# Oracle® Java CAPS HL7 Binding Component User's Guide

Part No: 821–2635
January 2011

ORACLE

120126@25097

# Contents

# Oracle HL7 Binding Component User's Guide

The HL7 Binding Component enables communication between HL7 servers and clients using a JBI environment. The document contains the following topics:



## About the HL7 Binding Component

The HL7 Binding Component enables Java CAPS to establish and maintain connection with HL7 messaging systems, manage message enveloping and routing, perform message validations, and implement optional HL7 protocol sequence numbering. The following topics provide overview information for the HL7 messaging standard and the HL7 BC:



### The Health Level 7 Messaging Standard

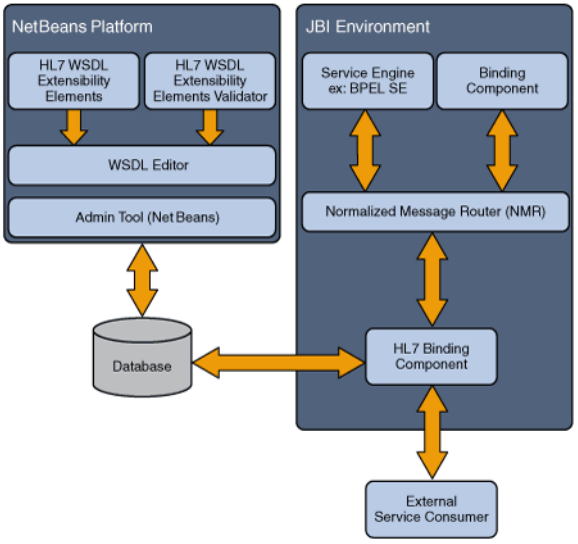
The HL7 Messaging Standard has been around for over 20 years, providing administrative, financial, and clinical messaging standards for the health care industry. This sharing of electronic health care information using the HL7 protocol allows hospitals to streamline all areas of patient care, communicating this information between numerous computer systems and applications from different departments and agencies. The problem is that these various

systems and applications are not designed to communicate with each other. The HL7 Binding Component provides a solution by configuring and connecting the HL7 servers and clients within a JBI environment.

# The HL7 Binding Component

The HL7 Binding Component supports HL7 Version 2 Messaging Standard (2.1, 2.2, 2.3, 2.3.1, 2.4, 2.5, 2.5.1, and 2.6). HL7 Version 3 can be implemented using other JBI components, such as the HTTP Binding Component using the SOAP protocol.

The HL7 Binding Component enables communication between HL7 servers and clients within a JBI environment. The binding component converts HL7 messages to XML messages within the JBI environment, and converts them back to HL7 messages. It provides both design-time and runtime functionality. The design-time portion uses a set of well-defined WSDL extensibility elements to configure HL7 service endpoints and message mapping. The runtime portion provides connectivity between the JBI framework and external HL7 clients and servers.

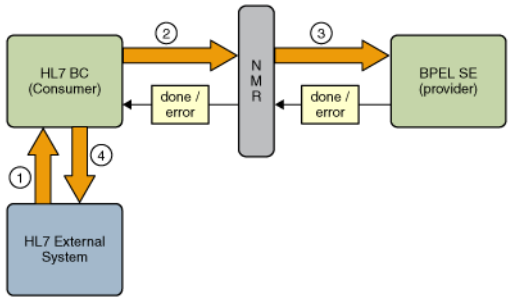



## HL7 Binding Component as a Consumer

Inbound, in the role of consumer, the HL7 Binding Component (BC) acts as follows:

1. The HL7 BC receives the HL7 message from an external HL7 system.
2. The HL7 BC parses the received message. If parsing fails, the HL7 BC sends a NAK to the external HL7 sender.

3. If validate MSH is enabled, the HL7 BC validates the MSH segment fields for message acceptance. If validation fails, the HL7 BC sends a NAK to the external HL7 sender.
4. If sequence number is enabled, the HL7 BC validates the MSH-13–Sequence Number field in the received message against the persisted sequence in the database for the inbound endpoint. If the sequence number does not match, the HL7 BC sends a NAK to the external HL7 message sender.
5. If the HL7 message proves acceptable, the HL7 BC sends the normalized message to the BPEL Service Engine. An ACK or NAK is sent to the external HL7 message sender, based on the message exchange status received from the Normalized Message Router, as follows:
   - An ACK is sent when the message exchange status is Done.
   - A NAK is sent when the message exchange status is Error.




1.HL7 Raw Message
2. Normalized XML HL7 Message
3. Normalized XML HL7 Message
4. HL7 Application ACK/NAK

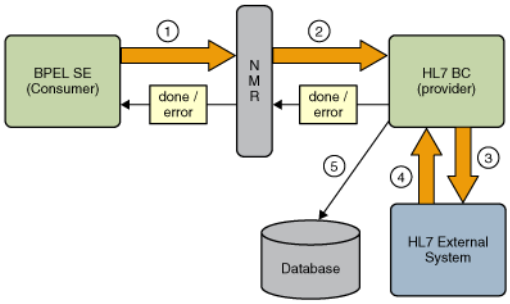
## HL7 Binding Component as a Provider

Outbound, in the role of provider, the HL7 Binding Component acts as follows:

1. The HL7 BC receives the HL7 message from the BPEL Service Engine.
2. If validate MSH is enabled, the HL7 BC validates the MSH segment fields, MSH-11–Processing ID and MSH-12–Version ID, against the configured Processing ID and Version ID. If validation fails, the HL7 BC sets the message exchange status to Error and sends the message back to the Normalized Message Router.
3. If sequence number is enabled, the HL7 BC inserts the sequence number in the message.
4. The HL7 BC converts the normalized message to the HL7 encoded form and sends it to the external HL7 system.
5. The HL7 BC receives the acknowledgement from the HL7 external system, and determines whether the acknowledgement is an ACK or NAK, and responds as follows:
   - An ACK is sent when the message exchange status is Done.

- A NAK is sent when the message exchange status is Error.

6. If the HL7 message proves acceptable, the HL7 BC sends the normalized message to the BPEL Service Engine. An ACK or NAK is sent to the external HL7 message sender, based on the message exchange status received from the Normalized Message Router, as follows:
   - For an ACK, the HL7 BC sets the message exchange status to Done and sends it to the BPEL Service Engine. The ACK is also stored in a database.
   - For a NAK, the HL7 BC sets the message exchange status to Error and sends it to the BPEL Service Engine. The NAK is also stored in a database.




1. Normalized XML HL7 Message
2. Normalized XML HL7 Message
3. HL7 Raw Message
4. HL7 Application ACK/NAK
5. Stores the ACK/NAK to DB (MCID | Message)

## HL7 Acknowledgements

The HL7 protocol specifies that the system that receives an HL7 message will acknowledge the message by sending an acknowledgement to the message sender. HL7 protocol supports two types of message acknowledgement, Original Mode and Enhanced Mode.

- **Original Acknowledgement Mode:** Original mode acknowledges message delivery to a receiving system by requiring an Application Acknowledgement. This is an acknowledgement from the system's application layer, sent after the application has successfully processed the message.
- **Application Acknowledgement:** The receiving system sends an acknowledgement to the message sender, acknowledging that the application has successfully processed the message.
- **Enhanced Acknowledgement Mode:**In case of Enhanced mode, the HL7 external system processes two acknowledgments, the Accept Acknowledgment and the Application Acknowledgment described above.

- **Accept Acknowledgement:** The receiving system sends an acknowledgement to the message sender, acknowledging that the message has been committed to safe storage and that there is no need to resend the message. The binding component can also be set to check the payload (MSH.15) of incoming HL7 messages to see if an acknowledgment needs to be generated.

Valid Acknowledgment condition values for MSH-15 and MSH-16 files in an HL7 messages are:

- **ALWAYS_CONDITION (AL)** – Always sends the ACK whether it is a success or not.
- **NEVER_CONDITION (NE)** – The sender does not require an ACK, so the receiver should not send any ACKs back to the sender.
- **ERROR_CONDITION (ER)** – If any errors occur during processing, the receiver sends an ACK; otherwise, the ACK is not sent.
- **SUCCESS_CONDITION** – The receiver sends an ACK only for a success.

### Sequence Numbering

The HL7 protocol uses the sequence number protocol to avoid duplication of messages between sending and receiving systems. The HL7 Binding Component can either play the role of the service consumer (that is, the inbound service endpoint) or the service provider (that is, the outbound service endpoint). In both roles, it supports the sequence numbering protocol for HL7 message transactions.

The negotiation and incrementing of the sequence number is automatically performed by the binding component. The HL7 Binding Component receives the message with the sequence number from the external system, validates the message based on the excepted sequence number stored in the database, and sends a positive or negative acknowledgment with the expected sequence number back to the external system. Sequence numbering is enabled or disabled as an attribute of the HL7 `protocolproperties` WSDL element.

## HL7 Binding Component Features

The HL7 Binding Component includes the following features:

- **HL7 Version Support:** Supports HL7 Versions 2.1, 2.2, 2.3, 2.3.1, 2.4, 2.5, 2.5.1, and 2.6.
- **Design Time WSDL Semantic Validation:** Supports WSDL validation at design-time for issues such as attribute values, relationship between extensibility elements, and so forth.
- **Symmetrical WSDL For Service Definition:** Both the service consumer and service provider use the same WSDL in their implementation of extensibility elements `hl7:binding`, `hl7:operation`, `hl7:message`, `hl7:address`, and `hl7:protocolproperties`.
- **Service Orchestration:** Business orchestration can specified as a service consumer or service provider.

- **Lower Layer Protocols:** Supports MLLP v1.0, MLLP v2.0, and HLLP Message Transport Protocols.
- **Transport Protocols:** Supports the TCP/IP Transport Protocol.
- **Acknowledgment Modes:** Supports both Original Mode Acknowledgement and Enhanced Mode Acknowledgement.
- **Message Validation and MSH Segment Validation:** Validates messages for syntactical correctness and MSH segment fields for message acceptance.
- **Accept Acknowledgment Restraint in Enhanced Mode:** Checks the payload (MSH.15) of incoming HL7 messages and determines whether an acknowledgment is required.
- **Journaling and Persistence Support:** Supports default Axion database, Derby, MySQL 5.1, and Oracle 11*g* databases.
- **HL7 XML Messages Support:** Supports processing the XML version of HL7 messages without depending on the HL7 Encoder when the "Literal" option in encoding details section is selected when the HL7 WSDL file is created.
- **HL7 Recourse Actions:** Supports communication controls and recourse actions.
- **Sequence Number Protocol:** Supports sequence number protocol when the binding component is set as a service consumer or service provider in the business orchestration.

---

**Note –** Since the HL7 Binding Component uses the Service Unit path as the primary key in the database table, and a Service Unit is a collection of service endpoints, only one sequence number enabled service endpoint is supported for each Service Unit.

---

# Using the HL7 Binding Component Wizard

The HL7 Binding Component is similar to other binding components in that you use a wizard that steps you through creating the binding WSDL document. Each of these processes bring you to the HL7 Binding Component Wizard.

---

**Note –**

There are two types of WSDL documents. The Concrete WSDL Document option is the only way to create an HL7 Binding Component.

- **Abstract WSDL Document:** An XML document that contains interface information that defines the structure of the method input properties, returned output types, exceptions or fault types, and the port type or interface. It does not contain any concrete data structures such as transport protocols or network address locations. The abstract type only requires that you fill out the Abstract Configuration page of the wizard.
- **Concrete WSDL Document:**An XML document that includes the bindings to protocols, concrete address locations and service elements, along with the abstract WSDL information. The concrete type requires you to select a Binding and a Binding Type. Additional configuration pages are added to the wizard, associated with the binding and type you select, in this case, the HL7 Binding.

The following topics provide information and instructions for accessing and using the HL7 Binding Component Wizard:


- "Accessing the HL7 Binding Component Wizard" on page 11
- "Configuring the HL7 Binding Component in the Wizard" on page 13


# Accessing the HL7 Binding Component Wizard

You can access the HL7 Binding Component Wizard and create an HL7 binding by performing any of the following procedures.


- "To Access the Wizard by Creating a New File" on page 11
- "To Access the Wizard by Creating a New Binding" on page 12
- "To Access the Wizard by Creating a New WSDL Document" on page 12


## ▼ To Access the Wizard by Creating a New File

1. **Select your project in the NetBeans IDE Projects window, then click the New File icon, or select File → New File from the NetBeans menu.**

   Ctrl+N also opens the New File Wizard.

2. **Select ESB as the Category, and select Binding as the File Type.**

3. **Click Next.**

   The Name and Location window appears.

4. **Enter a name for the BC, select HL7 in the Binding field, and select the type of HL7 binding in the Type field (inbound or outbound).**

5 **If necessary, you can also modify the Folder and Target Namespace fields.**
The target namespace is the URL String used as the target namespace.

6 **Continue to "Configuring the HL7 Binding Component in the Wizard" on page 13.**

## ▼ To Access the Wizard by Creating a New Binding

1 **Right-click your project in the Projects tree, and select New → Other.**
The New File Wizard appears.

2 **Select ESB as the Category, and Binding as the File Type.**

3 **Click Next.**
The Name and Location window appears.

4 **Enter a name for the BC, select HL7 in the Binding field, and select the type of HL7 binding in the Type field (inbound or outbound).**

5 **If necessary, you can also modify the Folder and Target Namespace fields.**
The target namespace is the URL String used as the target namespace.

6 **Continue to "Configuring the HL7 Binding Component in the Wizard" on page 13.**

## ▼ To Access the Wizard by Creating a New WSDL Document

1 **Right-click your project in the Projects tree, and select New → WSDL Document.**
The New WSDL Document Wizard appears.

2 **Enter a File Name, and select Concrete WSDL Document as the WSDL Type.**
This adds additional options to the wizard.

3 **In the Binding field, select HL7; and in the Type field, select the type of HL7 Binding you want to create (inbound or outbound).**

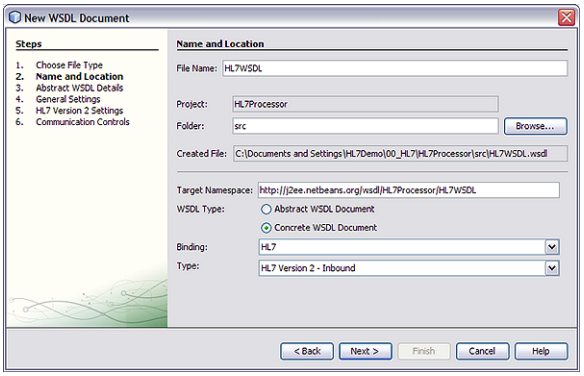
4 **If necessary, you can also modify the Folder and Target Namespace fields.**
The target namespace is the URL String used as the target namespace.

5 **Continue to "Configuring the HL7 Binding Component in the Wizard" on page 13.**

# Configuring the HL7 Binding Component in the Wizard

From the HL7 Binding Component Wizard you can configure an inbound or outbound HL7 Binding Component. The HL7 Binding Component supports the following types of binding:

- **Inbound Request**: A request message and MSH.9 field type, and inbound communication controls.
- **Inbound Request Reply**: A request message and MSH.9 field type, response message, and inbound communication controls.
- **Outbound Request**: A request message and outbound communication controls.
- **Outbound Request Reply**: A request message, a response message, and outbound communication controls.



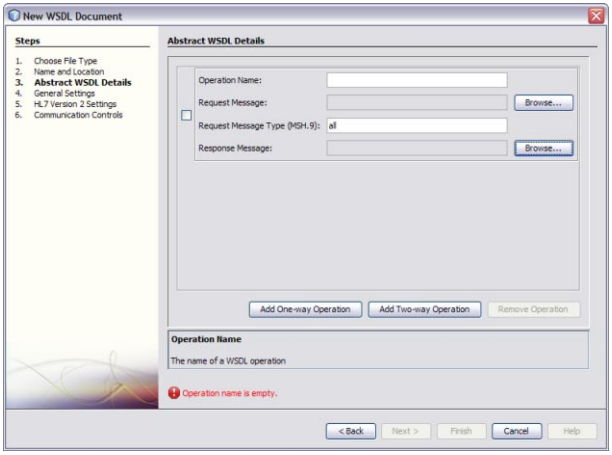

## To Configure the HL7 Binding Component in the Wizard

1 **Complete one of the procedures under "Accessing the HL7 Binding Component Wizard" on page 11.**

You should now be on the Name and Location window of the New WSDL Document Wizard, shown above.

2 **Click Next.**

The Abstract WSDL Details window appears.




3 **Do one of the following:**

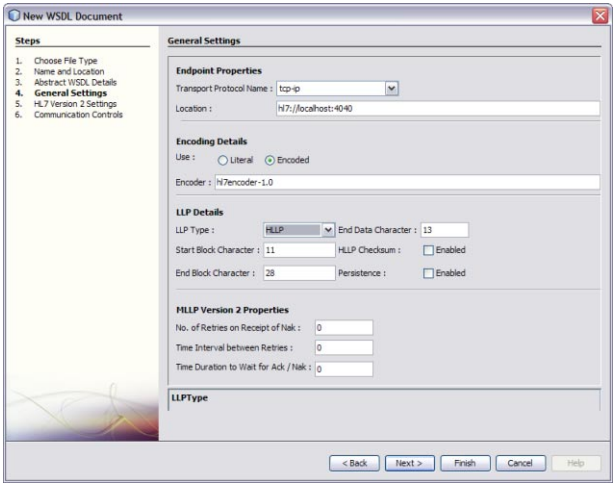
- **To implement an inbound or outbound request operation, select Add One-Way Operation.**
- **To implement an inbound or outbound request response operation, select Add Two-Way Operation.**

The fields to configure the type of operation you chose appear.

4 **Fill in the abstract WSDL properties as described in "Abstract WSDL Properties" on page 16.**

**5 Click Next.**

The General Settings windows appears.

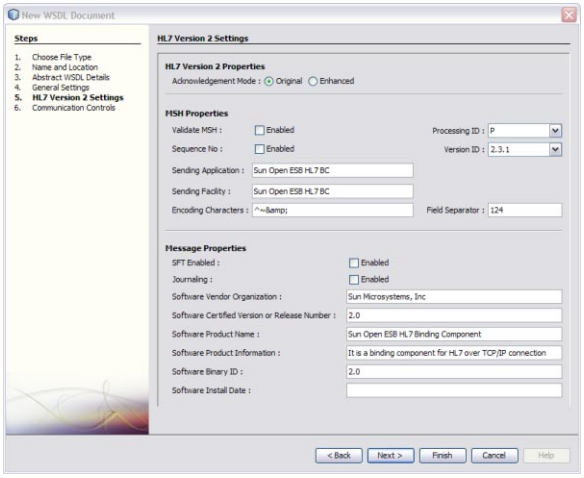



**Note –** The image above shows the properties for an outbound binding type. There are fewer fields for the inbound type.

**6 Fill in the general settings properties as described in "General Settings" on page 17.**

**7 Click Next.**

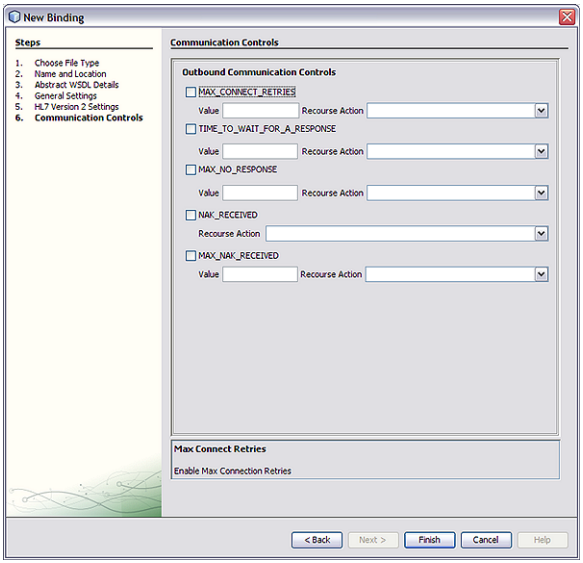
The HL7 Version 2 Settings window appears.



8 **Fill in the HL7 version 2 settings properties as described in "HL7 Version 2 Settings" on page 19.**

9 **Click Next.**

The Communication Controls window appears.




**Note –** The image above shows the properties for an outbound binding type. There are fewer properties for the inbound binding types.

10 **Fill in the communication controls properties as described in "Communication Controls" on page 21.**

11 **Click Finish.**

The WSDL file is created and appears inn the Projects window. The WSDL Editor appears, displaying the content of the file.

## Abstract WSDL Properties

The Abstract WSDL Details page of the wizard specifies the request and response messages and the message type specified as the MSH.9 field value. The options on this page vary depending on the HL7 Binding type. The Response Message Type field only appears for two-way operations.

| Property | Description |
|---|---|
| Operation Name | A name for the operation in the WSDL document. |

| Property | Description |
|---|---|
| Request Message | The HL7 XSD element or type used for the request message. |
| Request Message Type | The HL7 message type that the operation handles. This value should be the same as the message type entered in the MSH.9 field value. |
| Response Message Type | The HL7 XSD element or type used for the response message. |

## General Settings

The General Settings page of the HL7 Binding Wizard specifies the connection protocol and encoding details. The Outbound HL7 Bindings also include configuration properties for LLP message transport protocols.

The general settings are categorized into the following property types:


- "Endpoint properties" on page 17
- "Encoding Details" on page 17
- "LLP Details" on page 18
- "MLLP Version 2 Properties (Outbound Only)" on page 18


### Endpoint properties

The Endpoint properties specify the connectivity information to the HL7 external system, and include the following configuration properties.

| Property | Description |
|---|---|
| Transport Protocol Name: | The transport protocol used to transmit the HL7 message payload. For HL7 v2.X the only current transport protocol option is TCP/IP |
| Location | A URL used to connect to the HL7 external system. |

### Encoding Details

The Encoding Details specify the concrete properties associated with receiving or sending HL7 messages from or to the HL7 external system. The Encoding Details section include the following properties.

| Property | Description |
|---|---|
| Use | The use type, which affects how a message is interpreted. The options are Literal or Encoded. The Encoded option specifies that the message is encoded using a well-defined encoding style. |
| Encoder | The encoding type associated with the message. This also defines the encoder type used to process the encoded data. |

## LLP Details

The LLP Details section defines the Lower Layer Protocol (LLP) used for the HL7 messages, and includes the following properties.

| Property | Description |
|---|---|
| LLP Type: | The Lower Layer Protocol (LLP) type. Select one of the following types:<br>■ MLLP v1.0 (Minimal Lower Layer Protocol)<br>■ HLLP (Hybrid Lower Layer Protocol)<br>■ MLLP v2.0 (Minimal Layer Protocol v2.0) |
| Start Block Character | The Start Block Character value in a decimal ASCII number from 1 to 127. Unless there is a conflict, the value should be ASCII VT, which is decimal 11. |
| End Block Character | The End Block Character value in decimal ASCII number from 1 to 127. To be strictly comply with the HL7 standard, this parameter must be set to a carriage return, which is decimal 13. |
| End Data Character | The End Data Character value in decimal ASCII number from 1 to 127. To be strictly compliant , the value should be a carriage return, indicated by decimal 28. |
| HLLP Checksum | An indicator of whether the HLLP Checksum feature is enabled. This can only be enabled if the LLP Type is set to HLLP. |
| Persistence | An indicator of whether to store the HL7 message and ACKs into the persistence storage for recovery. Persistence can only be enabled for HLLP and MLLP v1.0; MLLP v2.0 has its own persistence. |

## MLLP Version 2 Properties (Outbound Only)

The MLLP Version 2 Properties are for outbound message only, and configure the Minimal Lower Layer Protocol (MLLP) version 2 property. This section includes the following properties.

| Property | Description |
|---|---|
| No. of Retries on Receipt of NAK | The maximum configured number of attempts to send a message after receiving a negative acknowledgement. |
| Time Interval Between Retries | The duration in milliseconds to wait between each retry attempt. |
| Time Duration to Wait for ACK / NAK | The duration in milliseconds to wait for MLLP V2 commit positive or negative acknowledgement. |

## HL7 Version 2 Settings

The HL7 Version 2 Settings are divided into the following property types:



### HL7 Version 2 Properties

The HL7 Version 2 Properties section includes the Acknowledgement Mode property.

| Property | Description |
|---|---|
| Acknowledgement Mode | The type of acknowledgement used with HL7 transaction. Choose one of the following options:<br>■ **Original:** An acknowledgement is sent from the receiving system's application layer once the application has successfully processed the message. This is also called the Application Acknowledgement.<br>■ **Enhanced:** Both the Application Acknowledgement (Original Acknowledgement) and an Accept Acknowledgement are sent from the receiving system. The Accept Acknowledgement is sent once the message has been committed to safe storage and acknowledges that there is no need to resend the message. |

### MSH Properties

The MSH Properties section provides the HL7 MSH Header segment settings. It includes the following properties.

| Property | Description |
|---|---|
| Validate MSH | An indicator of whether MSH segment validation is enabled. |
| Sequence No | An indicator of whether sequence number protocol is enabled. |
| Processing ID | The Processing ID value by which the MSH-11 segment in the received message is validated when validateMSH is enabled. MSH-11 is used to indicate whether a message is processed as defined in the HL7 Processing rules.<br>The Processing ID options include the following:<br>■ **D** - Debugging<br>■ **T**- Training<br>■ **P**- Production |
| Version ID | The HL7 Version ID value by which the MSH-12 segment in the received message is validated when validateMSH is enabled. The Version ID options are 2.1, 2.2, 2.3, 2.3.1, 2.4, 2.5, 2.5.1, and 2.6. |

| Property | Description |
|---|---|
| Sending Application | The sending application (MSH-03 segment) used to create the NAK for invalid HL7 messages. This is used to help identify this application from other participating applications within the network enterprise. |
| Sending Facility | The sending facility (MSH-04 segment) responsible for creating the NAK for invalid HL7 messages. This is used to help further identify participating facilities within the network enterprise. |
| Encoding Characters | The four encoding characters used to create the NAK for an invalid HL7 message. This attribute contains the four characters in the following order:<br>■ Component separator<br>■ Repetition separator<br>■ Escape character<br>■ Subcomponent separator<br>The recommended value is ^~\& (that is, ASCII 94, 126, 92, and 38, respectively). |
| Field Separator | 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. Enter the value as a decimal ASCII number ranging from 1 to 127. The default value is 124 which is the character "\|". |

## Message Properties

The Message Properties section contains the SFT configuration settings. HL7 versions 2.5, 2.5.1, and 2.6 add an SFT segment to every message. The binding not only sends and receives messages with the SFT segment, it can automatically create and populate the SFT configuration settings using information from the message properties for outbound mode, and from the ACK sent for inbound mode. This section includes the following properties.

| Property | Description |
|---|---|
| SFT Enabled | An indicator of whether SFT segment processing is enabled. |
| Journaling | An indicator of whether journaling of the HL7 message and ACK is enabled. |
| Software Vendor Organization | The Software Vendor Organization field (SFT-1-Software Vendor Organization), which identifies the vendor who is responsible for maintaining the application. This property only applies to HL7 versions 2.5, 2.5.1, and 2.6. |
| Software Certified Version or Release Number | The Software Certified Version or Release Number, which is the value of the HL7 segment SFT-02. The current 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 the HL7 messages. For example, if the sending system is Java CAPS 6.1, the value would be 6.1. This property only applies to HL7 versions 2.5, 2.5.1, and 2.6. |

| Property | Description |
|---|---|
| Software Product Name | The name of the software product that submitted the transaction, which is the value of the HL7 segment SFT-03. The software product name is a key component for identifying the sending application. This property only applies to HL7 versions 2.5, 2.5.1, and 2.6. |
| Software Product Information | The software product identification information, which is the value of the HL7 segment SFT-05. This may include a description of the software application, configuration settings, modifications made to the software, and so forth. This information is used for diagnostic purposes to help identify the application software. This property only applies to HL7 versions 2.5, 2.5.1, and 2.6. |
| Software Binary ID | The Software Binary ID, which is the value of HL7 segment SFT-04. 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. This property only applies to HL7 versions 2.5, 2.5.1, and 2.6. |
| Software Install Date | The Software Install Date, which is the value of HL7 segment SFT-06. This is the date on which the submitting software was installed at the sending site. The install date can provide key information in regard to the behavior of an application. The date format is YYYYMMDDHHSS. This property only applies to HL7 versions 2.5, 2.5.1, and 2.6. |

## Communication Controls

The last page of the HL7 Binding Wizard, the Communication Controls page, provides configuration properties for the communication controls and recourse actions that control the sending and receiving of messages over a TCP/IP connection. These properties are defined by the Communication Control WSDL Extensibility Element, and help establish the Quality of Service (QoS) between the HL7 external system and HL7 endpoints.

The communication controls differ for inbound or outbound endpoints, therefore different pages are provided for inbound or outbound.

The recourse actions supported by the HL7 Binding Component can include the following values:

- **Reset:** The binding component closes the connection with the HL7 external system and throws an alert.
- **Resend:** The binding component resends the last sent message to the HL7 external system.
- **Suspend:** The binding component closes the connection with the HL7 external system, suspends the endpoint from processing the messages, and throws an alert.
- **Skipmessage:** The binding component remains connected but writes the message to an error queue.
- **Error:** The binding component throws an Exception.

Each property in the table below allows you to enable or disable the control, select a value, and select a recourse action. If you do not enable the control, any setting you configure for that control are ignored.

**TABLE 1** Inbound Communication Controls

| Property | Description |
|---|---|
| MAX_NAK_SENT | The maximum number of NAKs, the binding component sends to the client before executing the recourse action. A negative acknowledgment is sent from the HL7 inbound endpoint when the received message from client is invalid. An invalid message, according to the HL7 specification, is one that fails MSH segment validation or has an invalid sequence number.<br>■ **Value:** An integer indicating the appropriate maximum number.<br>■ **Recourse Actions:** Suspend and Reset. |
| MAX_CANNED_NAK _SENT | The maximum number of canned NAKs, the binding component sends to the client before executing the configured recourse action. This communication control deals with the case where the message received from client is invalid as per the HL7 specification rules for that particular message.<br>■ **Value:** An integer indicating the appropriate maximum number.<br>■ **Recourse Actions:** Suspend and Reset. |

Each property in the table below allows you to enable or disable the control, select a value, and select a recourse action. If you do not enable the control, any setting you configure for that control are ignored.

**TABLE 2** Outbound Communication Controls

| Property | Description |
|---|---|
| MAX_CONNECT _RETRIES | The maximum number of connection retires the binding component will attempt before executing the recourse action.<br>■ **Value:** The value is presented in the format: n1;n2,n1;n2,..., where n1 is the number of retry attempts to connect, and n2 is the interval (in seconds) between retry attempts. If the value of n1 is –1, it indicates that the number of retry attempts is indefinite.<br>■ **Recourse Actions:** Suspend and Error. |
| TIME_TO_WAIT_FOR _A_RESPONSE | The amount of time the binding component waits for a response from the external system before executing the configured recourse action.<br>■ **Value:** An integer indicating the time interval in seconds.<br>■ **Recourse Actions:** Resend, Reset, and Suspend. |

**TABLE 2** Outbound Communication Controls *(Continued)*

| Property | Description |
|---|---|
| MAX_NO_RESPONSE | The maximum number of timeouts the binding component allows, while waiting for the response from external HL7 system, before executing the configured recourse action. This Communication Control is used in relationship with TIME_TO_WAIT_FOR_A_RESPONSE. The binding component resends the last sent message for each timeout until the configured maximum number of timeouts is exceeded.<br>■ **Value:** An integer indicating the maximum allowable number of "no-responses" from the external HL7 system.<br>■ **Recourse Actions:** Suspend and Reset. |
| NAK_RECEIVED | The recourse action the binding component will implement upon receiving a NAK from external system. There is no set value for this property other than enable and the selected recourse action.<br>■ **Recourse Action:** Resend, Reset, and Skipmessage. |
| MAX_NAK_RECEIVED | The maximum number of NAKs (negative acknowledgments) that the binding component accepts before it executes the configured recourse action.<br>■ **Value:** An integer indicating the maximum number of NAKs allowed.<br>■ **Recourse Actions:** Suspend, Reset, and Skipmessage. |

# HL7 Binding WSDL Extensibility Elements

WSDL, or Web Service Description Language, is an XML-based language used to define web services. The HL7 WSDL extensibility elements are used to construct HL7 messages by specifying the message parts, message formats, and other message related information used to properly map an HL7 message exchange. The HL7 WSDL elements also contain information that the HL7 Binding Component uses to establish connections and sessions with HL7 external systems. Various properties that affect the delivery of HL7 messages are included within the HL7 WSDL extensibility elements.

WSDL files are created using the WSDL Wizard and validated within the NetBeans IDE. The extensibility elements correspond to the properties you specify in the wizard. They are described here so you can add and modify them in the WSDL file directly.

For more information on how to use the WSDL Wizard to create and HL7 WSDL file see "Using the HL7 Binding Component Wizard" on page 10.

HL7 extensibility elements are divided into two sets of configuration elements:

- **Service Level** elements are used to configure the connectivity and protocol. The Service Level extensibility elements are:
  - "HL7 address Element" on page 26
  - "HL7 protocolproperties Element" on page 26

- "HL7 communicationcontrols Element" on page 30

- **Binding Level** elements are used to binding abstract WSDL messages to HL7 messages. The Binding Level extensibility elements are:
  - "HL7 binding Element" on page 32
  - " HL7 operation Element" on page 32
  - " HL7 message Element" on page 32

The following sections describe the HL7 extensibility elements. The tables describe the attributes for each extensibility element and its child elements. For each attribute or element the table lists the name, description, whether the attribute or element applies to both provider or consumer (Common), whether it is required, and an example of its usage.

# Adding HL7 Extensibility Attributes From the WSDL Editor

You can add HL7 extensibility elements from the wizard when you create the WSDL file. After the WSDL file is created, you can add extensibility attributes by entering the text directly or graphically in the WSDL Editor

## ▼ To add Service Level HL7 Extensibility Attributes

1 **Open the HL7 Binding WSDL file in the WSDL Editor and expand the Services → hl7wsdlService → hl7wsdlPort nodes.**

2 **If the type of element you want to add (address, protocolproperties, or communicationcontrols) does not appear in the list, right-click the hl7wsdlPort node, point to Add and then select the type of element to add.**

A new node is added under the hl7wsdlPort node.

3 **To add an address element, do the following:**

a. **Right-click the hl7wsdlPort node, point to Add and then select HL7 Address.**

A new hl7:address element appears in the list.

b. **Right-click the new element and then select Properties.**

c. **Configure the address extensibility attributes, and then click Close.**

For more information about these attributes, see "HL7 address Element" on page 26.

4 **To add a protocolproperties element, do the following:**

a. **Right-click the hl7wsdlPort node, point to Add and then select HL7 Protocol Properties.**

A new hl7:protocolproperties element appears in the list.

b. **Right-click the new element and then select Properties.**

c. **Configure the protocolproperties extensibility attributes, and then click Close.**

For more information about these attributes, see "HL7 protocolproperties Element" on page 26.

5 **To add communicationcontrol attributes, do the following:**

a. **If the communicationcontrol element does not appear in the list, right-click the hl7wsdlPort node, point to Add and then select HL7 Communication Control.**

A new hl7:communcationcontrols element appears in the list.

b. **Right-click the hl7:communicationcontrols node and then select Add communicationcontrol.**

A new hl7:communicationcontrol is added under the hl7:communicationcontrols node.

c. **Right-click the new element and then select Properties.**

d. **Configure the communcationcontrol extensibility attributes, and then click Close.**

Configure the following attribute properties. For more information about these attributes, see "HL7 communicationcontrols Element" on page 30.

- **name:** Specifies the attribute.
- **value:** Specifies the value associated with that attribute.
- **enabled:** Specifies if the attribute is enabled.
- **recourseAction:** Specifies the recourse action associated with the attribute.

  The recourse actions are:

  - **Reset:** Closes the connection with the HL7 external system and throws an alert.
  - **Resend:** Resends the last sent message to the HL7 external system.
  - **Suspend:** Closes the connection with the HL7 external system, suspends the endpoint from processing the messages, and throws an alert.
  - **Skipmessage:** Remains connected but writes the message to an error queue.
  - **Error:** Throws an Exception.

# Service Level HL7 Extensibility Elements

The following tables list and describe the Service Level HL7 extensibility elements:



## HL7 address Element

The HL7 address extensibility element specifies the information for connectivity to the HL7 external system.

**TABLE 3** HL7 address Element Attributes

| Name and Description | Required or Optional | Applies to Provider or Consumer | Example |
|---|---|---|---|
| **location:** Specifies the host and port used to connect to the HL7 external system. | Required | Common (both) | hl7://myhost:4040 |
| **transportProtocolName:** Specifies the transport protocol used to transfer the message payload. | Required | Common (both) | tcp-ip |

The following example illustrates the use of the HL7 address element defined for a service port.

```
<service name="SomeService">
   <port name="port1" binding="tns:someBinding">
       <hl7:address location="hl7://localhost:4040"
                    transportProtocolName="tcp-ip"/>
   </port>
 </service>
```

## HL7 protocolproperties Element

The HL7 protocolproperties extensibility element specifies the protocol specific information for connecting to the HL7 external system.

**TABLE 4** HL7 protocolproperties Element Attributes

| Name and Description | Required or Optional | Applies to Provider or Consumer | Example |
|---|---|---|---|
| **acknowledgeMode:** Indicates the acknowledge mode type: Original Mode or Enhanced Mode. | Optional | Common (both) | original |

**TABLE 4** HL7 protocolproperties Element Attributes *(Continued)*

| Name and Description | Required or Optional | Applies to Provider or Consumer | Example |
|---|---|---|---|
| **llpType:** Indicates the Lower Layer Protocol Type. MLLPv1, MLLPv2, or HLLP. | Optional | Common (both) | MLLPv1 |
| **startBlockCharacter:** Indicates the Start Block Character Value in a decimal ASCII number from 1 to 127. Unless there is a conflict, the value should be ASCII VT, which is decimal 11. | Optional | Common (both) | 11 |
| **endDataCharacter:** Indicates the End Data Character Value in decimal ASCII number from 1 to 127. Unless there is a conflict, the value should be ASCII VT, which is decimal 28. | Optional | Common (both) | 28 |
| **endBlockCharacter:** Indicates the End Block Character Value in decimal ASCII number from 1 to 127. To be strictly comply with the HL7 standard, this parameter must be set to a carriage Return, which is decimal 13. | Optional | Common (both) | 13 |
| **hllpChecksumenabled:** Specifies if HLLP CheckSum is enabled. “true” indicates enabled. | Optional | Common (both) | false |
| **seqNumberEnabled:** Specifies if sequence number protocol is enabled. “true” indicates enabled. | Optional | Common (both) | false |
| **validateMSH:** Specifies if the MSH segment in the HL7 message is validated against initiation rules. “true” indicates enabled. | Optional | Common (both) | false |
| **processingID:** Specifies the ProcessingID value against which the MSH-11-ProcessingID field in the received message is validated when validateMSH is set to “true”. Valid values are P (production), D (debugging), or T (training). | Optional | Common (both) | P |

**TABLE 4** HL7 protocolproperties Element Attributes *(Continued)*

| Name and Description | Required or Optional | Applies to Provider or Consumer | Example |
|---|---|---|---|
| **versionID:**<br>Specifies the versionID value against which MSH-12-VersionID field in the received message is validated when validateMSH is set to true. Valid values are 2.1, 2.2, 2.3, 2.3.1, 2.4, 2.5, 2.5.1 or 2.6. | Required | Common (both) | 2.3.1 |
| **fieldSeparator:**<br>Defines the Field Separator character value in a decimal ASCII number. This represents the separator between the segment ID and the first field, MSH-2-encoding characters. As such, it server as the separator and defines the character to be used as a separator for the rest of the message. The default setting is 124 which is the character "\|". The allowed range is 1 to 127. This attribute value is used in creating the NAK for invalid HL7 messages. | Optional | Common (both) | 124 |
| **encodingCharacters:**<br>Specifies the encoding characters to be used in creating the NAK for invalid HL7 messages. This attribute contains the four characters in the following order: the component separator, repetition separator, escape character, and subcomponent separator. Recommended values are ^~\& (that is, ASCII 94, 126, 92, and 38, respectively). | Optional | Common (both) | ^~\& |
| **sendingApplication:**<br>Specifies the MSH-03 Sending Application to be used in creating the NAK for invalid HL7 messages. | Optional | Common (both) | Java CAPS HL7 BC |
| **sendingFacility:**<br>Specifies the MSH-04 Sending Facility to be used in creating the NAK for invalid HL7 messages. | Optional | Common (both) | Java CAPS HL7 BC |
| **enabledSFT:**<br>Enables or disables SFT segment processing. “true” indicates enabled. | Optional | Common (both) | true |
| **softwareVendorOrganization:**<br>Specifies HL7 segment SFT-01, the software vendor organization field. This identifies the vendor who is responsible for maintaining the application. | Optional | Common (both) | Oracle Corporation |

**TABLE 4** HL7 protocolproperties Element Attributes *(Continued)*

| Name and Description | Required or Optional | Applies to Provider or Consumer | Example |
|---|---|---|---|
| **softwareCertifiedVersionOr ReleaseNumber:** Specifies HL7 segment SFT-02, the software certified version or release number. This is the current version or release number of the sending application. | Optional | Common (both) | 6.3 |
| **softwareProductName:** Specifies HL7 segment SFT-03, the name of the software product that submitted the transaction. The product name is a key component in identifying the sending application. | Optional | Common (both) | Java CAPS HL7 Binding Component |
| **softwareBinaryID:** Specifies HL7 segment SFT-04, the software binary ID. This property is available for HL7 version 2.5 and above. Software binary IDs are issued by a vendor for each unique software version instance. Binary 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. | Optional | Common (both) | 2.0 |
| **softwareProductInformation:** Specifies HL7 segment SFT-05, software product identification information. This can include a description of the software application, configuration settings, and software modifications. | Optional | Common (both) | Binding Component for HL7 over TCP/IP connection |
| **softwareInstallDate:** Specifies HL7 segment SFT-06, the software install date. This is the date on which the submitting software was installed at the sending site. Format as follows: YYYYMMDDHHSS. | Optional | Common (both) | 200909141020 |
| **journalingEnabled:** Enables or disables journaling. If enabled, it stores the HL7 messages and respective ACK in the flat file database. “true” indicates enabled. | Optional | Common (both) | true |

The following example illustrates the use of the HL7 `protocolproperties` element.

```
<hl7:protocolproperties acknowledgmentMode="original"
llpType="MLLPv1"
startBlockCharacter="11"
endDataCharacter="28"
endBlockCharacter="13"
```

```
hllpChecksumEnabled="false"
seqNumEnabled="false"
processingID="P"
versionID="2.3.1"
validateMSH="false"
sendingApplication="Java CAPS HL7 BC"
sendingFacility="Java CAPS HL7 BC"
enabledSFT="false"
softwareVendorOrganization="Oracle Corporation"
softwareCertifiedVersionOrReleaseNumber="6.3"
softwareProductName="HL7 Binding Component"
softwareBinaryID="6.3"
softwareProductInformation="It is a binding component for HL7 over TCP/IP connection"
journallingEnabled="false"
mllpv2RetriesCountOnNak="0"
mllpv2RetryInterval="0"
mllpv2TimeToWaitForAckNak="0"
encodingCharacters="^~&amp;"
softwareInstallDate=""
fieldSeparator="124"/>
```

## HL7 communicationcontrols Element

HL7 Communication controls and recourse actions together control the data transfer over a TCP/IP connection. The configuration of these elements is defined inside the HL7 `communicationcontrol` extensibility element. These attributes help to establish Quality of Service (QoS) between HL7 external systems and HL7 endpoints.

The HL7 communication control attributes can be added to an endpoint from the WSDL Editor or from the "Communication Controls" on page 21. The HL7 communicationcontrol attributes differ for inbound or outbound endpoints.

**TABLE 5** HL7 communicationcontrol Element Attributes

| Name and Description | Inbound or Outbound | Recourse Action |
|---|---|---|
| **MAX_NAK_SENT:**<br>Specifies the maximum number of NAKs, the binding component sends to the client before executing the recourse action. A negative acknowledgment is sent from the HL7 inbound endpoint when the received message from client is invalid. An invalid message, according to the HL7 specification, is one that fails MSH segment validation or has an invalid sequence number. | Inbound | Suspend and Reset. |
| **MAX_CANNED_NAK_SENT:**<br>Specifies the maximum number of canned NAKs, the binding component sends to the client before executing the configured recourse action. This communication control deals with the case where the message received from client is invalid as per the HL7 specification rules for that particular message. | Inbound | Suspend and Reset. |

**TABLE 5** HL7 communicationcontrol Element Attributes *(Continued)*

| Name and Description | Inbound or Outbound | Recourse Action |
|---|---|---|
| **MAX_CONNECT_RETRIES:**<br>Specifies the maximum number of connection retires the binding component will attempt before executing the recourse action.<br><br>The value is presented in the format: n1;n2,n1;n2,..., where n1 is the number of retry attempts to connect, and n2 is the interval (in seconds) between retry attempts. If the value of n1 is –1, it indicates that the number of retry attempts is indefinite. | Outbound | Suspend and Error. |
| **TIME_TO_WAIT_FOR_A_RESPONSE:**<br>Specifies the amount of time (in seconds) that the binding component waits for a response from the external system before executing the configured recourse action. | Outbound | Resend, Reset, and Suspend. |
| **MAX_NO_RESPONSE:**<br>Specifies the maximum number of timeouts the binding component allows, while waiting for the response from external HL7 system, before executing the configured recourse action.<br><br>This Communication Control is used in relationship with TIME_TO_WAIT_FOR_A_RESPONSE. The binding component resends the last sent message for each timeout until the configured maximum number of timeouts is exceeded. | Outbound | Suspend and Reset. |
| **MAX_RECEIVED:**<br>Specifies which recourse action the binding component will implement upon receiving a NAK from external system. There is no set value for this property other than enable and the selected recourse action. | Outbound | Resend, Reset, and Skipmessage. |
| **MAX_NAK_RECEIVED:**<br>Specifies the maximum number of NAKs (negative acknowledgments) that the binding component accepts before it executes the configured recourse action. | Outbound | Suspend, Reset, and Skipmessage. |

# Binding Level HL7 Extensibility Elements

The HL7 extensibility elements used to bind abstract WSDL messages to HL7 messages fall into three element types. Each type signifies how the binding occurs. At the binding level, the configuration applies to the entire port type. At the operation level, the configuration applies only to the operation. At the message level, the configuration applies to that particular message, whether it's input or output.

## HL7 binding Element

The HL7 binding extensibility element specifies a binding that is of interest to the HL7 Binding Component. It is essentially an empty element that serves as a marker, allowing the HL7 Binding Component to gather HL7 "binding" information described by the other HL7 extensibility elements. The HL7 binding extensibility element must be specified in the WSDL to define an HL7 protocol based binding.

The following example demonstrates how the HL7 binding extensibility element is used to associate a binding with a specific HL7 protocol.

```
<binding name="someBinding" type="tns:somePortType">
        <hl7:binding/>
    </binding>
```

## HL7 operation Element

The HL7 operation extensibility element specifies an operation binding that is of interest to the HL7 Binding Component.

The following example demonstrates how the HL7 operation extensibility element is used to associate an abstract operation with an HL7 operation.

```
<binding name="someBinding" type="tns:somePortType">
        <hl7:binding/>
        <operation name="someOperation">
            </hl7:operation/>
    </binding>
```

## HL7 message Element

The HL7 message element specifies the concrete properties associated with receiving or sending an HL7 message from or to the HL7 external system. To configure the attributes, right-click the message element and then select Properties.

**TABLE 6** HL7 message Element Attributes

| Name and Description | Required or Optional | Applies to Provider or Consumer | Example |
|---|---|---|---|
| **use:**<br>Specifies if the message (part) defined is encoded using a well defined encoding style. | Required | Common (both) | encoded |
| **encodingStyle:**<br>Specifies the encoding type associated with the message (part). This also defines the encoder type used to process the encoded data. | Optional | Common (both) | hl7encoder-1.0 |

**TABLE 6** HL7 message Element Attributes *(Continued)*

| Name and Description | Required or Optional | Applies to Provider or Consumer | Example |
|---|---|---|---|
| **part:**<br>Indicates the part of the message that contains the HL7 message to be sent. | Optional | Common (both) | part1 |

The following example demonstrates the HL7 message extensibility element defined for one-way operation.

```
<binding name="someBinding" type="tns:somePortType">
        <hl7:binding/>
        <operation name="oneWayOp">
            <hl7:operation/>
            <input>
                <hl7:message part="part1"
                 use="encoded"
                 encodingStyle="hl7encoder-1.0"/>
            </input>
        </operation>
    </binding>
```

# Dynamically Configuring HL7 Endpoints

Dynamic configuration is the ability to make changes at runtime, without reconfiguring a project's design-time configuration. WSDL files provide static configuration controls for the behavior of JBI endpoints.

To dynamically configure an HL7 project, there are three alternatives:

- "Application Variables" on page 33
- "Application Configurations" on page 34
- "Dynamic Addressing Using Normalized Message Properties" on page 34

The following topics provide additional information and instructions on using normalized message properties:

- "Enabling Dynamic Endpoint Configuration" on page 35
- "Using Normalized Message Properties in a BPEL Process" on page 36
- "Normalized Message Properties" on page 37

## Application Variables

Application Variables are name value pairs of a given type (String, number, Boolean, or password) and are defined at the binding component level. The Application Variable name acts as a token for a WSDL extensibility element attribute in a corresponding binding. For example,

if you were defining an application variable for the hostname as FOO, then the WSDL attribute would be ${FOO}. In the Application Variables property you would enter a String value of FOO for the name, and the desired attribute as the value. For more information on using Application Variables see "Using Application Variables to Define Name/Value Pairs" on page 46.

## Application Configurations

The Application Configuration property allows you to configure the connectivity and level parameters for an application that you have created, and without changing or rebuilding the application, deploy the same application into a different system. For example, you can take an application that is running in a test environment and deploy it to a production environment without rebuilding the application. A Composite Application's external connectivity parameters, which are normally defined in the WSDL service extensibility elements, can be configured in the runtime properties. For more information on using Application Configurations see "Defining an HL7 Binding Component Application Configuration" on page 43.

## Dynamic Addressing Using Normalized Message Properties

Dynamic Addressing uses Normalized Message Properties to dynamically override the static configuration of an endpoint and reroute messages accordingly. The message payload can contain the location of the message consumer as well as other binding protocol specific information to configure endpoint behavior.

Normalized Message properties are commonly used to specify metadata that is associated with message content. `javax.jbi.security.subject` and `javax.jbi.message.protocol.type` are two examples of standard normalized Message properties defined in the JBI Specification.

Normalized Message properties provide additional capabilities, including the following:

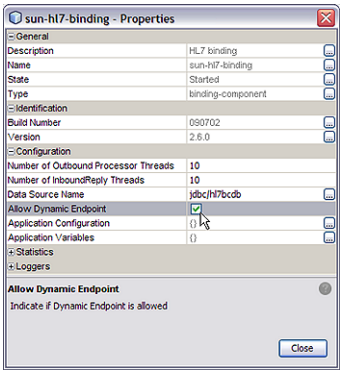
- Getting and setting transport context properties; for example, HTTP headers in the incoming HTTP request or file names read by the File Binding Component.
- Getting and setting protocol-specific headers or context properties (SOAP headers).
- Getting and setting additional message metadata; for example. a unique message identifier or an endpoint name associated with a message.
- Dynamic configurations; for example, to dynamically overwrite the statically configured destination file name at runtime.

Some of the use cases mentioned above require protocol and binding specific properties, typically used by a particular binding component. Other properties are considered common or

general purpose properties that all participating JBI components make use of, such as the message ID property, which can be utilized to uniquely identify or track a given message in the integration.

# Enabling Dynamic Endpoint Configuration

Normalized Message Properties are component specific, different for every binding component type. To enable HL7 Normalized Message properties to be applied to a Dynamic Endpoint configuration, set the HL7 runtime configuration property `Allow Dynamic Endpoint` to true (checked). You will also need to set the HL7 Normalized Message Property, `Use Dynamic Endpoint`, to true.




When Allow Dynamic Endpoint is enabled, the HL7 Binding Component behaves as follows:

- **Inbound Endpoint Behavior:**
  1. The HL7 Binding Component receives the message from the external system.
  2. It creates the Message Exchange and adds the normalized message.
  3. It checks to see if the runtime `Allow Dynamic Endpoint` property is enabled. If enabled, it populates the respective Normalized Message Properties with the specified values.
  4. It appends these Normalized Message Properties to the Message Exchange.
  5. It sends the Message Exchange to the delivery channel.
- **Outbound Endpoint Behavior:**
  1. The HL7 Binding Component receives the Message Exchange from the delivery channel.
  2. It checks to see if the runtime `Allow Dynamic Endpoint` property, and the Outbound Normalized Message property `Use Dynamic Endpoint` (org.glassfish.openesb.hl7.use.dynamic.endpoint) are enabled. If enabled, it retrieves the

Normalized Message properties from Message Exchange and uses them to overwrite the static binding configuration of the endpoint.

3. It uses the overwritten property information while communicating with the external system.

# Using Normalized Message Properties in a BPEL Process

The Normalized Message properties are accessed from the BPEL Designer Mapper view. When you expand a variable's Properties folder it exposes the variable's predefined NM properties, as well as the regular BPEL specific WSDL properties used in correlation sets, assigns, and to build expressions .

## Using Predefined Normalized Message Properties in a BPEL Process

Predefined Normalized Message properties are ready for use, from a variable's Properties file.

### ▼ To Use Predefined Normalized Message Properties in a BPEL Process

1. **From the Design View diagram, select the activity with the process you want to edit.**
2. **Click Mapper to switch to the Mapper view of the BPEL process.**
3. **Expand the Variable you want to edit and its Properties file.**
4. **To dynamically configure an endpoint, first expand the /Properties/General node to expose the General Normalized Message Properties.**
5. **Enter values for the Message ID property and the Endpoint Name property.**
6. **To enter a `java.lang.String` value for the General or HL7 Normalized Message Properties, do the following:**

   a. **From the BPEL Designer's Mapper window, select the Normalized Message Property you want to configure.**

      The property is highlighted with a blue field indicating that you are applying an action to this item.

   b. **From the String menu in the Mapper's toolbar select String Literal.**

      A String Literal method box is added to the properties action field.

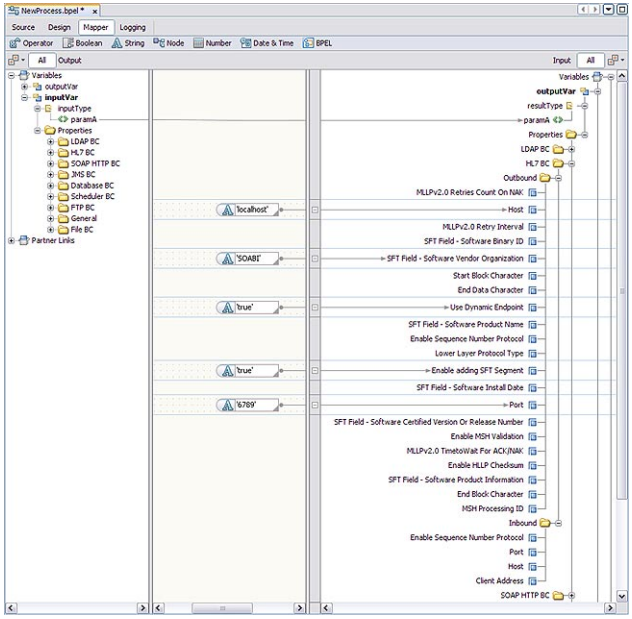
   c. **Double-click the value field of the String Literal box and enter the appropriate value.**

**d. Map the String Literal box to the property. To do this, click on the outbound handle of the String Literal box, and drag your cursor to the Normalized Message Property.**

A line now links the String Literal box to the property.

**7 Expand the /Properties/HL7 node to expose the HL7 Normalized Message Properties.**

**8 Set the String value of the Use Dynamic Endpoint property to `true`. Also add String values to any other HL7 properties you want to dynamically configure.**




**9 Save your changes.**

# Normalized Message Properties

The HL7 Binding Component uses the General Normalized Message properties, as well as the HL7 Normalized Message properties for outbound and inbound endpoints.

## General Normalized Message Properties

The following General NM properties are available to all binding components.

**TABLE 7** General Normalized Message Properties

| Property Name | Type | Description and Use |
|---|---|---|
| Endpoint Name<br>org.glassfish.openesb.exchange.endpointname | java.lang.String | Specifies the endpoint name set on the exchange. This represents the endpoint name of the "owner" of the message, and can be made available by JBI runtime. |
| Message ID<br>org.glassfish.openesb.messaging.messageid | java.lang.String | Specifies a unique identifier for a message. This might be a record number (for example, a particular record in a file), or a GUID. |
| Last Record<br>org.glassfish.openesb.messaging.lastrecord | java.lang.String | Specifies the last record in a group, e.g. the last record in an RM sequence for SOAP messages, or the last record in a file when multiple record processing is turned on for File BC. |
| Group ID<br>org.glassfish.openesb.messaging.groupid | java.lang.String | Specifies a unique identifier for a group to which a message belongs. For example, it applies the RM sequence group number for SOAP messages, or a time stamped file name (where the file record message comes from). |

## HL7 Binding Component Normalized Message Properties for Outbound Endpoints

The following table defines the HL7 Normalized Messages for outbound endpoints.

**TABLE 8** HL7 NM Properties for Outbound Endpoints

| Property Name | Type | Description and Use |
|---|---|---|
| Use Dynamic Endpoint<br>org.glassfish.openesb.hl7.use.dynamic.endpoint | java.lang.String | Specifies if the message uses "DynamicAddressing." This is a mandatory property if you are using NM properties for dynamic addressing.<br>This property has no equivalent WSDL attribute. |
| Host | java.lang.String | Specifies the host part of a URL used to connect to an HL7 external system.<br>The equivalent WSDL attribute is hl7:address → location(host). |
| Port<br>org.glassfish.openesb.hl7.outbound.address.port | java.lang.String | Specifies the port part of a URL used to connect to an HL7 external system.<br>The equivalent WSDL attribute is hl7:address → location(port). |

**TABLE 8** HL7 NM Properties for Outbound Endpoints *(Continued)*

| Property Name | Type | Description and Use |
|---|---|---|
| Lower Layer Protocol Type<br><br>org.glassfish.openesb.hl7 .outbound.protocolproperties. llptype | java.lang.String | Indicates the Lower Layer Protocol Type as MLLPv1 (Minimal Lower Layer Protocol release 1), MLLPv2 (Minimal Lower Layer Protocol release 2), and HLLP (Hybrid Lower Layer Protocol).<br><br>The equivalent WSDL attribute is hl7:protocolproperties → llpType. |
| Start Block Character<br><br>org.glassfish.openesb.hl7 .outbound.protocolproperties. startblockcharacter | java.lang.String | Indicates the Start Block Character Value in a decimal ASCII number from 1 to 127. Unless there is a conflict, the value should be ASCII VT, which is decimal 11.<br><br>The equivalent WSDL attribute is hl7:protocolproperties → startBlockCharacter. |
| End block Character<br><br>org.glassfish.openesb.hl7. outbound.protocolproperties. endblockcharacter | java.lang.String | Indicates the End Block Character Value in decimal ASCII number from 1 to 127. To be strictly comply with the HL7 standard, this parameter must be set to a carriage Return, which is decimal 13.<br><br>The equivalent WSDL attribute is hl7:protocolproperties → endBlockCharacter. |
| End Data Character<br><br>org.glassfish.openesb.hl7. outbound.protocolproperties. enddatacharacter | java.lang.String | Indicates the End Data Character Value in decimal ASCII number from 1 to 127. Unless there is a conflict, the value should be ASCII VT, which is decimal 28.<br><br>The equivalent WSDL attribute is hl7:protocolproperties → endDataCharacter. |
| Enable HLLP Checksum<br><br>org.glassfish.openesb.hl7. outbound.protocolproperties. hllpchecksumenabled | java.lang.String | Specifies if HLLP CheckSum is enabled. “true” indicates enabled.<br><br>The equivalent WSDL attribute is hl7:protocolproperties → hllpChecksumEnabled. |
| Enable Sequence Number Protocol<br><br>org.glassfish.openesb.hl7. outbound.protocolproperties. seqnumenabled | java.lang.String | Specifies if sequence number protocol is enabled. “true” indicates enabled.<br><br>The equivalent WSDL attribute is hl7:protocolproperties → seqNumEnabled. |

**TABLE 8** HL7 NM Properties for Outbound Endpoints *(Continued)*

| Property Name | Type | Description and Use |
|---|---|---|
| MSH Processing ID<br><br>org.glassfish.openesb.hl7.outbound.protocolproperties.processingid | java.lang.String | Specifies the ProcessingID value against which the MSH-11-ProcessingID field in the received message is validated when validateMSH is set to "true". Valid values are P (production), D (debugging), or T (training).<br><br>The equivalent WSDL attribute is hl7:protocolproperties → processingID. |
| Enable MSH Validation<br><br>org.glassfish.openesb.hl7.outbound.protocolproperties.validatemsh | java.lang.String | Specifies if the MSH segment in the HL7 message is validated against initiation rules. "true" indicates enabled.<br><br>The equivalent WSDL attribute is hl7:protocolproperties → validateMSH. |
| Enable adding SFT Segment<br><br>org.glassfish.openesb.hl7.outbound.protocolproperties.enablesft | java.lang.String | A toggle property to enable/disable SFT segment processing.<br><br>The equivalent WSDL attribute is hl7:protocolproperties → enabledSFT. |
| SFT Field — Software Vendor Organization<br><br>org.glassfish.openesb.hl7.outbound.protocolproperties.softwarevendororganization | java.lang.String | Defines the Software Vendor Organization field (SFT-1-Software Vendor Organization) which identifies the vendor who is responsible for maintaining the application.<br><br>The equivalent WSDL attribute is hl7:protocolproperties → softwareVendorOrganization. |
| SFT Field — Software Certified Version Or Release Number<br><br>org.glassfish.openesb.hl7.outbound.protocolproperties.softwarecertifiedversionorreleasenumber | java.lang.String | 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.<br><br>The equivalent WSDL attribute is hl7:protocolproperties → softwareCertifiedVersionOrReleaseNumber. |
| SFT Field — Software Product Name<br><br>org.glassfish.openesb.hl7.outbound.protocolproperties.softwareproductname | java.lang.String | 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.<br><br>The equivalent WSDL attribute is hl7:protocolproperties → softwareProductName. |

**TABLE 8** HL7 NM Properties for Outbound Endpoints *(Continued)*

| Property Name | Type | Description and Use |
|---|---|---|
| SFT Field — Software Binary ID<br><br>org.glassfish.openesb.hl7.outbound.protocolproperties.softwarebinaryid | java.lang.String | 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.<br><br>The equivalent WSDL attribute is hl7:protocolproperties → softwareBinaryID. |
| SFT Field — Software Product Information<br><br>org.glassfish.openesb.hl7.outbound.protocolproperties.softwareproductinformation | java.lang.String | Specifies HL7 segment SFT-05, software product identification information. This may include a description of the software application, configuration settings, or modifications made to the software.<br><br>The equivalent WSDL attribute is hl7:protocolproperties → softwareProductInformation. |
| SFT Field — Software Install Date<br><br>org.glassfish.openesb.hl7.outbound.protocolproperties.softwareinstalldate | java.lang.String | Specifies HL7 segment SFT-06, the Software Install Date. This is the date, in YYYYMMDDHHSS format, on which the submitting software was installed at the sending site.<br><br>The equivalent WSDL attribute is hl7:protocolproperties → softwareInstallDate. |
| MLLPv2.0 Retries Count On NAK<br><br>org.glassfish.openesb.hl7.protocolproperties.mllpv2retriescountonnak | java.lang.String | Specifies the maximum number of retries on receipt of MLLP V2 negative acknowledgement.<br><br>The equivalent WSDL attribute is hl7:protocolproperties → mllpv2RetriesCountOnNak. |
| MLLPv2.0 Retry Interval<br><br>org.glassfish.openesb.hl7.outbound.protocolproperties.mllpv2retryinterval | java.lang.String | Specifies the time duration to wait in milliseconds before each retry.<br><br>The equivalent WSDL attribute is hl7:protocolproperties → mllpv2RetryInterval. |
| MLLPv2.0 TimetoWait For ACK/NAK<br><br>org.glassfish.openesb.hl7.outbound.protocolproperties.mllpv2timetowaitforacknak | java.lang.String | Specifies the time duration to wait in milliseconds for receiving MLLP V2 commit acknowledgement / negative acknowledgement.<br><br>The equivalent WSDL attribute is hl7:protocolproperties → mllpv2TimeToWaitForAckNak. |

### HL7 Binding Component Normalized Message Properties for Inbound Endpoints

The following table defines the HL7 Normalized Messages for inbound endpoints.

**TABLE 9** HL7 NM Properties for Inbound Endpoints

| Property Name | Type | Description and Use |
|---|---|---|
| Host<br>org.glassfish.openesb.hl7.inbound.address.host | java.lang.String | A URL part, which specifies the connectivity information to connect to the HL7 external system.<br>The equivalent WSDL attribute is hl7:address → location (host). |
| Port<br>org.glassfish.openesb.hl7.inbound.address.port | java.lang.String | A URL part, which specifies the connectivity information to connect to the HL7 external system.<br>The equivalent WSDL attribute is hl7:address → location (port). |
| Client Address<br>org.glassfish.openesb.hl7.inbound.client.address | java.lang.String | Represents the client address and port information.<br>This property has no equivalent WSDL attribute. |
| Enable Sequence Number Protocol<br>org.glassfish.openesb.hl7.inbound.protocolproperties.seqnumenabled | java.lang.String | Specifies whether sequence number protocol is enabled or not.<br>The equivalent WSDL attribute is hl7:protocolproperties → seqNumEnabled. |

# HL7 Binding Component Runtime Properties

The runtime properties allow you to make changes to a project after the project's design time is completed, without touching the business logic. The runtime properties can be accessed from the Properties window or from the binding component's properties editor.

## Configuring the HL7 Binding Component Runtime Properties

The runtime configuration for the HL7 Binding Component affects all HL7 Binding Components that are deployed on the domain you are configuring.

### ▼ To Edit HL7 Binding Component Runtime Properties

**1 From the Services window of the NetBeans IDE, expand the Servers node.**

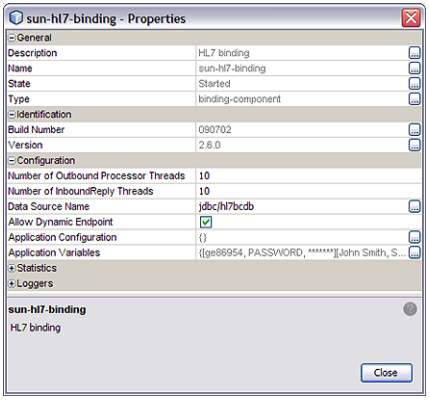
2 **If GlassFish is not started, start GlassFish. To do this, right-click GlassFish V2 and then select Start.**

3 **Under the application server, expand the JBI → Binding Components nodes and select the HL7 Binding Component (sun-hl7-binding).**

4 **If sun-hl7-binding is not started, right-click it and then select Start.**

5 **Double-click the sun-hl7-binding.**

The Properties Editor appears.




6 **Edit the properties as needed.**

Properties are described under "HL7 Binding Component Runtime Properties" on page 42.

7 **Once you are finished editing your properties, click Close.**

8 **On the NetBeans toolbar, click Save All.**

9 **If you have any deployed HL7 Binding Component projects, redeploy the projects so the changes take affect.**

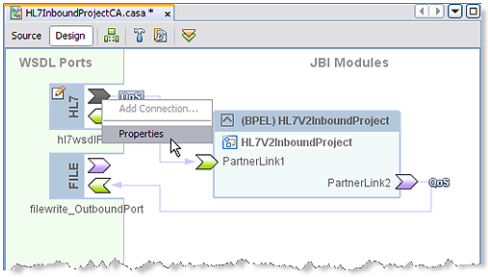
# Defining an HL7 Binding Component Application Configuration

The HL7 Binding Component's Application Configuration property allows you to change the configuration information for an endpoint without changing a projects business logic. The HL7 WSDL endpoint properties can be configured from the HL7 Binding Component Runtime Properties Editor.

## To Define the HL7 Application Configuration

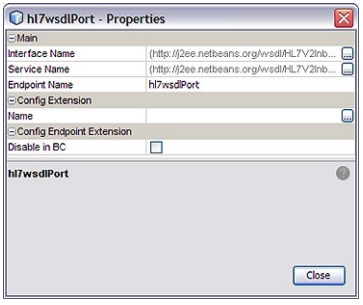
1. **To associate a Configuration Extension profile with the HL7 endpoint, open your project service assembly in the CASA Editor. To do this, from the Project window, right-click your composite application's Service Assembly node, and then select Edit.**

   The CASA Editor opens.




2. **Right-click the HL7 consumer port icon, and then select Properties.**

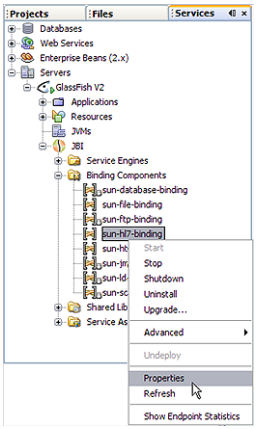
   The HL7 WSDL Port Properties Editor appears.




3. **From the Properties Editor, under Config Extension, enter a name for your profile, such as the name of the HL7 WSDL. Click Close.**

4. **From the NetBeans IDE Services window, make sure that GlassFish V2 server is started. If the GlassFish server is not started, right-click GlassFish V2 and then select Start. In the same way, make sure that the HL7 Binding Component (sun-hl7-binding) is also started.**

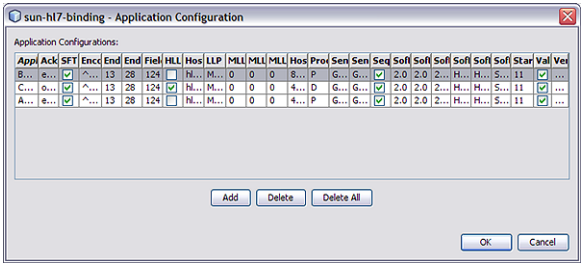
5 **Right-click the sun-hl7-binding and then select Properties.**




The sun-hl7-binding Properties Editor appears.

6 **From the sun-hl7-binding Properties Editor, click the edit button for the Application Configuration property.**

The sun-hl7-binding Application Configuration Editor appears.




7 **Click Add to add a new row to the Application Configuration Editor, representing all of the application configurable parameters for the HL7 Binding Component.**

8 **Enter the properties for your binding.**

For more information about Application Configuration properties, see Application Configuration under "Runtime Properties" on page 48.

9 **Click OK and Close to save properties.**

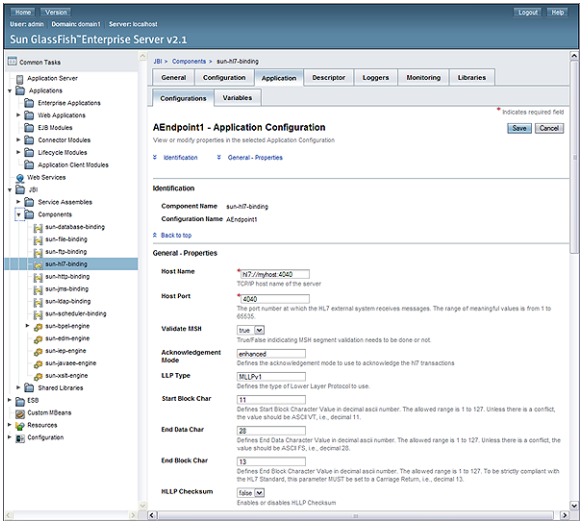
When the composite application is deployed, the HL7 Binding Component will use the new application configuration that has been defined for the respective endpoint.

### Defining the Application Configuration Using Other Tools

In addition to the NetBeans IDE, you can also use these other tools to edit the HL7 Binding Component Application Configuration.

- **GlassFish Admin Console:** To access the Admin Console, from the NetBeans Services window, right-click GlassFish V2 under Servers and then select View Admin Console. You can also access the Admin Console at `http://localhost:4848/login.jsf`, if this setting was retained during installation.

  To open the Application Configuration window from the Admin Console, select the sun-hl7-binding, under Common Tasks → JBI → Components. Select the Application tab and the Configuration sub-tab.

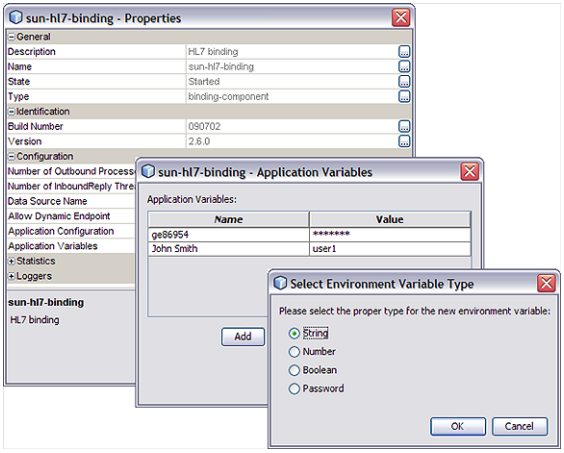



- **asadmin Command Line Interface:** For information on using the Command Line Interface (CLI) to create, edit, or delete and application, see create-jbi-application-configuration, update-jbi-application-configuration, or delete-jbi-application-configuration.

# Using Application Variables to Define Name/Value Pairs

The binding component Application Variables property allows you to define a list of name:value pairs for a given stated type. The application variable name can be used as a token for a WSDL extensibility element attribute in a corresponding binding. For example, if you were defining an application variable for the hostname as FOO, then the WSDL attribute would

be ${FOO}. In the Application Variables property you would enter a String value of FOO for the name and the desired attribute as the value. When you deploy an application that uses application variables, any variable that is referenced in the application's WSDL is loaded automatically.




The Application Variables configuration property offers four variable types:

- **String:** Specifies a string value, such as a path or directory.
- **Number:** Specifies a number value.
- **Boolean:** Specifies a Boolean value. The VALUE field provides a checkbox (checked = true).
- **Password:** Specifies a password value. The password is masked and displays only asterisks.

Variables also allow greater flexibility for your WSDL files. For example, you can use the same WSDL for different runtime environments by using application variables to specify system specific information. These values can then be modified from the binding component runtime properties as needed.

When you deploy an application that uses Application Variables, all of the Application Variables that are referenced in the application's WSDL files are loaded automatically. If you attempt to start an application and an Application Variables value is not defined (no value is specified for the Application Variable) an exception is thrown.

To change a property when the application is running, change your Application Variable property value, then right-click your application in the Services window under Servers → GlassFish → JBI → Service Assemblies, and click Stop in the popup menu. When you restart your project, your new settings will take effect.

# Using Application Variables for Password Protection

To protect passwords that would otherwise appear as clear text in your WSDL file, you can enter a password Application Variable as a token. In the following example, a password Application Variable is created that uses the name SECRET and the password PROTECT.

## ▼ Creating a Password Application Variable

1. **In the NetBeans Servers window, expand Servers → GlassFish V2 → JBI → Binding Components.**
2. **Right click sun-hl7-binding.**
   The sun-hl7-binding Properties appear in the Properties window.
3. **Click the Application Variables property ellipsis button.**
   The Application Variables editor appears.
4. **Click Add, select Password as your variable type, and then click OK.**
   A new row is added to the Application Variables editor.
5. **Enter SECRET as the name, and enter PROTECT as the value.**
   Because this is a password type, the characters of your password are displayed as asterisks.
6. **Use the application variable name ${SECRET} as your WSDL password attribute, using the dollar sign and curly braces as shown.**

# Runtime Properties

The HL7 Binding Component properties specify Thread Count, Application Configuration, Application Variables, Statistics, Loggers, and reference the Binding Component's description, name, type, and state.

| Header | Header |
|---|---|
| Description | Displays the description of the binding component. |
| Name | Displays the name of the binding component. |
| State | Indicates the state of the binding component as Started, or Stopped. |
| Type | Displays the component type. |
| Build Number | Displays the build version for the current component. |
| Version | Displays the components specification version. |

| Header | Header |
|---|---|
| Number of Outbound Processor Threads | Specifies the number of threads configured to concurrently process outbound HL7 requests. The value range is an integer from 1 to 2147483647. |
| Number of Inbound Reply Threads | Specifies the Maximum number of threads configured to process responses to HL7 client requests. The value range is an integer from 1 to 2147483647. |
| Data Source Name | Specifies the name of the data source where the sequence number and the HL7 acknowledgement message are persisted. |
| Allow Dynamic Endpoint | Specifies if Dynamic Endpoints are allowed for the HL7 Binding. For more information, see "Dynamically Configuring HL7 Endpoints" on page 33 |
| Application Configuration | Specifies the values for a Composite Application's external connectivity parameters, which are normally defined in the WSDL service extensibility elements. You can apply these values to a user-named endpoint ConfigExtension Property. The Application Configuration property editor includes fields for all of the connectivity parameters that apply to that component's binding protocol. When you enter the name of a saved ConfigExtension and define the connectivity parameters in the Application Configuration editor, these values override the WSDL defined connectivity attributes when your project is deployed. To change these connectivity parameters again, you simply change the values in the Application Configuration editor, then shutdown and start your Service Assembly to apply the new values.<br><br>The Application Configuration parameters include the following:<br>■ **Application Configuration Name:** Specifies the name of the Application Configuration.<br>■ **Acknowledgement Mode:** Indicates the acknowledge mode type: original or enhanced.<br>■ **SFT Enabled:** Enables or disables SFT segment processing.<br>■ **Encoding Characters:** Specifies the encoding characters to be used in creating the NAK for invalid HL7 messages. This attribute contains the four characters in the following order: the component separator, repetition separator, escape character, and subcomponent separator. Recommended values are ^~\& (that is, ASCII 94, 126, 92, and 38, respectively).<br>■ **End Block Char:** Indicates the End Block Character Value in decimal ASCII number from 1 to 127. To be strictly comply with the HL7 standard, this parameter must be set to a carriage Return, which is decimal 13.<br>■ **End Data Char:** Indicates the End Data Character Value in decimal ASCII number from 1 to 127. Unless there is a conflict, the value should be ASCII VT, which is decimal 28. |

| Header | Header |
|---|---|
| | ■ **Field Separator:** Defines the Field Separator character value in a decimal ASCII number. This represents the separator between the segment ID and the first field, MSH-2-encoding characters. As such, it serves as a separator and defines the character to be used as a separator for the rest of the message. The default setting is 124 which is the character "\|". The allowed range is 1 to 127. This attribute value is used in creating the NAK for invalid HL7 messages.<br>■ **HLLP Checksum:** Specifies if HLLP CheckSum is enabled.<br>■ **Host:** Specifies the host part of a URL used to connect to an HL7 external system. The equivalent WSDL attribute is hl7:address → location(host).<br>■ **LLP Type:** Indicates the Lower Layer Protocol Type as MLLPv1 (Minimal Lower Layer Protocol release 1), MLLPv2 (Minimal Lower Layer Protocol release 2), and HLLP (Hybrid Lower Layer Protocol). The equivalent WSDL attribute is hl7:protocolproperties → llpType.<br>■ **MLLPV2 Retries Count on Nak:** Specifies the maximum number of retries on receipt of MLLP V2 negative acknowledgement. The equivalent WSDL attribute is hl7:protocolproperties → mllpv2RetriesCountOnNak.<br>■ **MLLPV2 Retry Interval:** Specifies the time duration to wait in milliseconds before each retry. The equivalent WSDL attribute is hl7:protocolproperties → mllpv2RetryInterval.<br>■ **MLLPV2.0 Time to Wait For ACK/NAK:** Specifies the time duration to wait in milliseconds for receiving MLLP V2 commit acknowledgement / negative acknowledgement. The equivalent WSDL attribute is hl7:protocolproperties → mllpv2TimeToWaitForAckNak.<br>■ **Host Port:** A URL part, which specifies the connectivity information to connect to the HL7 external system. The equivalent WSDL attribute is hl7:address → location (port).<br>■ **Processing ID:** Specifies the ProcessingID value against which the MSH-11-ProcessingID field in the received message is validated when validateMSH is set to "true". Valid values are P (production), D (debugging), or T (training). The equivalent WSDL attribute is hl7:protocolproperties → processingID<br>■ **Sending Application:** Specifies the MSH-03 Sending Application to be used in creating the NAK for invalid HL7 messages.<br>■ **Sending Facility:** Specifies the MSH-04 Sending Facility to be used in creating the NAK for invalid HL7 messages. |

| Header | Header |
| --- | --- |
| | ■ **Sequence Number:** Specifies if sequence number protocol is enabled. The equivalent WSDL attribute is hl7:protocolproperties → seqNumEnabled.<br>■ **Software Binary ID:** 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.<br>The equivalent WSDL attribute is hl7:protocolproperties → softwareBinaryID.<br>■ **Software Version or Release Number:** 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.<br>The equivalent WSDL attribute is hl7:protocolproperties → softwareCertifiedVersionOrReleaseNumber.<br>■ **Software Install Date:** Specifies HL7 segment SFT-06, the Software Install Date. This is the date, in YYYYMMDDHHSS format, on which the submitting software was installed at the sending site.<br>The equivalent WSDL attribute is hl7:protocolproperties → softwareInstallDate.<br>■ **Software Product Information:** 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.<br>The equivalent WSDL attribute is hl7:protocolproperties → softwareProductInformation.<br>■ **Software Product Name:** 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.<br>The equivalent WSDL attribute is hl7:protocolproperties → softwareProductName.<br>■ **Software Vendor Organization:** Defines the Software Vendor Organization field (SFT-1-Software Vendor Organization) which identifies the vendor who is responsible for maintaining the application.<br>The equivalent WSDL attribute is hl7:protocolproperties → softwareVendorOrganization. |

| Header | Header |
|---|---|
| | ■ **Start Block Character:** Indicates the Start Block Character Value in a decimal ASCII number from 1 to 127. Unless there is a conflict, the value should be ASCII VT, which is decimal 11.<br>The equivalent WSDL attribute is hl7:protocolproperties → startBlockCharacter.<br>■ **Validate MSH:** Specifies if the MSH segment in the HL7 message is validated against initiation rules.<br>The equivalent WSDL attribute is hl7:protocolproperties → validateMSH.<br>■ **Version ID:** Specifies the versionID value against which MSH-12-VersionID field in the received message is validated when validateMSH is set to true. Valid values are 2.1, 2.2, 2.3, 2.3.1, 2.4, 2.5, 2.5.1 or 2.6. |
| Application Variable | The Application Variables configured for the binding component. Application Variables allow you to define a list of name:value pairs for a given stated type. The application variable name can be used as a token for a WSDL extensibility element attribute in a corresponding binding.<br>The Application Variables configuration property offers four variable types:<br>■ **String:** Specifies a string value, such as a path or directory.<br>■ **Number:** Specifies a number value.<br>■ **Boolean:** Specifies a Boolean value. The VALUE field provides a checkbox (checked = true).<br>■ **Password:** Specifies a password value. The password is masked and displays only asterisks.<br>For more information about using Application Variables, see "Using Application Variables to Define Name/Value Pairs" on page 46 and "Using Application Variables for Password Protection" on page 48. |

# Statistics Properties

The Statistics properties provide a record of key statistics for the HL7 Binding Component.

| | |
|---|---|
| Activated Endpoints | Tracks the number of activated endpoints. |
| Active Exchanges | Tracks the number of active exchanges. |
| Avg. Component Time | Tracks the average message exchange component time in milliseconds. |
| Avg. D.C. Time | Tracks the average message exchange delivery channel time in milliseconds. |

| | |
|---|---|
| Avg. Msg. Service Time | Tracks the average message exchange message service time in milliseconds. |
| Avg. Response Time | Tracks the average message exchange response time in milliseconds. |
| Completed Exchanges | Tracks the total number of completed exchanges. |
| Error Exchanges | Tracks the total number of error exchanges. |
| Received Dones | Tracks the number of received dones. |
| Received Errors | Tracks the number of received errors. |
| Received Faults | Tracks the number of received faults. |
| Received Replies | Tracks the number of received replies. |
| Received Requests | Tracks the number of received requests. |
| Sent Dones | Tracks the number of sent dones. |
| Sent Errors | Tracks the number of sent errors. |
| Sent Faults | Tracks the number of sent faults. |
| Sent Replies | Tracks the number of sent replies. |
| Sent Requests | Tracks the number of sent requests. |
| Up Time | Tracks the up time of this component in milliseconds. |

# Logger Properties

The HL7 Binding Component runtime Logger properties include 13 different component activities that can be monitored and recorded at user-designated levels.

Each logger can be set to record information at any of the following levels:

- **FINEST:** messages provide highly detailed tracing
- **FINER:** messages provide reasonably detailed tracing
- **FINE:** messages provide basic tracing
- **CONFIG:** provides static configuration messages
- **INFO:** provides informative messages
- **WARNING:** messages indicate a warning
- **SEVERE:** messages indicate a severe failure
- **OFF:** no logging messages

### HL7 Binding Component Loggers

The values for the HL7 Binding Component Loggers start with the location: sun-hl7-binding. The value text has been wrapped for display purposes.

| | |
|---|---|
| sun-hl7-binding | Specifies the logging level for the entire group of component loggers. Individual logger levels can then be changed as desired. `com.sun.jbi.hl7bc` |
| EndpointImpl | `com.sun.jbi.hl7bc.EndpointImpl` |
| HL7BindingComponent | `com.sun.jbi.hl7bc.HL7BindingComponent` |
| HL7BindingDeployer | `com.sun.jbi.hl7bc.HL7BindingDeployer` |
| HL7Denormalizer | `com.sun.jbi.hl7bc.HL7Denormalizer` |
| HL7Normalizer | `com.sun.jbi.hl7bc.HL7Normalizer` |
| InboundMessageProcessor | `com.sun.jbi.hl7bc.InboundMessageProcessor` |
| InboundReceiver | `com.sun.jbi.hl7bc.InboundReceiver` |
| OutboundMessageProcessor | `com.sun.jbi.hl7bc.OutboundMessageProcessor` |
| OutboundReciever | `com.sun.jbi.hl7bc.OutboundReciever` |
| ServiceUnitImpl | `com.sun.jbi.hl7bc.SchedulerEndpointManager` |
| DeploymentLookup | `com.sun.jbi.hl7bc.DeploymentLookup` |
| MessagingChannel | `com.sun.jbi.hl7bc.MessagingChannel` |

For more information about the HL7 Binding Component Logging Codes, see Logging Codes By Component.

# Using HL7 Quality of Service (QoS) Features

Quality of Service features are configured from the CASA Editor, and include properties used to configure retry (redelivery) and throttling.

This section contains the following topics:

# Quality of Service (QoS) Properties

The QoS attributes are configured from the QoS Properties Editor, accessed from the Composite Application Service Assembly (CASA) Editor. For an example of how to access the QoS Properties Editor, see "Using Message Throttling" on page 56.

Service Name (Consumer)
: Specifies the consumer service name. Provides a list of preexisting namespaces. Click the ellipses button to open the QName Editor.

  Example: `ns1:http://j2ee.netbeans.org/wsdl/HL7V2InboundProject/hl7receive`

Endpoint Name (Consumer)
: Indicates the consumer endpoint name.

Service Name (Provider)
: Specifies the provider service name. Provides a list of preexisting namespaces. Click the ellipses button to open the QName Editor.

  Example: `ns1:http://j2ee.netbeans.org/wsdl/HL7V2InboundProject/hl7receive`

Endpoint Name (Provider)
: Indicates the consumer endpoint name.

Max Attempts (Redelivery Extension)
: Specifies the number of retries to attempt before using the On Failure option.

  Example: 20

Wait Time (Redelivery Extension)
: Specifies time (in milliseconds) to wait between redelivery attempts.

  Example: 300

On Failure (Redelivery Extension)
: Specifies the type of action to be taken when message exchange (ME) redelivery attempts have been exhausted.

  The On Failure options are:

  - **Delete:** When the final defined delivery attempt has failed, the QoS utility abandons the message exchanges (ME) and returns a Done status to the JBI component, which proceeds to its next process instance. This option is only supported for In-Only message exchanges.
  - **Error:** When the final defined delivery attempt has failed, the QoS utility returns an Error status to the JBI component, and the JBI component throws an Exception. This is the default option, and is supported for both In-Only and In-Out message exchanges.

- **Redirect:** Similar to the Delete option, except that the QoS utility reroutes the ME to the configured redirect endpoint when the maxAttempts count has been exhausted. If the QoS utility is successful in routing the message to the redirect endpoint, a Done status is returned to the JBI component; otherwise, an Error status is returned. This option is supported for In-Only message exchanges only.
- **Suspend:** The QoS utility returns an Error status to the JBI component if it is not able to deliver the ME to the actual provisioning endpoint. After the re-delivery attempts have been exhausted, the JBI Component suspends the process instance. This option is only supported if monitoring is enabled in the JBI Component, since the user must use the monitoring tool to resume a suspended instance. This option is supported for both In-Only and In-Out message exchanges.

Example: Delete

Maximum Concurrency Limit (Throttling Extension)
: Specifies the maximum number of concurrent messages that can be processed on a specific connection. This number is used to set up the maximum number of concurrent messages that the internal endpoint sends to the provider endpoint.

  Example: 10

# Using Message Throttling

Throttling allows you to set the maximum number of concurrent messages that are processed by a particular endpoint. Increased message load and large message payloads can cause memory usage spikes that can decrease performance. Throttling limits resource consumption so that consistent performance is maintained.

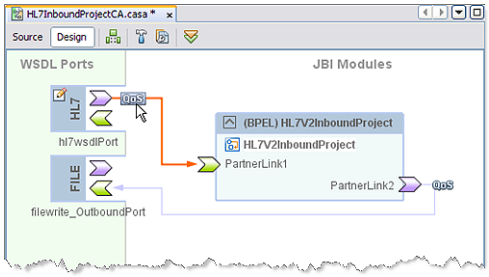
The HL7 Binding Component can manage the flow of messages by evaluating endpoints to determine when it is necessary to suspend requests and when to resume processing as usual.

Throttling for the HL7 Binding Component is a QoS feature configured from the CASA Editor.

## To Configure Throttling for an HL7 WSDL Port

1. **From the NetBeans IDE Projects window, right-click the Service Assembly node under your composite application, and then select Edit.**

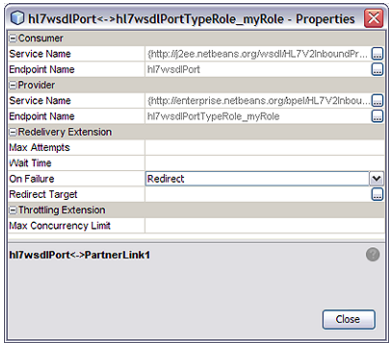
   The CASA Editor opens containing your composite application.




2. **In the CASA Editor, click the QoS icon located near the HL7 WSDL port you want to configure.**

   The QoS Properties Editor appears.

3. **In the QoS Properties Editor, click the property field for maximumConcurrencyLimit under ThrottlingExtension, and enter an integer for the maximum number of concurrent messages allowed for this endpoint.**




4. **Click Close.**

   The appropriate throttling configuration for the connection is generated in the project's `jbi.xml` file, when the service assembly is built.

# Using Redelivery

Redelivery is a Quality of Service mechanism that handles message delivery when first-time delivery fails. Redelivery allows you to define the number of attempts that the system makes to deliver a message, the time between attempts, and the final result for an undeliverable message or nonresponsive endpoint.

Redelivery is configured for a specific connection from the Composite Application Service Assembly (CASA) Editor, by clicking the QoS icon for that connection. From the RedeliveryExtension section of the QoS Properties Editor, configure the Redelivery properties.

The Redelivery configuration parameters are:

- **Max Attempts:** The number of times that the project attempts to re-deliver a message. An error status is returned to the JBI component for each failed attempt.
- **Wait Time:** The time, in milliseconds, that the project waits between redelivery attempts.
- **On Failure:** The actions taken and the message destination when the specified redelivery attempts have been exhausted. This parameter has four options: Delete, Redirect, Suspend, and Error. See the QoS Properties section for more information.

---

**Note –** The On Failure options: Delete and Redirect, cannot be applied to In-Out message exchanges because In-Out message exchanges require a specific response from the process instance to proceed further, and as such, the return value for these options does not suffice.

---