journal Destination. If the message fails to journal, the associated recourse action is taken.

If Sequence Numbering is enabled, the **processAckNakSequenceNumbering** method calculates the next sequence number and stores the number in the sequence number file, calling updateSequenceNumberFile to persist the next sequence number (see Figure 21.

Figure 21 Outbound Collaboration - Part 3



5.1.3 HL7 Outbound Test Collaboration

In addition to the Inbound and Outbound HL7 Collaborations, an Outbound test Collaboration, **jcdHL7OutboundTestDriver**, is provided to test the HL7 Outbound and HL7 Outbound Delayed ACK samples.

The **jcdHL7OutboundTestDriver** Collaboration simply picks up HL7 messages from the File eWay and sends the message to the JMS queue. This is used by the sample projects to test the number of HL7 messages processed per minute.

5.2 Creating a Copy of a Project

It is recommended that you retain the prjHL7Inbound and prjHL7Outbound project as they are when loaded and create copies of the projects to use as a basis for your new projects. To create a copy of a project, the original project is first exported to a file, the name of the original project is changed, and the new project is imported into the Repository.

Exporting a Project

To export a Project or Environment using Enterprise Designer do the following:

1. From the Enterprise Explorer tree, right-click the project you wish to export. Select **Export** from the shortcut menu. The **Export Manager** dialog box appears.
2. The selected project appears in the **Selected Projects** pane of the Export Manager; if not, select the project and click the arrow (>) button to add that project to the **Selected Projects** pane.
3. Click the Browse button and select an appropriate directory to save the exported project.
4. Click Export. The project is zipped and saved to the chosen directory.

Note: *The Enterprise Designer will not import identically named projects to the same root. Once your project has been exported, rename the original project (for example, prjHL7InboundSave) before importing the same project.*

Importing a Project

To import a Project or Environment using Enterprise Designer do the following:

1. Save any current changes before importing a new project. Any unsaved changes may be lost during import.
2. From the Enterprise Explorer tree, right-click the Repository and select **Import** from the shortcut menu. The Import Manager dialog box appears.
3. Click **Browse** and select the project that you want to import.
4. Click **Import**. The new project is imported to the repository.

5.3 Customizing Predefined Collaborations

The predefined Collaborations are designed to be extended and modified, however, for HL7 compliant systems this is not necessary. If you need to modify an HL7 Collaboration, it is strongly suggested that these "template" HL7 Collaborations be used as the basis for any new Collaborations. Therefore, it is important to maintain the original predefined **jcdHL7Inbound** and **jcdHL7Outbound** Collaborations in their initial form for future use.

There are two ways to create copies of the original HL7 Collaborations:

- Export and import a project (see **Creating a Copy of a Project** on page 109).
- Create a new Collaboration and add the HL7 Collaboration's Java code to the new Collaboration.

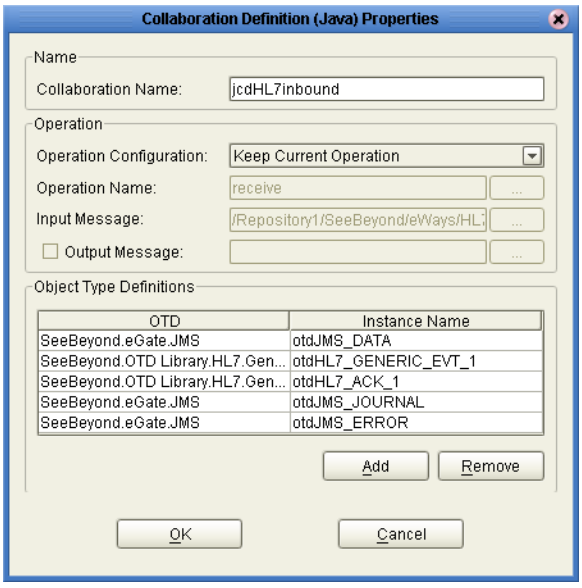
The process for creating a new Collaboration from an original predefined Collaboration includes the following.

- **Creating a Copy of the Inbound Predefined Collaboration** on page 110
- **Creating a Copy of the Outbound Predefined Collaboration** on page 116

Because the Collaboration code uses the OTD Instance Names explicitly, the OTD Instance Names must be entered exactly as those used in the original Collaborations (spelling and case). These names can be found in the Collaboration Properties dialog box.

To open the Collaboration Properties, right click the original Collaboration in the Project Explorer tree and select **Properties** from the shortcut menu. The Collaboration Definition (Java) Properties dialog box appears as displayed in **Figure 22 on page 110**. Figure 22 displays the jcdHL7inbound Collaboration Properties. To see the prjHL7 Outbound Collaboration Properties see **Figure 30 on page 119**.

Figure 22 Collaboration Properties



Copy the OTDs and the OTD Instance names from the Collaboration Properties to use when creating the new Collaborations.

5.3.1 Creating a Copy of the Inbound Predefined Collaboration

This section is a walk-through of the procedures for creating a copy of the inbound Collaboration, **jcdHL7Inbound**.

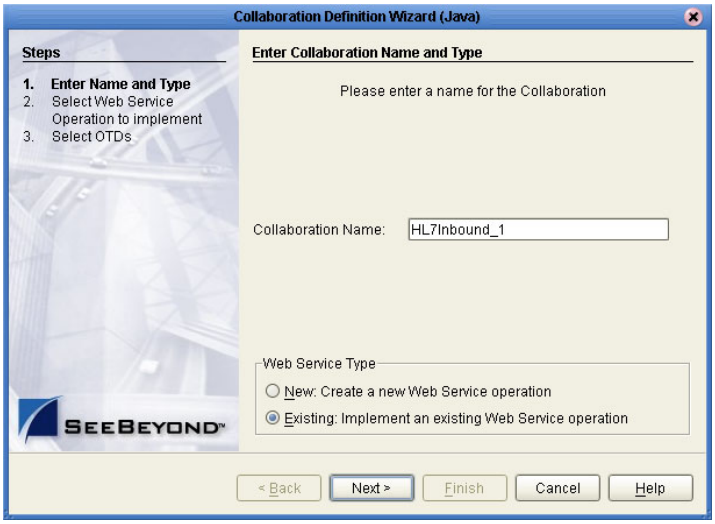
Save the original Collaboration's source code to a text file

1. Double-click the appropriate predefined Collaboration (For this example **jcdHL7Inbound**). The Collaboration opens in the Collaboration Editor.
2. From the Collaboration Editor toolbar, click the **Source Code mode** icon. The Java Source Editor pane appears.
3. Select and copy all of the Collaboration's Java source code, then close the Collaboration Editor.
4. Open a text editor program, and create a new file by pasting all the Java source code to that file. Save the file. This code will be used later in step 14.

Create a new Collaboration using the Collaboration Definition Wizard (Java)

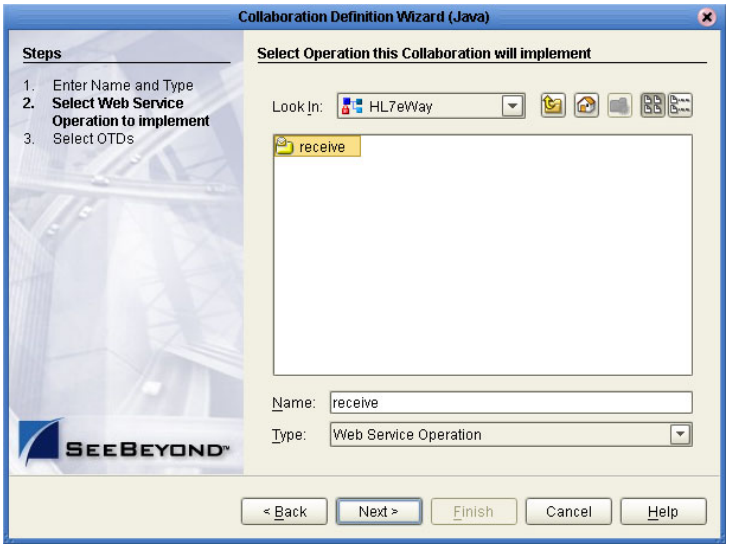
5. From the Enterprise Explorer tree, right-click the new project you are creating, and select **New > Collaboration Definition (Java)** from the shortcut menu. The **Collaboration Definition Wizard (Java)** appears.
6. For step 1 of the wizard, enter a unique name for the new Collaboration (for this example HL7Inbound_1) as displayed in **Figure 23 on page 111**. Click **Next**.

Figure 23 Collaboration Definition Wizard (Java) - Step 1



7. For step 2 of the wizard, select **SeeBeyond > eWays > HL7 > HL7 eWay > receive**. The **Name** field's value is now **receive** as displayed in Figure 24. Click **Next.**

Figure 24 Collaboration Definition Wizard (Java) - Step 2



8 For step 3 of the wizard, from the Select OTDs selection window, double-click **SeeBeyond > eGate > JMS**. A **SeeBeyond.eGate.JMS** OTD is added to the Selected OTD field with an Instance Name of **JMS_1**. Double-click the Instance name and rename it **otdJMS_DATA**.

Note: *The OTD and Instance Names must be identical to the original prjHL7Inbound Collaboration (spelling and case). These names can be found in the Collaboration Properties dialog box. To open the Collaboration Properties dialog box, right clicking the original Collaboration and select Properties from the shortcut menu.*

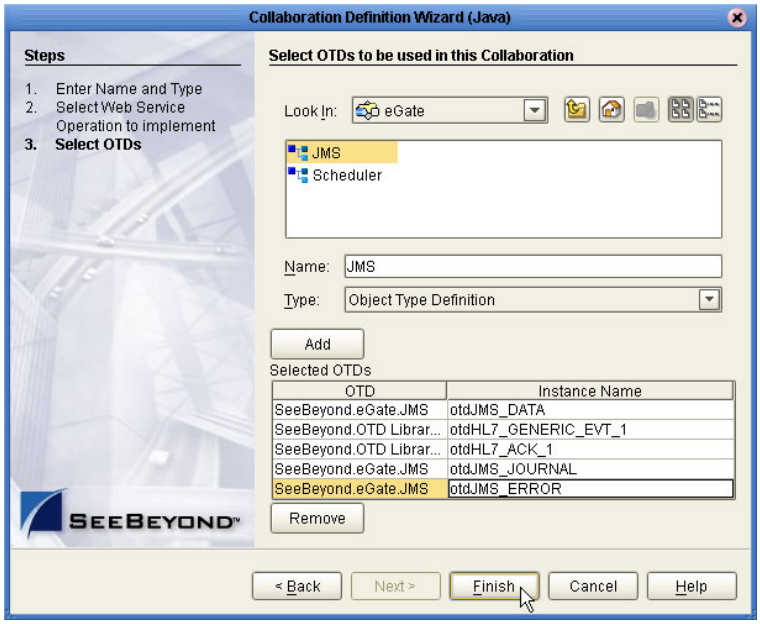
9 Click the **Up One Level** button to return to the Repository. Double-click **SeeBeyond > OTD Library > HL7 > Generic > HL7_GENERIC_EVT**. The **Seebeyond.OTD Library.HL7.Generic.HL7_GENERIC_EVT** OTD is added to the Selected OTD field with an Instance Name of **HL7_GENERIC_EVT_1** Double-click the Instance name and rename it **otdHL7_GENERIC_EVT_1**.

10 Again, click the **Up One Level** button to select **SeeBeyond > OTD Library > HL7 > Generic > HL7_ACK**. The **Seebeyond.OTD Library.HL7.Generic.HL7_ACK** OTD is added to the Selected OTD field with an Instance Name of **HL7_ACK_1** Double-click the Instance name and rename it **otdHL7_ACK_1**.

11 Again, click the **Up One Level** button to select **SeeBeyond > eGate> JMS**. A **SeeBeyond.eGate.JMS** OTD is added to the Selected OTD field with an Instance Name of JMS_2. Double-click the Instance name and rename it **otdJMS_JOURNAL**.

12 Again, click the **Up One Level** button to select **SeeBeyond > eGate > JMS**. A **SeeBeyond.eGate.JMS** OTD is added to the Selected OTD field with an Instance Name of JMS_3. Double-click the Instance name and rename it **otdJMS_ERROR**. The Collaboration Definition Wizard (Java) should now appear as displayed in **Figure 25 on page 113**. Click **Finish**.

Figure 25 Collaboration Definition Wizard (Java) - Step 3



13 The new Collaboration appears in the Enterprise Explorer tree and the Collaboration Editor open to the new Collaboration. From the Collaboration Editor toolbar, click the Source Code mode icon.

Copy the original Java Source Code to the new Collaboration

14 To create a copy of the original Collaboration you must cut and paste the code from the original Collaboration (that you saved to a text file in step 4), to the New Collaboration. Open the .txt file with the original Java source code.

15 From the original Collaboration's Java source code, copy all the code between:

```
public class jcdHL7inbound
```

and

```
public void receive( com.stc.connector.appconn...
```

This block of code starts with the following lines of code:

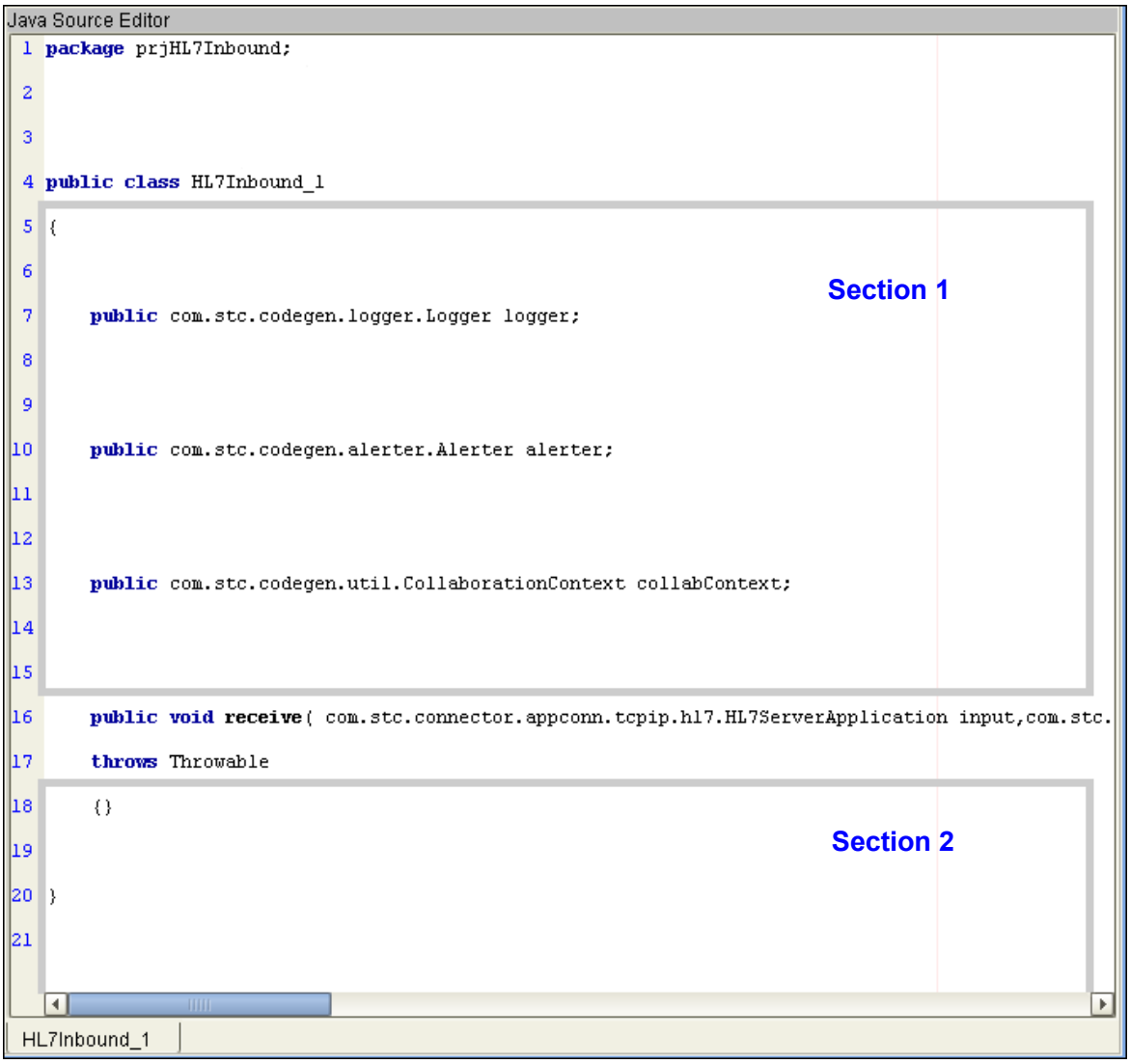
```
{

    private String hostId = null;

    public com.stc.codegen.logger.Logger logger;

    public com.stc.codegen.alerter.Alerter alerter;
```

and ends with the following lines of code:

```
// Component name for shutdown alert
private static final String SHUTDOWN_HL7_COMPONENT_NAME = "HL7";
```

Paste this copied block of code into **Section 1** as displayed in Figure 26 on page 114.

Figure 26 Java Source Code Editor - Inserting original Inbound Java source code



This replaces the current lines of code in the new Collaboration's Java source code between:

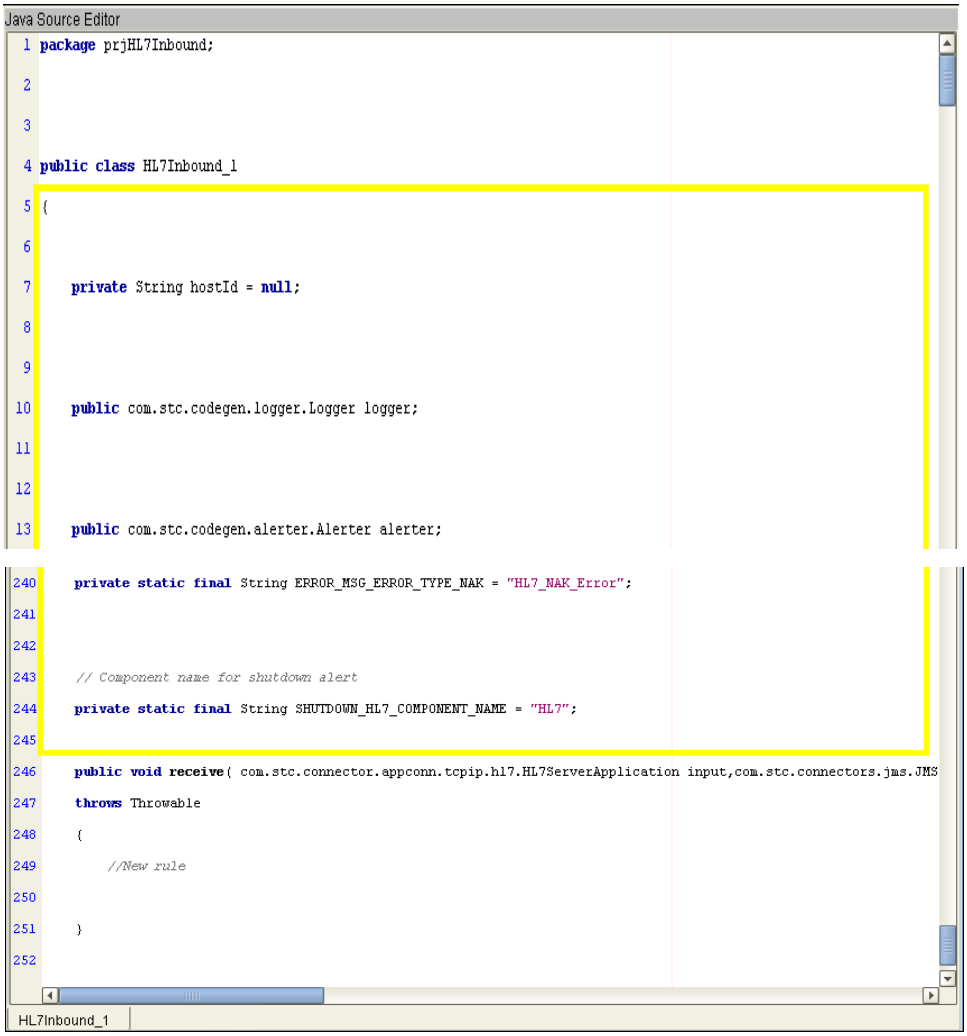
```
public class <name of your Collaboration>
```

and

```
public void receive( com.stc.connector.appconn.....
```

as displayed in Figure 27 on page 115.

Figure 27 Java Source Code Editor - Inserting original Inbound Java source code



16 Next, from the original Collaboration's Java source code, copy all the code after:

```
public void receive( com.stc.connector.appconn.tcpip.hl7.HL7ServerApplication
input,com.stc.connectors.jms.JMS
otdJMS_DATA,com.stc.SeeBeyond.OTD_Library.HL7.Generic.HL7_GENERIC_EVT.GENERIC_EVT
otdHL7_GENERIC_EVT_1,com.stc.SeeBeyond.OTD_Library.HL7.Generic.HL7_ACK.ACK
otdHL7_ACK_1,com.stc.connectors.jms.JMS otdJMS_JOURNAL,com.stc.connectors.jms.JMS
otdJMS_ERROR )
    throws Throwable
```

starting with the following lines:

```
{
        //@map:(LOG_LEVEL_DEBUG,"Entered Collaboration")
        log( LOG_LEVEL_DEBUG,"Entered Collaboration" );
```

to the end (all remaining code).

17 Paste this copied block of code into **Section 2** as displayed in Figure 26 on page 114.

This is in the new Collaboration's Java source code between:

```
throws Throwable
```

and the end (replacing all remaining lines of code).

18 From the Collaboration Editor toolbar, click the **Commit Changes** icon.

19 To use the new Collaboration, you must first incorporate it into a new or existing project, then drag and drop the Collaboration onto a Service and re-associate (bind) it with the other project components.

5.3.2 Creating a Copy of the Outbound Predefined Collaboration

This section is a walk-through of the procedures for creating a copy of the outbound Collaboration, **jcdHL7Outbound**.

Save the original Collaboration's source code to a text file

1 Double-click the appropriate predefined Collaboration (For this example **jcdHL7Outbound**). The Collaboration opens in the Collaboration Editor.

2 From the Collaboration Editor toolbar, click the **Source Code mode** icon. The Java Source Editor pane appears.

3 Select and copy all of the Collaboration's Java source code, then close the Collaboration Editor.

4 Open a text editor program and create a new file by pasting all the Java source code to that file. Save the file. This will be later used in step 14.

Create a new Collaboration using the Collaboration Definition Wizard (Java)

5 From the Enterprise Explorer tree, right-click the new project you are creating, and select **New > Collaboration Definition (Java)** from the shortcut menu. The **Collaboration Definition Wizard (Java)** appears.

6 For step 1 of the wizard, enter a unique name for the new Collaboration (for this example HL7Outbound_1). Click **Next**.

7 For step 2 of the wizard, select **SeeBeyond > eWays > File > FileClient > receive**. The **Name** field's value is now **receive**. Click **Next.**

8 For step 3 of the wizard, from the Select OTDs selection window, double-click **SeeBeyond > eWays > HL7 > HL7eWay**. A **SeeBeyond.eWays.HL7** OTD is added to the Selected OTD field with an Instance Name of **HL7eWay_1**. Double-click the Instance name and rename it **otdHL7eWay_1**.

Note: *The OTD and Instance Names must be identical to the original prjHL7Inbound Collaboration (spelling and case). These names can be found in the Collaboration Properties dialog box. To open the Collaboration Properties dialog box, right clicking the original Collaboration and select Properties from the shortcut menu.*

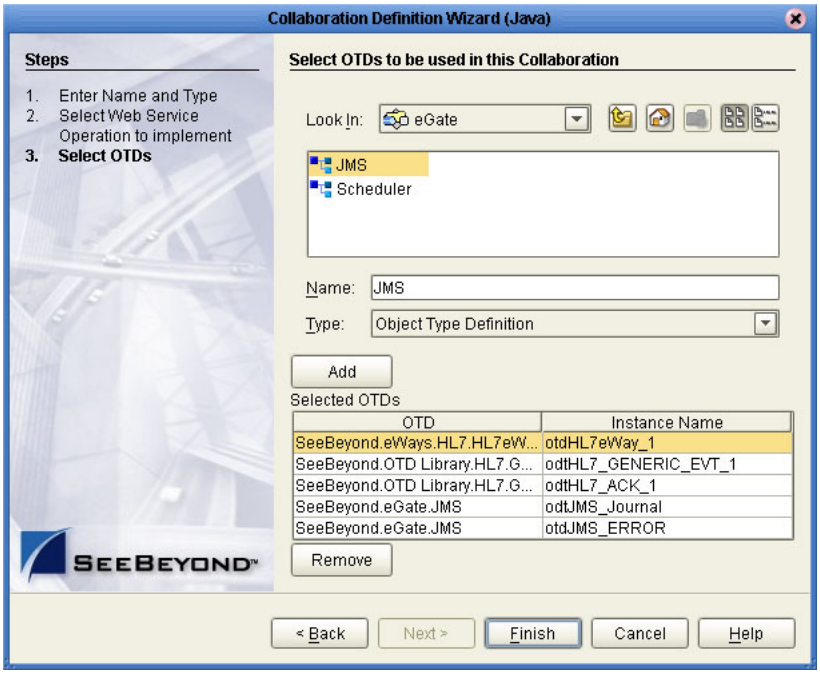
9 Click the **Up One Level** button to return to the Repository. Double-click **SeeBeyond > OTD Library > HL7 > Generic > HL7_GENERIC_EVT**. The **Seebeyond.OTD Library.HL7.Generic.HL7_GENERIC_EVT** OTD is added to the Selected OTD field with an Instance Name of **HL7_GENERIC_EVT_1** Double-click the Instance name and rename it **otdHL7_GENERIC_EVT_1**.

10 Again, click the **Up One Level** button to select **SeeBeyond > OTD Library > HL7 > Generic > HL7_ACK**. The **Seebeyond.OTD Library.HL7.Generic.HL7_ACK** OTD is added to the Selected OTD field with an Instance Name of **HL7_ACK_1** Double-click the Instance name and rename it **otdHL7_ACK_1**.

11 Again, click the **Up One Level** button to select **SeeBeyond > eGate > JMS**. A **SeeBeyond.eGate.JMS** OTD is added to the Selected OTD field with an Instance Name of JMS_2. Double-click the Instance name and rename it **otdJMS_JOURNAL**.

12 Again, click the **Up One Level** button to select **SeeBeyond > eGate > JMS**. A **SeeBeyond.eGate.JMS** OTD is added to the Selected OTD field with an Instance Name of JMS_3. Double-click the Instance name and rename it **otdJMS_ERROR**. The Collaboration Definition Wizard (Java) should now appear as displayed in Figure 28. Click **Finish**.

Figure 28 Collaboration Definition Wizard (Java) - Step 3



13 The new Collaboration appears in the Enterprise Explorer tree and the Collaboration Editor open to the new Collaboration. From the Collaboration Editor toolbar, click the Source Code mode icon.

Copy the original Java Source Code to the new Collaboration

14 To create a copy of the original Collaboration you must cut and paste the code from the original Collaboration (that you saved to a text file in step 4) to the new Collaboration. Open the .txt file with the Java source code.

15 From the original Collaboration's Java source code, copy all the Java code between:

```
public class jcdHL7Outbound
```

and

```
public void receive( com.stc.connectors.jms...
```

This block of code starts with the following lines of code:

```
{

    public com.stc.codegen.logger.Logger logger;
```

and ends with the following lines of code:

```
// HL7 versions
private static final String HL7_VERSION__2_5 = "2.5";
```

16 Paste these lines into Section 1 of the new Collaboration's Java source code, in the Collaboration Editor's Java Source Code Editor window, as displayed in Figure 29. This is in the new Collaboration's Java source code between:

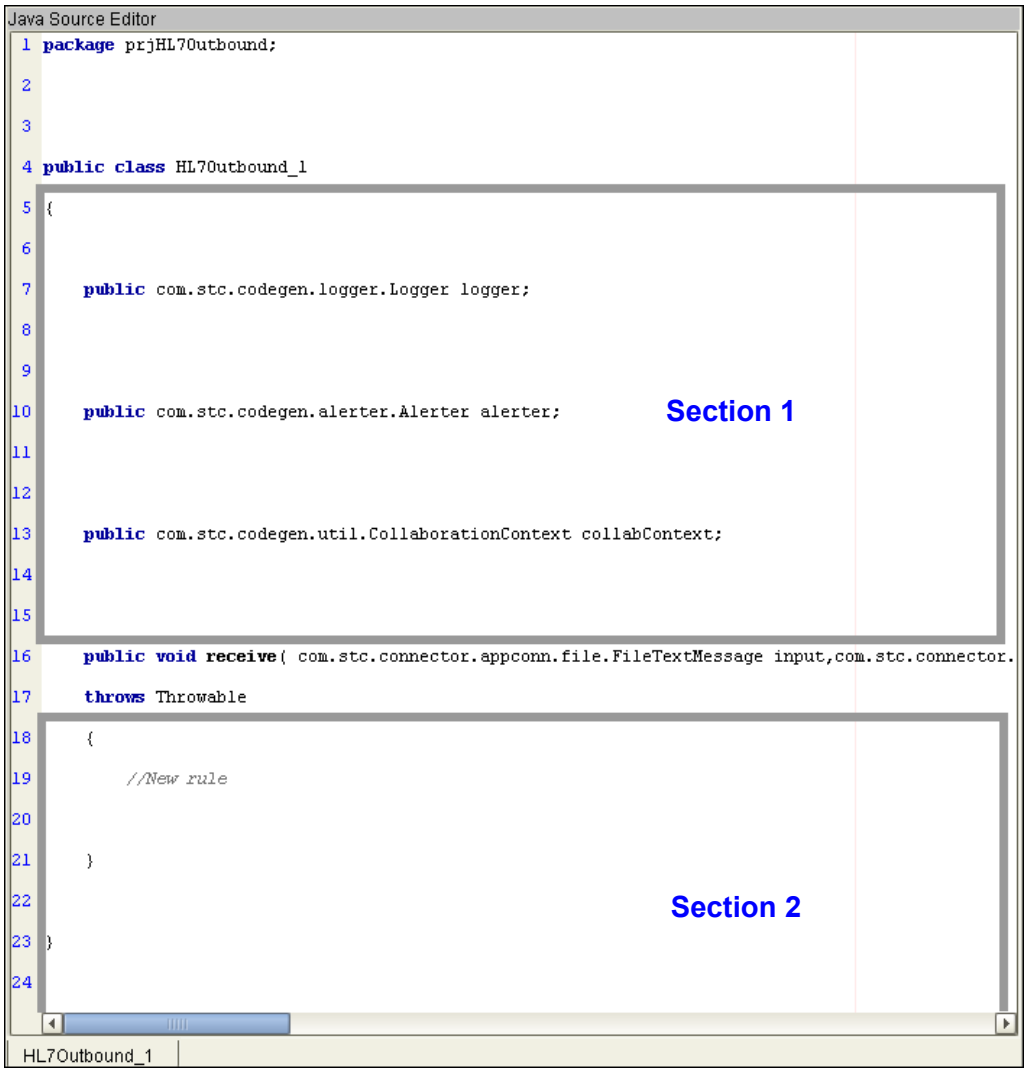
```
public class <name of your Collaboration>
```

and

```
public void receive( com.stc.connector.appconn...
```

replacing the current lines of Java code between those two lines, as displayed in Figure 29 on page 118

Figure 29 Java Source Code Editor - Inserting original Outbound Java source code



17 Next, from the original Collaboration's Java source code, copy all the code after:

```
public void receive( com.stc.connectors.jms.Message input...
throws Throwable
```

Starting with the following lines:

```
{
        // get the fully qualified collaboration name
        collabName = collabContext.getProjectPath() + "__...
```

to the end.

18 Paste this copied block of code into Section 2 as displayed in **Figure 29 on page 118**, replacing the new Collaboration's the remaining lines of code.

19 From the Collaboration Editor toolbar, click the **Commit Changes** icon.

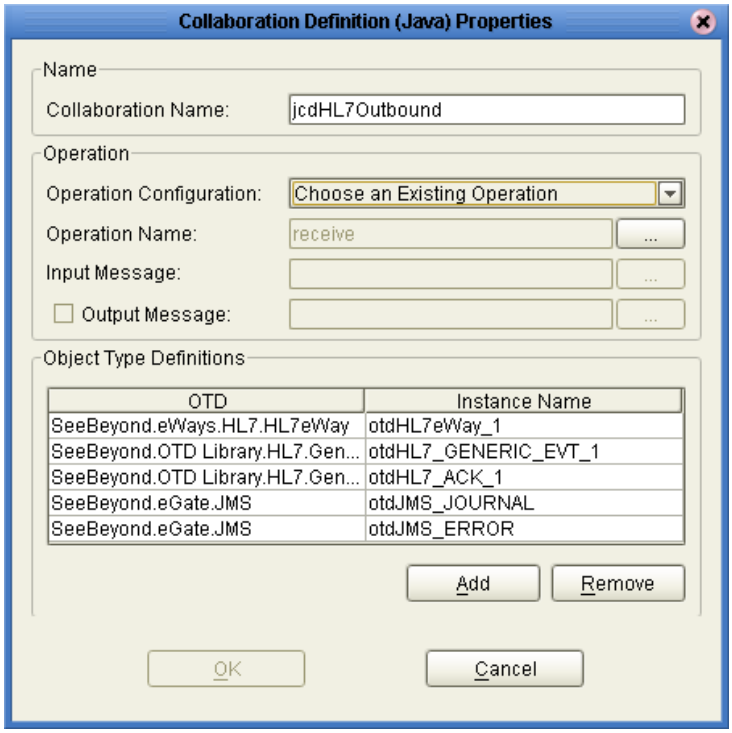
20 To use the new Collaboration, you must first incorporate it into a new or existing project, then drag and drop the Collaboration onto a Service and re-associate (bind) it with the other project components.

5.3.3 Adding an OTD to a Collaboration

In some cases, a specific HL7 message or messages may need to be added to the Collaboration. To add an HL7 OTD to an existing Collaboration do the following:

1 From the Enterprise Explorer tree, right-click the Collaboration to which you want to add the HL7 OTD. From the shortcut menu, select Properties. The Collaboration Definition (Java) Properties dialog box appears (for this example, the jcdHL7Outbound Collaboration is displayed in Figure 30).

Figure 30 Collaboration Properties



2 Select **Choose an Existing Operation** as the value for the **Operation Configuration** field.

3 Click **Add**. From the **Select** dialog box, browse to and select the appropriate HL7 OTD. The OTD is added to the Properties's OTDs.

4 Once the Collaboration has been modified and additional Java code or Business Rules have been added, it must be dragged and dropped again from the Enterprise Explorer tree to the associated Service, and all associations (bindings) must be re-mapped.

5 Once the Collaboration has been modified, you must first incorporate it into a new or existing project, then drag and drop the Collaboration onto a Service and re-associate (bind) it with the other project components.

Renaming a Collaboration

To rename an existing Collaboration do the following:

1 Right-click the Collaboration and select **Properties** from the shortcut menu.

2 Replace the name in the Collaboration Name field with your new name.

3 Select **Choose an Existing Operation** as the value for the **Operation Configuration** field and click **OK**.

4 Once the Collaboration has been modified, you must first incorporate it into a new or existing project, then drag and drop the Collaboration onto a Service and re-associate (bind) it with the other project components.

5.3.4 Adding HL7 Collaborations to a Connectivity Map

When creating a Connectivity Maps for an HL7-compliant system, the outbound and inbound Collaborations can be used "as is" without duplication. This ensures that there is only one copy for maintenance reasons.

It is recommended that you use the two sample projects, **prjHL7Inbound** and **prjHL7Outbound**, as templates for inserting an HL7 Collaboration into a flow. Each instance of an HL7 Collaboration needs to connect to JMS for data, journaling, and error messaging. The HL7 Collaboration must also connect to an HL7 RA. Each of the JMS queues can be shared across the Connectivity Maps, or each Connectivity Map can use its own unique queue (or some combination of the two).

For more information on how to add a Java Collaboration to a Service see the *eGate Integrator User's Guide* and the *eGate Integrator Tutorial*.

Chapter 6

Working With HL7 OTDs

This chapter provides information on viewing, testing and modifying OTDs using the OTD Editor.

What's in this Chapter

6.1 Generic HL7 OTDs

The eWays require special abstracted OTDs that represent all of the versions and types of HL7 messages. Two have been created:

- HL7_GENERIC_EVT: Parses any valid HL7 message.
- HL7_ACK: Parses any valid HL7 ACK or NAK.

The main purpose of these messages is to make the MSH fields available for implementation of the HL7 messaging protocol. It is assumed that other processes transform incoming and outgoing messages, to and from other formats.

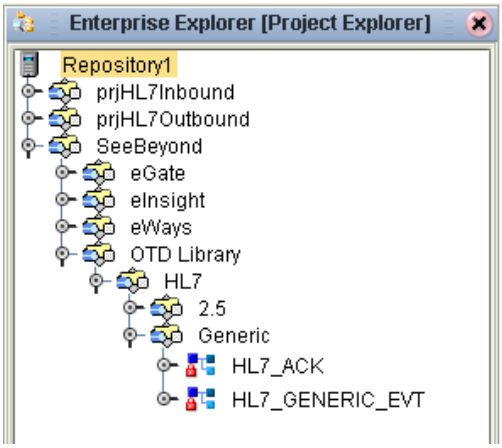
The OTDs are found on the Project Explorer tree under **SeeBeyond > OTD Library > HL7 > Generic.**

6.2 Viewing an OTD using the OTD Editor

To view the HL7 Generic OTS, or if you are using the SeeBeyond TCP/IP HL7 eWay in conjunction with the SeeBeyond HL7 OTD Library, the OTDs are accessed from the SeeBeyond Enterprise Designer. To open an OTD using the OTD Editor, do the following:

1. From the Enterprise Designer's Project Explorer tree, expand the **OTD Library** node, the **HL7** node, and the node for the appropriate HL7 version (see Figure 31). Only the version or versions you install will be displayed in your build.

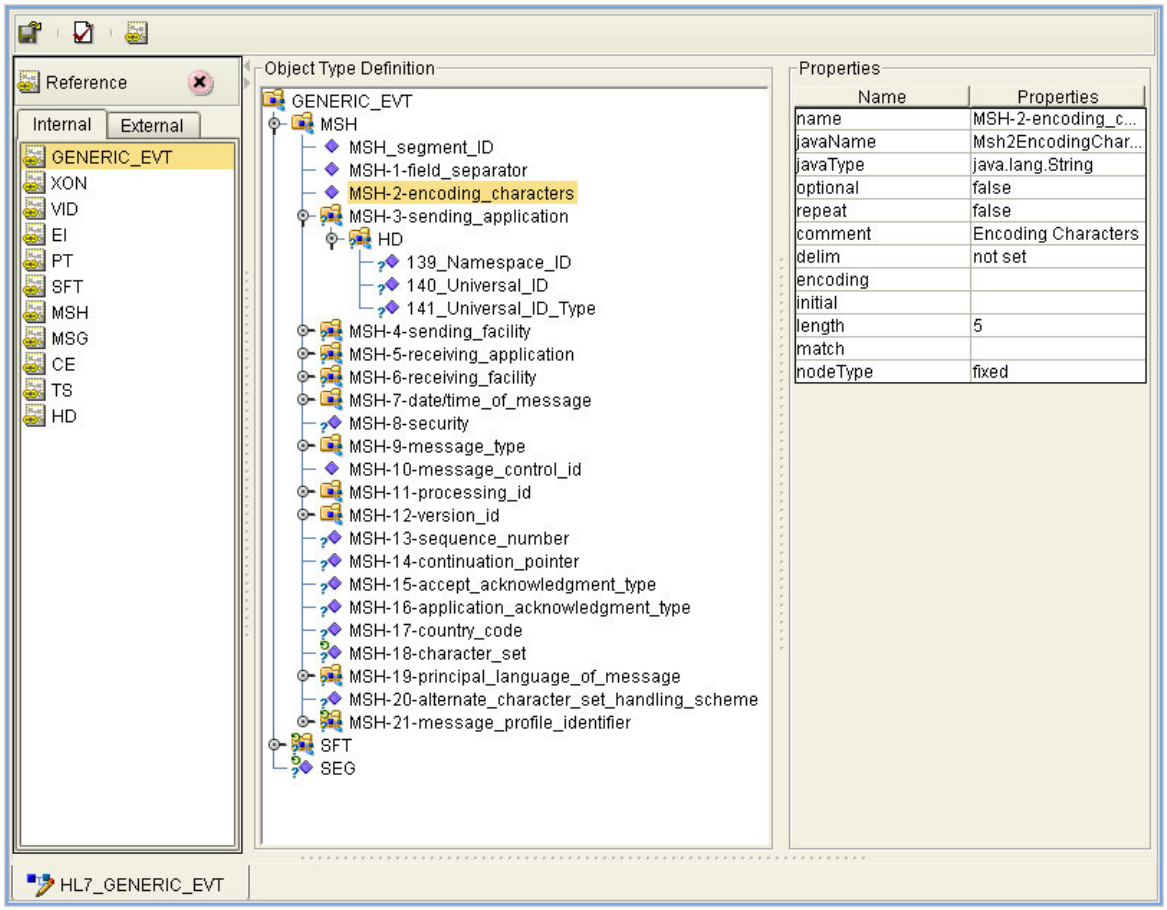
Figure 31 Selecting the OTD from the Project Explorer Tree



2 If you are opening an OTD to modify it, right-click that OTD and select **Check Out** from the shortcut menu. You can now modify the OTD using the OTD Editor.

3 Double-click the appropriate OTD (for this example **HL7_GENERIC_EVT**). The OTD Editor appears displaying the selected OTD. You can expand or contract or contract nodes from the OTD Editor by single-clicking the icon to its left or by double-clicking the node name (see Figure 32).

Figure 32 The OTD Editor - HL7_GENERIC_EVT



4 View and edit OTD Properties for any specific node from the Properties pane in the upper right section of the OTD Editor. Simply select a node and the node's properties are displayed in the Properties pane.

6.3 Modifying an OTD Using the OTD Editor

OTD Check Out and Check In

The generic OTD Library is distributed as a standard library from which variants may be made, but the original generic OTDs must not be modified, as the inbound and outbound Collaborations rely on these. Therefore, if you need to modify a generic OTD for your purposes, make a copy of the original OTD first, then modify the copy.

The OTDs are edited or modified using the OTD Editor. To edit an OTD it must first be checked out. To check out an OTD, from the Project Explorer tree, right-click the OTD and select **Check Out** from the shortcut menu.

Once an OTD has been edited, check in the OTD back to lock the OTD. The OTD cannot be edited until it is once again checked out. To check in an OTD, right click the OTD in the Project Explorer tree and select **Check In** from the shortcut menu. The OTD file icon now appears in the Project Explorer tree as "locked" (The OTD icon includes a red padlock).

6.3.1. Changing the OTD Delimiters

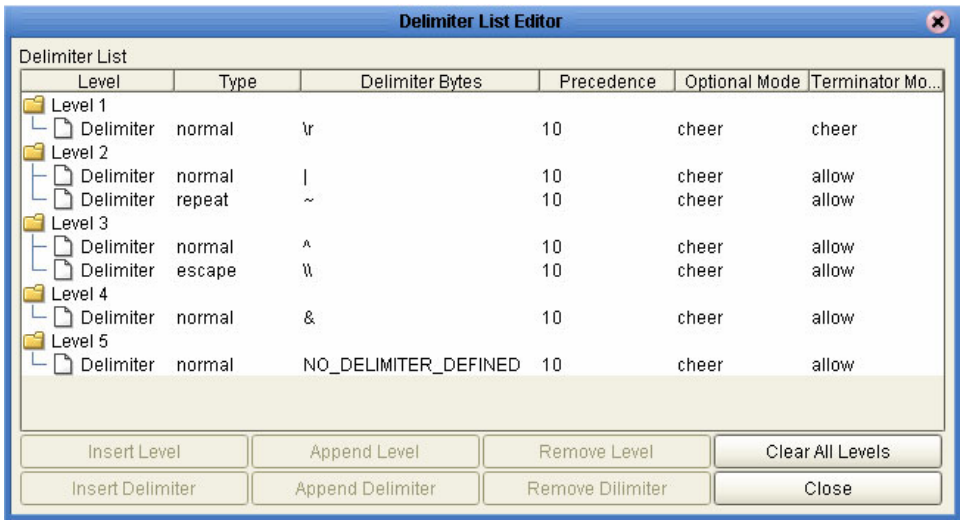
The most common reason to modify an HL7 OTD is to change the its delimiters. Typically, delimiters for all node levels are set (and modified) from the root node. Delimiters that are set from a lower level in the OTD will override any upper level settings for that specific node.

Be aware that the default level 1 delimiter character is a non-ASCII character. Once it has been changed it cannot be typed back in as a character (but can be pasted).

Delimiters are explained in greater detail later in this section (see **Specifying Delimiters** on page 129, but to quickly demonstrate how delimiters are changed, do the following:

1 This example uses the **HL7_25_ADT_A02** OTD. From the OTD Editor, select the root node in the Object Type Definition pane (for this example **ADT_A02**). From the Properties pane, double-click the **delim** properties field. An ellipsis (...) button appears in the field. Click the ellipsis button. The **Delimiter List Editor** appears (see **Figure 33 on page 124**).

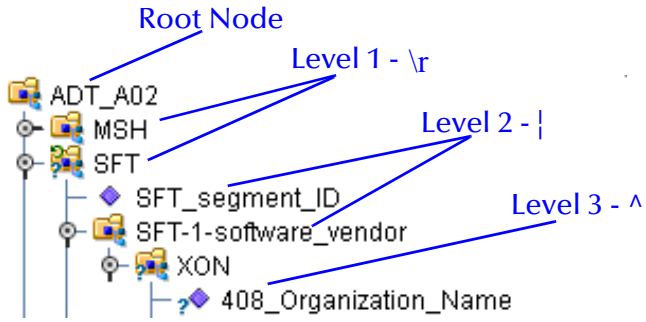
Figure 33 HL7 OTD Editor - Delimiter List Editor



2 Double-clicking a field for any level makes the field editable, or displays a list of options. Double-click the Delimiter Bytes field for level 3 (see Figure 38). Change the current delimiter character to a pound sign (#), **Tab** to the next field, and click **OK**.

The delimiter for all level 3 nodes in the OTD is now a pound sign (#), unless it is specified differently from a more direct level. For example, if you change the delimiter for any specific level 3 node from that node's properties, that value will override the value set from the root node. Figure 34 displays an example of various levels in the Object Type Definition tree, from the root node.

Figure 34 Delimiter Levels From the Root Node



When the Delimiter List Editor is opened from a different level node (for example, a level 3 node), that node appears as level 1 in the editor.

Creating a Delimiter List if one is not specified

To specify a different delimiter for a node, when the delimiter for that specific node is not set (meaning that the node gets its delimiter from an upper level node), do the following:

1 Select the node, and from the Properties pane double-click the **delim** Properties field and click the ellipsis button to open the Delimiter List Editor.

2 Right-click the Delimiter List Editor under the list headers, and select **Add Level To End** from the shortcut menu. Level 1 is added to the Delimiter List Editor.

3 Right-click Level 1 and select **Add Delimiter**. A delimiter is added to level 1.

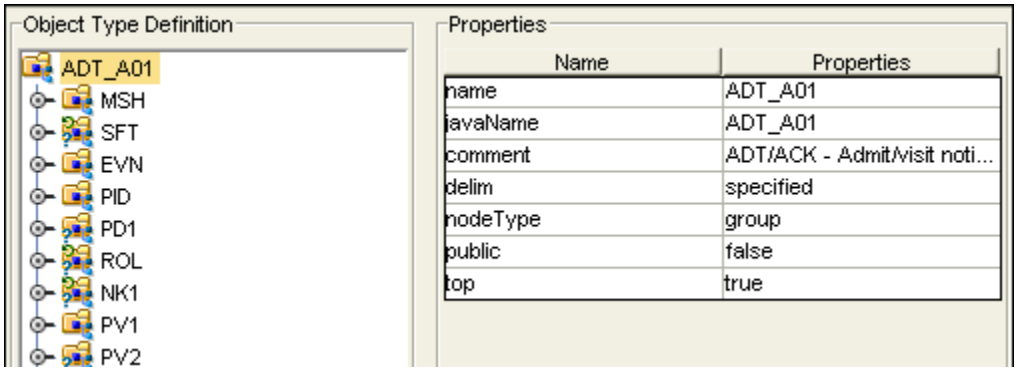
4 Double-click the Delimiter Bytes field and enter a percent sign (%). **Tab** to the next field (you must Tab to the next field or select another field before closing the editor to save your last change). Click **OK**.

5 The delimiter for that specific node is now **%**, as displayed in the node's **showDelim** field.

OTD Properties

The Object Type Definition pane (center pane) of the OTD Editor displays the nodes, elements, and fields of the OTD. When any of these are selected, the item's properties are displayed in the Properties pane.

Figure 35 HL7 OTD Editor - OTD Node Properties



Node Properties

When an HL7 OTD is opened in the OTD Editor, the properties of the root node are displayed in the Properties pane (see Figure 35). The configurable node properties are displayed in Table 3.

Table 3 Node Properties

Name	Description
name	Node display name. This can be a virtually-arbitrary string.
javaName	Property accessor basename. This is normally derived from the display name, modified to suit the restrictions on Java identifiers, and supplied automatically by eGate.
comment	Free-form text (no run-time effect).
delim	Specified delimiter. See **Delimiter Properties** on page 130.
nodeType	Governs the marshal/unmarshal format. See **Specifying the Node Type** on page 128.
public	Reserved for future development

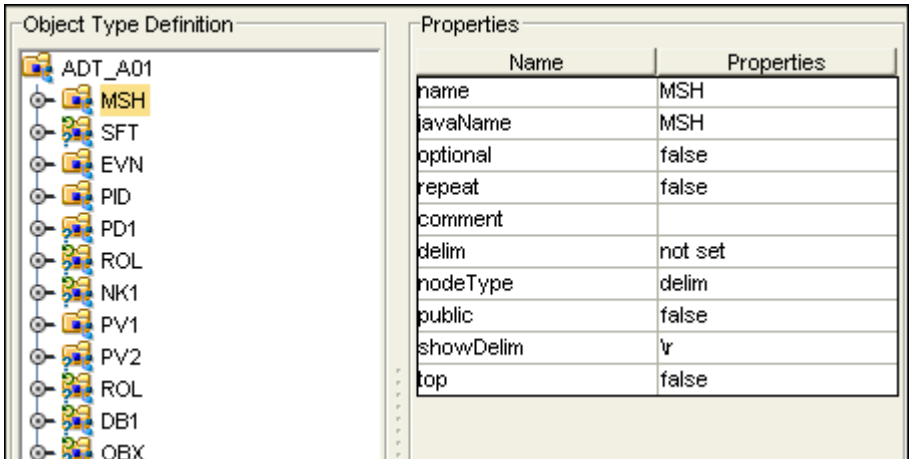
Name	Description
showDelim	If nodeType is delimited.
top	Flag on root node: support marshal/unmarshal (T/F).

Important: *Do not modify the javaName property.*

Element Properties

The set of properties associated with the element level is shown in Figure 36.

Figure 36 OTD Editor - OTD Element Properties



The configurable element properties are displayed in Table 4

Table 4 Element Properties

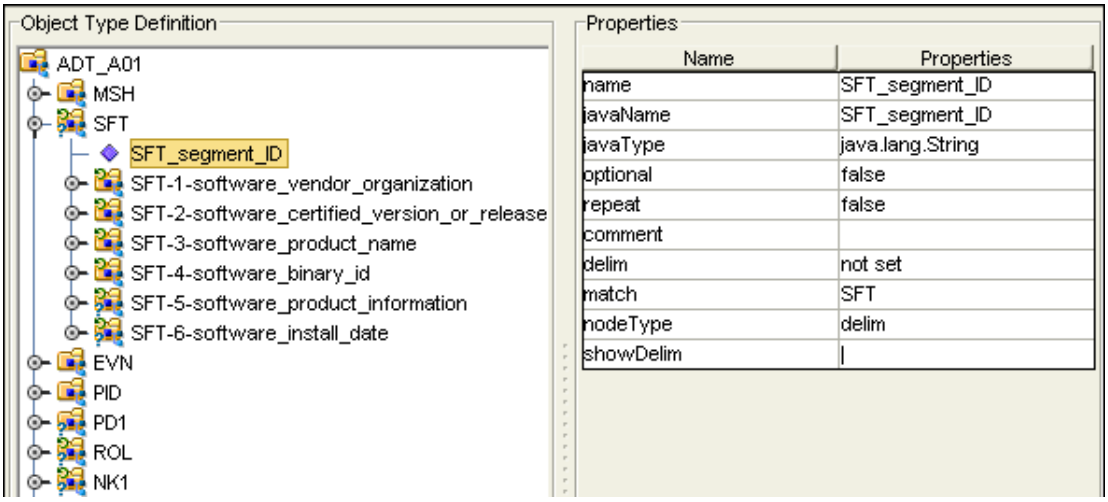
Name	Description
name	Element display name.
javaName	Property accessor basename.
optional	Flag: Can the element be absent? (T/F) Not applicable to root, or child of a choice Node.
repeat	Flag: Can the node appear multiple times? (T/F) Not applicable to root, or child of a choice Node.
comment	Free-form text (no run-time effect).
delim	Delimiter specification (see **Specifying Delimiters** on page 129).
nodeType	Governs the marshal/unmarshal format.
showDelim	If nodeType is delimited,

Important: *Do not modify the javaName property.*

Field Properties

The set of properties associated with the field level is shown in Figure 37.

Figure 37 OTD Editor - OTD Field Properties



The configurable field properties are displayed in Table 5

Table 5 Field Properties

Name	Description
name	Field display name.
javaName	Property accessor basename.
javaType	Java type: can be either java.lang.String or byte array (byte[]).
optional	Flag: Can the element be absent? (T/F) Not applicable to root, or child of a choice Element.
repeat	Flag: Can the node appear multiple times? (T/F) Not applicable to root, or child of a choice Element.
comment	Free-form text (no run-time effect).
delim	Delimiter specification (see **Specifying Delimiters** on page 129).
match	If **nodeType** is delimited, performs exact match to the data.
nodeType	Governs the marshal/unmarshal format.
showDelim	If **nodeType** is delimited, shows the delimiter.

Important: *Do not modify the javaName property.*

6.3.2 Specifying the Node Type

Double-clicking the nodeType properties field activates the field for editing. Click the button to display the selection menu. Descriptions of the property options are listed in Table 6.

Table 6 Node Type Property Options

Option	Description	Element	Field	Internal
alter	Alter (*alternate*) selects one child or the other. One child is always present after the unmarshal operation. Applies only to elements.	Yes	No	choice
array	Array is a delimited structure. If repeated, occurrences are separated by the *repeat* delimiter. The last occurrence may be terminated by a *normal* delimiter.	Yes	Yes	simple or group
delim	Delim (*delimited*) structure. If repeated, occurrences are separated by a *normal* delimiter.	Yes	Yes	simple or group
fixed	Fixed indicates a fixed length, which is specified by non-negative integer (or zero to indicate end of parent node data).	Yes	Yes	simple or group
group	Group provides organizational grouping for purposes such as repetition. Applies only to elements.	Yes	No	group
trans	Trans (*transient*) appears only in an internal tree as a scratch pad field. It does not appear in external data representation, and can only have *trans* nodeTypes as children.	Yes	Yes	choice, simple, or group

Note: *If you move a OTD node, you must reset the nodeType for that node.*

6.3.3 Specifying Delimiters

Any node can define a set of delimiters to be used in the external data representation for itself and its descendents in the hierarchical data structure. If a node defines a delimiter list, this negates any effect of any ancestor's delimiter list on itself and its descendents. The delimiter list is typically specified on the root node.

For example, if you want to parse the following data:

```
a^b|c^d|e
```

you might define an OTD as follows:

- demo-otd
  - element1
    - field1
    - field2
  - element2
    - field3
    - field4
  - field5

The delimiter list for this OTD will be specified on the *demo-otd* element, so that it applies to the entire OTD, and will have two levels:

- Level 1
  - Delimiter |
- Level 2
  - Delimiter ^

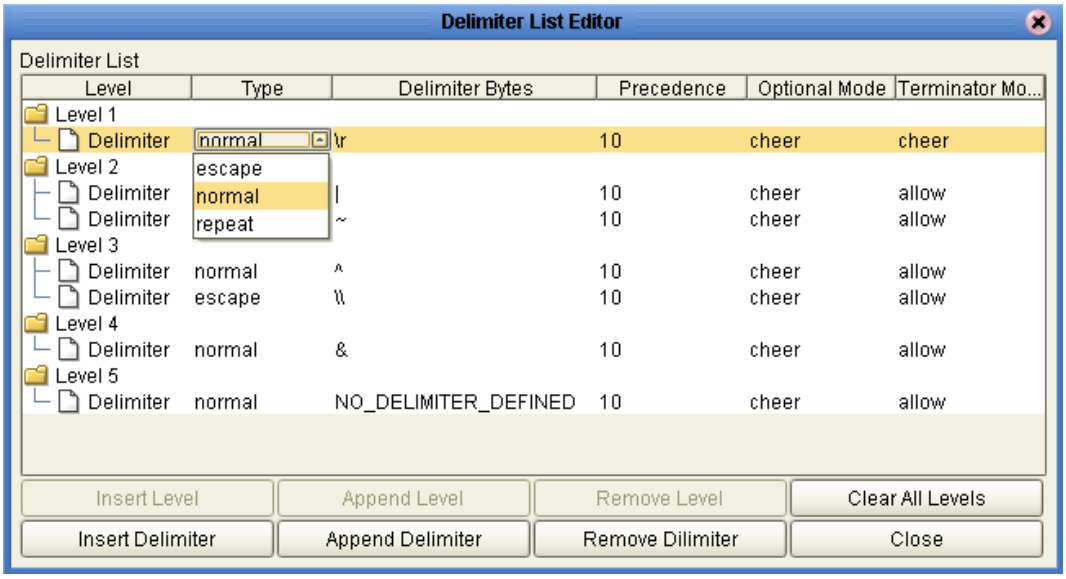
Level 1's delimiter applies to the two elements and field5, and level 2's delimiter applies to fields 1 through 4.

Delimiter lists can be much more complex than this very simple example. For instance, you can create multiple delimiters of different types at any given level, and you can specify a delimiter list on any node within the OTD—not only the root node as shown in the example. See **Modifying an OTD Using the OTD Editor** on page 123 for a description of the procedure for creating a Delimiter List.

Delimiter Properties

Delimiters are defined using the Delimiter List Editor (see Figure 38).

Figure 38 Delimiter List Editor



The Delimiter properties and values are displayed in Table 7.

Table 7 Delimiter Properties and Value Options

Property	Option	Description
Level		Child level beneath defining node.
Type	escape	Escape sequence.
	repeat	Array delimiter/separator.
	normal	Terminator.
Delimiter Bytes		Delimiter (single or multiple characters).
Precedence		See **Precedence** on page 134.
Optional Mode	never	Do not allow on input, do not emit on output (empty field between delimiters implies zero length data field).
	allow	Skip empty field if present; if absent, do not delimit on output.
	cheer	Skip empty field if present; if absent, do delimit on output.
	force	Require empty, delimited field on input; always delimit on output.

Table 7 Delimiter Properties and Value Options

Property	Option	Description
Terminator Mode	never	Do not allow on input, do not emit on output (pure separator).
	allow	Allow on input, do not emit on output.
	cheer	Allow on input, always emit on output.
	force	Require on input, always emit on output (pure terminator).

Type Property - *Escape* Option

An ***escape*** delimiter is simply a sequence that will be recognized *and ignored* during parsing. Its purpose is to allow the use of escape sequences to embed byte sequences in data that would otherwise be seen as delimiter occurrences.

For example, if there is a normal delimiter "**+**" at a given level, and we define an escape delimiter "**\+**", then **aaa+b\+c+ddd** will parse as three fields: **aaa**, **b\+c**, and **ddd**. If the escape delimiter were not defined, the sequence would then parse as four fields: **aaa**, **b**, **c**, and **ddd**.

If there is *only* an escape delimiter on a given level, however, it presents a *no delimiter defined* situation for **delim** and **array** nodes.

The following escape delimiters are allowed (see Table 8).

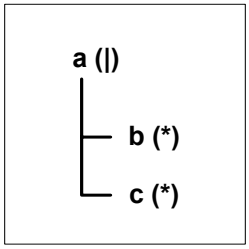
Table 8 Escape Delimiters

Delimiter	Description
\b	Backspace
\n	Newline
\r	Carriage return
\t	Tab
\0HH	Hexadecimal number

Terminator Mode Property

Consider the tree structure shown in Figure 39, where the node **a** has a pipe (|) as its delimiter, the sub-node **b** has a tilde (~) as its delimiter, and sub-node **c** has an asterisk (*) as its delimiter.

Figure 39 Terminal Type Property Example



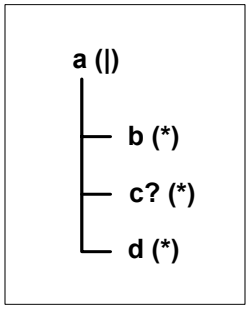
Option	Input	Output
never	**c**|	**c**|
allow	**c**| or **c***|	**c**|
cheer	**c**| or **c***|	**c***|
force	**c***|	**c***|

Optional Mode Property

Consider the tree structures shown in Figure 40 and Figure 41, where the node **a** has a pipe (|) as its delimiter, and the sub-nodes **b**, **c**, and **d** all have asterisks (*) as their delimiters.

- **Example 1:** Sub-node **c** is *optional*. (Sub-node **c** and sub-node **d** must have different values for the *match* parameter.)

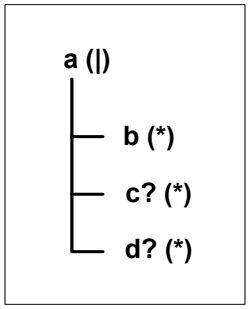
Figure 40 Optional Property (Example 1)



Option	Input	Output
never	**b*d|**	**b*d|**
allow	**b**d|**	**b*d|**
cheer	**b**d|**	**b**d|**
force	**b**d|**	**b**d|**

- **Example 2:** Both sub-node **c** and sub-node **d** are *optional*.

Figure 41 Optional Property (Example 2)



Option	Input	Output
never	**b|**	**b|**
allow	**b|**, **b*|**, or **b**|**	**b|**
cheer	**b|**, **b*|**, or **b**|**	**b**|**
force	**b**|**	**b**|**

Precedence

Precedence (see **Figure 38 on page 130**) indicates the priority of a certain delimiter, relative to the other delimiters. By default, all delimiters are at precedence 10, which means they are all considered the same; fixed fields are hard-coded at precedence 10. Delimiters on parent nodes are not considered when parsing the child fields; only the child's delimiter (or if it is a fixed field, its length).

Changing the precedence of a delimiter will cause them to be applied to the input data-stream in different ways. For example:

- *root node*
  - element (type delim, delimiter = "^", repeat)
    - field1 (type fixed, length = 5)
    - field2 (type fixed, length = 8, optional)

Although this will parse **'abcde12345678^zyxvuABCDEFGH'**, it will *not* parse the text **'abcde^zyxvuABCDEFGH'** even though the second fixed field is optional. The reason is that the element's delimiter is ignored within the fixed field because they have the same precedence. If you want the element's delimiter to be examined within the fixed field data, you must change its precedence, for example:

- *root node*
  - element (type delim, delimiter = "^", repeat, **precedence = 11**)
    - field1 (type fixed, length = 5)
    - field2 (type fixed, length = 8, optional)

This will successfully parse the text **'abcde^zyxvuABCDEFGH'**.

Node Management

The OTD Editor allows you to:

- **Add** nodes and elements to an OTD.
- **Delete** nodes and elements from an OTD.

  When a node is *deleted*, both the node and its associated *children* (data elements) are deleted.
- **Prune** nodes in an OTD.

  When a node is *pruned*, only its associated *children* (data elements) are deleted, while the node itself is preserved. Pruning can only be performed on nodes.

These commands are accessed from the node context menu.

6.4 Using the OTD Tester

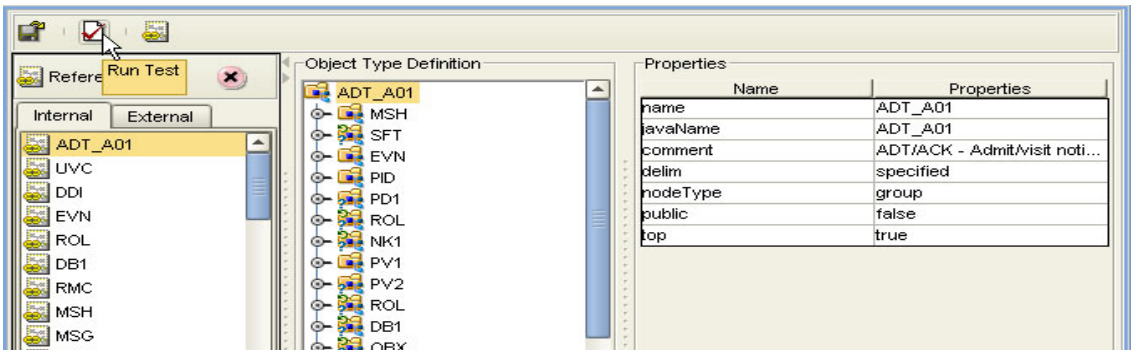
The OTD tester provides a facility to verify the correctness of an OTD, for example to:

- Prevent data errors at runtime.
- Verify that all required data elements are available.
- Verify that all used data formats are correct.

To use the OTD tester

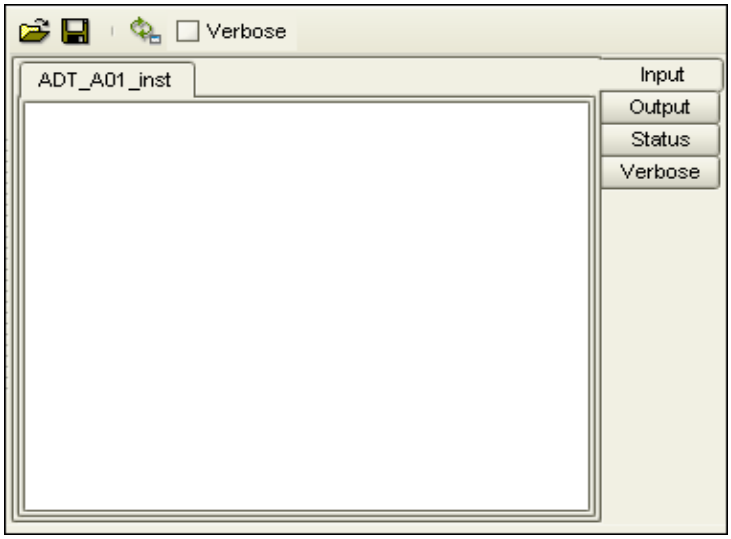
1. Open an OTD.
2. Click the **Tester** icon (see Figure 42).

Figure 42 OTD Tester



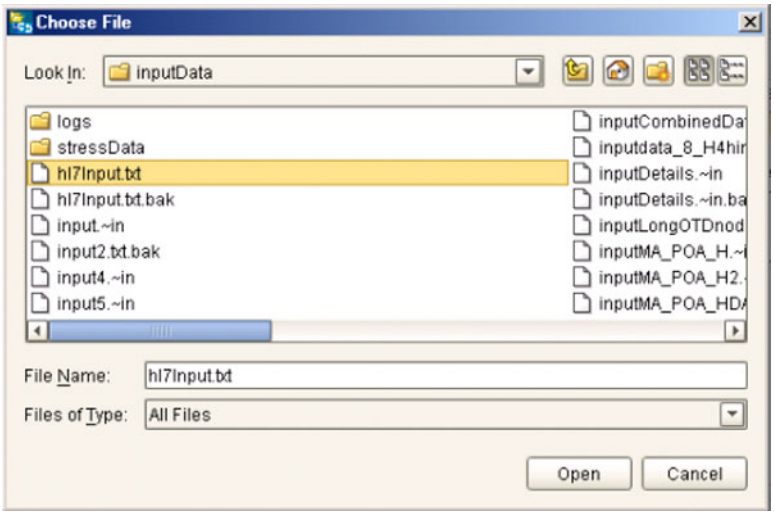
A test panel will appear below the OTD detail area of the editor. Note that there are four data display modes, selectable by tabs (see Figure 43). The Input tab is selected by default.

Figure 43 Test Panel Data Display



3 You can provide the input test data either by selecting a data file (see Figure 44), or by entering the data manually.

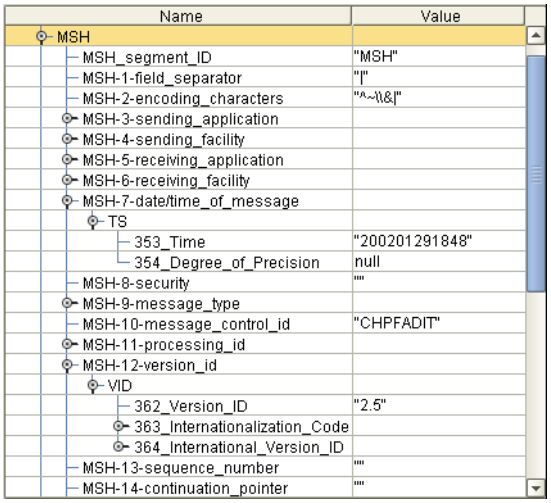
Figure 44 Select Data File



4 Click the **Run Tester** icon (green arrow) to test the selected OTD.

5 Verify the output by checking the values for each element for correctness (see Figure 45).

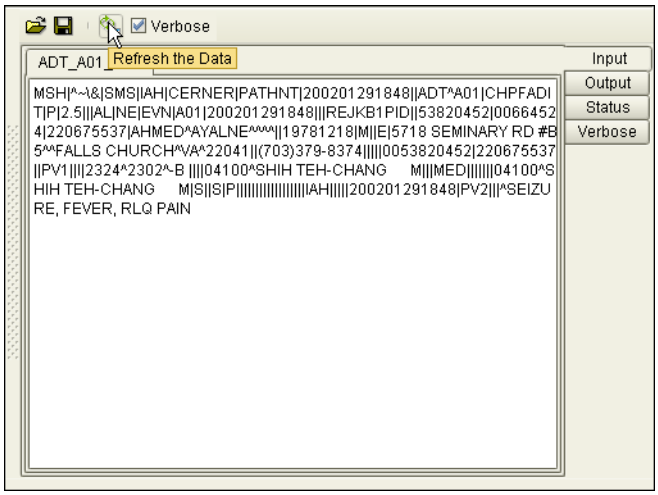
Figure 45 Object Elements and Values



6 Input test data can be saved to a file for re-use by selecting the **Input** data display and clicking the **Save** icon.

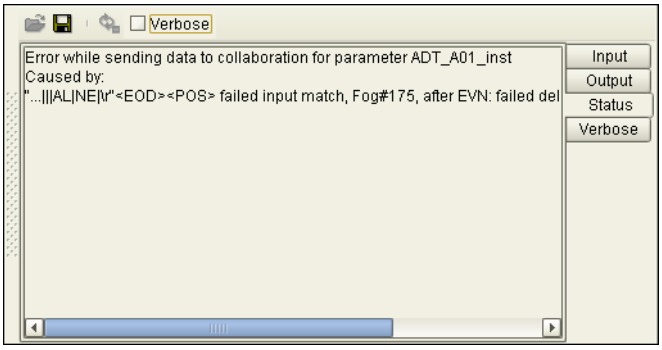
7 You can also change your test data in the Input data display, then re-test the OTD by clicking the **Refresh** icon (see Figure 46) to repopulate your OTD object elements with the new values.

Figure 46 Data Display: Refresh Icon



8 If there are errors in your input data, the **Status** data display is automatically invoked, showing the appropriate error messages (see Figure 47).

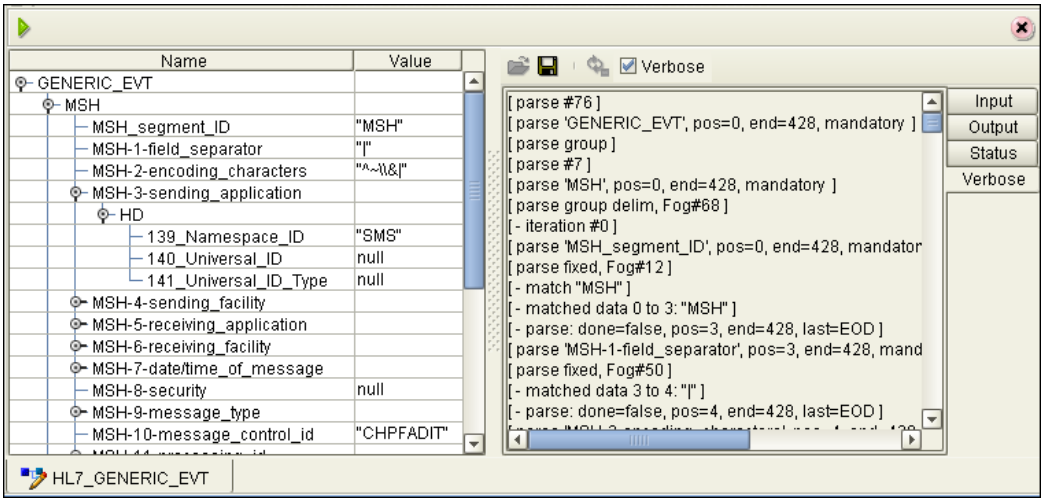
Figure 47 Status Data Display



Interpreting Failed Parse Messages Using the Verbose Option

9 The **Verbose** option provides a trace of parsing actions during the *unmarshal* process to aid in debugging the OTD structure. When this option is available, the OTD will activate the **Verbose** check box. Selecting the **Verbose** check box causes parsing information to appear on the **Verbose** data display (see **Figure 48 on page 138**). The format and content of the data display are OTD-specific.

Figure 48 Verbose Data Display



Chapter 7

Working with the TCP/IP HL7 eWay Projects

This chapter describes how to use the TCP/IP HL7 eWay template projects that are installed automatically as part of the TCP/IP HL7 eWay.

What's in This Chapter

7.1 TCP/IP HL7 Project Overview

The TCP/IP HL7 eWay projects provide predefined eWay components designed to be extended and modified for your specific project requirements. This chapter provides directions for completing the TCP/IP HL7 sample projects to test the eWay on your system and demonstrate how to implement the eWay.

7.1.1 Project Components

Projects are created using tools contained within the Enterprise Designer, and are deployed to specific Logical Hosts in specific Environments by means of Deployment Profiles. Components developed for use in one project can be used in another, and a project can internally reference another project. The components found in a typical project include:

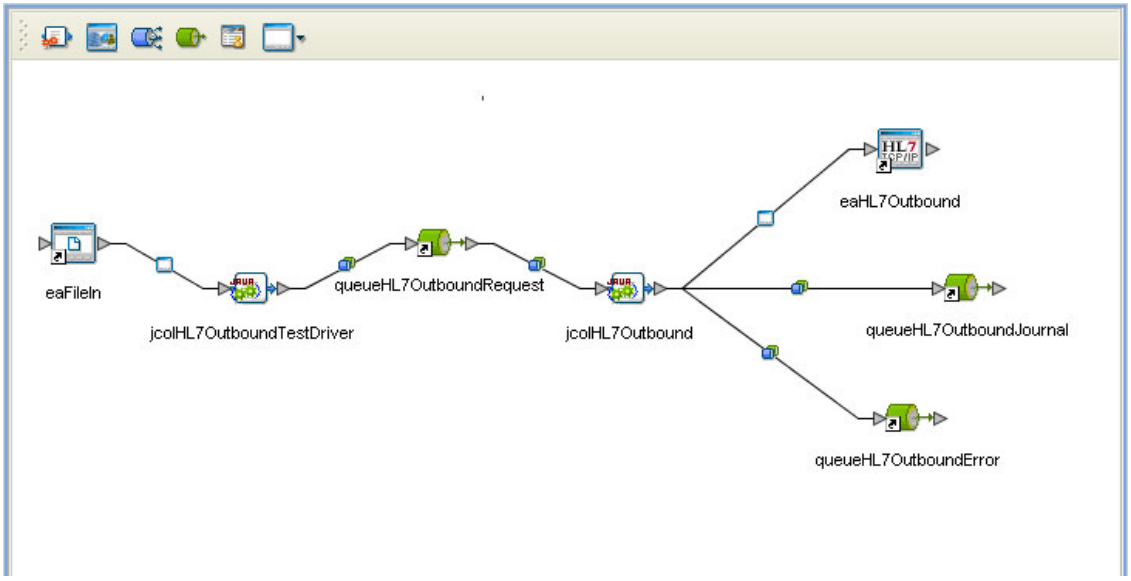
- **Services**: A service provides a framework for a process or a Collaboration.
- **External Applications**: External applications are logical representations of external software applications that are being integrated by the eGate system. These are linked to a Service by means of an eWay.
- **Schedulers**: A Scheduler allows a service to be performed at a prescribed interval.
- **Component Connections**: When you link two components on a Connectivity Map, the Enterprise Designer places either an eWay or JMS Client connection icon on the

link, depending upon the type of components you are linking. When an external application and a Collaboration are linked, the link contains an **eWay**. When a Service and a Message Destination (queue or topic) are linked, the link contains a **JMS Client Connection**.

- **Message Destinations**: A Message Destination is a container for stored data, and can follow either the topic or queue JMS model.
  - **Topic**: A topic is a message destination that conforms to the publish-and-subscribe messaging model (one to many).
  - **Queue**: A queue is a message destination that conforms to the point-to-point messaging model (one to one).

These components are graphically represented in a project's Connectivity Map: for example, the cmHL7Outbound Connectivity Map as displayed in Figure 49. This Connectivity Map does not contain a Scheduler or any Topics.

Figure 49 Connectivity Map - cmHL7Outbound



7.1.2 TCP/IP HL7 eWay Sample Projects

prjHL7Inbound

- **HL7Inbound Sample**: A standard inbound HL7 messaging project that receives HL7 messages from an external system, sends an acknowledgement of the message, provides sequence numbering, writes the HL7 message to a JMS data queue, and also writes the HL7 message and ACK to a JMS Journal queue. For more information see **Standard Inbound Message Mode Data Flow and Architecture** on page 25.
- **HL7ForwardMSG Inbound**: Inbound Forward Message mode is use with the Outbound Delayed ACK. Its purpose is to get a message from an outbound Forwarder and return and acknowledgement. For more information see **Inbound Receiver Message Mode** on page 25.

prjHL7Outbound

- **HL7Outbound Sample**: A standard outbound HL7 messaging project that receives the HL7 message from the JMS data queue, provides sequence numbering, sends the HL7 message to an external system, receives an acknowledgement from the external system, and writes the HL7 message and the ACK to a JMS Journal queue. For more information see **Outbound Standard Messaging Mode** on page 28.
- **HL7Forward Outbound**: Outbound Forward Message is use with the Delayed ACK. Its purpose is to get a message from a JMS queue and send it to an external system. No validation is preformed. For more information see **Outbound Forwarder Role** on page 30.
- **HL7OutboundDelayedAck**: Delayed Acknowledgement is similar to HL7Outbound, but the initial acknowledgement is received from the receiving system. After the Sender receives the first ACK, it waits for a second ACK that indicates that the message was received by the external HL7 system. For more information see **Outbound Delayed ACK Role** on page 29.

7.2 Completing a Project

To complete a TCP/IP HL7 eWay project do the following:

- Modify the Inbound or Outbound projects for your specific requirements.
- Create an Environment (see **Creating an Environment** on page 141).
- Configure the eWays for your specific requirements (see **Creating and Configuring the TCP/IP HL7 eWay** on page 38).
- Create a Deployment Profile (see **Creating and Activating the Deployment Profile** on page 145).
- Run the Projects.

7.3 Creating an Environment

Environments include the external systems, Logical Hosts, integration servers and message servers used by a project, and contain the configuration information for these components. Environments are created using the Enterprise Designer's Environment Explorer and Environment Editor.

Creating an Environment for the prjHL7Outbound Outbound Sample

To create the Environment for the **prjHL7Outbound** Outbound sample, do the following:

1. From the Enterprise Designer's Enterprise Explorer, click the **Environment Explorer** tab.

2 Right-click the Repository and select **New Environment**. A new Environment is added to the Environment Explorer tree.

3 Rename the new Environment to **HL7Outbound_Env**.

4 Right-click **HL7Outbound_Env** and select **New File External System**. Name the External System **FileExtSysIn** and select **Inbound File eWay** as the External System Type. Click **OK**. **FileExtSysIn** is added to the Environment Editor.

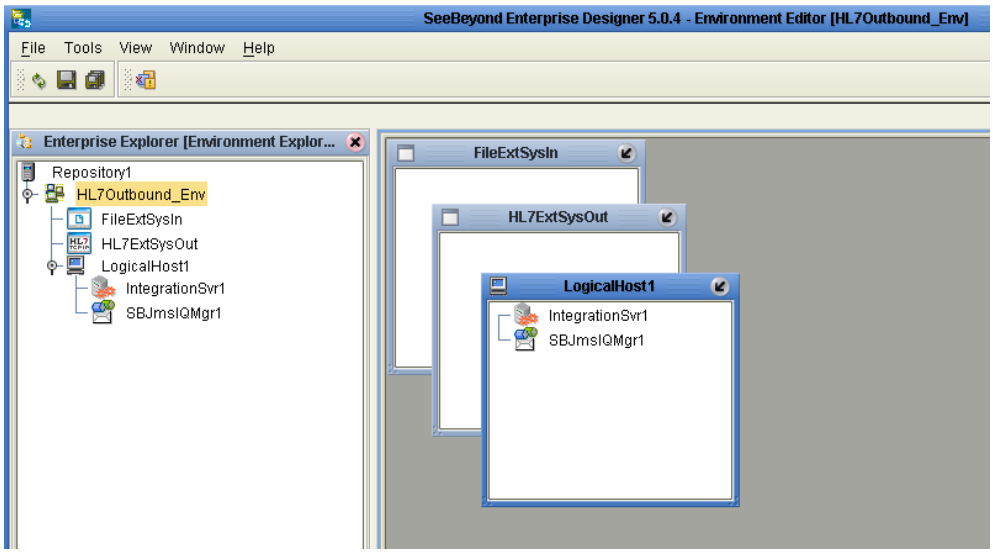
5 Right-click **HL7Outbound_Env** and select **New HL7 External System**. Name the External System **HL7ExtSysOut** and select **Outbound HL7 eWay** as the External System Type. Click **OK**. **HL7ExtSysOut** is added to the Environment Editor.

6 Right-click **HL7Outbound_Env** and select **New Logical Host**. **LogicalHost1** is added to the Environment Editor.

7 Right-click **LogicalHost1** and select **New SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under LogicalHost1.

8 Right-click **LogicalHost1** and select **New SeeBeyond JMS IQManager**. A new JMS IQ Manager (**SBJmsIQMgr1**) is added to the Environment Explorer tree under LogicalHost1. The Environment Explorer and Environment Editor appear as displayed in **Figure 50 on page 142**.

Figure 50 Environment Editor



9 Save your current changes.

7.3.1 Creating Environments for the HL7 Inbound and Outbound Samples

To create Environments for the remaining TCP/IP HL7 samples, follow the example given above to create the following:

HL7Inbound Inbound Sample

From the Environment Editor, create the **HL7Inbound_Env** Environment with the following components:

- New HL7 External System (Inbound) named **HL7ExtSysIn**
- New Logical Host
  - New SeeBeyond Integration Server
  - NewSeeBeyond JMS IQ Manager

HL7Inbound ForwardMSG Sample

From the Environment Editor, create the **HL7InboundForwardMSG_Env** Environment with the following components:

- New HL7 External System (Inbound) named **HL7ExtSysIn**
- New Logical Host
  - New SeeBeyond Integration Server
  - NewSeeBeyond JMS IQ Manager

HL7Outbound OutboundDelayedACK Sample

From the Environment Editor, create the **HL7OutboundDelayedACK_Env** Environment with the following components:

- New HL7 External System (Outbound) named **HL7ExtSysOut**
- New File External System (Inbound) named **FileExtSysIn**
- New Logical Host
  - New SeeBeyond Integration Server
  - NewSeeBeyond JMS IQ Manager

HL7Outbound Forward Sample

From the Environment Editor, create the **HL7OutboundForward_Env** Environment with the following components:

- New HL7 External System (Outbound) named **HL7ExtSysOut**
- New Logical Host
  - New SeeBeyond Integration Server
  - NewSeeBeyond JMS IQ Manager

7.4 Configuring the Properties

To run the inbound and outbound projects as samples, the following properties must be specified for you specific system. For additional information on the eWay properties, see **Creating and Configuring the TCP/IP HL7 eWay** on page 38.

HL7Inbound HL7 eWay Sample

The **HL7Inbound** sample contains one eWay, an Inbound TCP/IP HL7 eWay with both both the Connectivity Map and Environment properties. This project will work using the default settings.

HL7ForwardMSG Inbound HL7 eWay Sample

The **HL7ForwardMSG** Inbound sample contains one eWay, an Inbound TCP/IP HL7 eWay with both the Connectivity Map and Environment properties. This project will work using the default settings.

HL7Outbound HL7 eWay Sample

The **HL7Outbound** sample contains two eWays, an Inbound File eWay, and an Outbound TCP/IP HL7 eWay. The Outbound TCP/IP HL7 eWay has both Connectivity Map and Environment properties. The File eWay only contains Connectivity Map properties that require configuration.

The **eaFileIn** eWay will work with the default setting, except for the following that must be set for your specific system:

- **Connectivity Map Properties -> Parameter Settings -> Directory**
- **Connectivity Map Properties -> Parameter Settings -> Input file name**

The **HL7Outbound HL7** eWay will work with the default setting, except for the following that must be set for your specific system:

- **Environment Properties -> TCPIP Outbound Settings -> Host**
- **Environment Properties -> TCPIP Outbound Settings -> Port**

HL7Forward Outbound HL7 eWay Sample

The HL7Inbound sample contains one eWay, an Outbound TCP/IP HL7 eWay. The Outbound TCP/IP HL7 eWay has both the Connectivity Map and Environment properties. The File eWay only contains Connectivity Map properties that require configuration.

The **HL7Forward** Outbound HL7 eWay will work with the default setting, except for the following that must be set for your specific system:

- **Environment Properties -> TCPIP Outbound Settings -> Host**
- **Environment Properties -> TCPIP Outbound Settings -> Port**

HL7OutboundDelayedAck HL7 eWay Sample

The **HL7OutboundDelayedAck** sample contains two eWays, an Inbound File eWay, and an Outbound TCP/IP HL7 eWay. The Outbound TCP/IP HL7 eWay has both Connectivity Map and Environment properties. The File eWay only contains Connectivity Map properties that require configuration.

The **eaFileIn** eWay will work with the default setting, except for the following that must be set for your specific system:

- **Connectivity Map Properties -> Parameter Settings -> Directory**
- **Connectivity Map Properties -> Parameter Settings -> Input file name**

The **HL7OutboundDelayedAck** HL7 eWay will work with the default setting, except for the following that must be set for your specific system:

- **Environment Properties -> TCPIP Outbound Settings -> Host**
- **Environment Properties -> TCPIP Outbound Settings -> Port**

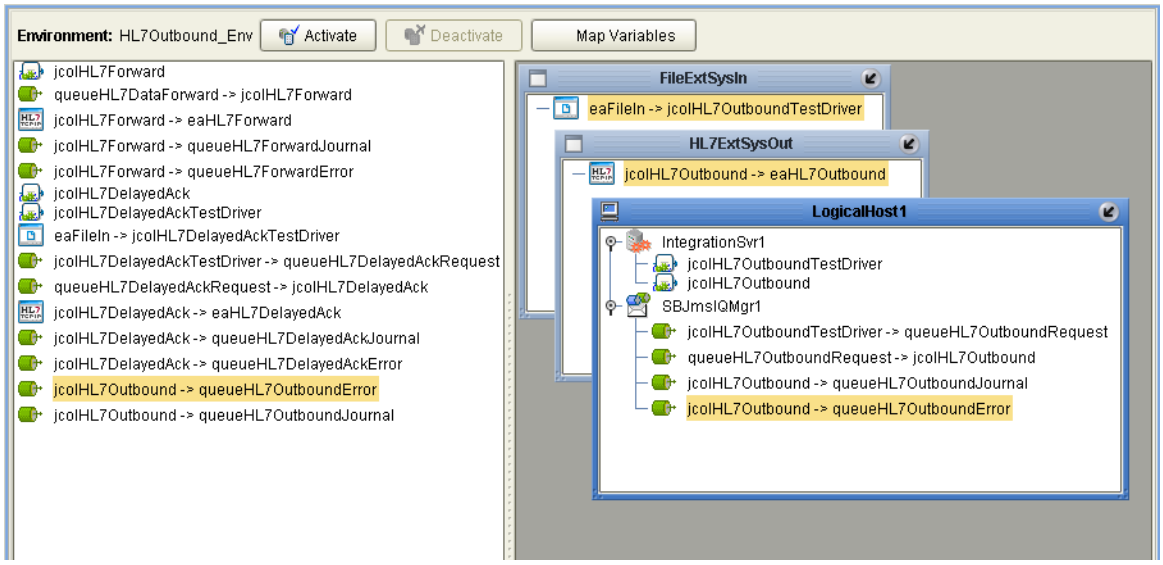
7.5 Creating and Activating the Deployment Profile

A Deployment Profile assigns Collaborations and message destinations to the Integration Server, and Topics and Queues to the IQ Manager. Deployment Profiles are created using the Deployment Editor.

Creating a Deployment for the prjHL7Outbound Outbound Sample

1. From the Enterprise Explorer's Project Explorer, right-click the project (**prjHL7Outbound**) and select **New** > **Deployment Profile**.
2. Enter a name for the Deployment Profile (for this sample **HL7Outbound_DP**). Make sure that the selected Environment is **HL7Outbound_Env**. Click **OK**. The Deployment Editor appears.
3. From the left pane of the Deployment Editor, drag the **eaFileIn** -> **jcolHL7OutboundTestDriver** (External Application) to the **FileExtSysIn** window.
4. From the left pane of the Deployment Editor, drag the **jcolHL7Outbound** -> **eaHL7Outbound** (External Application) to the **HL7ExtSysOut** window.
5. From the left pane of the Deployment Editor, drag the following Collaborations to **IntegrationSvr1** in the **LogicalHost1** window:
   - **jcolHL7OutboundTestDriver**
   - **jcolHL7Outbound**
6. Drag the following Queues to **SBJmsIQMgr1** in the **LogicalHost1** window (see Figure 51):
   - **jcolHL7OutboundTestDriver -> queueHL7OutboundRequest**
   - **queueHL7OutboundRequest -> jcolHL7Outbound**
   - **jcolHL7Outbound -> queueHL7OutboundJournal**
   - **jcolHL7Outbound -> queueHL7OutboundError**

Figure 51 HL7Outbound_DP Deployment Profile



Click **Activate**. When activation succeeds, save the changes to the Repository.

Creating Deployment Profiles for the HL7 Inbound and Outbound Samples

To create Deployment Profiles for the remaining TCP/IP HL7 samples, follow the prior example to create the following:

HL7Inbound Inbound Sample

Create the **HL7Inbound_DP** Deployment Profile using the **HL7Inbound_Env**. From the Deployment Editor, drag and drop the following components into the appropriate Environment windows:

- **eaHL7Inbound -> jcdHL7Inbound1** to the **HL7ExtSysIn** window.
- **jcdHL7inbound1** to **IntegrationSvr1** in the **LogicalHost1** window.
- **jcdHL7Inbound1 -> queueHL7DataInbound** to **SBJmsIQMgr1** in the **LogicalHost1** window.
- **jcdHL7Inbound1 -> queueHL7JournalInbound** to **SBJmsIQMgr1** in the **LogicalHost1** window.
- **jcdHL7Inbound1 -> queueHL7ErrorInbound** to **SBJmsIQMgr1** in the **LogicalHost1** window.

HL7Inbound ForwardMSG Sample

Create the **HL7InboundForwardMSG_DP** Deployment Profile using the **HL7InboundForwardMSG_Env**. From the Deployment Editor, drag and drop the following components into the appropriate Environment windows:

- **eaHL7ForwardMSG -> jcdHL7Inbound2** to the **HL7ExtSysIn** window.
- **jcdHL7inbound2** to **IntegrationSvr1** in the **LogicalHost1** window.

- **jcdHL7Inbound2 -> queueHL7DataForward** to **SBJmsIQMgr1** in the **LogicalHost1** window.
- **jcdHL7Inbound2 -> queueHL7JournalForward** to **SBJmsIQMgr1** in the **LogicalHost1** window.
- **jcdHL7Inbound2 -> queueHL7ErrorForward** to **SBJmsIQMgr1** in the **LogicalHost1** window.

HL7Outbound OutboundDelayedACK Sample

Create the **HL7OutboundDelayedACK_DP** Deployment Profile using the **HL7OutboundDelayedACK_Env**. From the Deployment Editor, drag and drop the following components into the appropriate Environment windows:

- **eaFileIn -> jcolHL7DelayedAckTestDriver** to the **FileExtSysIn** window.
- **jcolHL7DelayedAck -> eaHL7DelayedAck** to the **HL7ExtSysOut** window.
- **jcolHL7DelayedAckTestDriver** to **IntegrationSvr1** in the **LogicalHost1** window.
- **jcolHL7DelayedAck** to **IntegrationSvr1** in the **LogicalHost1** window.
- **jcolHL7DelayedAckTestDriver -> queueHL7DelayedAckRequest** to **SBJmsIQMgr1** in the **LogicalHost1** window.
- **queueHL7DelayedAckRequest -> jcolHL7DelayedAck** to **SBJmsIQMgr1** in the **LogicalHost1** window.
- **jcolHL7DelayedAck -> queueHL7DelayedAckJournal** to **SBJmsIQMgr1** in the **LogicalHost1** window.
- **jcolHL7DelayedAck -> queueHL7DelayedAckError** to **SBJmsIQMgr1** in the **LogicalHost1** window.

HL7Outbound Forward Sample

Create the **HL7OutboundForward_DP** Deployment Profile using the **HL7OutboundForward_Env**. From the Deployment Editor, drag and drop the following components into the appropriate Environment windows:

- **jcolHL7Forward -> eaHL7Forward** to the **HL7ExtSysOut** window.
- **jcolHL7Forward** to **IntegrationSvr1** in the **LogicalHost1** window.
- **queueHL7DataForward -> jcolHL7Forward** to **SBJmsIQMgr1** in the **LogicalHost1** window.
- **jcolHL7Forward -> queueHL7ForwardJournal** to **SBJmsIQMgr1** in the **LogicalHost1** window.
- **jcolHL7Forward -> queueHL7ForwardError** to **SBJmsIQMgr1** in the **LogicalHost1** window.

7.6 Running a Project

The following directions assume that the Enterprise Designer was downloaded to **C:\ican50**. If this is not the case, replace that location in the following directions with the appropriate location.

1. From the Enterprise Manager Downloads tab, download **Logical Host - for win32**.
2. Extract the file to the **ican50\logicalhost1** directory. You must specify the **logicalhost1** directory for it to be created.
3. Navigate to **C:\ican50\logicalhost1\logicalhost\bootstrap\config** directory and open the **logical-host.properties** file using a text editor.
4. Enter the following information in the appropriate fields:
   - Logical Host root directory: **ican50\logicalhost1\logicalhost**
   - Repository URL: **http://localhost:*port number/repository name***
   - Repository user name and password: ***Your user name and password***
   - Logical Host Environment name: (For example, **HL7Outbound_Env**)
   - Logical Host name: **logicalhost1**

   Save your changes to **logical-host.properties** and close the file.
5. Run the **bootstrap.bat** file in the **ican50\logicalhost1\logicalhost\bootstrap\bin** directory.

Testing the Project

1. To test a sample project, copy your sample input message to the input directory designated in the project's File eWay properties.
2. From the ICAN Monitor's Environment tab, check the project's IQ Manager to access the JMS topics and queues for Journal entries, error messages, and so forth.

For more information on the ICAN Monitor, see the *eGate Integrator System Administration Guide*.

For more information on how to run a project, see the *eGate Integrator Tutorial* and *eGate Integrator User's Guide*.

Chapter 8

Methods for the TCP/IP HL7 eWay

The TCP/IP HL7 eWay contains a number of Java methods that have been exposed to extend the functionality of the eWay. To view a list of the available methods see the TCP/IP HL7 eWay Javadoc.

What's in This Chapter

8.1 TCP/IP HL7 eWay Methods

The TCP/IP HL7 eWay classes expose a number of methods which are represented in the TCP/IP HL7 eWay OTDs. These classes are:

- Interface HL7AppMessage
- Interface HL7ClientApplication
- Interface HL7ServerApplication

8.1.1 TCP/IP HL7 eWay Javadoc

The TCP/IP HL7 eWay Javadoc is an API document in HTML format that provides information on the various methods available with the TCP/IP HL7 eWay. The Javadoc is accessed, along with the **TCP/IP HL7 eWay Intelligent Adapter User's Guide**, by selecting and uploading **HL7eWayDocs.sar** from the **ADMIN** tab of the Enterprise Manager. The Javadoc can then be accessed from the **Documentation** tab of the Enterprise Manager.

To access the full Javadoc, extract the Javadoc to an easily accessible folder, and double click the **index.html** file.

Index

G

H

I

J

L

M

N

O

P

R

S

T

V