Working with TCP/IP HL7 Collaborations

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 820–5508–10
June 2008

090925@22749

Contents

CHAPTER 1

Working with TCP/IP HL7 Collaborations

The following sections provide instructions on the method to Work with TCP/IP HL7 Collaborations. If you have any questions or problems, see the Java CAPS web site at `http://goldstar.stc.com/support`.

This chapter covers the following topics:

TCP/IP HL7 Adapter Collaborations Overview

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

TCP/IP HL7 Adapter Task Overview

The task included in this section allows you to perform the following,

- "To Export a Project" on page 18
- "To Create Copies of an HL7 V2 Collaborations" on page 23
- "To Add HL7 V2 OTD to an Existing Collaboration" on page 27
- "To Add HL7 V3 OTD to an Existing Collaboration" on page 59

TCP/IP HL7 V2 Adapter Collaborations

The TCP/IP HL7 V2 adapter includes one inbound Collaboration (**jcdHL7Inbound**) and one outbound Collaboration (**jcdHL7Outbound**), provided within the sample Projects for inbound and outbound HL7 V2 messaging. These template/sample Projects, **prjHL7Inbound** and **prjHL7Outbound** are downloaded from the Sun Java[TM] Composite Application Platform Suite Installer. These Collaborations are designed to work **as is** for HL7 V2 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 V2 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 application 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:** 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 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 V2 Inbound Collaboration publishes received data as a Byte message in JMS using the **sendBytes()** method. However, the HL7 V2 Outbound Collaboration expects a Text message from JMS. The adapter is not designed for the HL7 V2 Outbound Collaboration to subscribe to a JMS data queue created by the HL7 V2 Inbound Collaboration directly. HL7 V2 Inbound and Outbound Collaborations are designed to communicate through an HL7 V2 TCP/IP adapter connection.

Inbound HL7 V2 Collaboration Overview

The Inbound HL7 V2 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 HL7 V2 Collaboration - Part 1

The inbound Collaboration is triggered by an HL7 V2 message received from an external system or an outbound HL7 V2 Client. The Collaboration calls the HL7 V2 User Collaboration Rule by executing **receive()**.

The receive method is the entry point to the HL7 V2 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 V2 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 adapter sends a NAK.

The validated HL7 V2 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 MSH segment. The ACKs level of acknowledgement is set to A (acknowledgement is sent after the message is successfully processed) , or C (acknowledgement is sent when the message is successfully received).

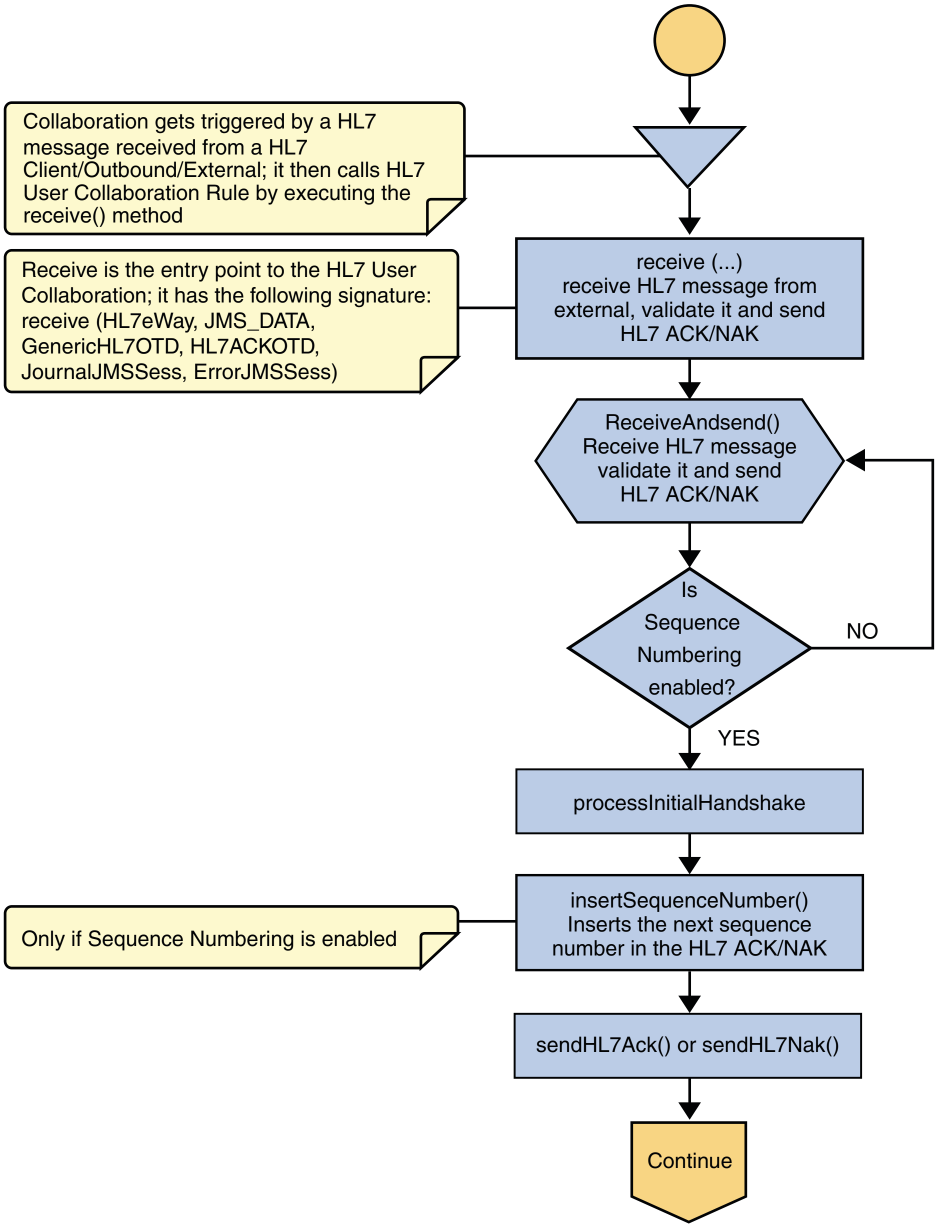

Collaboration gets triggered by a HL7 message received from a HL7 Client/Outbound/External; it then calls HL7 User Collaboration Rule by executing the receive() method
Receive is the entry point to the HL7 User Collaboration; it has the following signature: receive (HL7eWay, JMS_DATA, GenericHL7OTD, HL7ACKOTD, JournalJMSSess, ErrorJMSSess)
receive (...) receive HL7 message from external, validate it and send HL7 ACK/NAK
ReceiveAndsend() Receive HL7 message validate it and send HL7 ACK/NAK
Is Sequence Numbering enabled?
NO
YES
processInitialHandshake
insertSequenceNumber() Inserts the next sequence number in the HL7 ACK/NAK
Only if Sequence Numbering is enabled
sendHL7Ack() or sendHL7Nak()
Continue

FIGURE 1–1 Inbound HL7 V2 Collaboration - Part 1

Inbound HL7 V2 Collaboration - Part 2

The Collaboration receives the HL7 V2 message from the external using **receiveHL7message()**. If an exception occurs due to incomplete data, and the adapter 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, **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 V2 message.

If the HL7 V2 message does not pass validation, the Collaboration calls **makeNak()** and **sendHL7Nak()** to create and send the NAK to the external. The HL7 V2 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.

If the HL7 V2 message passes validation, the Collaboration calls **makeAck()** and **sendHL7Ack()** to create and send the ACK to the external.

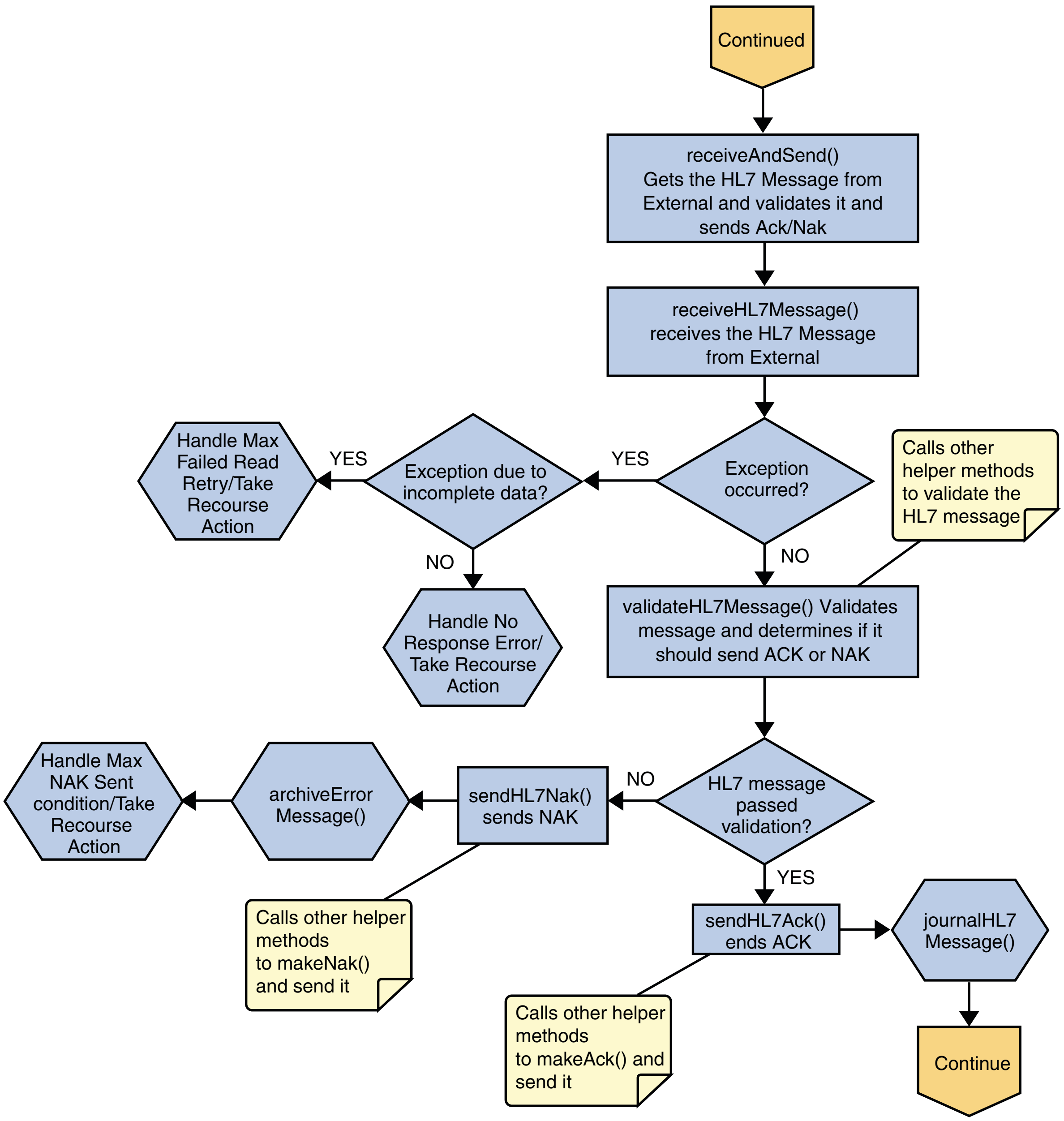

Continued
receiveAndSend() Gets the HL7 Message from External and validates it and sends Ack/Nak
receiveHL7Message() receives the HL7 Message from External
Exception occurred?
YES
Exception due to incomplete data?
YES
Handle Max Failed Read Retry/Take Recourse Action
NO
Handle No Response Error/ Take Recourse Action
NO
Calls other helper methods to validate the HL7 message
validateHL7Message() Validates message and determines if it should send ACK or NAK
HL7 message passed validation?
NO
sendHL7Nak() sends NAK
archiveError Message()
Handle Max NAK Sent condition/Take Recourse Action
Calls other helper methods to makeNak() and send it
YES
sendHL7Ack() ends ACK
journalHL7 Message()
Calls other helper methods to makeAck() and send it
Continue

FIGURE 1–2 Inbound HL7 V2 Collaboration - Part 2

Inbound HL7 V2 Collaboration - Part 3

After the ACK is sent, the HL7 V2 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.

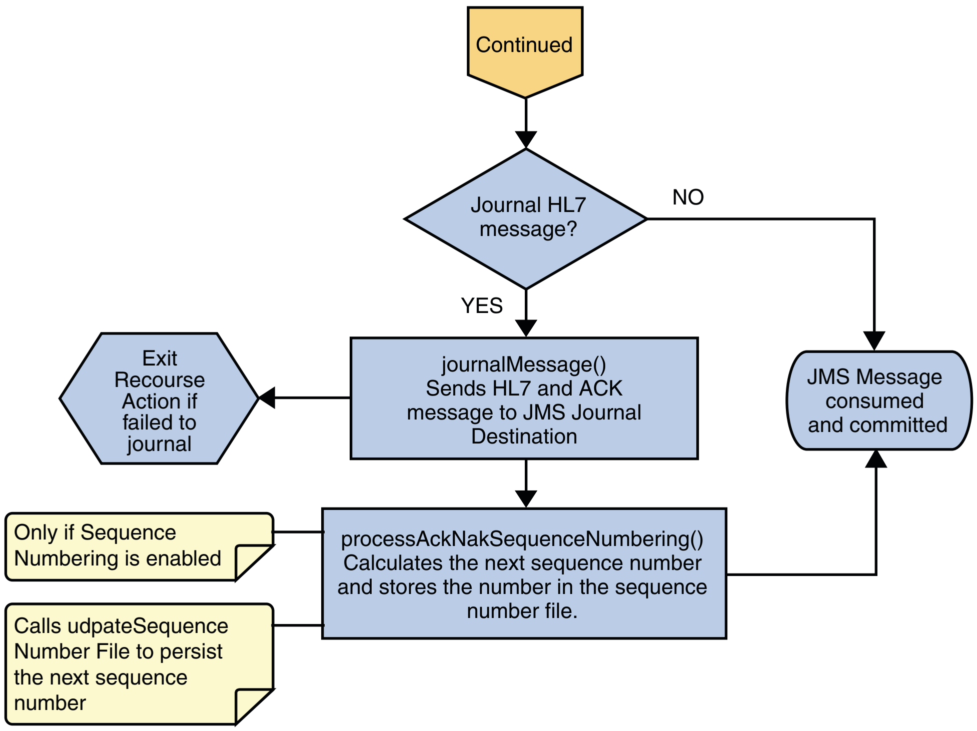

Continued
Journal HL7 message?
NO
YES
Exit Recourse Action if failed to journal
journalMessage() Sends HL7 and ACK message to JMS Journal Destination
JMS Message consumed and committed
Only if Sequence Numbering is enabled
processAckNakSequenceNumbering() Calculates the next sequence number and stores the number in the sequence number file.
Calls udpateSequence Number File to persist the next sequence number

FIGURE 1–3 Inbound HL7 V2 Collaboration - Part 3

Outbound HL7 V2 Collaboration Overview

The Outbound HL7 V2 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 V2 messages, so the data flow that feeds this Collaboration must ensure this.

Outbound HL7 V2 Collaboration - Part 1

The Collaboration is triggered by a JMS HL7 V2 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 V2 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 V2 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 V2 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 V2 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 V2 message to the external using the HL7 adapter OTD.

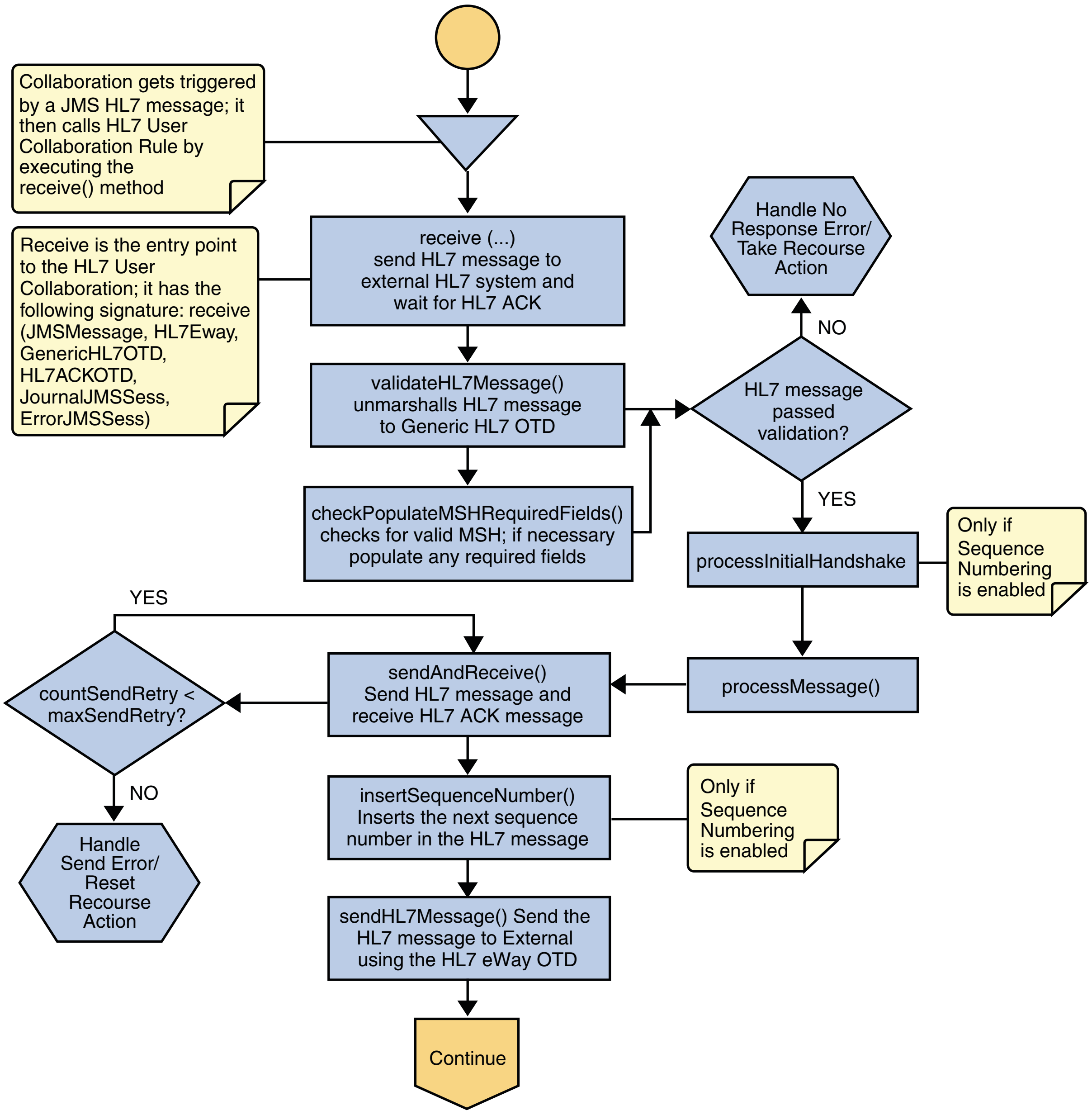

Collaboration gets triggered by a JMS HL7 message; it then calls HL7 User Collaboration Rule by executing the receive() method
Receive is the entry point to the HL7 User Collaboration; it has the following signature: receive (JMSMessage, HL7Eway, GenericHL7OTD, HL7ACKOTD, JournalJMSSess, ErrorJMSSess)
receive (...) send HL7 message to external HL7 system and wait for HL7 ACK
validateHL7Message() unmarshalls HL7 message to Generic HL7 OTD
checkPopulateMSHRequiredFields() checks for valid MSH; if necessary populate any required fields
HL7 message passed validation?
NO
Handle No Response Error/ Take Recourse Action
YES
processInitialHandshake
Only if Sequence Numbering is enabled
processMessage()
sendAndReceive() Send HL7 message and receive HL7 ACK message
countSendRetry < maxSendRetry?
YES
NO
Handle Send Error/ Reset Recourse Action
insertSequenceNumber() Inserts the next sequence number in the HL7 message
Only if Sequence Numbering is enabled
sendHL7Message() Send the HL7 message to External using the HL7 eWay OTD
Continue

FIGURE 1–4 Outbound HL7 V2 Collaboration - Part 1

Outbound HL7 V2 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 adapter 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

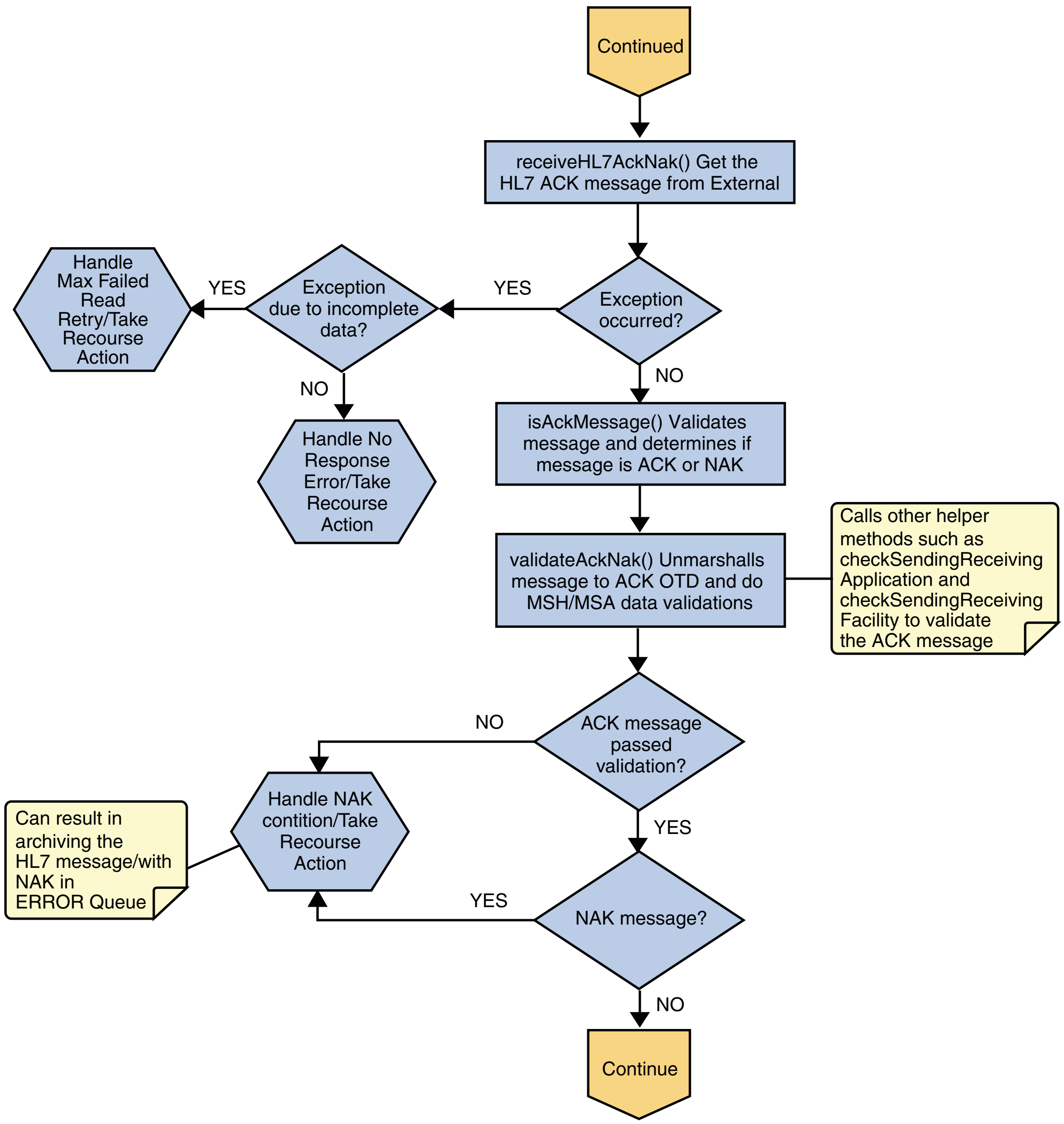

Continued
receiveHL7AckNak() Get the HL7 ACK message from External
Exception occurred?
YES
Exception due to incomplete data?
YES
Handle Max Failed Read Retry/Take Recourse Action
NO
Handle No Response Error/Take Recourse Action
NO
isAckMessage() Validates message and determines if message is ACK or NAK
validateAckNak() Unmarshalls message to ACK OTD and do MSH/MSA data validations
Calls other helper methods such as checkSendingReceiving Application and checkSendingReceiving Facility to validate the ACK message
ACK message passed validation?
NO
Handle NAK contition/Take Recourse Action
Can result in archiving the HL7 message/with NAK in ERROR Queue
YES
NAK message?
YES
NO
Continue

FIGURE 1–5 Outbound HL7 V2 Collaboration - Part 2

Outbound HL7 V2 Collaboration - Part 3

If the ACK message validates, the HL7 V2 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.

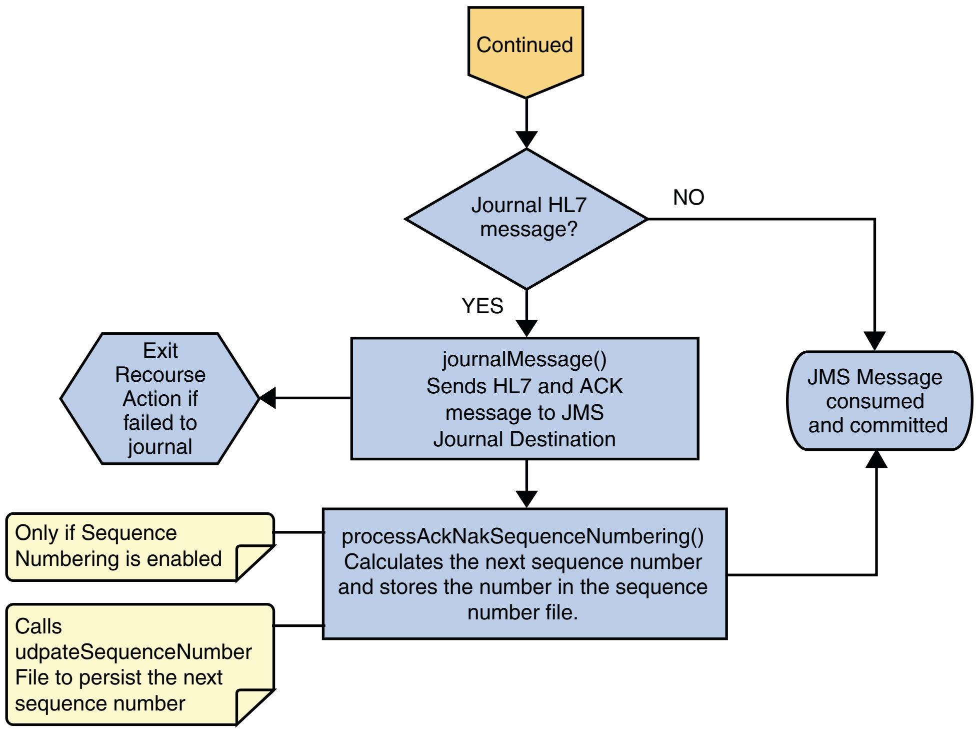

Continued
Journal HL7 message?
NO
YES
Exit Recourse Action if failed to journal
journalMessage() Sends HL7 and ACK message to JMS Journal Destination
JMS Message consumed and committed
Only if Sequence Numbering is enabled
processAckNakSequenceNumbering() Calculates the next sequence number and stores the number in the sequence number file.
Calls udpateSequenceNumber File to persist the next sequence number

FIGURE 1–6 Outbound HL7 V2 Collaboration - Part 3

HL7 V2 Outbound Test Collaboration

In addition to the Inbound and Outbound HL7 V2 Collaborations, an Outbound test Collaboration, **jcdHL7OutboundTestDriver**, is provided to test the HL7 V2 Outbound and HL7 V2 Outbound Delayed ACK samples.

The **jcdHL7OutboundTestDriver** Collaboration simply picks up HL7 V2 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 V2 messages processed per minute.

Limitations of Version 2.x

Limitations of Version 2.x can be categorized as follows:

- Implicit information model, not explicit
- Need for controlled vocabularies
- Limited to a single encoding syntax
- No explicit support for object technologies
- No explicit support for security functions
- Optionality is ubiquitous and troublesome

Creating a Copy of an HL7 V2 Project

It is recommended that you retain the **prjHL7Inbound** and **prjHL7Outbound** Projects as they are 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.

▼ To Export a Project

To export a Project or Environment using NetBeans IDE, perform the following:

1 **Start Repository.**

Follow this path to start the repository **root:\JavaCAPS6\start_repository**

2 **Start NetBeans IDE.**

Follow this path to start the NetBeans IDE: **root:\JavaCAPS\start_netbeans**

3 **Click on Tools menu, select CAPS Repository. Then, click Connect....**

This displays the Connect to CAPS Repository.

You can follow either of these steps to Connect to the Repository.

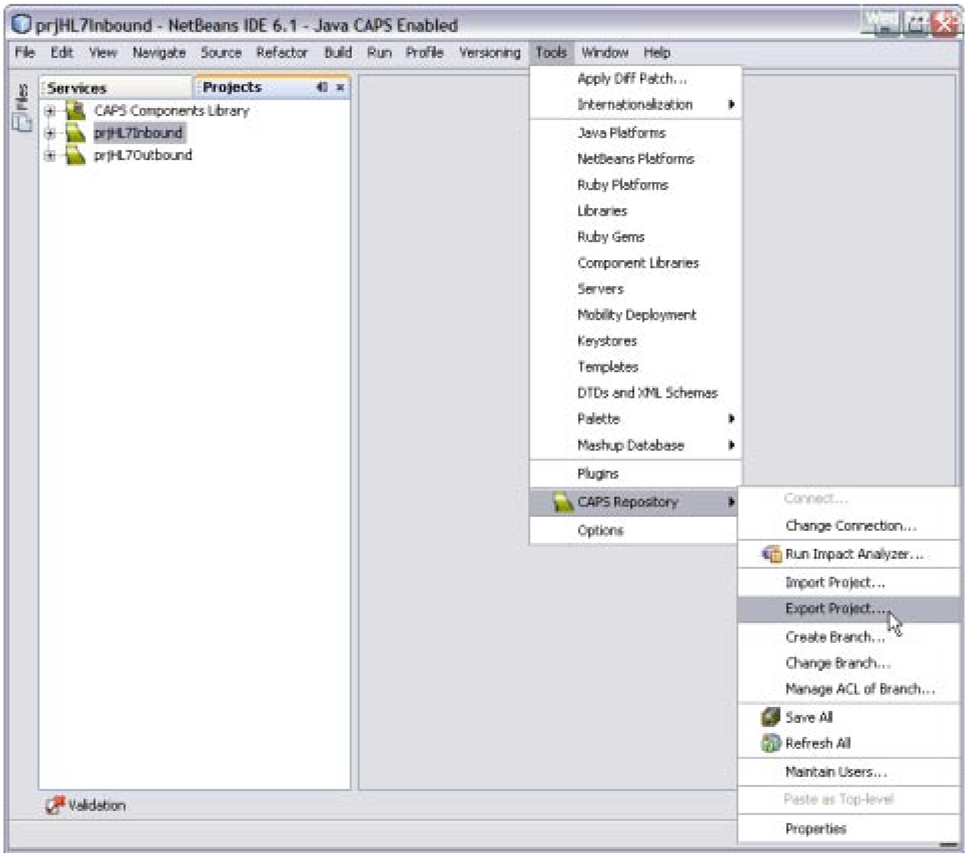
- From the Services tab, right-click CAPS Environment to Connect to CAPS Repository.....
- Click on the *icon* at the bottom right hand of the screen to connect to the repository. Here, the Environments are defined in the Repository.

Note – When connected the icon turns from red to green.

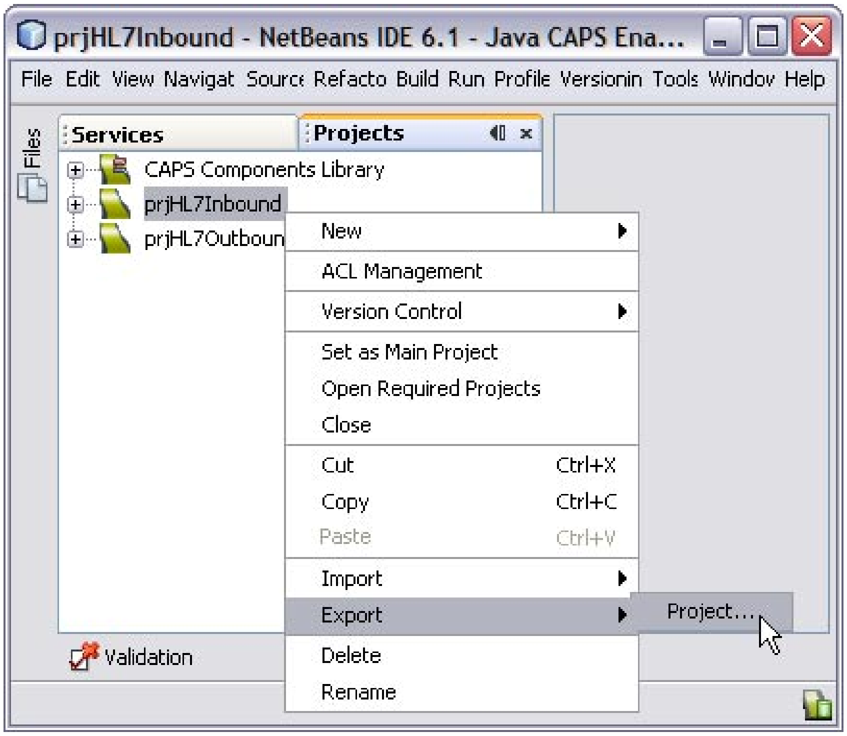
Click Connect....

4 **Select Export Project... from either of the following methods:**

- Click Tools menu and select CAPS Repository from the drop-down menu. Choose Export Project...



- Click Projects tab and right-click the Project you wish to export (for example, prjHL7Inbound). Select Export from the drop—down menu. Select Project...



The Export Manager dialog box appears.

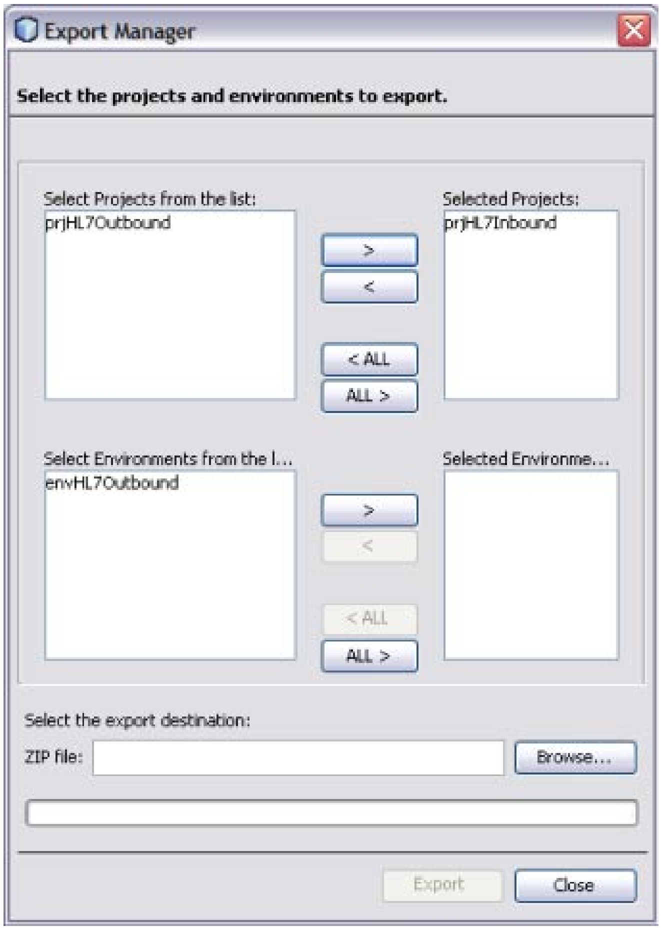

Export Manager
Select the projects and environments to export.
Select Projects from the list:
prjHL7Outbound
Selected Projects:
prjHL7Inbound
>
<
< ALL
ALL >
Select Environments from the l...
envHL7Outbound
Selected Environme...
Select the export destination:
ZIP file:
Browse...
Export
Close

FIGURE 1–7 Export Manager

5 **Select the Projects and Environments to export.**

The selected Project appears in the Selected Projects: pane of the Export Manager.

6 **Select the Project or the Environment from the left pane and click the arrow > button to add the Project to the Selected Projects: pane.**

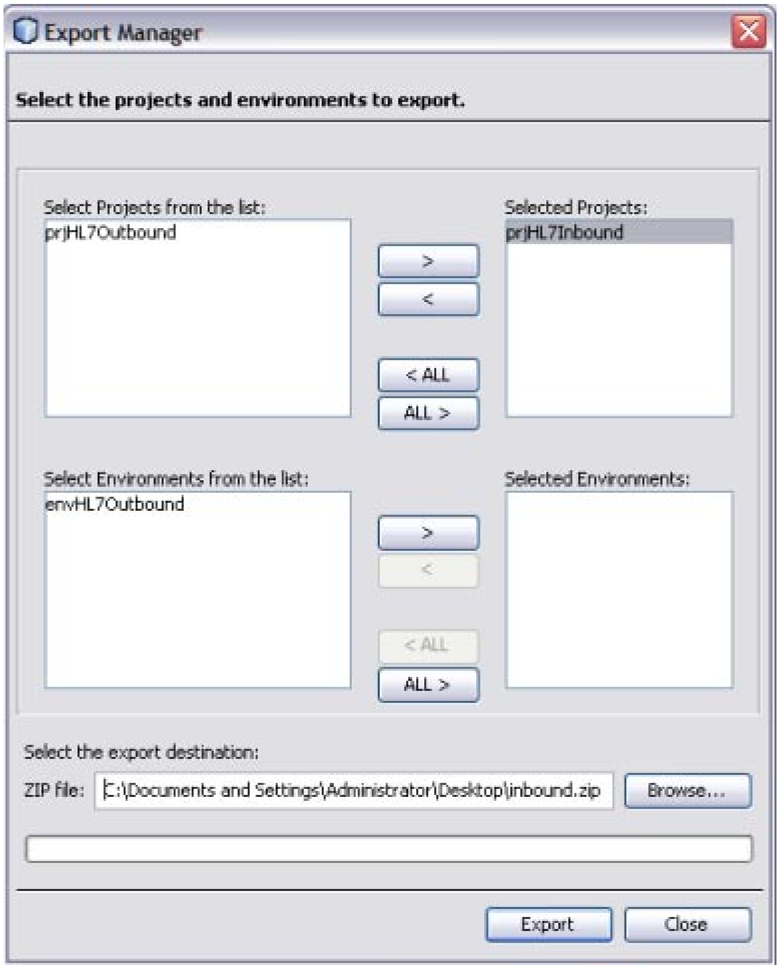
Note – Click the arrow <, if you want to move the selected projects from the right pane to the left pane.

Click the arrow **ALL>** to move all the projects from the left pane to the right pane.

Click the arrow **<ALL** to move all the projects from the right pane to the left pane

---

7 **Click Browse and select an appropriate directory to save the exported Project.**


Export Manager
Select the projects and environments to export.
Select Projects from the list:
prjHL7Outbound
Selected Projects:
prjHL7Inbound
>
<
< ALL
ALL >
Select Environments from the list:
envHL7Outbound
Selected Environments:
Select the export destination:
Export
Close
Browse...

ZIP file: C:\Documents and Settings\Administrator\Desktop\inbound.zip

FIGURE 1–8 Export Manager — Browse Option

8 **Click Export button.**

The Project is zipped and saved to the chosen directory.

Note – The NetBeans IDE 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.

More Information

To Import a Project

To Import a Project, follow the steps from "Installing the TCP/IP HL7 Adapter and Sample Projects" in *Sun Adapter for TCP/IP HL7 Tutorial*.

Customizing Predefined Collaborations for HL7 V2

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. 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.

Two ways are involved in creating copies of the HL7 Collaborations

- "Creating Copies of an HL7 V2 Collaborations" on page 23
- "Adding HL7 V2 OTD to an Existing Collaboration" on page 27

Creating Copies of an HL7 V2 Collaborations

Production Projects contain several Collaborations. To create multiple copies of the HL7 V2 Collaborations for your Project, copy and paste the original Collaborations into your project as follows:

▼ To Create Copies of an HL7 V2 Collaborations

1 **Right-click the Collaboration you want to copy and select Copy from the drop—down menu.**

For Example, from the **prjHL7Inbound** Project, right-click and copy **jcdHL7Inbound**.

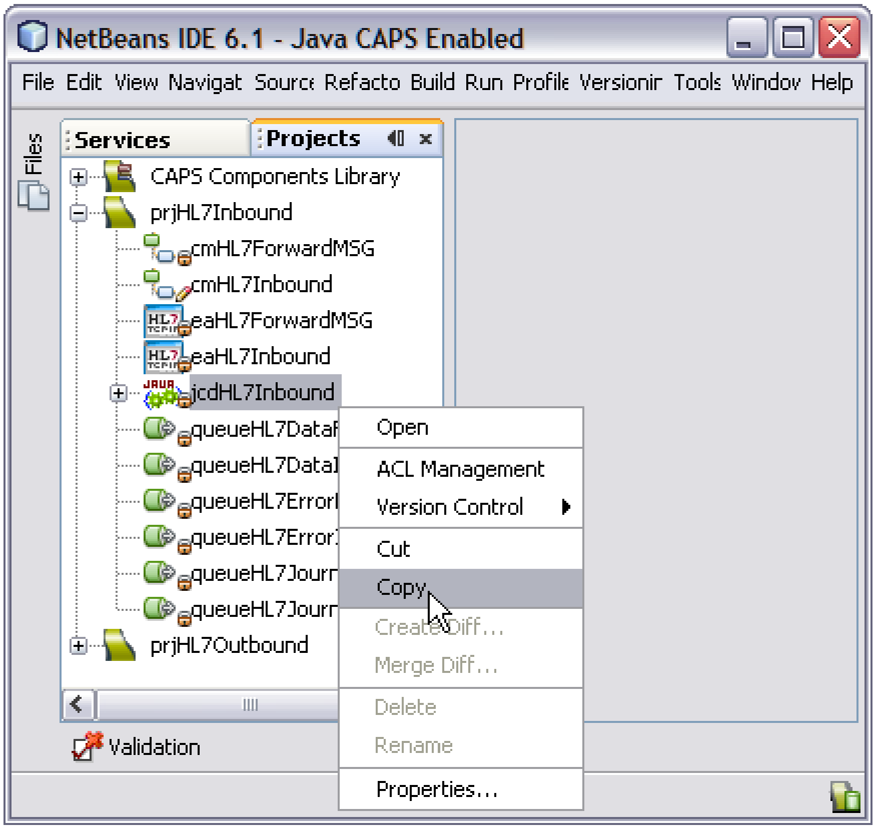


FIGURE 1–9 Collaboration — Copy Project

2 Right-click the Project to which you are copying the Collaboration, and select Paste from the drop—down menu.

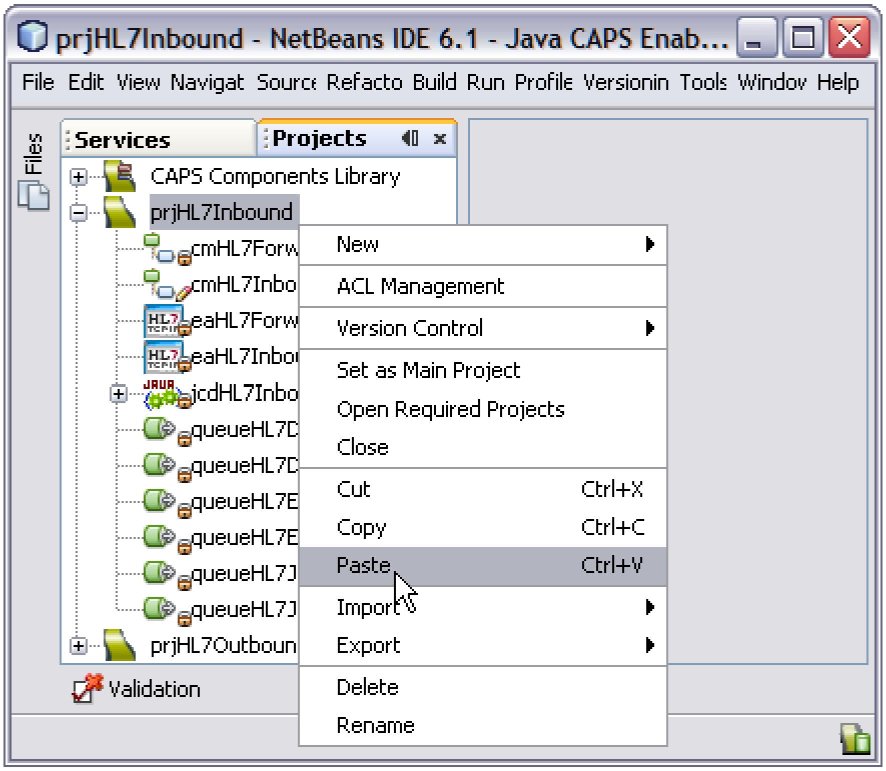


FIGURE 1–10 Collaboration — Paste Project

Note – A numbered copy of the Collaboration appears in your Project. The Collaboration can be copied within the Project or to another Project.

For Example, **jcdHL7Inbound_1**. Repeat this step for additional copies of the Collaboration.

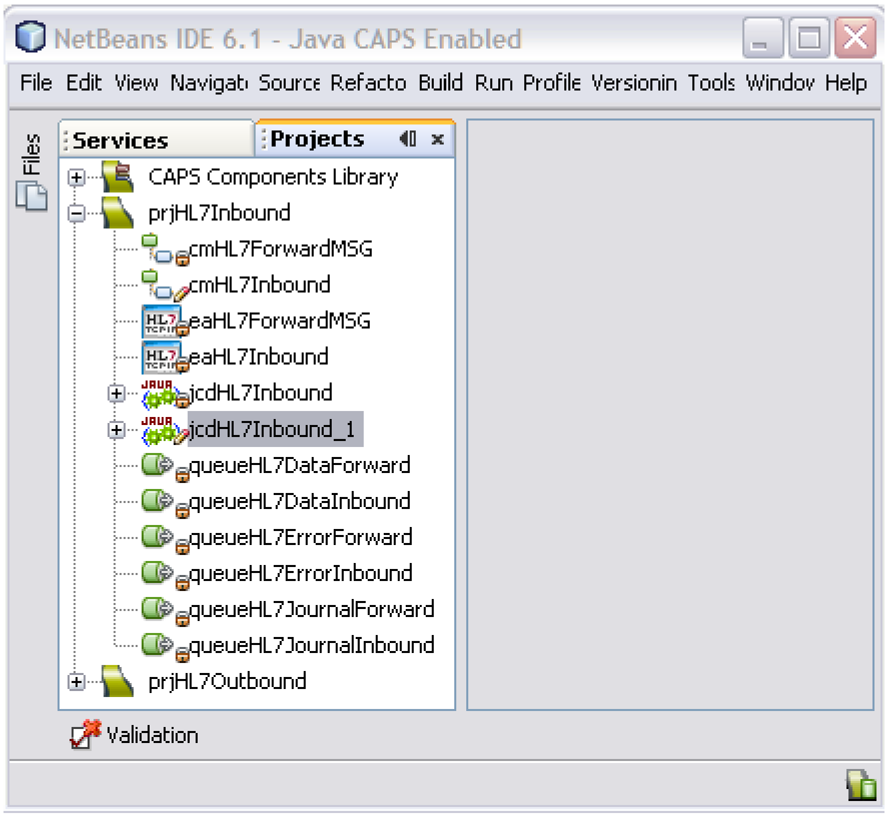


FIGURE 1–11 Collaboration — Copied to Project Tree

3 **Drag and drop each new Collaboration from the Projects Explorer tree to add the new Collaboration to the existing Connectivity Map (cmHL7Inbound) to the Connectivity Map Editor.**

For Example, **cmHL7Inbound_jcdHL7Inbound_11**. The new Collaborations can now be associated (bind) with the Project's components.

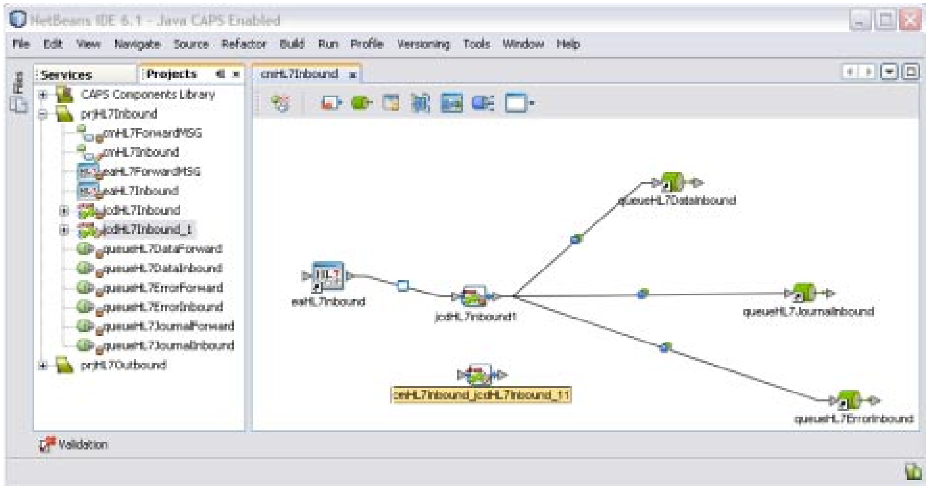


FIGURE 1–12 Connectivity Map — Newly Added Collaborations

Adding HL7 V2 OTD to an Existing Collaboration

In some cases, a specific HL7 V2 message or messages may need to be added to the Collaboration. Perform the following to add a HL7 V2 OTD to an existing Collaboration.

▼ To Add HL7 V2 OTD to an Existing Collaboration

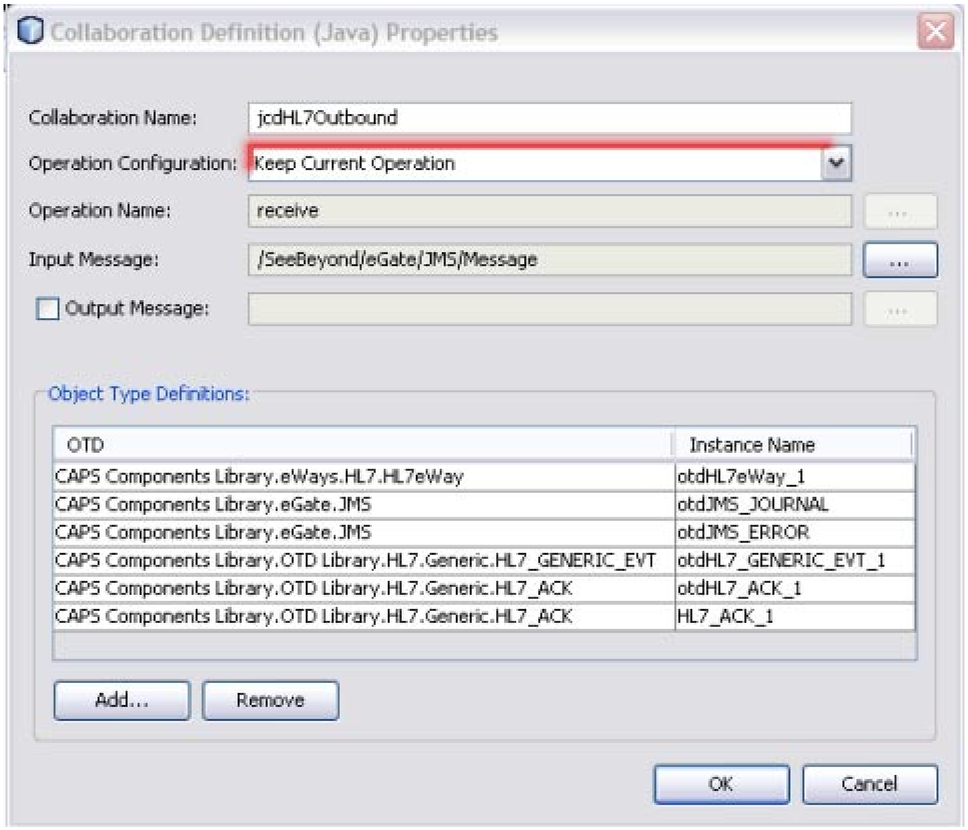
1 **Right-click the Collaboration from the Projects tab to which you want to add the HL7 OTD.**

For Example, **jcdHL7Outbound**.

2 **Select Properties from the drop-down menu.**

The Collaboration Definition (Java) Properties dialog box appears (for this example, the **jcdHL7Outbound** Collaboration.

3 **Select Keep Current Operation as the value for the Operation Configuration field.**



Collaboration Name: jcdHL7Outbound

Operation Configuration: Keep Current Operation

Operation Name: receive

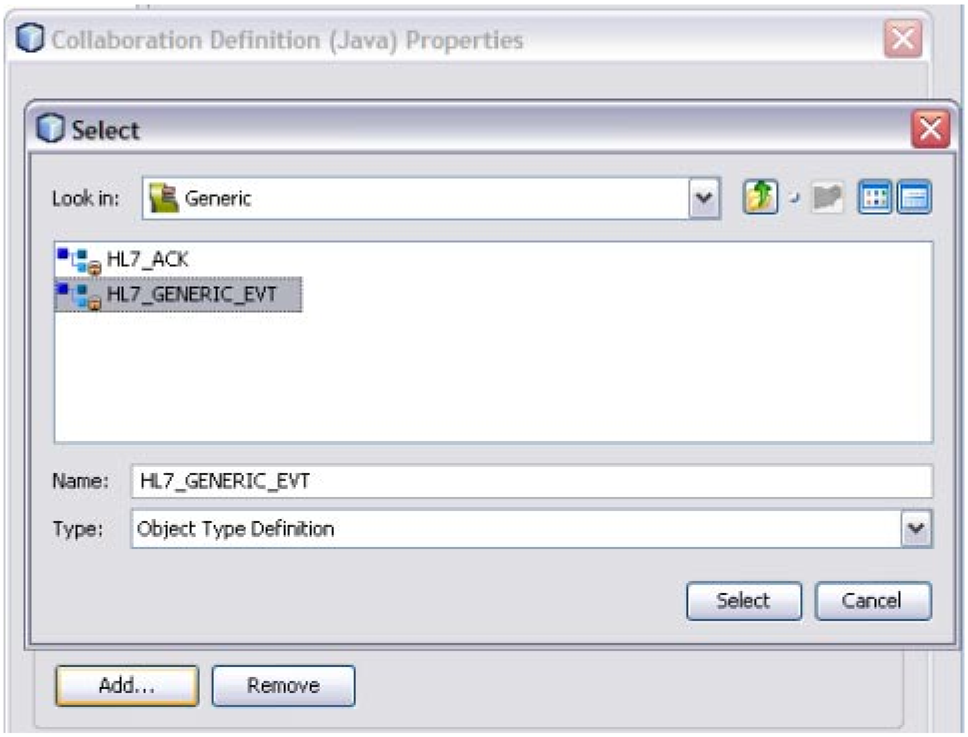
Input Message: /SeeBeyond/eGate/JMS/Message

Output Message:

OTD	Instance Name
CAPS Components Library.eWays.HL7.HL7eWay	otdHL7eWay_1
CAPS Components Library.eGate.JMS	otdJMS_JOURNAL
CAPS Components Library.eGate.JMS	otdJMS_ERROR
CAPS Components Library.OTD Library.HL7.Generic.HL7_GENERIC_EVT	otdHL7_GENERIC_EVT_1
CAPS Components Library.OTD Library.HL7.Generic.HL7_ACK	otdHL7_ACK_1
CAPS Components Library.OTD Library.HL7.Generic.HL7_ACK	HL7_ACK_1

FIGURE 1–13 Collaboration Definition (Java) Properties

4 Click Add to select the object type definition.



Name: HL7_GENERIC_EVT

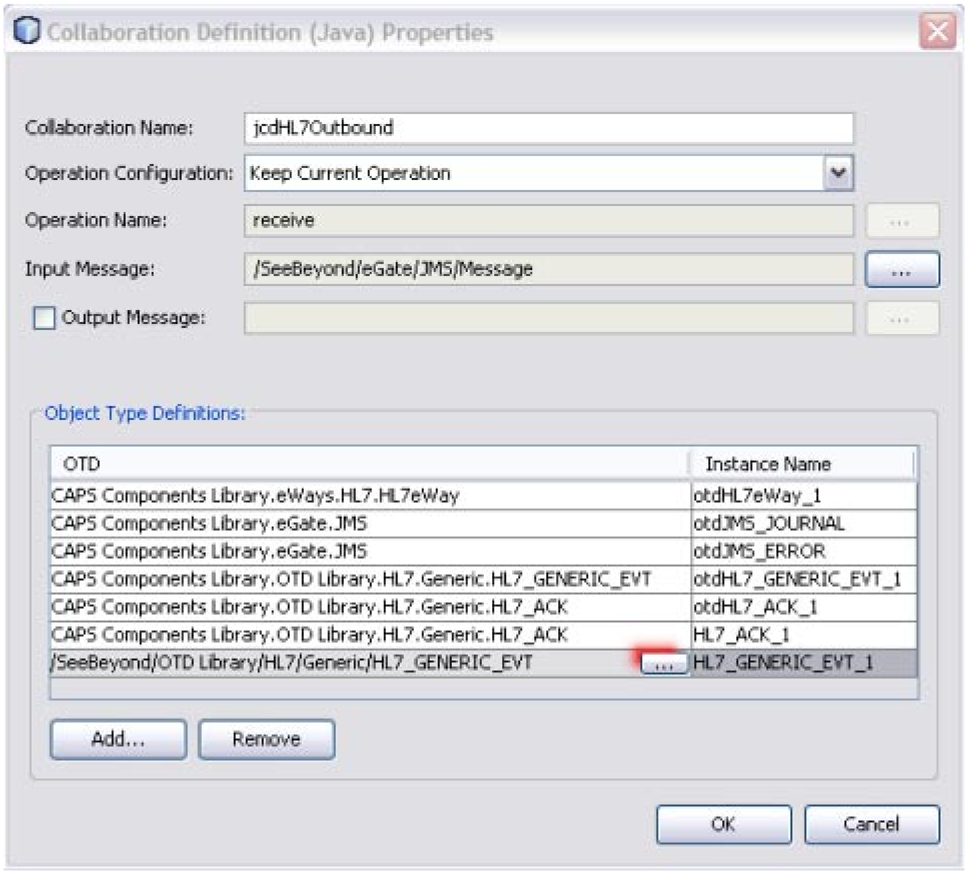
Type: Object Type Definition

FIGURE 1–14 Select Object Type Definition

5 **Select the OTD (for example, HL7_GENERIC_EVT) from the Select dialog box and click Select.**

Note – Click Remove to delete the OTD from the list.

The OTD is added to the list.



Collaboration Name: jcdHL7Outbound
Operation Configuration: Keep Current Operation
Operation Name: receive
Input Message: /SeeBeyond/eGate/JMS/Message
Output Message:

OTD	Instance Name
CAPS Components Library.eWays.HL7.HL7eWay	otdHL7eWay_1
CAPS Components Library.eGate.JMS	otdJMS_JOURNAL
CAPS Components Library.eGate.JMS	otdJMS_ERROR
CAPS Components Library.OTD Library.HL7.Generic.HL7_GENERIC_EVT	otdHL7_GENERIC_EVT_1
CAPS Components Library.OTD Library.HL7.Generic.HL7_ACK	otdHL7_ACK_1
CAPS Components Library.OTD Library.HL7.Generic.HL7_ACK	HL7_ACK_1
/SeeBeyond/OTD Library/HL7/Generic/HL7_GENERIC_EVT	HL7_GENERIC_EVT_1

FIGURE 1–15 OTD Added to the List

Note – To Modify the Existing HL7 V2 OTD

Click the ellipses and select appropriate Projects folder (for example, CAPS Components Library) from the Select dialog box to modify the existing HL7 V2 OTD.

TCP/IP HL7 V3 Adapter Collaborations

HL7 V3, like V2.x, is a standard for exchanging messages among information systems that implement healthcare applications. However, V3 strives to improve the V2 process and its outcomes. The original process for defining HL7 messages was established in 1987. It has served well since. However, as HL7 membership grew and its standards became more widely used, HL7 has become aware of opportunities to revolutionize healthcare interface computing. HL7 interfaces substantially reduce costs and implementation times when compared to the industry's experience with proprietary interfaces. The development principles behind HL7 V3 lead to a more robust, fully specified standard.

V3 introduces a new approach to HL7 message development.

While the HL7 V2 standard was created mostly by clinical interface specialists, the V3 standard has been influenced strongly by work from volunteers representing the government and medical informatist users. This means that the level of formal modeling, complexity, and internal consistency is radically higher in V3 when compared to V2.

To assist with the comprehension of this new approach, the domain begins with a high-level overview and progressively drill down to lower levels of messaging detail (including storyboards, applications roles, trigger events, D-MIMs, R-MIMs, HMDs, message types, and interactions). Since a *big picture* understanding of the particular domain is crucial to understanding the messages defined within it, each domain provides a Storyboard.

Introducing the Methodology

The Domain Information Model (D-MIM) is presented first in each domain, providing a graphical and narrative detail of the scope of the domain. The D-MIM is the first in the series of information models. The D-MIM is a subset of the RIM but uses specific conventions to represent artifacts. Once the classes, attributes, and associations for a particular domain have been isolated through the D-MIM, the next logical step is to extract the set of classes, attributes, and associations required for an HMD or set of HMDs that originate from the same root class. These extractions are referred to as R-MIMs (Refined Message Information Models), which are a subset of and use the same conventions as the domain D-MIM.

The second element in the domain is one or more domain-level storyboards. Each storyboard includes a diagrammatic representation, called an **Interaction Diagram**.[1]

There are three modes of Acknowledgement Process in HL7 V3.

- Immediate Mode
- Deferred Mode
- Queued Mode

Outlined below is a summary assessment of the two HL7 versions.

[1] Interactions describe the purpose of information flow. Create Patient Billing Account and Delete Patient Billing Account are examples of interactions found in the HL7 V3 messaging standard. Once the overall picture of data exchange is presented, the Storyboard continues with a narrative describing how the interactions might unfold in real life. Following the narrative is a description of each application role. Storyboards may be represented for the entire domain, or may be specific within the domain.

TABLE 1–1 Summary Assessment

Standard	Benefits	Challenges
HL7 V2	Reflects the complex **everyone is special** world of healthcare	Provides a **one size fits none** standard
	Much less expensive to build HL7 interfaces compared to custom interfaces	**Loose and optional ridden** HL7 definitions lead to discrepancies in HL7 interfaces
	Provides 80 percent of the interface and a framework to negotiate the remaining 20 percent on an interface-by-interface basis	Not inclusive of international needs No compatibility with HL7 V3
	Historically built in an ad hoc way, allowing the most critical areas to be defined first Generally provides compatibility between 2.x versions	Defining a detailed list of items to be discussed and negotiated before interfacing can occur is required
HL7 V3	More of a **true standard** and less of a **framework for negotiation**	For clinical interface specialists
	Model-based standard provides consistency across entire standard	No compatibility with HL7 V2
	Application roles well defined	Adoption will be expensive and takes time
	Much less message optionality	Long adoption cycle, unless storage business case or regulatory requirement changes
	Less expensive to build and maintain mid-to-long term interfaces	Retraining and retooling necessary

What's New with HL7 V3

1. HL7 V3 adopts an Object Oriented (OO) approach using Unified Modeling Language (UML) principles.
2. Reduced Optionality
3. Conformance to HL7 V3 will be testable

Artifact Identification System

Within the HL7 V3 standards the components that make up the HL7 V3 are each referred to as Artifacts. This includes,

- Interactions
- RIM
- D-MIMs
- R-MIMs
- HMDs
- Storyboards
- Application Roles
- Trigger Events
- Message Types

HL7 V3 Message Development Process

The V3 message development process is based on information models produced by the development groups. An information model represents data in object oriented way. It has classes with relationships, properties, states and constraints. The information models are refined and localized to come up with messages that can be exchanged with different systems.

- **Reference Information Model (RIM):** The RIM is an information model collectively developed by the HL7 working group. It is the information model that encompasses the HL7 domain of interest as a whole. The RIM is a coherent, shared information model that is the source for the data content of all HL7 messages. The RIM is intentionally abstract to represent the HL7 data in a standard way across all the domains of Healthcare. It is the root of all information models and structures developed as part of the V3 development process.
- **Domain Message Information Model (D-MIM):** The classes, attributes, state-machines, and relationships in the RIM are used to derive domain-specific information models called D-MIMs. A D-MIM is a refined subset of the RIM that includes a set of class clones, attributes and relationships that can be used to create messages for a particular domain (a particular area of interest in healthcare)

  Domains under Administrative Management

  - Accounting and Billing
  - Claims & Reimbursements
  - Patient Administration
  - Personnel Management
  - Scheduling

  Domains under Health and Clinical Management:

  - Clinical Document Architecture
  - Medical Records

  - Public Health Reporting
  - Clinical Genomics
  - Specimen Domain
  - Regulated Studies
- **Refined Message Information Model (R-MIM):** The R-MIM is a subset of a D-MIM that is used to express the information content for a message or set of messages with annotations and refinements that are message specific. The D-MIM is used as a common base upon which all R-MIMs within a domain are built.
- **Hierarchical Message Descriptions (HMD):** The HMDs are abstract message structures which represent the information from R-MIMs. The HMD represents R-MIM in an organized way which can be exchanged between systems. The HMDs are called abstract because they define the message structure without reference to the implementation technology.
- **XML Schemas:** HL7 V3 defines Implementation technology specifications (ITS). These ITS define how the abstract message types should be transmitted using an underlying technology. V3 currently defines XML ITS and UML ITS. The XML ITS defines data types and structures. The XML datatypes represents HL7 datatypes in XML specific way. And the XML structures represent the structures defined by HMDs. Thus, for every message type there is a corresponding HMD and XSD .

Artifact Codes

As pet the process of V3 message development, where V3 information models are refined to come up with XML schemas which can be used to construct V3 XML messages. The specification development process identifies various artifacts for a particular domain and submits them. For every domain, the artifacts are organized in the same structure. The following Artifact Codes have been assigned.

TABLE 1–2 Artifact Codes

Artifact	Code
Application Role	AR
Domain Information Model (D-MIM)	DM
Hierarchical Message Descriptor (HMD)	HD
Interaction	IN
Message Type	MT
Refined Message Information Model (R-MIM)	RM
Storyboard	ST

TABLE 1–2 Artifact Codes *(Continued)*

Artifact	Code
Storyboard Narrative	SN
Trigger Event	TE

The Patient Administration domain submits an application role with the following unique artifact identifier:

PRPA_AR00001UV00

Where:

PR Subsection: Practice
PA Domain: Patient Administration
AR Artifact: Application Role
00001 6 digit non-meaningful number assigned by the TC to ensure uniqueness
UV Realm (the only current value is UV for universal)
00 Current version number

Application Role (Code: AR)

Application roles represent a set of communication responsibilities an application can implement. Thus, they describe system components or sub-components that send and/or receive interactions.

TABLE 1–3 Patient Billing Account Informer

Structured Name	Artifact Identifier	Description
Patient Billing Account Event Informer	FIAB_AR010001UV01	Informs a tracker of patient billing account creates (activate), update (revise), closes (complete), deletes (nullify), and merges (obsolete)

Trigger Event (Code: TE)

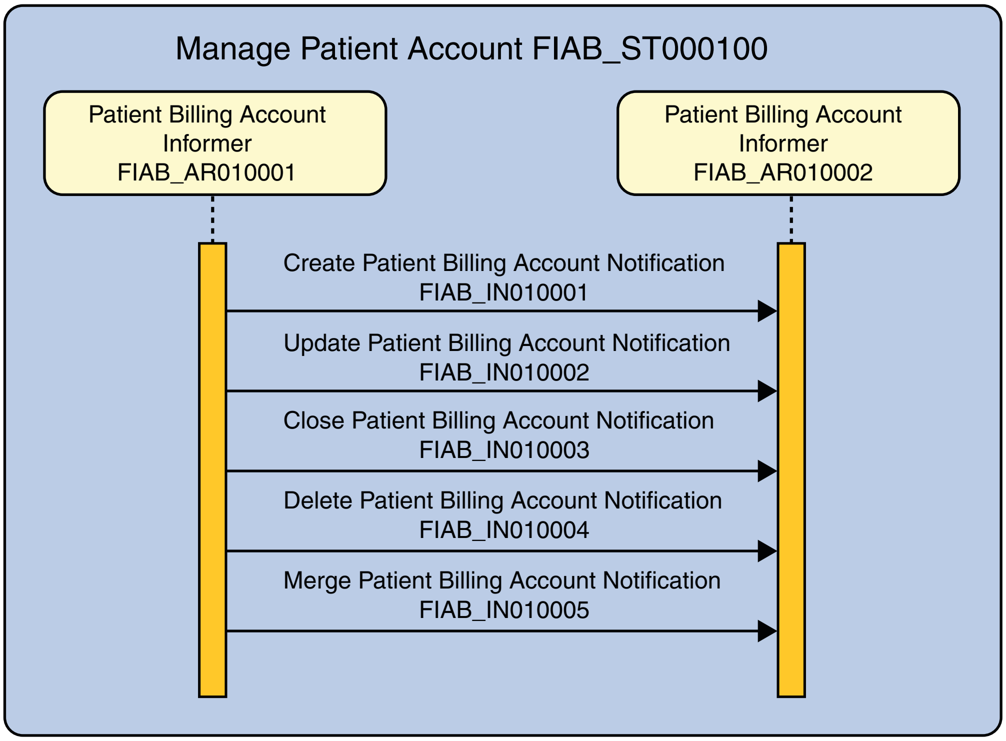
A trigger event is a condition on which an action is initiated. HL7 adapter covers interaction based trigger events.

TABLE 1–4 Close Patient Billing Account

Structured Name	Artifact Identifier	Description
Patient Billing Account Event Complete Notification **Type:** State-transition based	FIAB_TE010003UV01	This trigger event signals that the status of a patient billing account is rendered closed. This means it is no longer eligible for financial transaction posting.

Storyboard (Code: ST)

A storyboard explains the series of actions in a particular scenario. Its primary purpose is to identify the various scenarios. Each storyboard is represented as sequence diagrams. It also lists the interactions between two or more systems playing different Application Roles.


Manage Patient Account FIAB_ST000100
Patient Billing Account Informer FIAB_AR010001
Patient Billing Account Informer FIAB_AR010002
Create Patient Billing Account Notification FIAB_IN010001
Update Patient Billing Account Notification FIAB_IN010002
Close Patient Billing Account Notification FIAB_IN010003
Delete Patient Billing Account Notification FIAB_IN010004
Merge Patient Billing Account Notification FIAB_IN010005

Storyboard Narration (Code: SN)

A Storyboard narrative provides explanation for the storyboard.

TABLE 1–5 Close Patient Account

Structured Name	Artifact Identifier	Description
Patient Account Close	FIAB_SN000106UV01	Ms Smith's account for a previous visit is at a zero balance and has not had any activity for some specified period of time. The account is closed, declared inactive and not available for further activity, preparatory to archiving and/or removal from the administrative system.

Domain Information Model (D-MIM) (Code: DM)

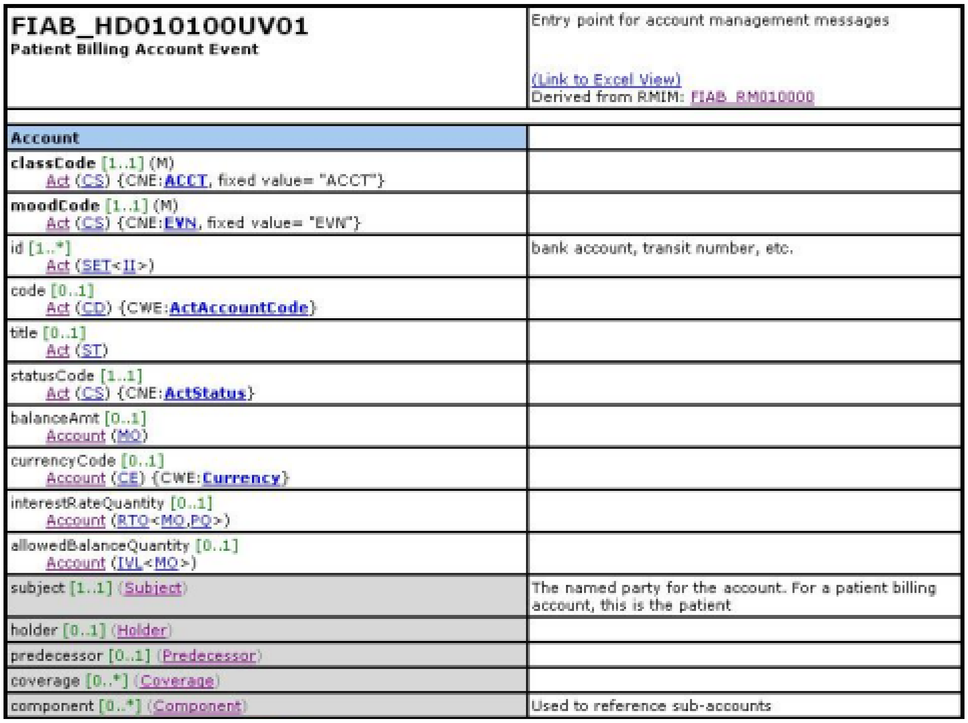
Domain specific classes represented in a diagram with relationships.

Refined Message Information Model (R-MIM) (Code: RM)

A subset of D-MIM classes represented in a diagram with relationships. These classes specifically represent information content for one or more HMDs.

Hierarchical Message Descriptor (HMD) (Code: HD)

The Message Structures or Message types which form the payload of the data transmitted. The base HMD is depicted in a tabular format or in excel view. It also lists the various message types built based on this HMD.



FIAB_HD010100UV01 **Patient Billing Account Event**	Entry point for account management messages (Link to Excel View) Derived from RMIM: FIAB_RM010000
Account	
classCode [1..1] (M) Act (CS) {CNE:**ACCT**, fixed value= "ACCT"}	
moodCode [1..1] (M) Act (CS) {CNE:**EVN**, fixed value= "EVN"}	
id [1..*] Act (SET<II>)	bank account, transit number, etc.
code [0..1] Act (CD) {CWE:**ActAccountCode**}	
title [0..1] Act (ST)	
statusCode [1..1] Act (CS) {CNE:**ActStatus**}	
balanceAmt [0..1] Account (MO)	
currencyCode [0..1] Account (CE) {CWE:**Currency**}	
interestRateQuantity [0..1] Account (RTO<MO,PQ>)	
allowedBalanceQuantity [0..1] Account (IVL<MO>)	
subject [1..1] (Subject)	The named party for the account. For a patient billing account, this is the patient
holder [0..1] (Holder)	
predecessor [0..1] (Predecessor)	
coverage [0..*] (Coverage)	
component [0..*] (Component)	Used to reference sub-accounts

FIGURE 1–16 Hierarchical Message Descriptor

Message Type (Code: MT)

A message type represents a unique set of constraints applied against the common message. It is represented in Table view or in XML schema view.

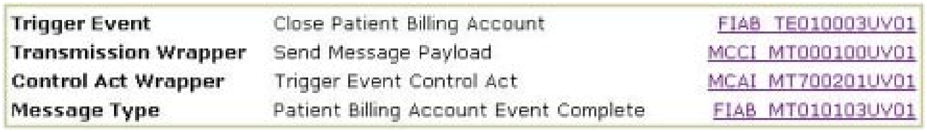
Interaction (Code: IN)

Interactions are at the heart of messaging. Interactions define the interaction between two systems. The documentation of interaction defines the following interactions.

1. Trigger Event which initiated the interaction.
2. Message Type which should be used for interaction (the payload).
3. Transmission Wrapper: A Message Type to wrap the payload.
4. Control Act Wrapper: A Message Type to hold the administration information about the payload being transmitted.
5. Sender: The Sender Role for this interaction.
6. Receiver: The Receiver Role for this interaction.

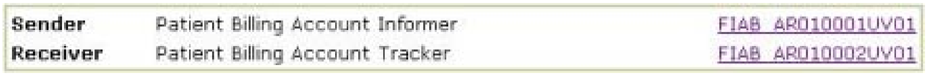
TABLE 1–6 Close Patient Billing Account Notification

Structured Name	Artifact Identifier	Description
Patient Billing Account Event Complete Notification	FIAB_IN010003UV01	Close patient billing account



Trigger Event	Close Patient Billing Account	FIAB_TE010003UV01
Transmission Wrapper	Send Message Payload	MCCI_MT000100UV01
Control Act Wrapper	Trigger Event Control Act	MCAI_MT700201UV01
Message Type	Patient Billing Account Event Complete	FIAB_MT010103UV01

FIGURE 1–17 Close Patient Billing Account



Sender	Patient Billing Account Informer	FIAB_AR010001UV01
Receiver	Patient Billing Account Tracker	FIAB_AR010002UV01

FIGURE 1–18 Sending and Receiving Roles

Transmission Wrapper and Control Act Wrapper

This specification defines HL7 V3 Composite message is composed of:

1. An **HL7 Transmission wrapper** (always)
2. A **Trigger Event Control Act** (required for all messages except accept level acknowledgements, for which it is not permitted)
3. The **HL7 Domain Content** specified by an HL7 domain specific technical committee (required for each Trigger Event Control Act)

The HL7 Transmission wrapper is specified in domain: Transmission Infrastructure. This domain specification describes the details on composing a V3 composite message and exchanging the composite message between systems. It covers features like acknowledgements, sequence no. protocol support, error handling and security. Trigger Event Control Act is detailed in domain: Message Control Act Infrastructure. This details the administrative information about the data being transmitted. Thus the interactions define the complete set of information that is required to exchange HL7 data between the systems.

Comparison between HL7 V2.x and HL7 V3

The HL7 2.x specifications are

- Segments that imply information entities
- Events that indicate implied behaviors

The HL7 V3 specifications are

Uses object-oriented analysis methodology to

- Improve the internal consistency of HL7
- Provide sound semantic definitions
- Enable future architectures
- Produce an evolution not a revolution

Benefits of V3 to HL7

Following are the benefits.

- Reduces optionality that results in more specific messages
- Uncovers hidden assumptions about application boundaries
- Facilities defining clear, fine-grained, conformance claims

The HL7 V3 specification contains the XSDs for Interactions which can be used to construct the composite HL7 V3 message. This composite message then can be transmitted between the systems. The V3 Specification says that a system can claim for V3 conformance based on the Application Role it will play in a particular domain. The system playing the Application Role will recognize the Trigger Events, the message types and the data content of these messages. Thus a system whose sole responsibility is to exchange the V3 data between systems reliably should claim conformance for the Application Roles defined under Transmission Infrastructure Domain.

Inbound HL7 V3 Collaboration

The Inbound HL7 V3 Collaboration is classified into two types:

1. Inbound HL7 V3 Immediate Collaboration
2. Inbound HL7 V3 Deferred Collaboration

Inbound HL7 V3 Immediate Collaboration Overview

The Inbound HL7 V3 Collaboration, **jcdHL7V3Inbound**, contains OTDs for the HL7 V3 Resource Adapter, JMS Data, JMS Journal, and JMS Error, as well as the HL7 V3 Patient

Administration Domain Interaction Event (PRPA_IN403001UV01) and the corresponding HL7 V3 Acknowledgements (MCCI_IN000004UV01). The Collaboration works with its own internal code and the Properties Configuration files.

Note – Immediate Mode: Contains only one type of Acknowledgement OTD, that is, MCCI_IN000004UV01

HL7 V3 Standard Inbound Message Mode Data Flow (For Immediate Mode of ACK Process) — Part 1

A HL7 V3 message triggers an inbound Collaboration received from an external system or an outbound HL7 V3 Client. Execute the following Collaboration calls of the HL7 V3 User Collaboration Rule **receive()**.

The receive method is the entry point to the HL7 V3 User Collaboration, with the following signature:

```
public void

receive(com.stc.connector.appconn.tcpip.hl7.HL7ServerApplication input,

com.stc.connectors.jms.JMS otdJMS_DATA,

xsd.hl7v3.PRPA_IN403001UV01.PRPA_IN403001UV01_ otd_PRPA_IN403001UV01_1

xsd.hl7v3.MCCI_IN000004UV01.MCCI_IN000004UV01_ otd_MCCI_IN000004UV01

otdHL7_ACK_1,com.stc.connectors.jms.JMS otdJMS_JOURNAL,

com.stc.connectors.jms.JMS otdJMS_ERROR) throws Throwable.
```

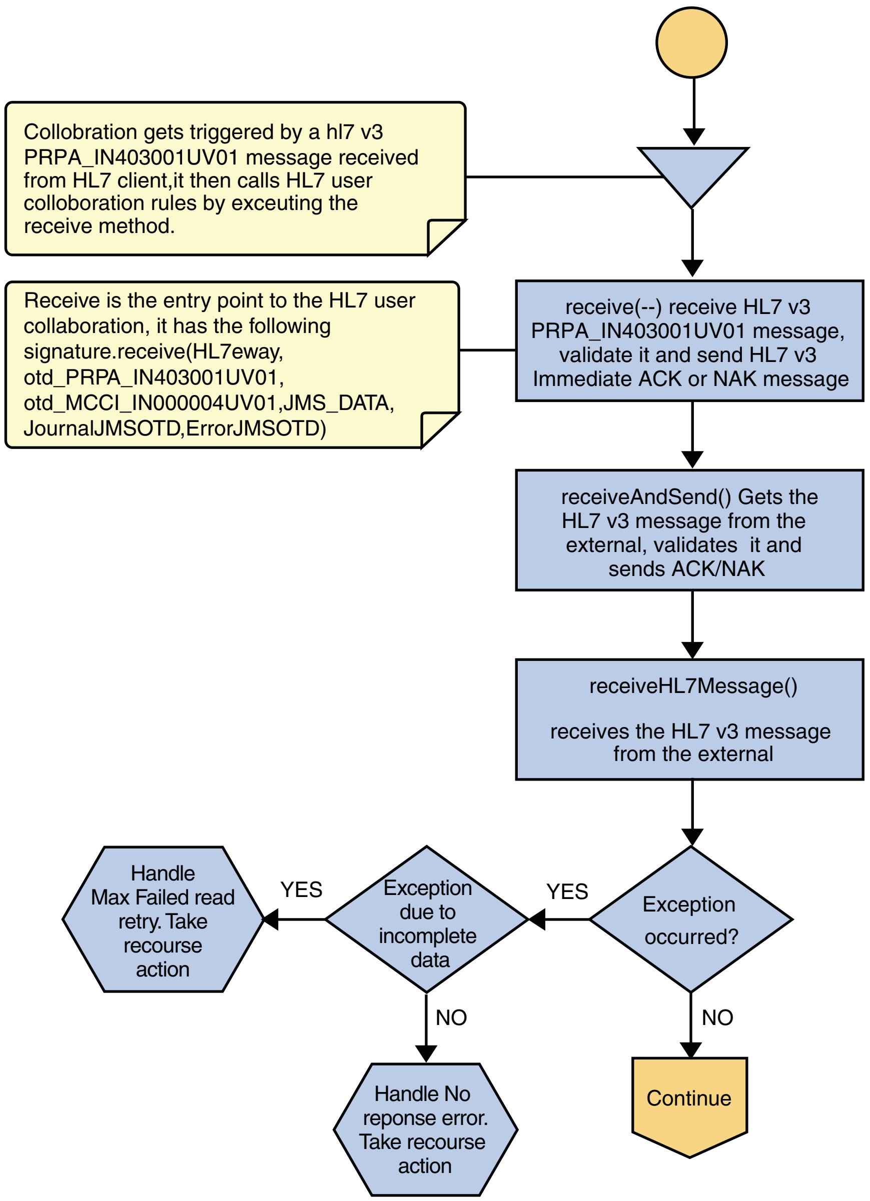

Collobration gets triggered by a hl7 v3 PRPA_IN403001UV01 message received from HL7 client,it then calls HL7 user colloboration rules by exceuting the receive method.
Receive is the entry point to the HL7 user collaboration, it has the following signature.receive(HL7eway, otd_PRPA_IN403001UV01, otd_MCCI_IN000004UV01,JMS_DATA, JournalJMSOTD,ErrorJMSOTD)
receive(--) receive HL7 v3 PRPA_IN403001UV01 message, validate it and send HL7 v3 Immediate ACK or NAK message
receiveAndSend() Gets the HL7 v3 message from the external, validates it and sends ACK/NAK
receiveHL7Message()
receives the HL7 v3 message from the external
Exception occurred?
YES
Exception due to incomplete data
YES
Handle Max Failed read retry. Take recourse action
NO
Handle No reponse error. Take recourse action
NO
Continue

FIGURE 1–19 Immediate Mode of ACK Process — Part 1

Once the message is received, the Collaboration determines whether the message needs to be validated. The HL7 V3 message is then validated, making sure that the message structure is correct. Various fields in the Transmission Wrapper of the message are also validated, such as

Version Code, Processing Code, Processing Mode Code, and Interaction ID. If these fields do not match the configuration, a NAK is returned.

The Collaboration receives the HL7 V3 message from the external using **receiveHL7Message()**. If an exception occurs due to incomplete data, and the adapter 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.

HL7 V3 Standard Inbound Message Mode Data Flow (For Immediate Mode of ACK Process) — Part 2

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 V3 message.

If the HL7 V3 message does not pass validation, the Collaboration calls **makeNak()** and **sendHL7Nak()** to create and send the NAK to the external. The HL7 V3 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.

If the HL7 V3 message passes validation, the Collaboration calls **makeAck()** and **sendHL7Ack()** to create and send the ACK to the external.

After the ACK is sent, the HL7 V3 message and the ACK are journaled to the JMS Queue Journal destination. If the message fails to journal the associated recourse action is taken.

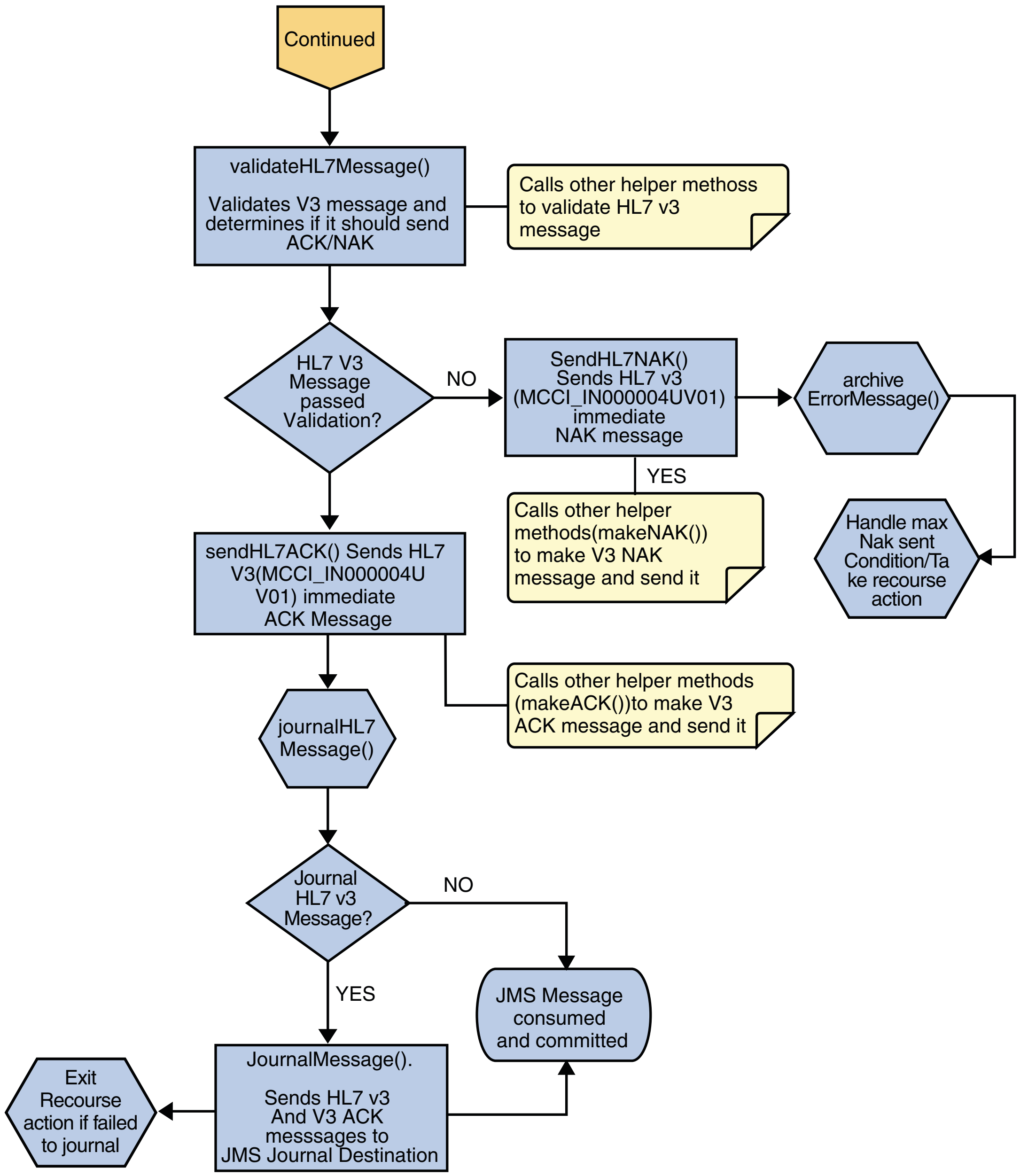

Continued
validateHL7Message()
Validates V3 message and determines if it should send ACK/NAK
Calls other helper methoss to validate HL7 v3 message
HL7 V3 Message passed Validation?
NO
SendHL7NAK() Sends HL7 v3 (MCCI_IN000004UV01) immediate NAK message
archive ErrorMessage()
YES
Calls other helper methods(makeNAK()) to make V3 NAK message and send it
Handle max Nak sent Condition/Take recourse action
sendHL7ACK() Sends HL7 V3(MCCI_IN000004UV01) immediate ACK Message
Calls other helper methods (makeACK())to make V3 ACK message and send it
journalHL7 Message()
Journal HL7 v3 Message?
NO
YES
JMS Message consumed and committed
JournalMessage().
Sends HL7 v3 And V3 ACK messsages to JMS Journal Destination
Exit Recourse action if failed to journal

FIGURE 1–20 Immediate Mode of ACK Process — Part 2

Inbound HL7 V3 Deferred Collaboration Overview

The Inbound HL7 V3 Collaboration, **jcdHL7V3DeferredInbound**, contains OTDs for the HL7 V3 Resource Adapter, JMS Data, JMS Journal, JMS Error, and JMS APP ACK as well as the HL7 V3 Patient Administration Domain Interaction Event (PRPA_IN403001UV01) and the corresponding HL7 V3 Commit Acknowledgement (MCCI_IN000006UV01) and Application Acknowledgement (MCCI_IN000007UV01). The Collaboration works with its own internal code and the Properties Configuration files.

Note – Deferred Mode: Contains two types of Acknowledgement OTDs,

1. Commit Acknowledgement (MCCI_IN000006UV01)
2. Application Acknowledgement (MCCI_IN000007UV01)

The above JCD (jcdHL7V3DeferredInbound) will also work for both Immediate Mode and Deferred Mode.

HL7 V3 Standard Inbound Message Mode Data Flow (For Deferred Mode of ACK Process) — Part 1

A HL7 V3 message triggers an inbound Collaboration received from an external system or an outbound HL7 V3 Client. Execute the following Collaboration calls of the HL7 V3 User Collaboration Rule **receive()**.

The receive method is the entry point to the HL7 V3 User Collaboration, with the following signature:

```
public void

receive(com.stc.connector.appconn.tcpip.hl7.HL7ServerApplication input,

com.stc.connectors.jms.JMS otdJMS_DATA,

xsd.hl7v3.PRPA_IN403001UV01.PRPA_IN403001UV01_ otd_PRPA_IN403001UV01_1

xsd.hl7v3.MCCI_IN000004UV01.MCCI_IN000004UV01_ otd_MCCI_IN000004UV01

xsd.hl7v3.MCCI_IN000006UV01.MCCI_IN000006UV01_ otd_MCCI_IN000006UV01

xsd.hl7v3.MCCI_IN000007UV01.MCCI_IN000007UV01_ otd_MCCI_IN000007UV01

com.stc.connectors.jms.JMS otdJMS_APPACK

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 V3 message is then validated, making sure that the message structure is correct. Various fields in the Transmission Wrapper of the message are also validated, such as Version Code, Processing Code, Processing Mode Code, and Interaction ID. 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 adapter sends a NAK. The validated HL7 V3 message moves on to **processInitialHandshake()**.

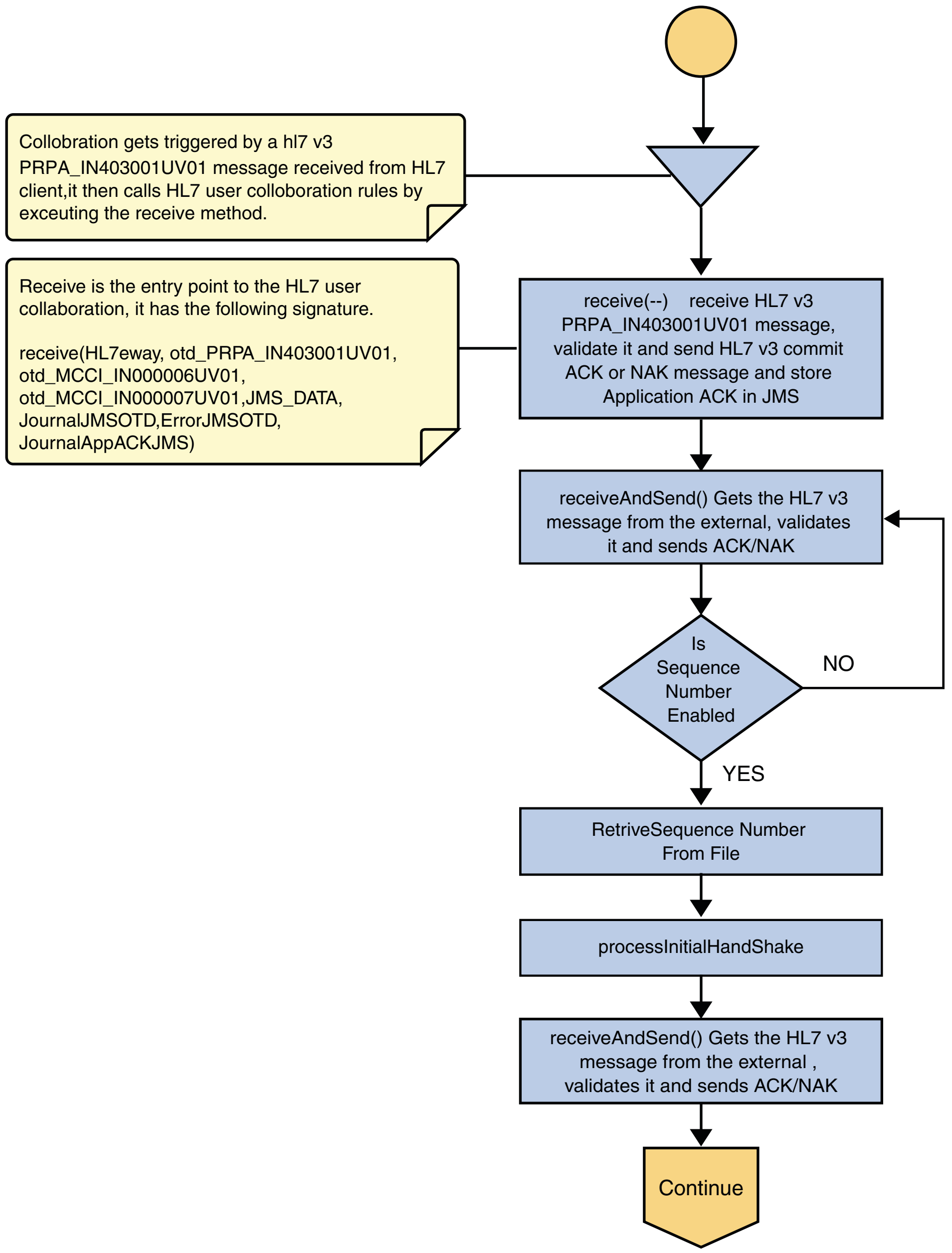

Collobration gets triggered by a hl7 v3 PRPA_IN403001UV01 message received from HL7 client,it then calls HL7 user colloboration rules by exceuting the receive method.
Receive is the entry point to the HL7 user collaboration, it has the following signature.
receive(HL7eway, otd_PRPA_IN403001UV01, otd_MCCI_IN000006UV01, otd_MCCI_IN000007UV01,JMS_DATA, JournalJMSOTD,ErrorJMSOTD, JournalAppACKJMS)
receive(--) receive HL7 v3 PRPA_IN403001UV01 message, validate it and send HL7 v3 commit ACK or NAK message and store Application ACK in JMS
receiveAndSend() Gets the HL7 v3 message from the external, validates it and sends ACK/NAK
Is Sequence Number Enabled
NO
YES
RetriveSequence Number From File
processInitialHandShake
receiveAndSend() Gets the HL7 v3 message from the external , validates it and sends ACK/NAK
Continue

FIGURE 1–21 Deferred Mode of ACK Process — Part 1

HL7 V3 Standard Inbound Message Mode Data Flow (For Deferred Mode of ACK Process) — Part 2

The Collaboration receives the HL7 V3 message from the external using **receiveHL7message()**. If an exception occurs due to incomplete data, and the adapter 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, **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 V3 message.

If the HL7 V3 message passes validation, the Collaboration calls **makeCommitAck()** and **sendHL7CommitAck()** to create and send the Commit ACK (MCCI_IN000006UV01) to the external. It then calls **journalHL7AppAck()** to create Application ACK (MCCI_IN000007UV01) and store it into the JMS.

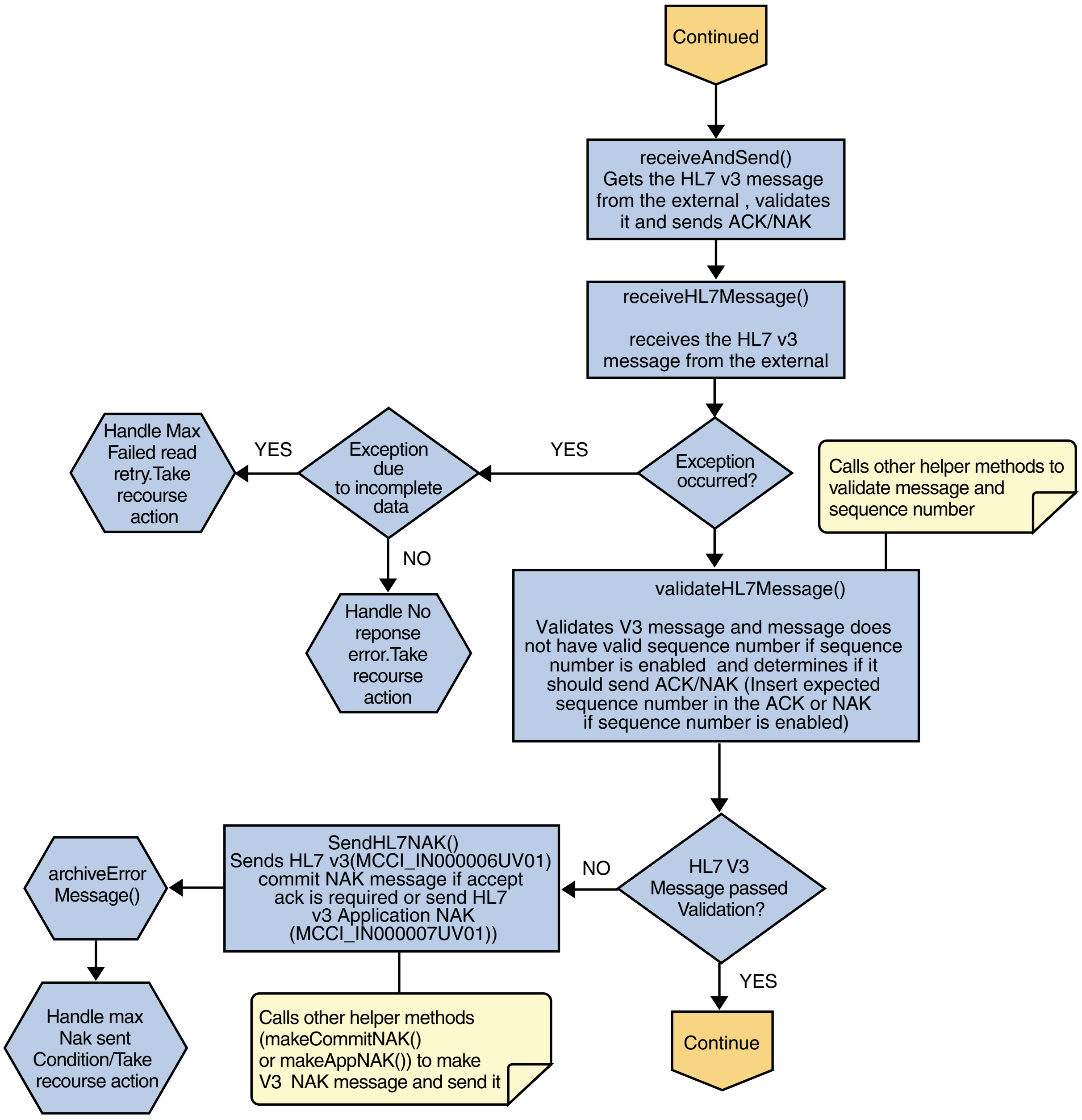

Continued
receiveAndSend()
Gets the HL7 v3 message from the external , validates it and sends ACK/NAK
receiveHL7Message()
receives the HL7 v3 message from the external
Exception occurred?
YES
Exception due to incomplete data
YES
Handle Max Failed read retry.Take recourse action
NO
Handle No reponse error.Take recourse action
Calls other helper methods to validate message and sequence number
validateHL7Message()
Validates V3 message and message does not have valid sequence number if sequence number is enabled and determines if it should send ACK/NAK (Insert expected sequence number in the ACK or NAK if sequence number is enabled)
HL7 V3 Message passed Validation?
NO
SendHL7NAK()
Sends HL7 v3(MCCI_IN000006UV01) commit NAK message if accept ack is required or send HL7 v3 Application NAK (MCCI_IN000007UV01))
archiveError Message()
Handle max Nak sent Condition/Take recourse action
Calls other helper methods (makeCommitNAK() or makeAppNAK()) to make V3 NAK message and send it
YES
Continue

FIGURE 1–22 Deferred Mode of ACK Process — Part 2

HL7 V3 Standard Inbound Message Mode Data Flow (For Deferred Mode of ACK Process) — Part 3

After the Commit ACK is sent and the Application ACK is stored in JMS, the HL7 V3 message and the ACKs are journaled to the JMS Queue Journal destination. If the message fails to journal the associated recourse action is taken.

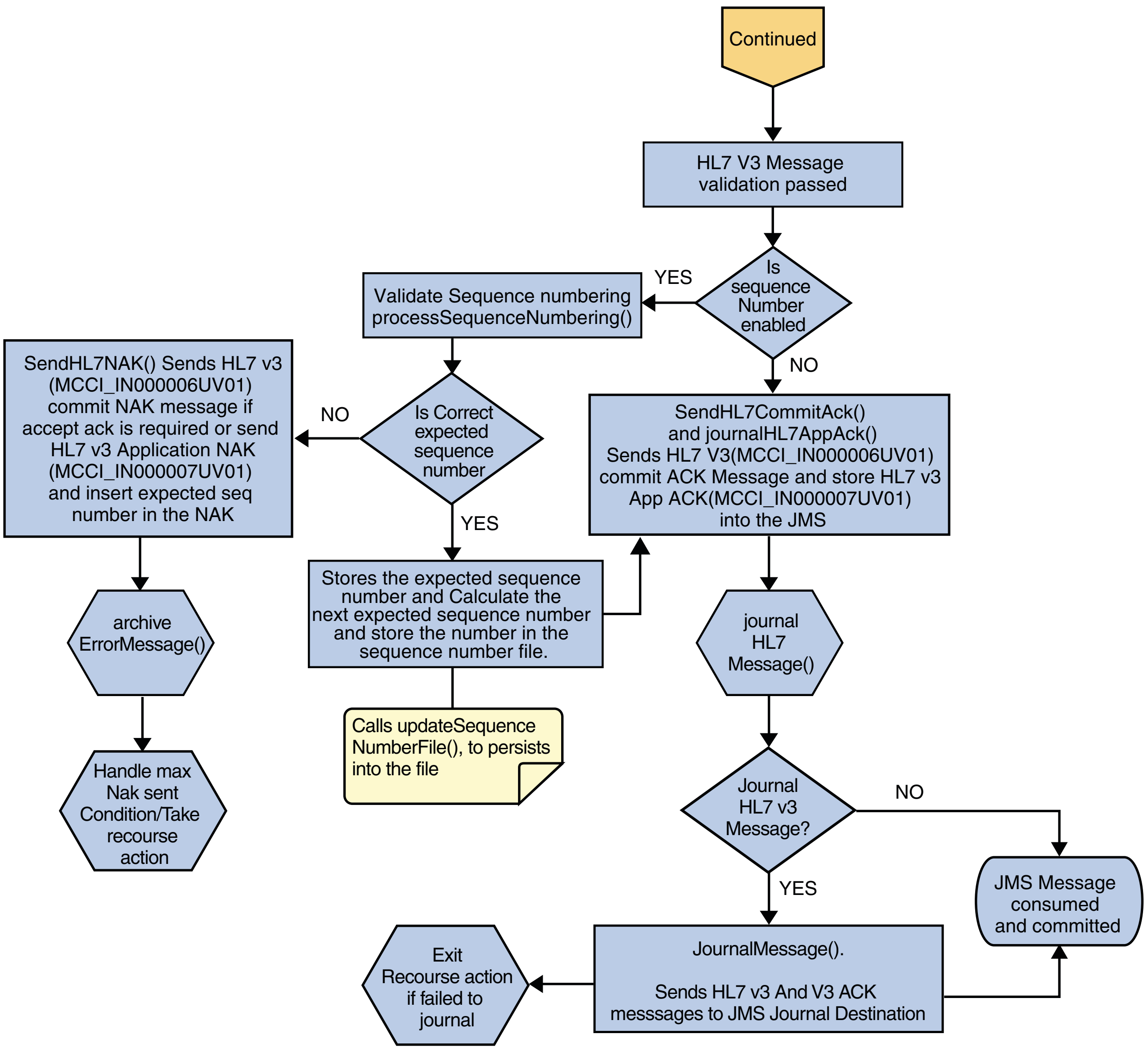

Continued
HL7 V3 Message validation passed
Is sequence Number enabled
YES
NO
Validate Sequence numbering processSequenceNumbering()
Is Correct expected sequence number
NO
YES
SendHL7NAK() Sends HL7 v3 (MCCI_IN000006UV01) commit NAK message if accept ack is required or send HL7 v3 Application NAK (MCCI_IN000007UV01) and insert expected seq number in the NAK
archive ErrorMessage()
Handle max Nak sent Condition/Take recourse action
Stores the expected sequence number and Calculate the next expected sequence number and store the number in the sequence number file.
Calls updateSequence NumberFile(), to persists into the file
SendHL7CommitAck() and journalHL7AppAck() Sends HL7 V3(MCCI_IN000006UV01) commit ACK Message and store HL7 v3 App ACK(MCCI_IN000007UV01) into the JMS
journal HL7 Message()
Journal HL7 v3 Message?
NO
YES
JMS Message consumed and committed
JournalMessage().
Sends HL7 v3 And V3 ACK messsages to JMS Journal Destination
Exit Recourse action if failed to journal

FIGURE 1–23 Deferred Mode of ACK Process — Part 3

Outbound HL7 V3 Collaboration

The Outbound HL7 V3 Collaboration, **jcdHL7V3Outbound**, contains OTDs for the HL7 V3 Resource Adapter, JMS Data, JMS Journal, and JMS Error, as well as the HL7 V3 Patient Administration Domain Interaction Event (PRPA_IN403001UV01) and the corresponding

HL7 V3 Acknowledgements. MCCI_IN000004UV01 for Immediate Mode and MCCI_IN000006UV01 for Deferred Mode. The Collaboration works with its own internal code and the Properties Configuration files. The outbound Collaboration assumes that it is reading valid HL7 V3 messages, so the data flow that feeds this Collaboration must ensure this.

Outbound HL7 V3 Collaboration Overview

The Outbound Collaboration is for both Immediate and Deferred Mode.

HL7 V3 Standard Outbound Message Mode Data Flow (For Immediate and Deferred Mode of ACK Process) — Part 1

The Collaboration is triggered by a JMS HL7 V3 message. The Collaboration then calls the HL7 V3 User Collaboration Rule by executing the receive method. Receive is the entry point to the HL7 V3 User Collaboration, with the following signature:

```
receive (input, otdHL7eWay_1, otdJMS_JOURNAL, otdJMS_ERROR,
otd_MCCI_IN000004UV01_1, otd_MCCI_IN000006UV01_1, otd_PRPA_IN403001UV01_1)
```

The incoming HL7 V3 message is then validated, making sure that the message structure is correct. Various fields in the Transmission Wrapper of the message are also validated, such as Version Code, Processing Code, Processing Mode Code, and Interaction ID.

If the message does not pass validation, an error occurs and the associated recourse action is applied. If the HL7 V3 message passes validation, the message moves on to **processInitialHandshake()** to receive a sequence number (if sequences numbering is enabled only for Deferred Mode). 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 V3 message.

Next, the message moves on to **processMessage()**, which calls the helper method, **sendAndReceive()**. The **sendAndReceive** method sends the HL7 V3 message, waits for an HL7 V3 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 V3 message up to the configured number of retries, at which time an error occurs and the associated recourse action is taken.

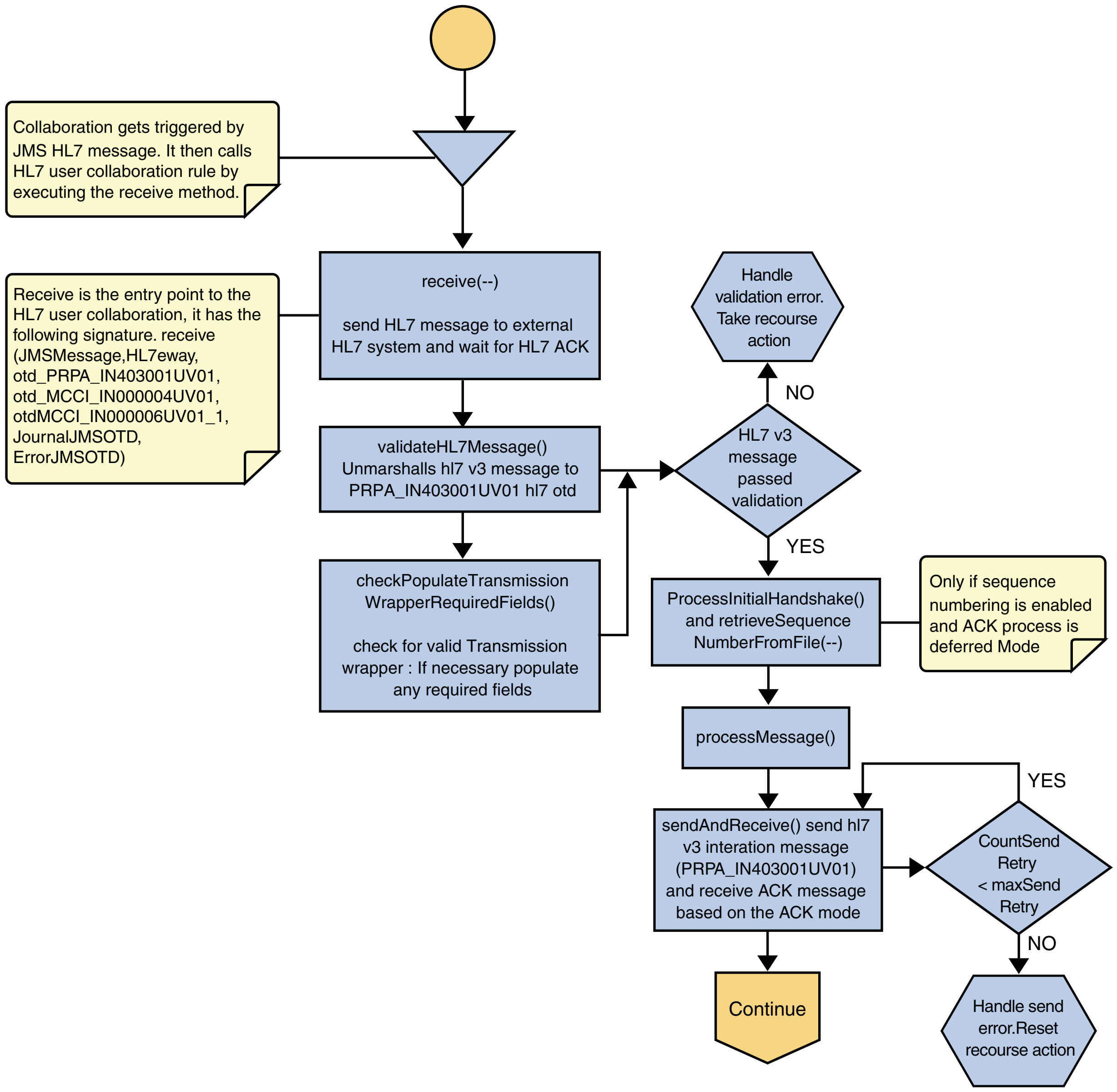

Collaboration gets triggered by JMS HL7 message. It then calls HL7 user collaboration rule by executing the receive method.
Receive is the entry point to the HL7 user collaboration, it has the following signature. receive (JMSMessage,HL7eway, otd_PRPA_IN403001UV01, otd_MCCI_IN000004UV01, otdMCCI_IN000006UV01_1, JournalJMSOTD, ErrorJMSOTD)
receive(--)
send HL7 message to external HL7 system and wait for HL7 ACK
validateHL7Message() Unmarshalls hl7 v3 message to PRPA_IN403001UV01 hl7 otd
checkPopulateTransmission WrapperRequiredFields()
check for valid Transmission wrapper : If necessary populate any required fields
Handle validation error. Take recourse action
NO
HL7 v3 message passed validation
YES
ProcessInitialHandshake() and retrieveSequence NumberFromFile(--)
Only if sequence numbering is enabled and ACK process is deferred Mode
processMessage()
YES
sendAndReceive() send hl7 v3 interation message (PRPA_IN403001UV01) and receive ACK message based on the ACK mode
CountSend Retry < maxSend Retry
NO
Continue
Handle send error.Reset recourse action

FIGURE 1–24 Immediate and Deferred Mode of ACK Process — Part 1

HL7 V3 Standard Outbound Message Mode Data Flow (For Immediate and Deferred Mode of ACK Process) — Part 2

In the **processMessage()**, the message moves on to **insertSequenceNumber()**. If sequence numbering is enabled and is in Deferred Mode, the **insertSequenceNumber** method inserts the sequence number and call **sendHL7Message()**. The **sendHL7Message** method sends the HL7 V3 message to the external using the HL7 V3 adapter OTD

The Collaboration receives the HL7 V3 ACK or NAK from the external using **receiveHL7AckNak()**. If an exception occurs due to incomplete data, and the adapter 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 in MCCI_IN000004UV01 Otd for Immediate Mode and MCCI_IN000006UV01 Otd for Deferred Mode.

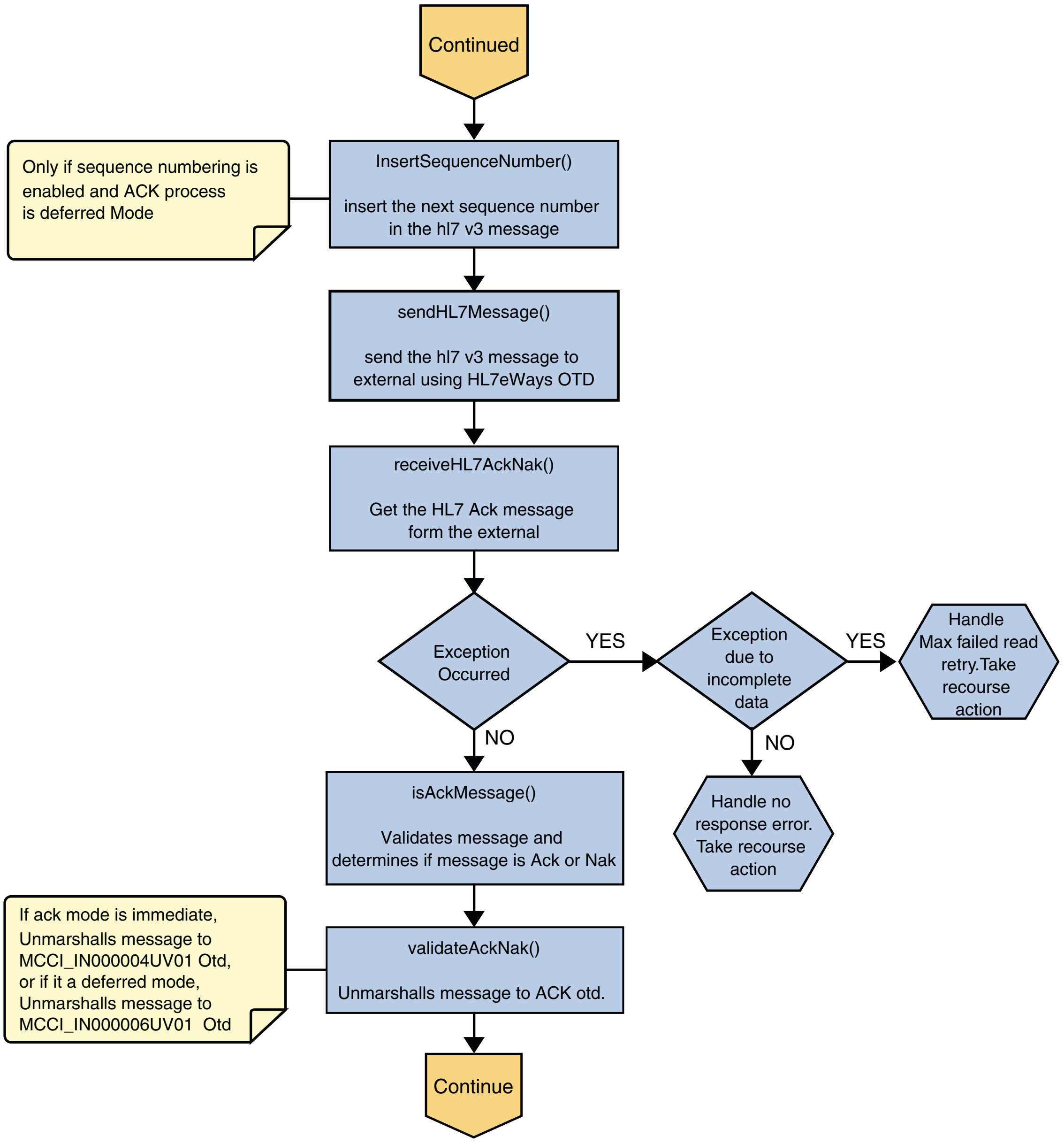

Continued
Only if sequence numbering is enabled and ACK process is deferred Mode
InsertSequenceNumber()
insert the next sequence number in the hl7 v3 message
sendHL7Message()
send the hl7 v3 message to external using HL7eWays OTD
receiveHL7AckNak()
Get the HL7 Ack message form the external
Exception Occurred
YES
Exception due to incomplete data
YES
Handle Max failed read retry.Take recourse action
NO
Handle no response error. Take recourse action
NO
isAckMessage()
Validates message and determines if message is Ack or Nak
If ack mode is immediate, Unmarshalls message to MCCI_IN000004UV01 Otd, or if it a deferred mode, Unmarshalls message to MCCI_IN000006UV01 Otd
validateAckNak()
Unmarshalls message to ACK otd.
Continue

FIGURE 1–25 Immediate and Deferred Mode of ACK Process — Part 2

HL7 V3 Standard Outbound Message Mode Data Flow (For Immediate and Deferred Mode of ACK Process) — Part 3

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.

If the ACK message validates, the HL7 V3 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 for Deferred Mode, 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.

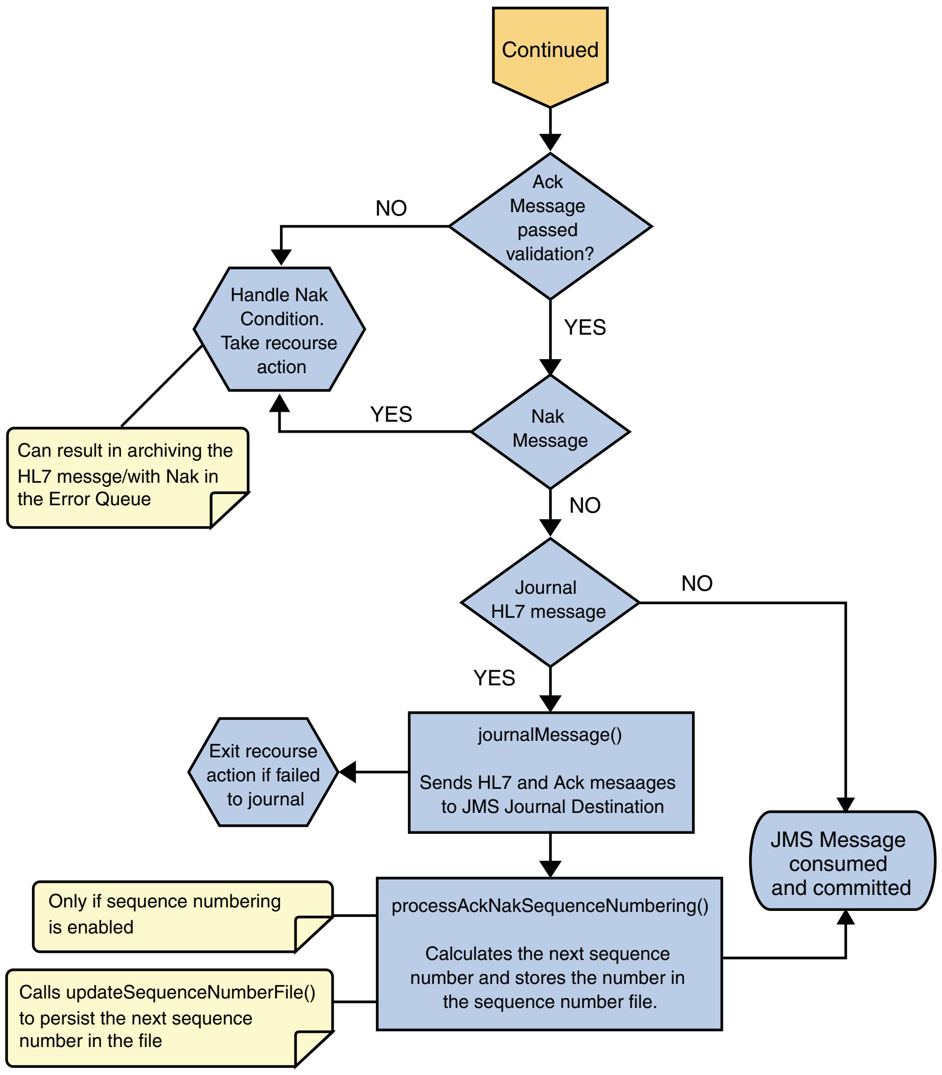

Continued
Ack Message passed validation?
NO
Handle Nak Condition. Take recourse action
YES
Can result in archiving the HL7 messge/with Nak in the Error Queue
YES
Nak Message
NO
Journal HL7 message
NO
YES
journalMessage()
Sends HL7 and Ack mesaages to JMS Journal Destination
Exit recourse action if failed to journal
JMS Message consumed and committed
Only if sequence numbering is enabled
processAckNakSequenceNumbering()
Calculates the next sequence number and stores the number in the sequence number file.
Calls updateSequenceNumberFile() to persist the next sequence number in the file

FIGURE 1–26 Immediate and Deferred Mode of ACK Process — Part 3

HL7 V3 Outbound Test Collaboration

For information, see "HL7 V2 Outbound Test Collaboration" on page 17.

Creating a Copy of an HL7 V3 Project

1. For more information and steps on the method to export a project, see "To Export a Project" on page 18.
2. For more information and steps on the method to import a project, see "Importing the TCP/IP Adapter Sample Projects" in *Sun Adapter for TCP/IP HL7 Tutorial*.

Customizing Predefined Collaborations for HL7 V3

The predefined Collaborations are designed to be extended and modified, however, for HL7 V3 compliant systems this is not necessary. If you need to modify an HL7 V3 Collaboration, it is strongly suggested that these template. Collaborations be used as the basis for any new Collaborations. Therefore, it is important to maintain the original predefined **jcdHL7V3Inbound** and **jcdHL7V3Outbound** Collaborations in their initial form for future use.

Note – Ensure that the required HL7 V3 SAR files are uploaded to the repository.

1. HL7eWay.sar
2. HL7OTDLibrary.sar
3. HL7V32006ACCTBilling.sar
4. HL7V32006ClaimsAndReimb.sar
5. HL7V32006ClinicalGenomics.sar
6. HL7V32006MedicalRecords.sar
7. HL7V32006MsgContActInfra.sar
8. HL7V32006PatientAdmin.sar
9. HL7V32006PersonnelManagement.sar
10. HL7V32006PublicHealthRepot.sar
11. HL7V32006QueryInfra.sar
12. HL7V32006RegulateStudies.sar
13. HL7V32006Scheduling.sar
14. HL7V32006SharedMessages.sar
15. HL7V32006TransInfra.sar

For complete instructions on installing the TCP/IP HL7 Adapter to the Repository and NetBeans, and on downloading the sample projects, see "Installing the TCP/IP HL7 Adapter and Sample Projects" in *Sun Adapter for TCP/IP HL7 Tutorial*. Download these sample projects: HL7 eWay Inbound Collaboration Project for HL7V3 PRPA IN403001UV01 Interaction over MLLPV2 and HL7 eWay Outbound Collaboration Project for HL7V3 PRPA IN403001UV01 Interaction over MLLPV2. Save the files to a local directory.

Creating Copies of an HL7 V3 Collaborations

For more information on the method to create copies of the HL7 V3 Collaborations, see "To Create Copies of an HL7 V2 Collaborations" on page 23.

The Connectivity Map of the newly added Collaborations can now be associated (bind) with the Project's components and is as shown.

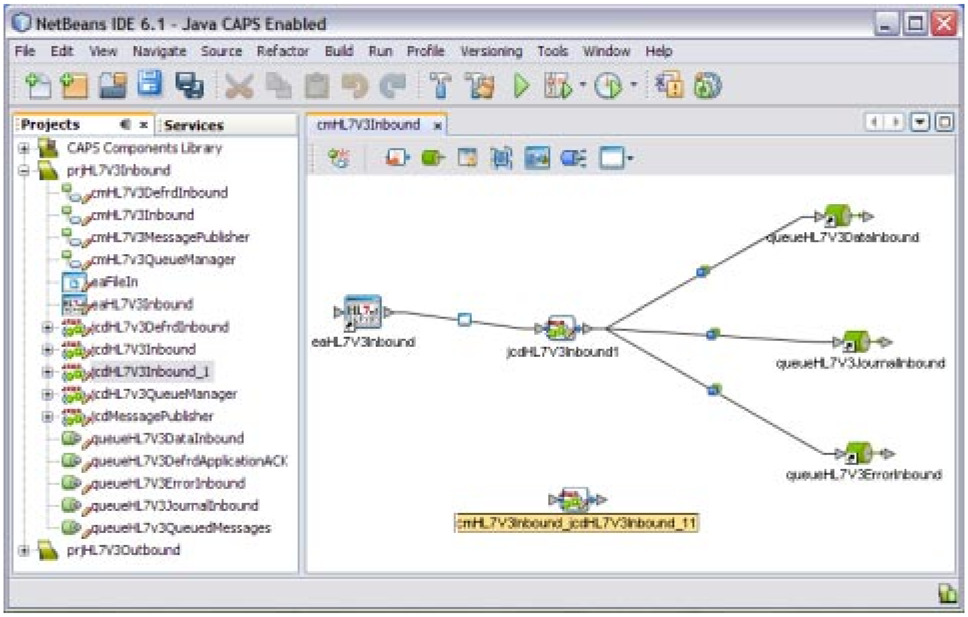


FIGURE 1–27 Connectivity Map — Newly Added Collaborations

Adding HL7 V3 OTD to an Existing Collaboration

In some cases, a specific HL7 V3 message or messages may need to be added to the Collaboration. Perform the following to add an HL7V3OTD to an existing Collaboration.

▼ To Add HL7 V3 OTD to an Existing Collaboration

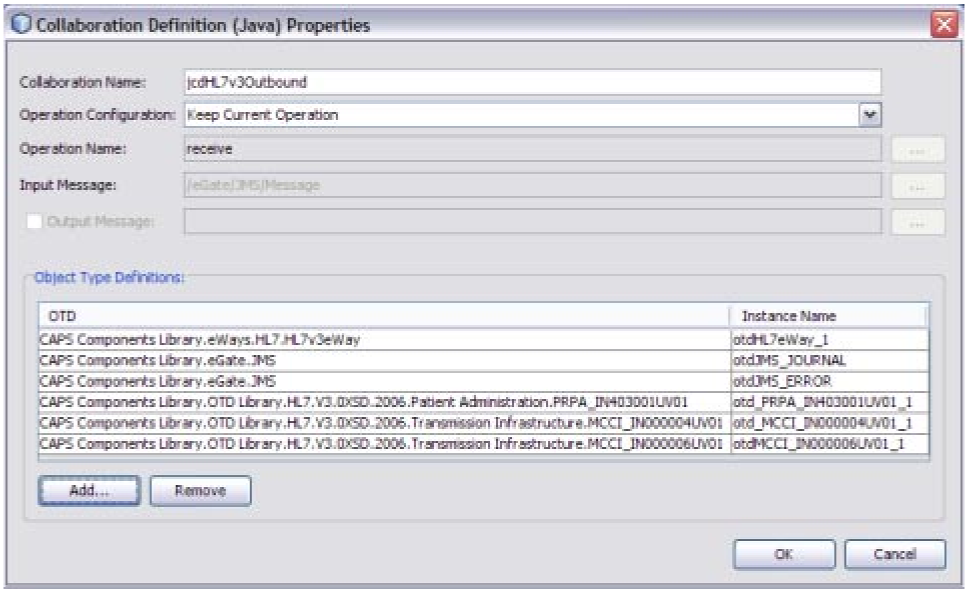
1 **Right-click the Collaboration from the Projects tab to which you want to add the HL7 OTD.**

For Example, **jcdHL7v3Outbound**.

2 **Select Properties from the drop-down menu.**

The Collaboration Definition (Java) Properties dialog box appears (for this example, the **jcdHL7v3Outbound** Collaboration.

3 **Select Keep Current Operation as the value for the Operation Configuration field.**


Collaboration Definition (Java) Properties

Collaboration Name: jcdHL7v3Outbound
Operation Configuration: Keep Current Operation
Operation Name: receive
Input Message: /eGate/JMS/Message
Output Message:

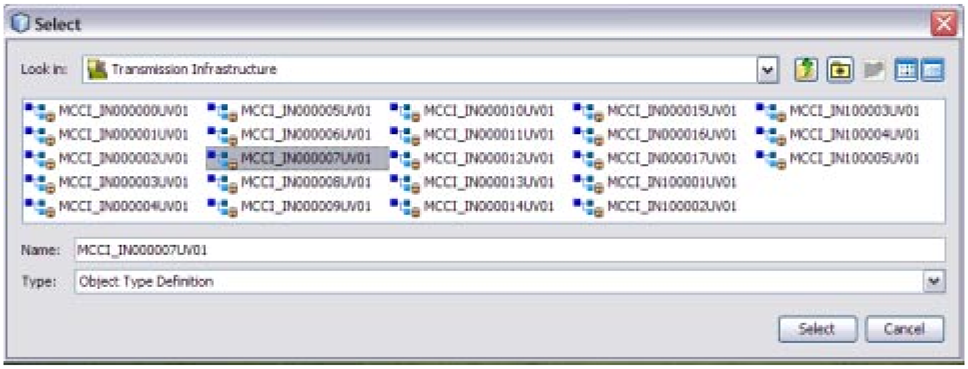
Object Type Definitions:

OTD	Instance Name
CAPS Components Library.eWays.HL7.HL7v3eWay	otdHL7eWay_1
CAPS Components Library.eGate.JMS	otdJMS_JOURNAL
CAPS Components Library.eGate.JMS	otdJMS_ERROR
CAPS Components Library.OTD Library.HL7.V3.0XSD.2006.Patient Administration.PRPA_IN403001UV01	otd_PRPA_IN403001UV01_1
CAPS Components Library.OTD Library.HL7.V3.0XSD.2006.Transmission Infrastructure.MCCI_IN000004UV01	otd_MCCI_IN000004UV01_1
CAPS Components Library.OTD Library.HL7.V3.0XSD.2006.Transmission Infrastructure.MCCI_IN000006UV01	otdMCCI_IN000006UV01_1

FIGURE 1–28 Collaboration Definition (Java) Properties

4 **Click Add to select the object type definition.**

Note – Click Remove to delete the OTD from the list.


Select

Look in: Transmission Infrastructure

MCCI_IN000000UV01	MCCI_IN000005UV01	MCCI_IN000010UV01	MCCI_IN000015UV01	MCCI_IN100003UV01
MCCI_IN000001UV01	MCCI_IN000006UV01	MCCI_IN000011UV01	MCCI_IN000016UV01	MCCI_IN100004UV01
MCCI_IN000002UV01	MCCI_IN000007UV01	MCCI_IN000012UV01	MCCI_IN000017UV01	MCCI_IN100005UV01
MCCI_IN000003UV01	MCCI_IN000008UV01	MCCI_IN000013UV01	MCCI_IN100001UV01	
MCCI_IN000004UV01	MCCI_IN000009UV01	MCCI_IN000014UV01	MCCI_IN100002UV01	

Name: MCCI_IN000007UV01

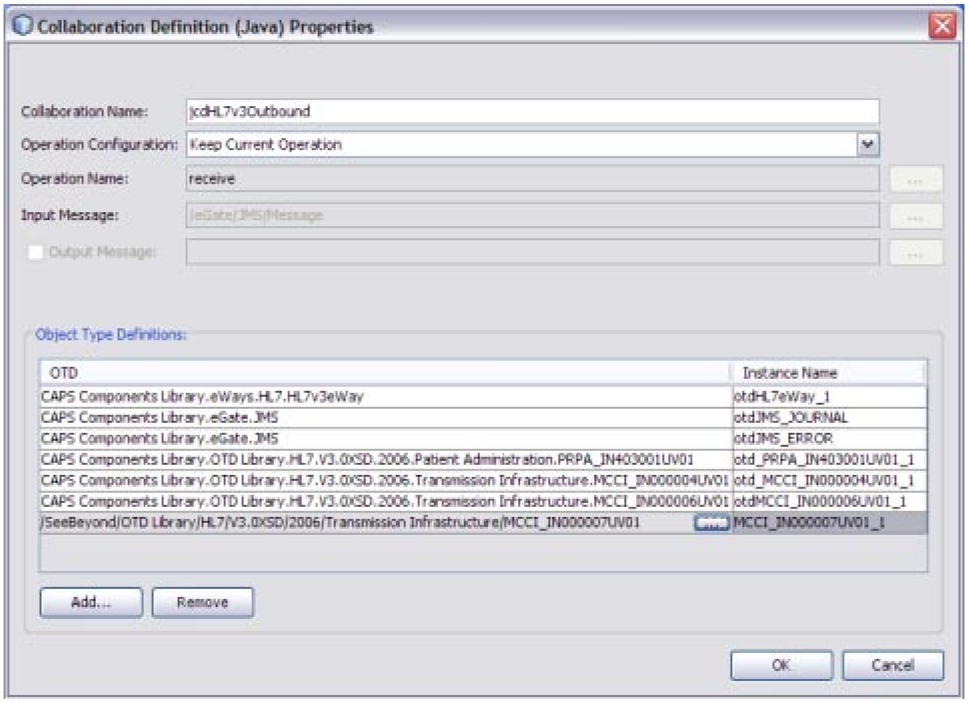
Type: Object Type Definition

Select Cancel

FIGURE 1–29 Select Object Type Definition

5 Select the OTD (for example, MCCI_IN000007UV01) from the Select dialog box and click Select.

The OTD is added to the list.


Collaboration Definition (Java) Properties

Collaboration Name: jcdHL7v3Outbound

Operation Configuration: Keep Current Operation

Operation Name: receive

Input Message: ;eGate/JMS/Message

Output Message:

Object Type Definitions:

OTD	Instance Name
CAPS Components Library.eWays.HL7.HL7v3eWay	otdHL7eWay_1
CAPS Components Library.eGate.JMS	otdJMS_JOURNAL
CAPS Components Library.eGate.JMS	otdJMS_ERROR
CAPS Components Library.OTD Library.HL7.V3.0XSD.2006.Patient Administration.PRPA_IN403001UV01	otd_PRPA_IN403001UV01_1
CAPS Components Library.OTD Library.HL7.V3.0XSD.2006.Transmission Infrastructure.MCCI_IN000004UV01	otd_MCCI_IN000004UV01_1
CAPS Components Library.OTD Library.HL7.V3.0XSD.2006.Transmission Infrastructure.MCCI_IN000006UV01	otdMCCI_IN000006UV01_1
/SeeBeyond/OTD Library/HL7/V3.0XSD/2006/Transmission Infrastructure/MCCI_IN000007UV01	MCCI_IN000007UV01_1

Add... Remove

OK Cancel

FIGURE 1–30 OTD Added to the List

Note – To Modify the Existing HL7 V3 OTD

Click the ellipses and select appropriate Projects folder (for example, CAPS Components Library) from the Select dialog box to modify the existing HL7 V3 OTD.

MLLP V2

The abbreviation of MLLP V2 is Minimal Lower Layer Protocol Release 2. MLLP V2 is a requirement for all Message Transport protocols used to transport HL7 Version 3 content.

MLLP V2 Content Exchange Model

MLLP Release 2 is a reliable Message Transport protocol. It guarantees In Order delivery and At Least Once delivery of HL7 Content. HL7 Content is framed in a Block and sent to the Destination system. The Destination system acknowledges the receipt of the message by returning a Commit Acknowledgement message. The MLLP V2 acknowledgement protocol is synchronous: the Source system shall not send new HL7 content until an acknowledgement for the previous HL7 Content has been received.

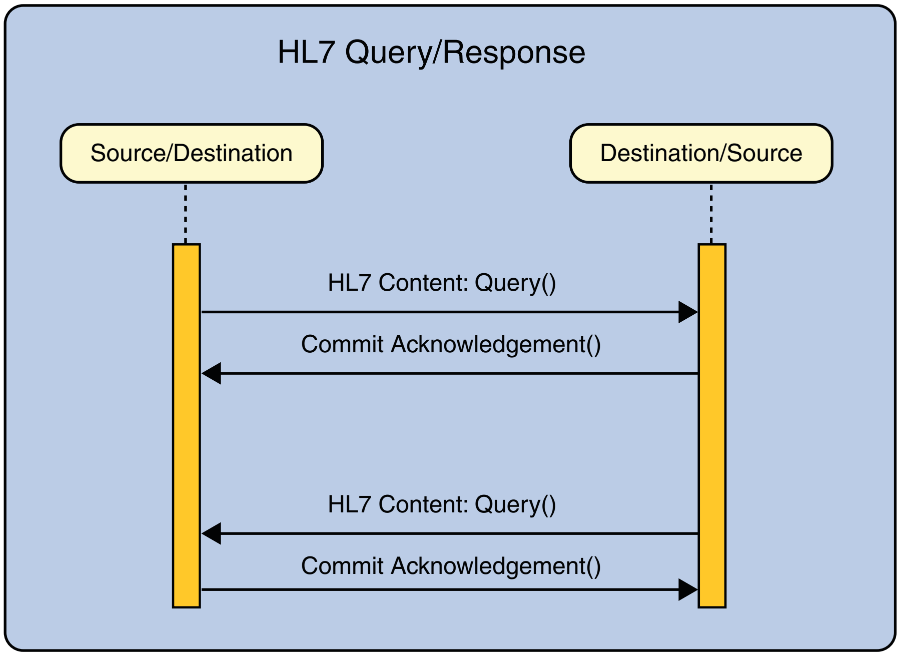


FIGURE 1–31 Interaction Diagram

All HL7 Content (of any kind or type) is framed in a Block and sent to the Destination system. The Destination system acknowledges the receipt of the Block by returning a Commit Acknowledgement message. If the HL7 Content (a query in the example below) triggers the sending of HL7 Content (a Response) by the Destination system, then this HL7 Content is framed in a Block and sent. MLLP has no knowledge of the HL7 Content, nor does it base any part of its behavior on HL7 Content.

In HL7 adapter, a database is used to persist the message.

In Inbound Mode, once the message is received by the HL7 adapter, it is persisted into the database and a commit acknowledgement is sent to the sender before the HL7 message is passed to the collaboration level. If the persistence fails, a negative acknowledgement is sent to the sender and the collaboration receives null as its message. Inside the collaboration, the message received is null, the collaboration returns without proceeding.

If the message received is a duplicate message, and if an ACK/NAK is already available in the persistence database, the same is retrieved from the database sent to the sender. The collaboration receives null as the received message.

In Outbound Mode, the message is sent to the sender and waits for the commit acknowledgement and/or negative acknowledgement from the receiver. If the adapter receives a negative acknowledgement, the message is sent again until the configured Maximum Number of Retries is made.

Standard Inbound HL7 V2 Collaboration Overview over MLLPV2

The Inbound HL7 V2 Collaboration, **jcdHL7Inbound**, contains OTDs for the HL7 Resource Adapter, JMSData, 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.

HL7 V2 Standard Inbound Message Mode Data Flow over MLLPV2 — Part 1

Once the message is received, the Collaboration determines whether the message needs to be validated. The HL7 V2 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 adapter sends a NAK.

The validated HL7 V2 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 processed).

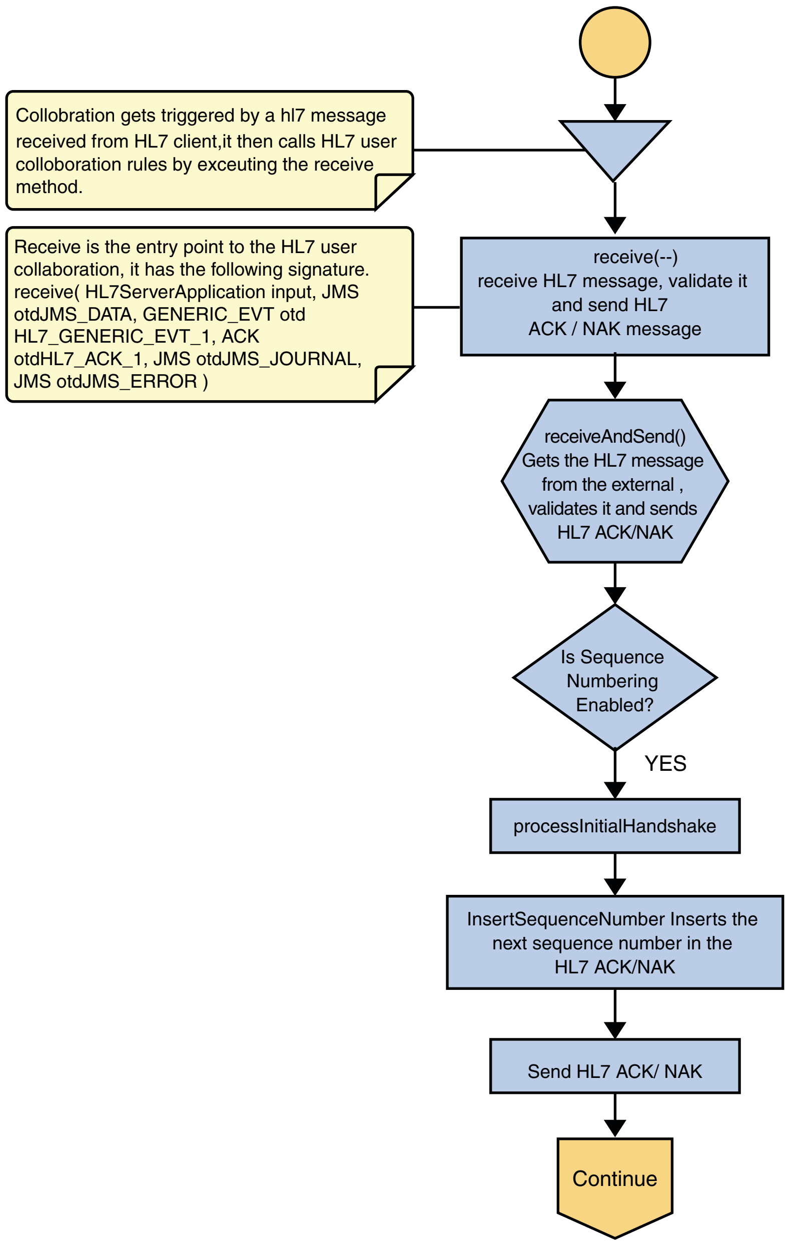

Collobration gets triggered by a hl7 message received from HL7 client,it then calls HL7 user colloboration rules by exceuting the receive method.
Receive is the entry point to the HL7 user collaboration, it has the following signature. receive(HL7ServerApplication input, JMS otdJMS_DATA, GENERIC_EVT otd HL7_GENERIC_EVT_1, ACK otdHL7_ACK_1, JMS otdJMS_JOURNAL, JMS otdJMS_ERROR)
receive(--) receive HL7 message, validate it and send HL7 ACK / NAK message
receiveAndSend() Gets the HL7 message from the external , validates it and sends HL7 ACK/NAK
Is Sequence Numbering Enabled?
YES
processInitialHandshake
InsertSequenceNumber Inserts the next sequence number in the HL7 ACK/NAK
Send HL7 ACK/ NAK
Continue

FIGURE 1–32 HL7 V2 Standard Inbound Message Mode Data Flow over MLLPV2 — Part 1

HL7 V2 Standard Inbound Message Mode Data Flow over MLLPV2 — Part 2

The Collaboration receives the HL7 V2 message from the external using **receiveHL7message()**. If an exception occurs due to incomplete data, and the adapter 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.

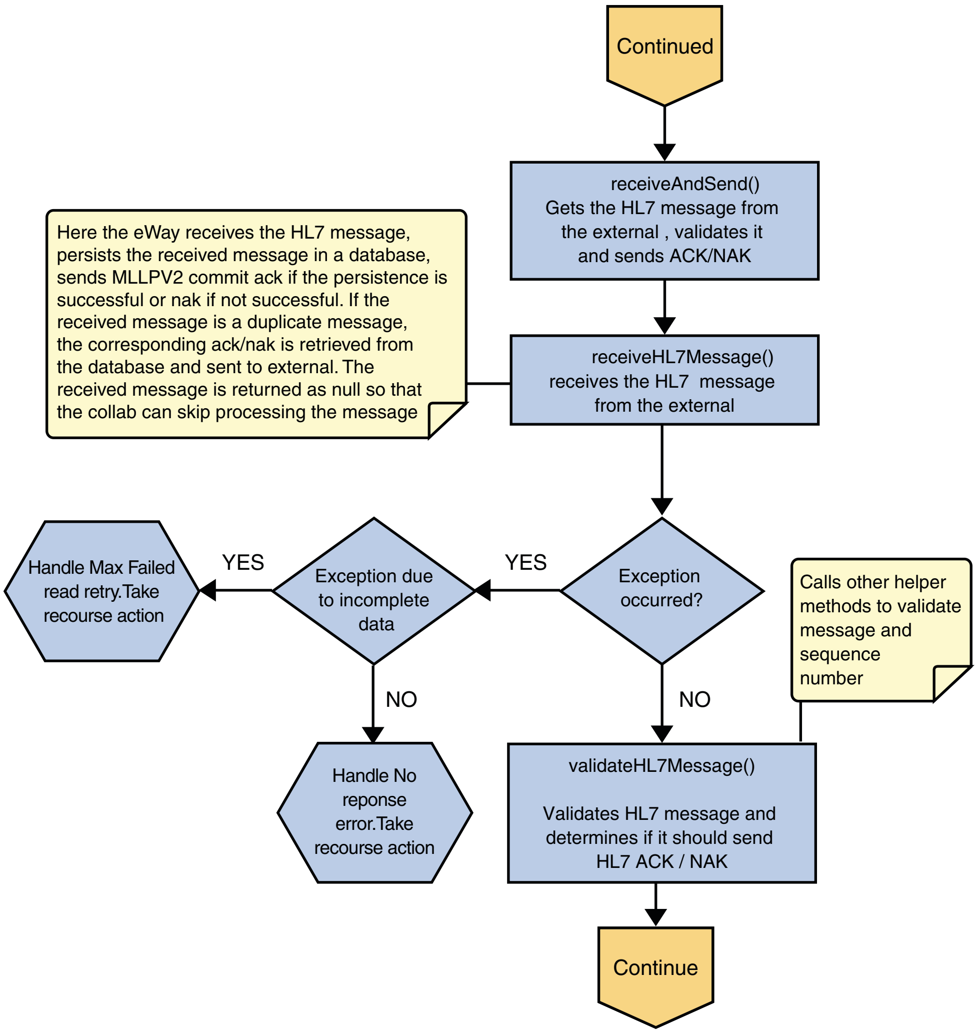

Continued
receiveAndSend()
Gets the HL7 message from the external , validates it and sends ACK/NAK
Here the eWay receives the HL7 message, persists the received message in a database, sends MLLPV2 commit ack if the persistence is successful or nak if not successful. If the received message is a duplicate message, the corresponding ack/nak is retrieved from the database and sent to external. The received message is returned as null so that the collab can skip processing the message
receiveHL7Message()
receives the HL7 message from the external
Handle Max Failed read retry.Take recourse action
YES
Exception due to incomplete data
YES
Exception occurred?
Calls other helper methods to validate message and sequence number
NO
NO
Handle No reponse error.Take recourse action
validateHL7Message()
Validates HL7 message and determines if it should send HL7 ACK / NAK
Continue

FIGURE 1–33 HL7 V2 Standard Inbound Message Mode Data Flow over MLLPV2 — Part 2

HL7 V2 Standard Inbound Message Mode Data Flow over MLLPV2 — Part 3

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.

If the HL7 message passes validation, the Collaboration calls **makeAck()** and **sendHL7Ack()** to create and send the ACK to the external.

The HL7 ACK/NAK are stored in the persistent database. This ACK/NAK is stored against the Inbound HL7 message received.

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.

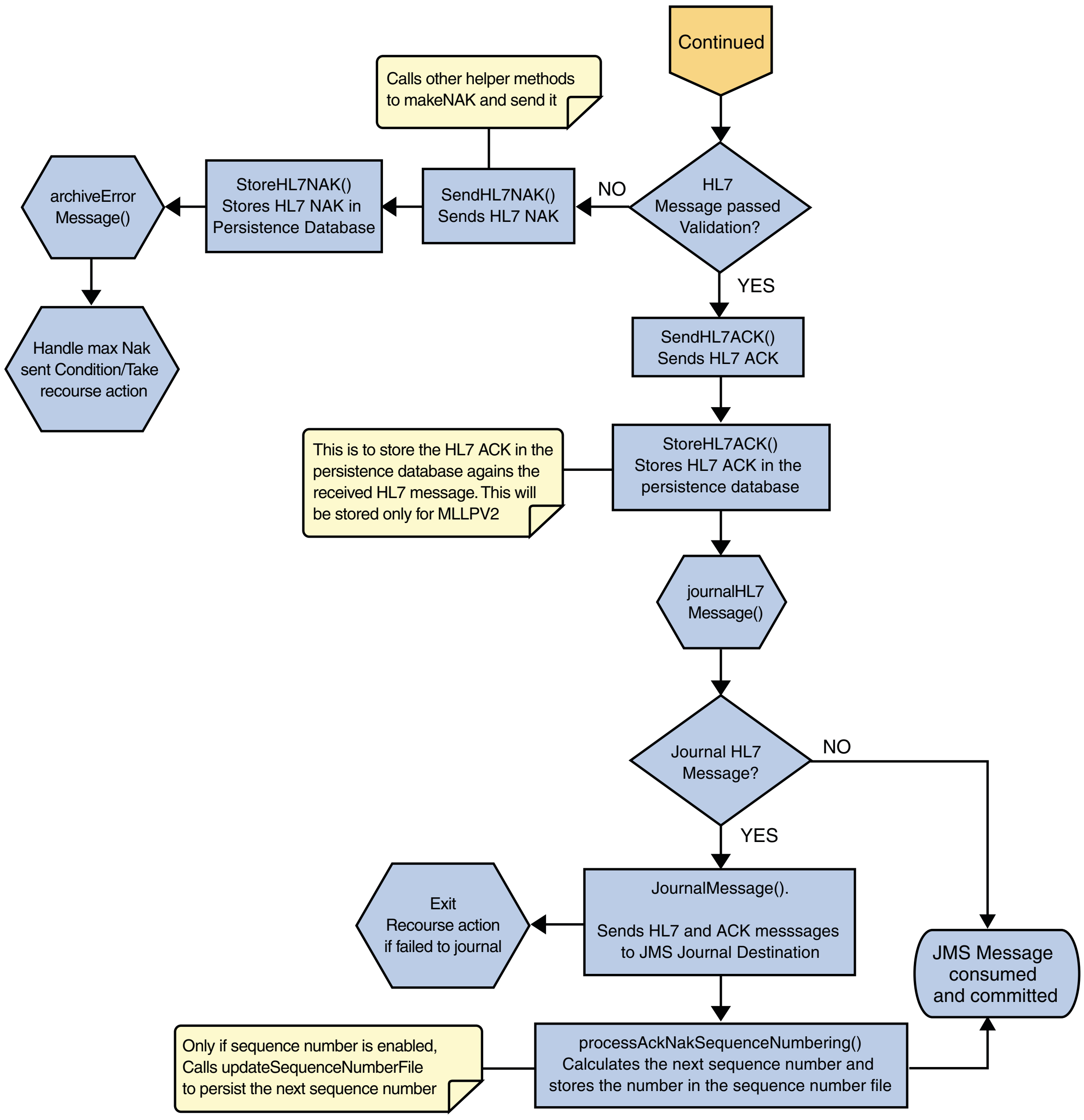

Continued
HL7 Message passed Validation?
NO
YES
Calls other helper methods to makeNAK and send it
SendHL7NAK() Sends HL7 NAK
StoreHL7NAK() Stores HL7 NAK in Persistence Database
archiveError Message()
Handle max Nak sent Condition/Take recourse action
SendHL7ACK() Sends HL7 ACK
This is to store the HL7 ACK in the persistence database agains the received HL7 message. This will be stored only for MLLPV2
StoreHL7ACK() Stores HL7 ACK in the persistence database
journalHL7 Message()
Journal HL7 Message?
NO
YES
JournalMessage(). Sends HL7 and ACK messsages to JMS Journal Destination
Exit Recourse action if failed to journal
JMS Message consumed and committed
Only if sequence number is enabled, Calls updateSequenceNumberFile to persist the next sequence number
processAckNakSequenceNumbering() Calculates the next sequence number and stores the number in the sequence number file

FIGURE 1–34 HL7 V2 Standard Inbound Message Mode Data Flow over MLLPV2 — Part 3