webMethods HL7 Module Installation and User's Guide

Version 7.1

July 2013

WEBMETHODS

Document ID: ESTD-HL7-IUG-71-20200925

Table of Contents

About this Guide

This guide describes how to install, configure, and use webMethods HL7 Module. It contains information for administrators and application developers who want to exchange HL7 messages with other trading partners using webMethods HL7 Module.

To use this guide effectively, you should be familiar with:

- webMethods Integration Server and Integration Server Administrator, and understand the concepts and procedures in the *webMethods Integration Server Administrator's Guide*.
- webMethods Trading Networks and understand the concepts and procedures described in the various Trading Networks guides.
- Software AG Designer and Developer, and understand the concepts and procedures described in the *webMethods Service Development Help* and the *webmethods Developer User's Guide*.

  Note:
  Procedures for creating flow services and running webMethods HL7 Module services are similar in Developer and Designer.

- My webMethods Server and its interface My webMethods, and understand the concepts and procedures described in the appropriate My webMethods documentation for your release.
- Have a basic knowledge of HL7 standards and HL7 terminology.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies service names and locations in the format *folder.subfolder.service*, APIs, Java classes, methods, properties.
Italic	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.

Convention	Description
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at http://documentation.softwareag.com.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at https://empower.softwareag.com/.

You can find product information on the Software AG Empower Product Support website at https://empower.softwareag.com.

To submit feature/enhancement requests, get information about product availability, and download products, go to Products.

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the Knowledge Center.

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at https://empower.softwareag.com/public_directory.aspx and give us a call.

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at http://techcommunity.softwareag.com. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

1 Overview of webMethods HL7 Module

What is HL7?

Health Level Seven (HL7) is one of several American National Standards Institute (ANSI) -accredited Standards Developing Organizations (SDOs) operating in the area of healthcare. HL7 develops standards for the electronic interchange of clinical, financial, and administrative information among independent healthcare-oriented computer systems, such as hospital information systems. For more information about HL7, see http://www.hl7.org.

HL7 Messaging Standards

HL7 specifies a number of flexible standards, guidelines, and models by which various healthcare systems can communicate with each other. Such guidelines or data standards are a set of rules that enable healthcare organizations to share and process clinical information in a uniform and consistent manner. For information about the types of HL7 standards, see http://www.hl7.org.

HL7 messaging standards version 2.x and 3.0 determine how the information is captured and communicated between two parties. These messaging standards have evolved from textual, non-XML and delimiter-based representation to XML-based syntax.

HL7 Version 2.x Messaging Standard

The HL7 version 2 messaging standard defines a series of electronic messages to support administrative, logistical, financial, and clinical processes. The standard has undergone frequent updates and has resulted in a series of ANSI approved versions, collectively know as version 2.x.

The HL7 version 2.x messaging standard uses two types of encoding:

- **Standard encoding.** The standard encoding is a textual, non-XML encoding syntax based on delimiters, also known as encoding rule-7 format (ER7).
- **HL7 XML encoding.** The HL7 XML encoding is defined by XML encoding rules that represent HL7 message structures, components and fields as XML elements.

For examples of HL7 version 2.x messages of the two encoding types, see “Example: HL7 ER7-Encoded Message” on page 84 and “Example: HL7 XML-Encoded Message” on page 84.

HL7 Version 3.0 Messaging Standard

The HL7 version 3.0 messaging standard defines a series of electronic messages (called interactions) to support all healthcare workflows. HL7 version 3.0 messages are based on an XML encoding syntax. HL7 version 3.0 messages include the concepts of message wrappers, sequential interactions, and model-based message payloads.

What is webMethods HL7 Module?

webMethods HL7 Module provides support for the popular version 2.x HL7 Messaging Standards. HL7 Module runs on top of webMethods Integration Server. When you use HL7 Module along with other webMethods components, such as Trading Networks, you can extend its capabilities. Using HL7 Module, you create, parse and validate supported versions of HL7 messages. In addition, HL7 Module provides the functionality to send and receive HL7 messages from other partners, as well as send the appropriate message acknowledgment.

HL7 Module has the following features:

- **Version 2.x HL7 messages.** Supports the HL7 Messaging Standards version 2.x. For a list of supported HL7 message versions, see *webMethods eStandards Modules System Requirements* .

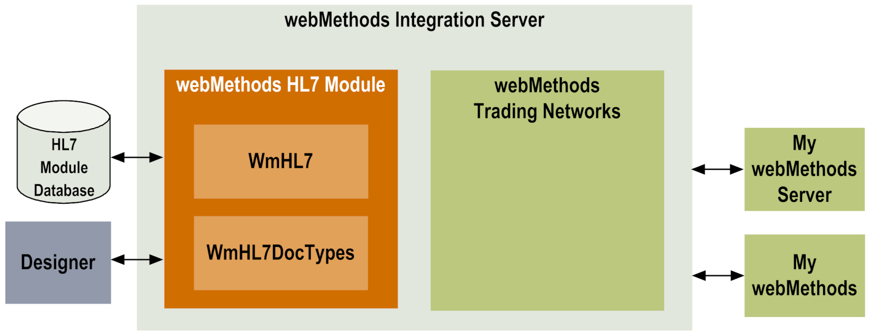
  Note:HL7 Module supports processing of only those message versions for which you have configured the module. For example, if Integration Server is configured for HL7 message versions 2.5 and 2.6, sending an HL7 message version 2.5 or 2.6 to Integration Server will succeed, but sending an HL7 message version 2.3 will fail.

- **Version 3.0 HL7 Messages.**Provides the schema for the HL7 Messaging Standard version 3.0 Normative Edition 2010, supplied by the Health Level Seven standards organization.
- **HL7 Message Processing.** Generates HL7 messages into HL7 version 2.x ER7 or XML format and parses ER7 and XML-encoded HL7 version 2.x messages.
- **Custom Message Schemes.** Supports multiple instances of the same message scheme with the same message version, but different in structure. The custom message schemes are specific to a trading partner.
- **Message Validation.**Supports validation of HL7 version 2.x messages for structural and semantic consistency, as well as verification of the codes in the HL7 version 2.x messages against the code tables. With HL7 Module, you can perform partner specific code table validation.
- **HL7 Code Tables.** Supports managing and using code tables for encoding data elements and validating HL7 version 2.x messages. Code tables are used to check for compliance during the message transaction. HL7 Module also supports managing and using code tables for a specific Trading Networks trading partner.
- **Standard Communication Protocols.** Supports all standard transport protocols to exchange messages with partners, such as HTTP(S), FTP, and E-mail.
- **MLLP Communication Protocol.** Supports communication via the Minimum Lower Level Protocol (MLLP) that is widely used among HL7 applications.

- **Error Handling.** Reports errors encountered during the processing of the MSH segment or the message itself.
- **Message Persistence.** Saves all incoming and outgoing messages in the Trading Networks database.
- **Auto Acknowledgment.** Supports sending an auto acknowledgment to the sending application.

Architecture and Components

The following diagram illustrates the HL7 Module architecture and components, and the component relationships. For further explanation, see the table that follows the diagram.



Component	Description
webMethods HL7 Module	webMethods HL7 Module enables your enterprise to construct, parse, and transport HL7 version 2.x messages. HL7 Module is installed and runs on Integration Server. HL7 Module contains the following packages: ■ WmHL7 ■ WmHL7DocTypes For a description of the packages, see " webMethods HL7 Module Packages" on page 13.
webMethods Integration Server	webMethods Integration Server is the underlying foundation of the webMethods architecture. It processes requests from and relays responses to a back-end system. HL7 Module uses the Integration Server WmART package to manage MLLP connections and related services. HL7 Module uses the Integration Server built-in infrastructure services for managing the IS documents generated from the HL7 Schema definition. In addition, HL7 Module uses the WmFlatFile package for processing the schema.

Component	Description
	The webMethods data format supported on Integration Server is based on the following elements: ▪ *IData object.* The IData object is the universal container that services use to receive input from and deliver output to other programs. It contains an ordered collection of key/value pairs on which a service operates. Each element stored in an IData object corresponds to a data type. ▪ *An IS document type.* An IS document type contains a set of fields used to define the structure and type of data in a document (IData object). You can use an IS document type to specify input or output parameters for a service or specification. You can also use an IS document type to build a document or document list field and as the blueprint for pipeline validation and document (IData object) validation. For more information about IData objects and IS document types, see the *webMethods Service Development Help*.
webMethods Trading Networks	webMethods Trading Networks enables your enterprise to link with other companies (buyers, suppliers, strategic partners) and marketplaces to form a business-to-business trading network. Trading Networks enhances the functionality of HL7 Module and facilitates the exchange of HL7 messages.
My webMethods Server and My webMethods	My webMethods Server is a Web-based monitoring and administration user interface for managing your webMethods components. You use My webMethods Server (and its user interface, My webMethods) with HL7 Module to define and manage Trading Networks partner profiles, trading partner agreements (TPAs), custom attributes, TN document types, and processing rules. You also use My webMethods to monitor and manage HL7 message transactions. For information about using My webMethods to manage HL7 Module, see "Defining Trading Networks Information" on page 67.
Designer	At design time, use Designer or Developer to edit HL7 Module services and create customized solutions. It also provides tools for testing and debugging the solutions you create.
HL7 Module Database	The HL7 Module database is a relational database that HL7 Module uses to store code tables.

webMethods HL7 Module Packages

HL7 Module contains the following packages (sets of services and related files) that you install on Integration Server.

Package	Description
WmHL7	Contains built-in services to: ■ Generate IS document types for the supported HL7 version 2.x messages. ■ Parse an ER7 or XML message for the supported HL7 version 2.x messages. ■ Create an ER7 or XML-encoded message from the IS Document format. ■ Manage code tables. ■ Validate HL7 version 2.x messages. ■ Facilitate interaction between the WmHL7 package and Trading Networks that enables you to exchange HL7 version 2.x messages with your trading partners.
WmHL7Doc Types	Contains the IS document types generated for the supported HL7 version 2.x messages using the wm.ip.hl7.utils:generateMessageScheme service.

For detailed information about the HL7 Module services and generated IS document types, see "webMethods HL7 Module Services" on page 139.

Message Schemes and IS Document Types

Before you use HL7 Module to parse or create HL7 version 2.x messages, you need to generate the data model and the IS document type for the supported HL7 message versions. The generated IS document types serve as input to the service that you use to create HL7 messages.

For HL7 messages version 2.x, HL7 Module generates IS document types for each HL7 message type based on message definitions (XML schemas) across HL7 versions using HL7 Module services.

For HL7 messages version 3.0, HL7 Module provides only valid HL7 version 3 schemas. The HL7 version 3 XML schemas are installed with HL7 Module and located in the *IntegrationServer_directory*\packages\WmHL7\data\v3 directory. You must use the Integration Server built-in services for generating IS document types for HL7 3.0 messages as described in the *webMethods Integration Server Administrator's Guide* for your release.

Note:
The terms *message definition* and *XML schema* (XSD) are used interchangeably throughout the guide.

Default Message Scheme

Each set of message definitions is identified by a message scheme ID. HL7 Module provides a set of XML schemas (supplied by the HL7 standards organization) for each supported HL7 2.x message version. These XML schemas represent the default message scheme for HL7 Module. The scheme ID for the HL7 Module default message scheme is `wm.ip.hl7.docType`. The default XML schemas are installed with HL7 Module and located in the *Integration Server_directory* \packages\WmHL7\data\xsd\wm.ip.hl7.docType\ v*messageVersion* directory. For information about how to configure HL7 Module to use the default message scheme, see "Generating Message Schemes" on page 33.

Custom Message Scheme

In some cases you may need to customize the default message scheme to meet your business requirements. For example, you may need to add an additional field to an existing segment or define a new custom Z-segment. With HL7 Module, you can customize the XML schemas for the HL7 message definitions by making copies of the default message definitions (XML schemas) provided with the module and modifying them as required. These customized XML schemas represent the custom message scheme. For information about how to configure HL7 Module to use the custom message scheme, see "Creating a Custom Message Scheme" on page 34.

Using Trading Networks, two trading partners can exchange messages that conform to a specific HL7 message scheme. To use a specific HL7 message scheme between two trading partners, you must generate the message scheme in HL7 Module and then associate the message scheme ID with the trading partner agreement that you use for the message exchange. For information about associating a message scheme ID with a trading partner agreement, see "Associating a Trading Partner Agreement with a Message Scheme ID" on page 39.

HL7 Code Tables

Data fields and their components in HL7 messages are related to HL7 code tables that contain groups of code sets. HL7 Module provides services that you use to:

- Insert and manage code tables in the HL7 Module database.
- Customize code tables for specific Trading Networks partners.
- Validate HL7 version 2.x messages based on the trading partner.

Managing Standard and Customized Code Tables

HL7 Module supports both standard and customized code tables. The standard code tables contain standard HL7 code sets that are not specific to any Trading Networks trading partner. They are used for all Trading Networks partners for which you do not require customized HL7 code tables. Customized code tables contain modified code sets that are specific to a Trading Networks partner and that you use only in the message exchange with that partner.

You must configure the HL7 Module database before you can manage or use code tables. For more information about how to do that, see "Configuring the HL7 Module Database" on page 32.

You manage standard and custom code tables and code table values using the HL7 Module user interface in Integration Server Administrator.

For more information about:

- Using the HL7 Module user interface to manage code tables, see "Managing HL7 Code Tables" on page 95.
- The operations that you can perform with code tables and code table values, see "Using Code Tables" on page 16.
- The HL7 Module code table services, see " webMethods HL7 Module Services" on page 139.

Using Code Tables

The following table lists the operations that you can perform with code tables and code table values using HL7 Module:

Operation	Use to:
Add a code table.	Include a new code table.
Add a set of code values.	Incorporate the code sets of a new HL7 standard into an existing HL7 code table.
Delete a code table.	Update the table list in the HL7 Module database.
Delete code table values.	Remove code sets that are not in use from an existing code table.
Load a code table and its values from an input file.	Update or upgrade a code table.
Enable or disable a code table.	■ Optimize your system by enabling only code tables that you need. ■ Disable code table validation of HL7 version 2.x messages by disabling all code tables.
Customize a code table.	■ Modify one or more values for a code table for a specific trading partner. ■ Modify a set of code tables and their properties for a specific trading partner.
Get a list of code tables	Obtain a list of code tables available in the HL7 Module database.
Get code table values	Obtain a list of code table values associated with an existing code table.

For more information about how to perform different operations with code tables and code table values from the HL7 Module user interface, see "Managing HL7 Code Tables" on page 95.

Processing HL7 Version 2.x Messages with Trading Networks

To process HL7 version 2.x messages with Trading Networks, you use the facilities provided by HL7 Module as follows:

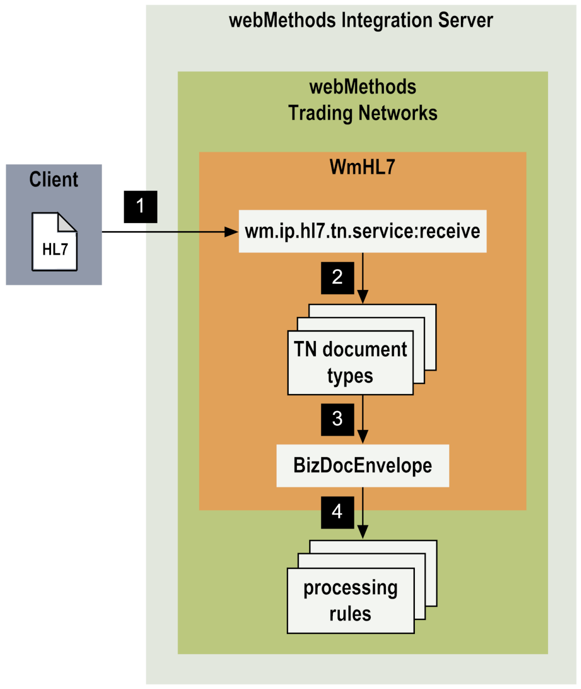
- **Inbound processing.** HL7 Module receives HL7 version 2.x messages from a business partner and then sends them to Trading Networks for processing.
- **Outbound processing**. HL7 Module generates HL7 version 2.x messages from the internal representation (the IS document format supported by Integration Server) and sends them to other business partners via Trading Networks.

Processing Inbound HL7 Version 2.x Messages

When you install HL7 Module, the WmHL7 package of the HL7 Module enables Trading Networks to recognize and to process HL7 version 2.x messages. For inbound processing, you create clients that send HL7 version 2.x messages to the HL7 Module wm.ip.hl7.tn.service:receive service and you set up information in Trading Networks to process the messages.

Note:
You can send HL7 messages directly to the Trading Networks receive service. However, HL7 Module creates the BizDocEnvelope only if the incoming message is a valid HL7 message. When an HL7 message is sent directly to the Trading Networks receive service, HL7 Module does not perform the tasks of persisting the message, routing, and assigning process or user status.

To start the run-time processing of the HL7 version 2.x message in Trading Networks, HL7 Module sends the message to Trading Networks. The following diagram illustrates the steps in which Trading Networks processes inbound HL7 version 2.x messages.


webMethods Integration Server
webMethods Trading Networks
WmHL7
Client
HL7
1
wm.ip.hl7.tn.service:receive
2
TN document types
3
BizDocEnvelope
4
processing rules

Step	Description
1	The client sends the HL7 version 2.x message to Trading Networks, invoking the wm.ip.hl7.tn.service:receive service to pass on the HL7 message to Trading Networks for processing.
2	When HL7 Module receives the HL7 version 2.x message, it first determines which TN document type to use. After determining the TN document type, HL7 Module extracts field values from the HL7 message and fills in the appropriate attributes in the BizDocEnvelope.
3	HL7 Module then forms a BizDocEnvelope that contains: ■ The original HL7 version 2.x message as the `contentPart` of the BizDocEnvelope. ■ The extracted fields from the MSH segment of the HL7 version 2.x message added as attributes to the BizDocEnvelope. ■ Information required for persisting the message in Trading Networks. ■ Information required for routing and processing the HL7 version 2.x message.
4	After forming the BizDocEnvelope, HL7 Module determines the processing rule to use to process the document and executes the processing rule. Processing rules define the pre-processing and processing actions you want performed on each type of document. For more information about processing rules, see the following section, "Processing Rules to Process HL7 Version 2.x Messages" on page 78.

Processing Outbound HL7 Version 2.x Messages

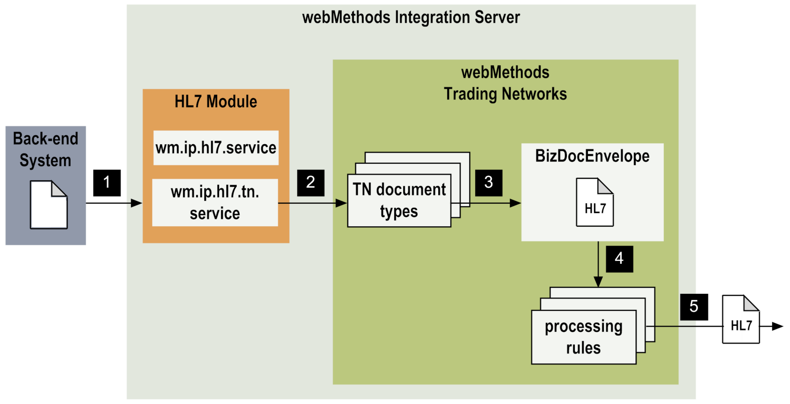
For outbound processing, you create an HL7 version 2.x message that can be sent to other trading partners. For example, you might use data from an internal document (such as a document from a back-end system) to form the HL7 version 2.x message. To create an HL7 version 2.x message from the IS document, you use the HL7 Module built-in services.

To deliver the HL7 version 2.x messages you create using HL7 Module, you have the following options:

- Send the HL7 version 2.x message using the HL7 Module wm.ip.hl7.tn.service:send service. This service obtains the BizDocEnvelope, then persists the message in Trading Networks, and executes the processing rules for the BizDocEnvelope. The send service sends the HL7 message using one of the Trading Networks transport protocols.
- Submit the outbound HL7 version 2.x messages to Trading Networks document recognition. With this method, you must have a TN document type for the outbound HL7 message, and a processing rule to deliver the outbound HL7 message. The outbound HL7 message is always saved to the Trading Networks database before it is delivered.

Delivering Outbound HL7 Version 2.x Messages Using the HL7 Module Send Service

The following diagram illustrates the process of receiving an internal-format document, creating an HL7 version 2.x message from that document, and submitting the outbound HL7 message to Trading Networks for processing and delivery using the HL7 Module wm.ip.hl7.tn.service:send service.

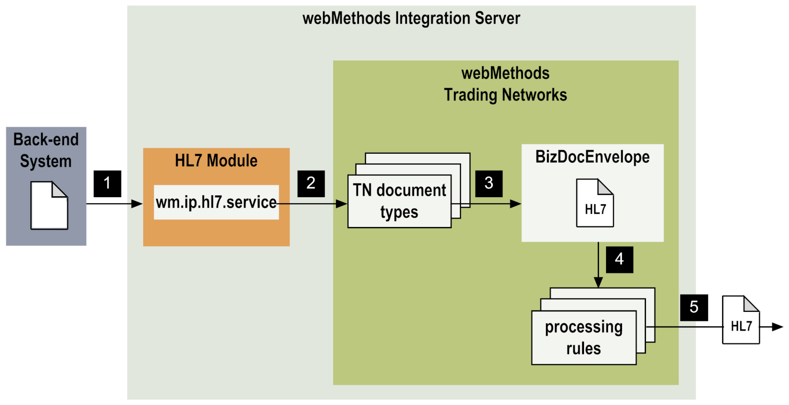


Step	Description
1	A back-end system or client sends an internal-format document to HL7 Module, invoking the wm.ip.hl7.service:convertIDataToHL7 service to convert the IS document into an HL7 version 2.x message. For information about creating an HL7 version 2.x message, see "Creating Outgoing HL7 Version 2.x Messages" on page 83.
2	After creating the HL7 version 2.x message, HL7 Module invokes the wm.ip.hl7.tn.service:send service to submit the outbound HL7 version 2.x message into Trading Networks document processing.
3	During the initial processing of the HL7 version 2.x message, the HL7 Module wm.ip.hl7.tn.service:send service either: ■ Obtains the BizDocEnvelope created by the `createBizDoc` service. or ■ Generates the BizDocEnvelope internally using the information from the HL7 message fields. The HL7 Module send service selects the BizDocEnvelope that matches the required TN document type for the outbound HL7 message. The BizDocEnvelope contains the outbound HL7 message as the content, and the values from the message fields as attributes.
4	After the HL7 Module send service takes the BizDocEnvelope, the service searches and executes the appropriate processing rules for the respective TN document type for the HL7 version 2.x message, to perform a required action before sending the message.
5	The HL7 Module send service sends the HL7 version 2.x message using one of the Trading Networks transport protocols.

Step	Description
	For more information about configuring the Trading Networks transport protocols, see "Configuring Communication Protocols" on page 42 and the *webMethods Trading Networks Administrator's Guide* for your release.

Submitting Outbound HL7 Version 2.x Messages to Trading Networks for Delivery

The following diagram illustrates the process of receiving an internal-format document, creating an HL7 version 2.x message from that document, and submitting the outbound HL7 version 2.x message to Trading Networks to process and deliver the outbound HL7 message through Trading Networks.



Step	Description
1	A back-end system or client sends an internal-format document to HL7 Module, invoking the wm.ip.hl7.service:convertIDataToHL7 service to convert the IS document into an HL7 message. For information about creating an HL7 message, see "Creating Outgoing HL7 Version 2.x Messages" on page 83.
2	After creating the HL7 version 2.x message, the service invokes the wm.ip.hl7.tn.service:receive service to submit the outbound HL7 version 2.x message into Trading Networks document recognition. Trading Networks document recognition determines the TN document type to use for the HL7 message. For more information, see step 2 in "Processing Inbound HL7 Version 2.x Messages" on page 17.
3	You can use a user-defined utility service to create a BizDocEnvelope that contains the HL7 message, the attribute values that match the TN document type for the particular message, and the information for processing this document in Trading Networks.
4	The user-defined utility service then invokes the wm.tn.route:routeBizDoc service to send the BizDocEnvelope that contains the HL7 message to Trading Networks to select a processing rule.

Step	Description
5	Trading Networks searches its processing rules to find the appropriate rule to use to deliver the HL7 version 2.x message. You need to create a processing rule for the HL7 version 2.x message that uses the **Deliver Document By** processing action. For more information on the delivery options that you specify with the **Deliver Document By**processing action in a processing rule, see the *webMethods Trading Networks Administrator's Guide* for your release.

webMethods HL7 Module MLLP Connections

HL7 Module supports creating MLLP connections that enable Integration Server to connect to the HL7 backend at run time when using the MLLP transport protocol. You create one or more connections at design time to use in integrations. The number of connections you create depends on your integration needs. You configure connections using Integration Server Administrator. You must have webMethods administrator privileges to access the administrative screens of HL7 Module. For instructions on configuring, viewing, editing, enabling, and disabling MLLP connections, see "MLLP Connections" on page 49.

For information about setting user privileges, see the *webMethods Integration Server Administrator's Guide* for your release.

For a list of tasks that you must do before you can create your connections, see "Before Configuring or Managing MLLP Connections" on page 49.

Connection Types

HL7 Module supports two types of MLLP connections:

- Permanent connections are configured as part of a connection pool and can be reused. For information about connection pools, see "Connection Pools" on page 21.
- Transient connections last during the execution of an HL7 Module service and are then terminated.

For information about configuring permanent and transient connections, see "Configuring MLLP Connections" on page 49.

Connection Pools

Integration Server includes a connection management service that dynamically manages connections and connection pools based on configuration settings that you specify for the connection. A connection pool is a collection of connections with the same set of attributes. Integration Server maintains connection pools in memory. Connection pools improve performance by enabling HL7 Module services to reuse open connections instead of opening new connections.

For details, see the *webMethods Integration Server Built-In Services Reference* for your release.

Runtime Behavior of Connection Pools

When you enable a connection, Integration Server initializes the connection pool, creating the number of connection instances you specified in the connection's **Minimum Pool Size** field. Whenever an HL7 Module service needs a connection, Integration Server provides a connection from the pool. If no connections are available in the pool, and the maximum pool size has not been reached, the server configures one or more new connections (according to the number specified in **Pool Increment Size**) and adds them to the connection pool. If the pool is full (as specified in **Maximum Pool Size**), the requesting service will wait for the Integration Server to obtain a connection, up to the length of time specified in the **Block Timeout** field, until a connection becomes available. Periodically, Integration Server inspects the pool and removes inactive connections that have exceeded the expiration period that you specified in **Expire Timeout**.

If the connection pool initialization fails (due to a network connection failure or some other type of exception), you can enable the system to retry the initialization any number of times, at specified intervals.

For information about configuring connections, see "MLLP Connections" on page 49.

2 Installing webMethods HL7 Module

Overview

This chapter explains how to install, upgrade, and uninstall webMethods HL7 Module 7.1 SP1. The instructions use the Software AG Installer and the Software AG Uninstaller wizards. For complete information about the wizards or other installation methods, or to install other webMethods products, see the *Installing webMethods Products On Premises* guide for your release.

Requirements

For a list of the HL7 messaging standards, operating systems, and webMethods products supported by HL7 Module 7.1 SP1, see *webMethods eStandards Modules System Requirements* .

HL7 Module supports the same databases as its host Integration Server. For a list of supported databases, see *System Requirements for Software AG Products*.

HL7 Module 7.1 SP1 has no hardware requirements beyond those of its host Integration Server.

Installing webMethods HL7 Module 7.1 SP1

Before You Begin

1. If you are going to install webMethods HL7 Module 7.1 SP1 on an already installed Integration Server, shut down the Integration Server.

2. Download Software AG Installer from the Empower Product Support website at http://empower.softwareag.com.

Install webMethods HL7 Module 7.1 SP1

To install webMethods HL7 Module

1. Start Software AG Installer.

2. Choose the webMethods release that includes the Integration Server on which to install the module. For example, if you want to install the module on Integration Server 7.1.2, choose the 7.1.x release.

3. Specify the installation directory, as follows:

- If you are installing webMethods HL7 Module 7.1 SP1 on an existing Integration Server, specify the webMethods installation directory that contains the host Integration Server.
- If you are installing both the host Integration Server and the module, specify the installation directory to use.

4. In the product selection list, select **eStandards > webMethods HL7 Module 7.1 SP1**.
5. Install any required webMethods components you have not installed, such as Integration Server, Trading Networks, and Designer. For a list of products and their versions required by webMethods HL7 Module, see *webMethods eStandards Modules System Requirements* .
6. To install documentation for webMethods HL7 Module, on the Documentation panel in Installer, select **eStandards Module Readmes and Documentation**.

   Alternatively, you can download the documentation at a later time from the Software AG Documentation website.
7. After Installer completes the installation, close Installer.
8. Start the Integration Server on which you installed webMethods HL7 Module 7.1 SP1.

Uninstalling webMethods HL7 Module 7.1 SP1

To uninstall webMethods HL7 Module

1. Shut down the Integration Server that hosts webMethods HL7 Module 7.1 SP1.
2. Start Software AG Uninstaller, selecting the webMethods installation directory that contains the host Integration Server. In the product selection list, select **eStandards > webMethods HL7 Module 7.1 SP1** and any other products and items you want to uninstall.
3. Restart the Integration Server from which you uninstalled webMethods HL7 Module 7.1.
4. Uninstaller does not delete files that you have created or configuration files associated with webMethods HL7 Module 7.1 SP1, nor does it delete the directory structure that contains those files. If you do not want to save those files, go to the *Integration Server_directory* \packages directory and delete the WmHL7 and WmHL7DocTypes package directories.

3 Steps to Set Up webMethods HL7 Module

Overview

This chapter lists the tasks you perform to set up webMethods HL7 Module to process and exchange HL7 version 2.x messages with other business partners.

Step 1: Configure the HL7 Module Database

Configure the HL7 Module database and populate the code tables as needed. You must configure the HL7 Module database so that you can use HL7 Module to:

- Manage the standard and partner specific code tables. For information about managing code tables, see "Managing HL7 Code Tables" on page 95.
- Use the code tables for validating HL7 version 2.x messages.

For the steps to configure the HL7 Module database, see "Configuring the HL7 Module Database" on page 32.

Step 2: Create Trading Partner Profiles and Trading Partner Agreements

Define profiles for your enterprise and the trading partners with whom you want to exchange HL7 version 2.x messages. A trading partner is any person or organization with whom you want to conduct business electronically. In HL7 Module, a trading partner is defined by several criteria that you specify in a trading partner profile, including company name and identifying information, contact information, and delivery methods. In addition to specifying trading partner profiles for all of your trading partners, you must specify a profile for your own organization. For more information, see "Defining Trading Networks Profiles" on page 68.

At start up, HL7 Module creates a default trading partner agreement (TPA) that contains HL7-specific information. For information about defining TPAs, see "Defining HL7 Trading Partner Agreements" on page 71.

Step 3: Configure the Trading Networks Database

HL7 Module registers with Trading Networks the following HL7-specific items that identify how Trading Networks will process HL7 version 2.x messages:

- TN document types for HL7 version 2.x messages
- Custom TN Attributes for HL7 version 2.x messages
- Processing rules for HL7 version 2.x messages

For more information about these Trading Networks items for transaction processing, see "Defining Trading Networks Information" on page 67.

Step 4: Prepare to Send and Receive Messages

The following lists the tasks you perform to send and receive messages using HL7 Module, and provides references to the chapters in this book where you can find more information about each task:

- **Generate IS document types.** To parse or create HL7 version 2.x messages using HL7 Module, you need to generate the IS document type for the supported HL7 message versions. HL7 Module provides default XML schemas (supplied by the HL7 standards organization) that you use to create the default message scheme for the module. For information about using the default message scheme to generate IS document types, see "Generating Message Schemes" on page 33. In some cases you may need to create and use a custom message scheme that meets your specific business requirements. For information about using a custom message scheme to generate IS document types, see "Creating a Custom Message Scheme" on page 34.
- **Parse HL7 version 2.x messages.** The WmHL7 package of HL7 Module contains built-in services that you use to convert HL7 version 2.x messages to IS document format and to create HL7 version 2.x messages from IS documents. For more information, see "Processing HL7 Messages" on page 81.
- **Validate HL7 version 2.x messages.** When you parse HL7 version 2.x messages, you use the wm.ip.hl7.service:validate service to validate the HL7 messages.

  For information about how to use the validate service, see "Validating HL7 Version 2.x Messages" on page 87 and "Handling Validation Errors" on page 108. For details about the service parameters, see " webMethods HL7 Module Services" on page 139.
- **Create clients and configure communication protocols.** To exchange messages using HL7 Module, you create clients that use the supported communication protocols. For a list of the supported transports and the configuration steps for each transport, see "Configuring webMethods HL7 Module " on page 31.
- **Generate accept acknowledgement.** HL7 Module generates accept acknowledgments for incoming HL7 version 2.x messages when the sender of the message has requested an accept acknowledgment. For details about generating accept acknowledgments, see "Sending an Accept Acknowledgment" on page 61.

Step 5: View Document Transactions

When you use HL7 Module with Trading Networks, you can view document transaction information about all HL7 version 2.x messages that have been sent or received through Trading Networks. You use My webMethods to query the Trading Networks database run-time message transactions that HL7 Module performs. For the type of transaction information you can view about HL7 message transactions, see "Viewing Information about HL7 Messages" on page 91.

4 Configuring webMethods HL7 Module

Overview

Before you can process HL7 messages using webMethods HL7 Module, you must first configure HL7 Module. This chapter describes the configuration tasks you need to complete before you can use HL7 Module.

Configuring the HL7 Module Database

You must configure the HL7 Module database to store, manage, and use code tables for validation of HL7 version 2.x messages.

To configure the HL7 Module database

1. In Integration Server Administrator, connect Integration Server to the HL7 Module database component, as follows:

   a. Create a new Pool Alias Definition for one of the supported databases. HL7 Module can use any of the supported databases for its database. For a list of all supported RDBMSs, see *System Requirements for Software AG Products*.

   b. Associate the HL7 functional alias with the Pool Alias Definition.

   c. Restart the HL7 functional alias for the changes to take effect.

   For the steps to create a new Pool Alias Definition and associate it with the HL7 functional alias, see the *Installing webMethods Products On Premises* guide for your release.

2. Create the database tables required by the HL7 Module database, as follows:

   a. In Designer, in the Package Navigator view, expand the WmHL7 package and locate the wm.ip.hl7.codetables:runScripts service.

   b. In Designer, select **Run > Run As > Run Service**.

   c. In the **Enter Input for 'runScripts'** dialog box, specify the following input values:

- In the **database** input field, select the database that is associated with the HL7 functional alias.
- In the **mode** input field, select `create` to create the database tables required by the HL7 Module database.

  For more information about the service parameters, see the wm.ip.hl7.codetables:runScripts service.

d. Click **OK**.

   Designer runs the service and displays the results in the Service Result view.

3. Populate the code table and code table values in the HL7 Module database, as follows:

   a. In Designer, in the Package Navigator view, expand the WmHL7 package and locate the `wm.ip.hl7.codetables:loadData` service.

   b. In Designer, select **Run > Run As > Run Service**.

   Designer runs the service and displays the results in the Service Result view.

   Note:
   The wm.ip.hl7.codetables:loadData service loads the standard code table and code table values from two files (codeTables.tsv and codeTableValues.tsv) located in the *Integration Server_directory* \packages\WmHL7\data\codetables directory.

Generating Message Schemes

To parse or create HL7 messages using HL7 Module, you need to generate the IS document type for the supported HL7 message versions from XML schemas. To generate the IS document types for the HL7 version 2 messages, you must create a message scheme.

You can use either Developer or Designer to create the default HL7 Module message scheme. The following procedure describes how to do that using Designer. The steps in Developer are similar.

To create the default message scheme

1. Start Integration Server and Designer, if they are not already running.

2. Make sure Designer is using the Service Development perspective. If not, switch to it by selecting **Window > Open Perspective > Service Development**.

3. To connect to Integration Server:

   a. In Package Navigator view, select the Integration Server to which you want to connect.

   b. Right click, and select **Connect to server**.

For more information about how to work with Integration Server in Designer, see the *webMethods Service Development Help*.

4. In Package Navigator view, expand the WmHL7 package and locate the wm.ip.hl7.utils:generateMessageScheme service.

5. In Designer, select **Run > Run As > Run Service**.

6. In the **Enter Input for 'generate Message Scheme'** dialog box, specify:

Parameter	Value
message Version	The message version for which you want to generate IS document types.
message Types	List of valid HL7 message types that belong to the specified *message Version* for which the IS document types will be generated. **Note:** If you do not specify a value for the *message Types* parameter, HL7 Module generates IS document types for all the HL7 message types valid for the specified *message Version*.
scheme ID	`wm.ip.hl7.docType` or do not specify a value. **Note:** If you do not specify a message scheme ID, HL7 Module uses the default message definitions provided with the module.

For details about the utility service parameters, see the wm.ip.hl7.utils:generateMessageScheme service.

7. Click **OK**.

   Designer runs the service and displays the results in the Service Result view.

Creating a Custom Message Scheme

To create a custom message scheme, follow these steps:

1. Customize the default XML schemes provided with HL7 Module as described in "Customizing the XML Schemes" on page 34.

2. Generate the custom message scheme as described in "Generating the Custom Message Scheme" on page 36.

Customizing the XML Schemes

Important:

Before you can customize a Z segment, you must add a definition for the Z segment to the corresponding message scheme and make changes where applicable.

To customize the XML schemes provided with HL7 Module

1. Create a directory for the XML schema that you want to customize, analogical to the default XML schema directory.

   The default XML schema is located in the *Integration Server_directory* \packages\WmHL7\data\xsd\wm.ip.hl7.docType\ v<*version_number*> directory.

   The customized XML schema should be located in the *Integration Server_directory* \packages\WmHL7\data\xsd\custom_message_scheme_id\v<*version_number*>directory.

   where *custom_message_scheme_id* is the scheme ID for your custom message scheme and <*version_number*> is the HL7 message version for which you want to use this custom message scheme.

   For example, *Integration Server_directory* \packages\WmHL7\data\xsd\com.example.custom.scheme\v231

2. Copy the XML schema you want to customize from the default XML message scheme into the customized XML schema directory. The new directory should:

   - Contain the complete definitions for generating the message structure (such as segments, fields, data types, and messages).
   - Contain a messages.properties file that lists all messages for the particular message version and message scheme.

     Each key in the messages.properties file refers to the message schema file, and the values denote the message types that the schema can represent.

     For example:

     ADT_A01=ADT_A01,ADT_A04,ADT_A08

     where ADT_A01 denotes that the message schema definition can represent message structures ADT_01, ADT_A04, and ADT_A08.

     A messages.properties file exists for each message version and describes the schemas and message types supported for that version. The file should be located in the same directory as the schema definitions. For details about the content and structure of a message property file, see the messages.properties file for one of the versions in the default XML message scheme.

3. Open the XML schema in an editor and include your custom changes.

   Important:
   Do not customize the schema for the MSH and MSA segments in the HL7 message. These segments are not open for customizing.

An HL7 message must have only one MSH segment and it should be the first segment in the message.

Important:
Do not use Java reserved key words, such as INT, TRY, and NEW, as segment names in the custom XML schema.

4. Edit each of the schema definition files, and update only the xmlns and targetNamespace attributes for your custom message scheme as follows:

   custom_message_scheme_id.v<*version_number*>

   For example:

   Replace

   ```
   xmlns="urn:hl7-org:v231xml"
   ```

   with

   ```
   xmlns="com.example.custom.scheme.v231"
   ```

   and

   ```
   targetNamespace="urn:hl7-org:v231xml"
   ```

   with

   ```
   targetNamespace="com.example.custom.scheme.v231"
   ```

5. To ensure that the folder contains all the required schema files and the correct structures, you can optionally validate the structure of the customized XML schemas using any third party parser.

Generating the Custom Message Scheme

You can use either Developer or Designer to create a custom HL7 Module message scheme. The following procedure describes how to do that using Designer. The steps in Developer are similar.

To create the custom message scheme

1. Start Integration Server and Designer, if they are not already running.

2. Make sure Designer is using the Service Development perspective. If not, switch to it by selecting **Window > Open Perspective > Service Development**.

3. To connect to Integration Server:

   a. In Package Navigator view, select the Integration Server to which you want to connect.

b. Right click, and select **Connect to server**.

   For more information about how to work with Integration Server in Designer, see the *webMethods Service Development Help*.

4. In Package Navigator view, expand the WmHL7 package and locate the wm.ip.hl7.utils:generateMessageScheme service.

5. In Designer, select **Run > Run As > Run Service**.

6. In the **Enter Input for 'generateMessageScheme'** dialog box, specify:

Parameter	Value
messageVersion	The message version for which you want to generate IS document types.
messageTypes	List of valid HL7 message types that belong to the specified *messageVersion* for which the IS document types will be generated. **Note:** If you do not specify a value for the *messageTypes*parameter, HL7 Module generates IS document types for all the HL7 message types listed in the messages.properties file, for the selected version and message scheme ID.
schemeID	Specify the message scheme ID for the custom message scheme. For example, `com.example.custom.scheme`

For details about the utility service parameters, see the wm.ip.hl7.utils:generateMessageScheme service.

7. Click **OK**.

   Designer runs the service and displays the results in the Service Result view.

Custom Message Schemes for HL7 Message Types with anyHL7Segment or anyZSegment

A number of HL7 message types contain references to *anyHL7Segment* and *anyZSegment*, the two placeholders defined in the hl7.org-supplied XML Schema to serve as indicators that you must provide a valid HL7 segment or Z segment in their place. The XML Schema for the HL7 message types with such references is included in HL7 Module. However, by default HL7 Module does not generate IS document types for these message types, because they are not enabled in the message.properties file. The messages.properties file is included with the XML Schema for every supported HL7 version under the *Integration Server_directory* \packages\WmHL7\data\xsd\wm.ip.hl7.docTypes\v<*version_number*\> directory and determines the default generation of HL7 messages. HL7 Module does not generate IS document types for HL7 message types not enabled in the messages.properties file.

To generate IS document types for HL7 message types with references to *anyHL7Segment* and *anyZSegment*, you must first customize the XML Schema definition for these message types by replacing the reference to *anyHL7Segment* or *anyZSegment* with a valid segment identifier for the respective HL7 version. After you customize the XML Schema of any of the message type definitions, you can uncomment the HL7 message type name or include it in the messages.properties file. The wm.ip.hl7.utils:generateMessageScheme service will then be able to generate the IS document types.

Note:
If you attempt to generate IS document types for HL7 message types with references to *anyHL7Segment* and *anyZSegment* without customizing the XML Schema, HL7 Module returns an error.

Following is an example for HL7 message version 2.3.1. The message.properties file includes the following definition:

```
## ===================================================================
## v2.3.1 Message types containing anyHL7Segment or anyZSegment,
## requires customization before usage
##
## MFN_M01=MFN_M01
## MFN_M03=MFN_M03
## MFR_M01=MFR_M01
```

This definition means that HL7 Module will not generate an IS document type for message types MFN_M01, MFN_M03, and MFR_M01 because they contain the placeholders *anyHL7Segment* or *anyZSegment*. To generate IS document types for these message types, you must edit the corresponding XML Schema definition for each message type. For example, for the MFN_M01 message type, you must edit the MFN_M01.xsd file and replace the occurrences of *anyHL7Segment* and/or *anyZSegment* with a valid segment identifier, defined in the segments.xsd file.

For the steps in customizing an XML Schema and generating a custom message scheme, see "Creating a Custom Message Scheme" on page 34.

The following table lists the HL7 Module message types that contain direct references to *anyHL7Segment* and *anyZSegment*:

HL7 Version	anyHL7Segment	anyZSegment
2.1	None.	None.
2.2	None.	MFN_M01, MFN_M02, MFN_M03, MFR_M01, MFR_M02, MFR_M03
2.3	MFN_M03	None.
2.3.1	MFN_M03	MFN_M01, MFR_M01
2.4	MFN_M03	MFN_M01, MFR_M01, QBP_Q13, QBP_Q15, QVR_Q17, RSP_K11

HL7 Version	anyHL7Segment	anyZSegment
2.5	MFN_M01, MFN_M03, MFR_M01, PGL_PC6, PPG_PCG, PPP_PCB, PPT_PCL, PPV_PCA, PRR_PC5, PTR_PCF, QBP_Q15, QVR_Q17	None.
2.5.1	MFN_M01, MFN_M03, MFR_M01, PGL_PC6, PPG_PCG, PPP_PCB, PPT_PCL, PPV_PCA, PRR_PC5, PTR_PCF, QVR_Q17	None.
2.6	MFN_M01, MFN_M03, MFR_M01, PGL_PC6, PPG_PCG, PPP_PCB, PPT_PCL, PPV_PCA, PRR_PC5, PTR_PCF, QVR_Q17	None.
2.7	MFN_Znn, PGL_PC6, PPG_PCG, PPP_PCB, PPR_PC1, PPT_PCL, PPV_PCA, PRR_PC5, PTR_PCF, QBP_Q11, QBP_Q13, QBP_Q15, QVR_Q17, RSP_K11, RTB_Knn	None.

Associating a Trading Partner Agreement with a Message Scheme ID

To use a specific HL7 message scheme to exchange messages between two trading partners, you must associate the message scheme ID with the trading partner agreement that you use for the message exchange.

To associate a trading partner agreement with a message scheme ID

1. Create a custom message scheme as described in "Creating a Custom Message Scheme" on page 34.

2. Create a trading partner agreement as described in "Defining HL7 Trading Partner Agreements" on page 71.

3. In the "wm.ip.hl7.tn.tpa.rec:HL7TPA" on page 153 document, specify the message scheme ID for your custom message scheme in the *hl7MessageScheme/schemeID* parameter. For example, `com.example.custom.scheme`.

4. In Designer or Developer, run the wm.ip.hl7.tn.tpa:initTPA service to populate the TPA with the values for the specified IS document type.

Viewing the IS Document Types for HL7 Version 2.x Messages

You view the generated IS document types for HL7 version 2.x messages using Designer.

To view the IS document types

1. In Designer, in Package Navigator view, expand the WmHL7DocTypes package and locate the wm.ip.hl7.docType folder.

   The generated IS document types for the message versions are located under the wm.ip.hl7.docType.*version*.reference namespace in that folder. For example, the generated IS document types for message version 2.3.1 are located under the wm.ip.hl7.docType.v231.reference folder.

2. Expand the reference folder for the respective message version and click on the IS document type that you want to view. The document type opens in the Document Type Editor window.

3. To view the wrapper IS document that corresponds to the generated IS document type that you want to view:

   a. Navigate to wm.ip.hl7.docType.*version*folder, where *version* is the message version for the generated IS document type.

   b. Click on the namespace with the HL7_ prefix attached to the message type corresponding to the generated IS document type. For example, the wrapper IS document for message type ADT_A01 is located under the wm.ip.hl7.docType.v231 folder, and designated wm.ip.hl7.docType.v231:HL7_ADT_A01.

      The wrapper IS document opens in the Document Type Editor window.

For information about the generated IS document types, see "WmHL7DocTypes Package" on page 162.

Deleting the Message Schemes and IS Document Types for HL7 Version 2.x Messages

HL7 Module provides the wm.ip.hl7.utils:deleteMessageScheme service that you use to delete generated message schemes and IS document types for HL7 message versions that you no longer use.

Note:
The deleteMessageScheme service does not check for dependencies or references before deleting the generated IS document types. Ensure that there are no dependencies or references in the IS documents before you run the service. For information about how to configure dependency checking for elements, see the *webmethods Developer User's Guide* or the *webMethods Service Development Help* for your release.

To delete the generated message schemes and IS document types

1. In Designer, in Package Navigator view, expand the WmHL7 package and locate the wm.ip.hl7.utils:deleteMessageScheme service.

2. In Designer, select **Run > Run As > Run Service**.

3. In the **Enter Input for 'deleteMessageScheme'** dialog box, specify the message version for which you want to delete the generated message schemes and IS document types.

4. Click **OK**.

   Designer runs the service and displays the results in the Service Result view.

For information about the service parameters, see the wm.ip.hl7.utils:deleteMessageScheme service.

Generating HL7 Version 3 Document Types

You generate HL7 version 3 document types using the Integration Server document types services in Designer or Developer.

To generate HL7 version 3 document types

1. Start Integration Server and Designer, if they are not already running.

2. Make sure Designer is using the Service Development perspective. If not, switch to it by selecting **Window > Open Perspective > Service Development**.

3. To create a package for the HL7 version 3 document types in Designer, select **File > New > Package**. Enter a name for the new package and click **Finish**.

4. Right click the new package and select **New > Folder**. Enter a name for the new folder and click **Finish**.

5. Right click the new folder, and select **New > Document Type**.

6. In the **Element name** field, enter a name for the new HL7 version 3 document type and click **Next**.

7. Under **Select a Source**, select **XML schema**.

8. Under **Import from Source**, browse to the *Integration Server_directory* \packages\WmHL7\data\v3\2010NE\domains\ directory and select the required schema.

9. Click **Open**.

10. Complete the remaining steps as described in the *webMethods Service Development Help*.

    Integration Server generates the HL7 version 3 document types and IS schema and saves them in the new package you created.

You can view the generated HL7 version 3 document types in the Package Navigator view in Designer.

Creating HL7 Clients

To send documents to Integration Server to be processed through Trading Networks, you can use a client program suitable for the communication protocol you use. HL7 Module supports the following communication protocols to send and receive HL7 version 2.x messages:

- HTTP and HTTPS
- FTP
- E-mail
- MLLP

For information about how to configure the supported transport protocols, see "Configuring Communication Protocols" on page 42.

The client sends the HL7 version 2.x messages to the HL7 Module wm.ip.hl7.tn.service:receive service. When your client sends the HL7 message to the HL7 Module receive service using HTTP or HTTPS, the client must associate the document with the `text/plain` content type that HL7 Module recognizes. The HL7 Module receive service receives and passes the HL7 version 2.x message to the content handler for the `text/plain` content type. This content handler is part of the Integration Server WmFlatFile package. The content handler performs initial processing on the document, which includes creating the pipeline with the variable *ffdata*that contains the HL7 message. After the initial processing by the content handler, the HL7 Module receive service processes the stream to extract the HL7 message.

Configuring Communication Protocols

HTTP and HTTPS

Sending HL7 Messages Using Integration Server

You can create a client that uses the HTTP or HTTPS transport to send an HL7 version 2.x message to Integration Server. When using HTTP, the client must include the following logic:

- Submit a POST request to Integration Server.
- Address the request to the URL of the service that is to process the HL7 message (for example, `http://localhost:5555/invoke/wm.ip.hl7.tn.service/receive`)
- In the HTTP request header, set the value of the content type field to `text/plain`.

- Put the HL7 document to process in the body of the message. The document must be the only text that appears in the body of the request.

Because most browsers do not allow you to modify the content type header field, they are not suitable clients for this type of submission. Clients that you might use to submit an HL7 message in this manner include PERL scripts (which allow you to build and issue HTTP requests) and the webMethods pub.client:http service. The following table describes the values that you provide for the pub.client:http service to POST an HL7 version 2.x message to Integration Server:

Parameter	Description
url	A string that contains the URL of the service that you want to invoke to process the HL7 message. For example, `http://localhost:5555/invoke/wm.ip.hl7.tn.service/receive`
method	Specify `post`.
loadAs	A string that contains the data type of the input data source. Specify either `bytes` or `stream`. ■ `bytes` if the document data source is a byte[]. ■ `stream` if the document data source is an InputStream.
data/string	A string that contains the HL7 message that you want to post.
headers	An IData object that contains the following: ■ *Name*Specify`Content-type` ■ *Value* The content type for the document, for example, `text/plain`.

The client can also set other optional HTTP variables, such as authorization information, that are required by your application. For a complete description of the pub.client.http service, see the *webMethods Integration Server Built-In Services Reference* for your release.

Sending HL7 Version 2.x Messages Using Trading Networks

To send HL7 version 2.x messages using Trading Networks, when you create trading partner profiles, you define the HTTP or HTTPS delivery method as part of the partner profile. For more information about creating profiles, see "Defining Trading Networks Profiles" on page 68 and the *webMethods Trading Networks Administrator's Guide* for your release.

To define HTTP or HTTPS as the delivery method in trading partner profiles

1. In My webMethods, go to **Applications > Administration > Integration > B2B > Partner Profiles > Create Enterprise Profile**.

2. On the **Delivery Settings** tab,click **Add Delivery Method**.

3. In the **Delivery Method** field, select **HTTP** or **HTTPS**. Set the following values:

Field	Description
Host	The partner's host name or IP address. For example, `AcmeMedical.com`.
Port	The port number on which the partner listens for incoming requests, for example, `5555`.
Location	`/invoke/wm.ip.hl7.tn.service/receive`
User Name	The user name that your Trading Networks system is to supply when connecting to the partner's system. This field is for a partner profile only.
Password	The password that your Trading Networks system is to supply (along with **User Name**). This field is for a partner profile only.

   For information about other fields on the Add Delivery Methods screen, see the *webMethods Trading Networks Administrator's Guide* for your release.

4. Select the **Use as preferred protocol** checkbox if you want to use HTTP or HTTPS as your preferred method to deliver HL7 messages to your partner.

5. Click **OK**.

Receiving HL7 Version 2.x Messages via HTTP or HTTPS

To receive HL7 version 2.x messages, configure an HTTP or HTTPS port on Integration Server. For information about configuring an HTTP port, see the *webMethods Integration Server Administrator's Guide* for your release.

FTP

Sending HL7 Version 2.x Messages Using Integration Server

You can create a client that uses the FTP transport to send an HL7 version 2.x message to Integration Server. The following table describes the values that you provide for the pub.client:ftp service to send an HL7 message to Integration Server:

Parameter	Data Type/Description
serverhost	A String that contains the name of the machine running Integration Server.
serverport	A String that contains the port on which Integration Server listens for FTP requests.

Parameter	Data Type/Description
username	A String that contains a valid user name of an Integration Server user account.
password	A String that contains the password for the user name.
command	Specify `put`
dirpath	Specify `/ns/wm/ip/hl7/tn/service/receive`
localfile	A String that contains the name of the source file containing the HL7 message.
remotefile	A String that contains the name to assign the file on Integration Server and the content type. Use the following format: *filename;content type:content sub-type* For example, `ADR_A19_251_091217-184854;text:plain`
secure	A Document that indicates whether the FTP session is with a secure FTP server. The variable *auth* specifies the kind of authentication mechanism to use (`SSL`, `TLS`, or `TLS-P`), and the variable *securedata* specifies whether the client is sending PROT C (Data Channel Protection Level Clear) or PROT P (Data Channel Protection Level Private).

For a complete description of the pub.client.ftp service, see the *webMethods Integration Server Built-In Services Reference* for your release.

Sending HL7 Version 2.x Messages Using Trading Networks

To send HL7 version 2.x messages using Trading Networks, when you create trading partner profiles, you define the FTP delivery method as part of the partner profile. For more information about creating profiles, see “Defining Trading Networks Profiles” on page 68 and the *webMethods Trading Networks Administrator’s Guide* for your release.

To define FTP as the delivery method in trading partner profiles

1. In My webMethods, go to **Applications > Administration > Integration > B2B > Partner Profiles > Create Enterprise Profile**.
2. On the **Delivery Settings** tab,click **Add Delivery Method**.
3. In the **Delivery Method** field, select **FTP**. Set the following values:

Field	Description
Host	The partner's host name or IP address. For example, `AcmeMedical.com`.

Field	Description
Port	The port number on which the partner listens for incoming requests, for example, `5555`.
Location	`\ns\wm\tn\receive`
User Name	The user name that your Trading Networks system is to supply when connecting to the partner's system. This field is for a partner profile only.
Password	The password that your Trading Networks system is to supply (along with **User Name**). This field is for a partner profile only.

For information about other fields on the Add Delivery Methods screen, see the *webMethods Trading Networks Administrator's Guide* for your release.

4. Select the **Use as preferred protocol** checkbox if you want to use FTP as your preferred method to deliver HL7 messages to your partner.

5. Click **OK**.

Sending HL7 Version 2.x Messages Using a Third Party Application

To send an HL7 version 2.x message via FTP using a third party application

1. Initiate an FTP connection to the Integration Server's FTP listening port (for example, `8021`).

2. Change the directory that represents the service you want to invoke to `/ns/wm/ip/hl7/tn/service/receive` using the `cd` command.

3. Send the HL7 message to this directory using the following `put` command:

```
put   localFileName filename;content type:content sub-type
```

   For example, `put ADR_A19_251_091217-184854 ADR_A19_251_091217-184854;text/plain:`

Receiving HL7 Version 2.x Messages via FTP

To receive HL7 version 2.x messages, configure an FTP port on Integration Server. By default the FTP port is assigned to port 8021, but you can configure a different assignment. For information about configuring an FTP port, see the *webMethods Integration Server Administrator's Guide* for your release.

E-mail

Sending HL7 Version 2.x Messages Using Integration Server

To send an HL7 version 2.x message via E-mail, you must configure Integration Server to use your local E-mail server. Ensure that an e-mail account and listening services are configured to route the incoming e-mails to the HL7 services. You can specify this information in Integration Server Administrator under **Security > Ports > Add Port > webMethods/E-mail**. For detailed instructions, see the *webMethods Integration Server Administrator's Guide* for your release.

To send an HL7 version 2.x message via E-mail using Integration Server

1. In Integration Server Administrator, select **Settings > Extended**.
2. Click **Edit Extended Settings**.
3. Specify the following values:
    a. Set the value of the watt.server.smtpServer property to your local E-mail server.
    b. Set the value of the watt.server.smtp.serverPort property to the port of the E-mail server. The default value is 25.

   For more information about setting the properties, see the *webMethods Integration Server Administrator's Guide* for your release.
4. Click **Save Changes**.
5. Restart Integration Server for the changes to take effect.

Sending HL7 Version 2.x Messages Using Trading Networks

To send HL7 version 2.x messages using Trading Networks, when you create trading partner profiles, you define the E-mail delivery method as part of the partner profile. For more information about creating profiles, see "Defining Trading Networks Profiles" on page 68 and the *webMethods Trading Networks Administrator's Guide* for your release.

To define E-mail as the delivery method in trading partner profiles

1. In My webMethods, go to **Applications > Administration > Integration > B2B > Partner Profiles > Create Enterprise Profile**.
2. On the **Delivery Settings** tab,click **Add Delivery Method**.
3. In the **Delivery Method** field, select **Primary E-mail**. Set the following values in the **E-mail** field:
    a. When you define your enterprise profile (**My Enterprise**), you must specify your e-mail address.

   b. When you define your partner's profile, enter the e-mail address provided by your trading partner to receive HL7 messages that you send to them.

4. Select the **Use as preferred protocol** checkbox if you want to use E-mail as your preferred method to deliver HL7 messages to your partner.

5. Click **OK**.

Sending HL7 Version 2.x Messages Using a Third Party Application

To send HL7 version 2.x messages using a third party application, you must send the HL7 message as an attachment. The name of the attached file must contain `hl7data`. HL7 Module processes only the first attachment that satisfies the above criteria and ignores all other attachments and the body of the e-mail.

Receiving HL7 Version 2.x Messages via E-mail

To receive an HL7 Version 2.x message via POP or IMAP

1. Add an e-mail account with any e-mail service provider that supports either POP3 or IMAP protocol. For instructions on adding an e-mail port, see the *webMethods Integration Server Administrator's Guide* for your release.

2. In Integration Server Administrator:

   a. Set up an e-mail listener that points to the "wm.ip.hl7.tn.service:receive" on page 149 service. Specify `Global Service`as the name of the listener.

   b. Select **Ports > Add Port > webMethods/E-mail**. For more information about setting the POP or IMAP ports, see the *webMethods Integration Server Administrator's Guide* for your release.

   c. Set the **Invoke service for each part of multipart message** to **No**.

   d. Set **Include e-mail headers when passing message to content handler** to **No**.

   e. Click **Save Changes**.

   f. From the Ports page, select **Access Mode > Edit.**

   g. Click **Set Access Mode to Allow By Default**.

MLLP

MLLP Connections

Before Configuring or Managing MLLP Connections

To prepare to configure or manage an MLLP connection

1. Install webMethods Integration Server and HL7 Module on the same machine. For more information about the installation steps, see "Installing webMethods HL7 Module 7.1 SP1" on page 24.

2. Make sure you have webMethods administrator privileges so that you can access the HL7 Module's administrative screens. For information about setting user privileges, see the *webMethods Integration Server Administrator's Guide* for your release.

3. Start Integration Server and Integration Server Administrator, if they are not already running.

4. Using Integration Server Administrator, make sure the WmHL7 package is enabled. For more information about working with packages, see the *webMethods Integration Server Administrator's Guide* for your release.

5. Using Software AG Designer, create a user-defined package to contain connections, if you have not already done so. For more information about creating packages, see the *webMethods Service Development Help*.

Configuring MLLP Connections

When you configure MLLP connections, you specify information that Integration Server uses to connect to an HL7 system. You configure MLLP connections using Integration Server Administrator.

To configure a connection

1. In Integration Server Administrator, go to **Solutions > HL7 Module** and click **MLLP Connections**.

2. On the MLLP Connections screen, click **Configure New MLLP Connection**.

3. On the Connection Types screen, click **MLLP Server Connection** to display the Configure Connection Type screen.

4. In the **MLLP Connections** section, use the following fields:

Field	Description/Action
Package	The package in which to create the connection.You must create the package using Developer or Designer before you can specify it using this parameter. For general information about creating packages, see the *webmethods Developer User's Guide* or the *webMethods Service Development Help* for your release. **Note:** Configure the connection in a user-defined package rather than in the WmHL7 package.
Folder Name	The folder in which to create the connection.
Connection Name	The name you want to give the connection. Connection names cannot have spaces or use special characters reserved by Integration Server, Developer, or Designer. For information about the use of special characters in package, folder, and element names, see the *webmethods Developer User's Guide* or the *webMethods Service Development Help* for your release.

5. In the **Connection Properties** section, use the following fields:

Field	Description
Connection Type	The type of the MLLP connection that you want to configure. Select one of the following connection types: ■ **Transient** The connection is terminated after the execution of the HL7 Module send service. This is the default. **Note:** When you configure a transient MLLP connection, you cannot specify values in the **Connection Management Properties** section. ■ **Permanent** The connection is added to the connection pool and can be reused.
Host Name	The name of the MLLP server that hosts the HL7 backend system.
Host Port	The port number that the connection will use to connect to the MLLP server.
Connection Timeout	The time in milliseconds for the client to wait before cancelling a connection attempt to the target server. If you do not specify a value in this field, the client uses the default value: **30000** (msec).

Field	Description
Ack Timeout	The time in milliseconds the client waits for an acknowledgement response after it has sent the HL7 2.x message, before closing the communication channel. If no response is received within the specified timeout, HL7 Module logs a warning in the Integration Server logs. The default is **-1** ■ When you enter a negative value or do not enter an **ackTimeout** value, the client does not wait for the ACK response. ■ When you enter a `0` value, the client waits for the ACK response infinitely. ■ When you enter a positive value, the client waits for the ACK response for the specified number of milliseconds. For more information about sending an accept acknowledgment, see "Sending an Accept Acknowledgment" on page 61.

6. In the **Connection Management Properties** section, use the following fields to configure a permanent MLLP connection:

Field	Description
Minimum Pool Size	The number of connections to create when the connection is enabled. The module will keep open the number of connections you configure here regardless of whether these connections become idle. The default is **1** **Important:** This field cannot take a zero value. When you configure a permanent connection, you must specify a value of 1 or higher. In addition, the value should be less than or equal to the value specified for the **Maximum Pool Size**.
Maximum Pool Size	The maximum number of connections that can exist at one time in the connection pool. The default is **10**
Pool Increment Size	The number of connections by which the pool will be incremented if connections are needed, up to the maximum pool size. The default is **1**
Block Timeout (msec)	The number of milliseconds that the Integration Server will wait to obtain a connection with the database before it times out and returns an error. For example, you have a pool with **Maximum Pool Size** of 20. If you receive 30 simultaneous requests for a connection, 10 requests will be waiting for a

Field	Description
	connection from the pool. If you set the **Block Timeout** to 5000, the 10 requests will wait for a connection for 5 seconds before they time out and return an error. If the services using the connections require 10 seconds to complete and return connections to the pool, the pending requests will fail and return an error message stating that no connections are available. If you set the **Block Timeout** value too high, you may encounter problems during error conditions. If a request contains errors that delay the response, other requests will not be sent. This setting should be tuned in conjunction with the **Maximum Pool Size** to accommodate such bursts in processing. The default is **1000**
Expire Timeout (msec)	The number of milliseconds that an inactive connection can remain in the pool before it is closed and removed from the pool. The connection pool will remove inactive connections until the number of connections in the pool is equal to the **Minimum Pool Size**. The inactivity timer for a connection is reset when the connection is used by the adapter. If you set the **Expire Timeout** value too high, you may have a number of unused inactive connections in the pool. This consumes local memory and a connection on your backend resource. This could have an adverse effect if your resource has a limited number of connections. If you set the **Expire Timeout** value too low, performance could degrade because of the increased activity of creating and closing connections. This setting should be tuned in conjunction with the **Minimum Pool Size** to avoid excessive opening/closing of connections during normal processing. The default is **1000**
Startup Retry Count	The number of times that the system should attempt to initialize the connection pool at startup if the initial attempt fails. The default is **0**
Startup Backoff Timeout (sec)	The number of seconds that the system should wait between attempts to initialize the connection pool. The default is **10**

7. Click **Save Connection**.

   The connection you created appears on the adapter's Connections screen and in Developer or Designer.

You can enable a connection only if the parameters for the connection are valid.

Enabling MLLP Connections

An MLLP connection must be enabled before an HL7 Module service can use the connection at run time. You enable MLLP connections using Integration Server Administrator.

Note:
When you reload a package that contains enabled connections, the connections will automatically be enabled when the package reloads. If the package contains connections that are disabled, they will remain disabled when the package reloads.

As you create user-defined packages in which to store connections, use the package management functionality provided in Developer or Designer and set the user-defined packages to have a dependency on the WmHL7 package. That way, when the WmHL7 package loads or reloads, the user-defined packages load automatically.

To enable a connection

1. In the Solutions menu in Integration Server Administrator, click **HL7 Module**.

2. On the Connections screen, click **No** in the **Enabled** column for the connection you want to enable.

   Integration Server Administrator enables the adapter connection and displays **Yes** in the **Enabled** column.

Disabling MLLP Connections

MLLP connections must be disabled before you can edit or delete them. You disable MLLP connections using Integration Server Administrator.

To disable a connection

1. In the Solutions menu in Integration Server Administrator, click **HL7 Module**.

2. On the Connections screen, click **Yes** in the **Enabled** column for the connection you want to disable.

   Integration Server Administrator disables the adapter connection and displays **No** in the **Enabled** column.

Viewing MLLP Connections

You can view MLLP connections and each connection's parameters from Integration Server Administrator or Designer.

To view the parameters for a connection using Integration Server Administrator

1. In the Solutions menu in Integration Server Administrator, click **HL7 Module**.
2. On the Connections screen, click the **View** icon for the connection you want to see.

   The View Connection screen displays the parameters for the connection.
3. Click **Return to MLLP Connections** to return to the main connections screen.

For information about viewing MLLP connection details in Designer, see the *webMethods Service Development Help*.

Editing MLLP Connections

You can update a connection's parameters using Integration Server Administrator.

To edit a connection

1. In the Solutions menu in Integration Server Administrator, click **HL7 Module**.
2. Make sure the connection is disabled before editing it. For instructions, see "Disabling MLLP Connections" on page 53.
3. On the Connections screen, click the **Edit** icon for the connection you want to edit.

   The Edit Connection screen displays the current parameters for the connection. Update the connection's parameters by typing or selecting the values you want to specify.
4. Click **Save Changes** to save the connection and return to the Connections screen.

Copying MLLP Connections

You can copy an existing MLLP connection to configure a new connection with the same or similar connection properties without having to re-type all of the properties for the connection. You copy MLLP connections using the Integration Server Administrator.

To copy a connection

1. In the Solutions menu in Integration Server Administrator, click **HL7 Module**.
2. On the Connections screen, click the **Copy** icon for the connection you want to copy.

   The Copy Connection screen displays the current parameters for the connection you want to copy. Name the new connection, specify a package name and folder name, and edit any connection parameters as needed by typing or selecting the values you want to specify.

> **Note:**
> When you copy a connection, the new connection does not save the password of the original connection. You must enter and then retype the password before you can save the new connection.

3. Click **Save Connection Copy** to save the connection and return to the Connections screen.

Deleting MLLP Connections

If you no longer want to use a particular MLLP connection, you can delete it by following the instructions in this section. You delete MLLP connections using Integration Server Administrator.

To delete a connection

1. In the Solutions menu in Integration Server Administrator, click **HL7 Module**.
2. Make sure the connection is disabled before deleting it. For instructions, see "Disabling MLLP Connections" on page 53.
3. On the Connections screen click the **Delete** icon for the connection you want to delete.

   Integration Server deleted the MLLP connection.

Sending HL7 Version 2.x Messages

HL7 Module provides the wm.ip.hl7.tn.transport:mllp service that acts as an MLLP client to deliver HL7 version 2.x messages using the MLLP transport protocol. The MLLP delivery service registers with Trading Networks during the startup of HL7 Module as one of the supported delivery methods.

To send HL7 version 2.x messages using Trading Networks, when you create trading partner profiles, you define the MLLP delivery method as part of the partner profile. For more information about creating profiles, see "Defining Trading Networks Profiles" on page 68 and the *webMethods Trading Networks Administrator's Guide* for your release.

To define MLLP as the delivery method in trading partner profiles

1. In My webMethods, go to **Applications > Administration > Integration > B2B > Partner Profiles > Add Profile**.
2. On the **Delivery Settings** tab,click **Add Delivery Method**.
3. In the **Delivery Method** field, select **MLLP**. Set values for the following required fields:

   > **Important:**

When configuring the delivery settings for the MLLP transport, do not specify values in the **Host, Port**, **User Name**, and **Password** fields. HL7 Module uses the value in the **connectionAlias** field to obtain the details of the MLLP server.

Field	Description
connectionAlias	The qualified namespace of the MLLP connection to the MLLP server, in the format *folder:connectionName*. For example, `connections:mllp_v2_host`
charsetEncoding	Used to determine the encoding when reading the HL7 message. If no value is specified, the **charsetEncoding** to be used is determined based on the information in the TPA. For more information, see the flow charts in "Determining the Encoding of HL7 Messages" on page 171. Valid values for this parameter are the Java charset encoding values supported by the HL7 standard. For information about the charset encoding supported by the HL7 standard, see Code Table: 0211 (Alternate Character Sets). **Note:** The **charsetEncoding** input parameter is different from the encoding mentioned in the HL7 message MSH.18 field. HL7 Module does not use the MSH.18 field encoding to parse or process the HL7 message.

4. Select the **Use as preferred protocol** checkbox if you want to use MLLP as your preferred method to deliver HL7 messages to your partner.

5. Click **OK**.

Receiving HL7 Version 2.x Messages

You receive HL7 version 2.x messages via the MLLP transport protocol using the HL7 Module MLLP listener. To manage the lifecycle of the HL7 Module MLLP listener, you configure an MLLP port where Integration Server listens for requests. The MLLP port is part of the WmHL7 package functionality and is only available after you install HL7 Module and enable the WmHL7 package. When configuring an MLLP port, you can specify which Integration Server service will be invoked by the MLLP listener when it receives an HL7 message via the MLLP transport protocol.

Configuring an MLLP Port

To configure an MLLP port

1. Open Integration Server Administrator if it is not already open.

2. Go to **Security** > **Ports**.

3. Click **Add Port**.

4. In the **Add Port** area of the screen, select **webMethods/MLLP**.

5. Click **Submit**. Integration Server Administrator displays a screen requesting information about the port. Enter the following information:

Field	Description
Port	The port on which Integration Server listens for MLLP requests. For example, `6000`. **Note:** If your Integration Server runs on a UNIX system, using a port number below 1024 requires that the server run as 'root.' For security reasons, Software AG discourages this practice. Instead, run your Integration Server using an unprivileged user ID on a high number port (for example 1024 or above) and use the port remapping capabilities present in most firewalls to move requests to the higher numbered ports.
Package Name	The name of the package associated with the MLLP port. The default is **WmHL7**. When you enable the package, the server enables the port. When you disable the package, the server disables the port and the MLLP port option is not visible on the Integration Server Administrator Add Port screen. **Important:** When you specify a package name other than the default value, the package associated with the MLLP port must have package dependency set to either the WmHL7 package, or to a package that is dependent on the WmHL7 package. In this way you ensure that when the WmHL7 package is disabled, the MLLP port package will be also disabled. For information about specifying package dependencies, see the *webMethods Service Development Help*. **Note:** If you replicate this package, Integration Server creates a port with this number and the same settings on the target server. If a port with this number already exists on the target server, its settings remain intact. HL7 Module will continue to work after it is replicated to another server.

Field	Description
Bind Address (optional)	The IP address to which to bind this port. For example, `10.60.25.217` or `myhl7.server.com`. Specify a bind address if your machine has multiple IP addresses and you want the port to use this specific address. If you do not specify a bind address, the server picks one for you.
Service	The name of the Integration Server service that the MLLP listener invokes when it receives an HL7 message via the MLLP transport. The default value is **wm.ip.hl7.tn.service:receive**. **Note:** When you use a custom service, the input and output signature of the custom service should have *contentStream* as an input parameter and *response* as an output parameter (similar to the wm.ip.hl7.tn.service:receive service). When the MLLP Listener receives the HL7 message, it sets the received content to the *contentStream* parameter and picks the response from the *response* output parameter. The listener sends this response as an acknowledgement for the received message when you use the synchronous delivery mode.
Synchronous ACK	Specify whether HL7 Module sends an acknowledgement synchronously or asynchronously. The default is **No**. Valid values: ■ **Yes** HL7 Module sends a synchronous ACK using the same connection channel. ■ **No** HL7 Module sends an asynchronous ACK via a new channel. For more information about sending an accept acknowledgment, see "Sending an Accept Acknowledgment" on page 61.
Run As User	The name of the Integration Server User that the MLLP listener uses to invoke the service specified in the **Service** field. By default no user is specified in the field. To specify a user, click the 🔍 icon and select a User Name from the **Select User** box.

6. Click **Save Changes**.

7. On the Ports screen, click **Edit** in the IP Access column to restrict the IP addresses that can connect to the MLLP port as follows:

   - **Allow by Default**. Set up the port to allow requests from all hosts except for ones you explicitly deny. Use this approach if you want to allow most hosts and deny a few. This is the default setting.

- **Deny by Default**. Set up the port to deny requests from all hosts except for ones you explicitly allow. Use this approach if you want to deny most hosts and allow a few.

Viewing and Editing the MLLP Port Configuration

To view and edit details about the MLLP port

1. Open the Integration Server Administrator if it is not already open.

2. Go to **Security** > **Ports**.

3. Find the MLLP port in the Port List and click the number in the **Port** column.

4. On the View MLLP Listener Details page, you can view or edit the MLLP listener configuration. To edit the MLLP listener configuration, click **Edit MLLP Listener Configuration**.

   Integration Server Administrator displays the Edit MLLP Listener Configuration screen, where you can edit the MLLP listener configuration as described in "Configuring an MLLP Port" on page 56.

For more information about configuring ports on Integration Server, see the *webMethods Integration Server Administrator's Guide* for your release.

Managing the MLLP Listener

When you want to start the MLLP listener, you enable the MLLP port. When you want to shut down the MLLP listener, you disable the MLLP port. Disabling the MLLP port blocks incoming requests from reaching the MLLP listener. When the MLLP port is disabled, clients receive an error message when they issue requests to it.

You can use the Integration Server Administrator to enable or disable the MLLP port. On the **Security > Ports** screen, find the MLLP port in the list of ports and change the status in the **Enabled** column to **Yes**when you want to enable the port. When you want to disable the port, change the status to **No**.

Another way to enable or disable the MLLP port is to enable or disable the package associated with the port. For the procedure how to enable or disable a port, see the *webMethods Integration Server Administrator's Guide* for your release.

HL7 Module Support for Multiple MLLP Listeners

You can configure more than one MLLP port on the same Integration Server and consequently you can have multiple MLLP listeners running on one Integration Server. The WmHL7 package of HL7 Module contains the wm.ip.hl7.tn.transport:mllpListenerStatus service that you can use to obtain the current status of all MLLP listeners.

You can run the HL7 Module MLLP listener status service in Developer or Designer.

To view MLLP listener status using the listener status service in Designer

1. Start Integration Server and Designer, if they are not already running.

2. Make sure Designer is using the Service Development perspective. If not, switch to it by selecting **Window > Open Perspective > Service Development**.

3. To connect to Integration Server:

   a. In Package Navigator view, select the Integration Server to which you want to connect.

   b. Right click, and select **Connect to server**.

      For more information about how to work with Integration Server in Designer, see the *webMethods Service Development Help*.

4. In the Package Navigator view, expand the WmHL7 package and locate the wm.ip.hl7.tn.transport:mllpListenerStatus service.

5. In Designer, select **Run > Run As > Run Service**.

6. In the **Enter Input for 'mllpListenerStatus'** dialog box, specify input values for the service.

   For information about the service parameters, see the wm.ip.hl7.tn.transport:mllpListenerStatus service.

7. Click **OK**. Designer runs the service and displays the results in the Service Result view.

Receiving More Than One HL7 Version 2.x Message via a MLLP Listener

HL7 Module supports receiving one or more HL7 version 2.x messages via an MLLP listener. When using the MLLP protocol, HL7 messages are transmitted in single data blocks. The HL7 version 2.x message content should be framed in a block before sending the message to the destination system via the MLLP transport protocol.

The HL7 content is enclosed by special characters to form a block in the following format:

```
<SB>HL7message<EB><CR>
```

where:

<SB> is the Start Block character (1 byte). ASCII `<VT>` character, that is `<0x0B>`.

<EB> is the End Block character (1 byte). ASCII`<FS>`character, that is `<0x1C>`.

<CR> is the Carriage Return (1 byte). ASCII `<CR>` character, that is `<0x0D>`.

The MLLP listener expects the incoming HL7 message or messages to be formatted in a content block. When receiving more than one message, the MLLP listener expects the messages to be formatted as a continuous series of HL7 content blocks.

For example:

- Format when sending a single message:

```
<SB>
MSH|^~\&|Sender||Receiver||20110614142226||ADT^A02|MSG00001|P|2.3|||||WINDO
WS-1252<CR>
EVN|A03<CR>
PID|||1221||TestMessage<CR>
PV1||C<CR>
<EB><CR>
```

- Format when sending more than one messages:

```
<SB>
MSH|^~\&|Sender||Receiver||20110614142226||ADT^A02|MSG00001|P|2.3|||||WINDO
WS-1252<CR>
EVN|A03<CR>
PID|||1221||TestMessage1<CR>
PV1||C<CR>
<EB><CR>
<SB>
MSH|^~\&|Sender||Receiver||20110614142226||ADT^A02|MSG00002|P|2.3|||||WINDO
WS-1252<CR>
EVN|A03<CR>
PID|||1221||TestMessage2<CR>
PV1||C<CR>
<EB><CR>
<SB>
MSH|^~\&|Sender||Receiver||20110614142226||ADT^A02|MSG00003|P|2.3|||||WINDO
WS-1252<CR>
EVN|A03<CR>
PID|||1221||TestMessage3<CR>
PV1||C<CR>
<EB><CR>
```

When HL7 Module receives multiple messages in a single request, the order in which the HL7 version 2.x messages are processed is not necessarily the order in which they appear in the incoming request. Therefore the processing of HL7 version 2.x messages in a sequence is not guaranteed. The MLLP listener treats each message as independent and sends it to the Integration Server service (the default is the wm.ip.hl7.tn.service:receive service) that you specified when configuring the MLLP listener Port in Integration Server Administrator. For more information about configuring the MLLP listener port, see "Configuring an MLLP Port" on page 56.

Sending an Accept Acknowledgment

To deliver an acknowledgment message, a TPA with an Agreement ID value of `HL7TPA` should be present in Trading Networks. When HL7 Module receives an HL7 message and identifies the trading partner, the module checks the *sendACK* parameter from the TPA and responds back with an acknowledgement based on the value selected for the *sendACK* parameter in the TPA. The *sendACK* field can take the following values:

- When you set the *sendACK* field in the TPA to `Default`, HL7 Module sends an acknowledgment based on the value set in the Accept Acknowledgment field (MSH-15). The Accept Acknowledgment field (MSH-15) can have one of the following values:
  - **AL -** Always. HL7 Module always sends an accept acknowledgment.
  - **NE -**Never. HL7 Module never sends an accept acknowledgment.
  - **ER -**Error conditions only. HL7 Module sends an accept acknowledgment only when it encounters errors.
  - **SU -**Successful completion only. HL7 Module sends an accept acknowledgment only when it receives the message successfully.
- When you set the *sendACK* field in the TPA to `Always`, HL7 Module always sends an acknowledgment regardless of the value set in the Accept Acknowledgment field (MSH-15).
- When you set the *sendACK* field in the TPA to `Never`, HL7 Module does not send an acknowledgment.

For information about the *sendACK* parameter and its values, see wm.ip.hl7.tn.tpa.rec:HL7TPA.

For more information about how to create a TPA and modify the default values in the TPA, see "Defining HL7 Trading Partner Agreements" on page 71 and the *webMethods Trading Networks Administrator's Guide* for your release.

Original and Enhanced Acknowledgement Mode

HL7 Module supports both the Original and Enhanced acknowledgment modes based on the MSH.15 and MSH.16 fields, following the HL7 standard. According to the HL7 standard, when both the MSH-15 (Accept acknowledgment type) and the MSH-16 (Application acknowledgment type) fields are null or not present, HL7 Module sends the acknowledgment using the Original acknowledgment mode. When at least one of the MSH-15 or MSH-16 fields is not null, the module sends the acknowledgment in the Enhanced acknowledgment mode.

The HL7 version 2.x message can be either accepted (CA), or rejected (CR) by the Receiver. The incoming HL7 version 2.x message is accepted when:

- The version of the received HL7 Message is supported by HL7 Module.
- The message type is valid for the given HL7 Message version.
- The Processing ID (MSH-11) for the received HL7 message is not null.

If the HL7 message does not meet the above criteria, the message is rejected.

When using the Original acknowledgment mode, HL7 Module includes in the MSA segment of the HL7 message the acknowledgment codes for the Original mode as defined by the HL7 standard. The following table describes the acknowledgment codes for the Original mode:

MSA Segment Value	Description
AA	Application Accept The message was accepted by the receiver.
AR	Application Reject The message has an application error and was not accepted by the receiver.

To identify the trading partner from the received HL7 Acknowledgment, HL7 Module processes the MSH Header of the acknowledgment, and interprets the MSH-3/MSH-4 and MSH-5/MSH-6 field pairs. For more information on partner identification, see "Extracting the Sender and the Receiver from the HL7 Message Header" on page 167.

Synchronous and Asynchronous Mode of Sending Acknowledgments

Using the MLLP Transport To Send Acknowledgments

When using MLLP as the transport protocol, HL7 Module sends an Accept Acknowledgment to the Sender using either a synchronous, or an asynchronous mode based on how you configure the **Synchronous ACK** parameters of the MLLP listener. For information about the parameters and the steps to configure them, see "Configuring an MLLP Port" on page 56.

If you configure the MLLP port to send acknowledgments synchronously, HL7 Module does not close the communication channel after receiving an HL7 message. HL7 Module processes the HL7 MSH segment to extract the Sender details and sends back an Accept Acknowledgment using the same request channel used to receive the HL7 message. The synchronous acknowledgment mode does not require an explicit partner identification or delivery settings on the Sender's side to send an acknowledgement back on the same requesting port.

If you configure the MLLP port to send acknowledgments asynchronously, HL7 Module opens a new communication channel to send back the acknowledgment. To send the acknowledgment asynchronously, HL7 Module requires a valid Sender delivery protocol. HL7 Module does not use the transport protocol in the request for the received HL7 message to send a response. The module closes the request after receiving the HL7 message, and creates a separate channel for sending the response with the accept acknowledgment. The transport protocol that HL7 Module uses to deliver the acknowledgment message is determined by the value set for the **ackProtocol** parameter in the TPA. If no value for **ackProtocol** is specified, HL7 Module uses the preferred protocol of the Sender trading partner profile.

Regardless of which delivery mode of the Accept Acknowledgement you specify in the MLLP port, synchronous or asynchronous, Trading Networks recognizes any ACK message as an HL7 Acknowledgement document, and processes the document based on the document recognition, document relation, and processing rule execution defined for HL7 Module documents.

Using the HTTP Transport To Send Acknowledgments

When using HTTP as the transport protocol, HL7 Module sends an Accept Acknowledgment to the Sender using either a synchronous, or an asynchronous mode based on how you configure the **ackMode** parameter when creating a TPA. In My webMethods, in the TPA Data section of the Trading Partner Agreements > Trading Partner Agreement Details screen, you can specify the following values for the **ackMode** parameter:

- **Async** This is the default. HL7 Module acknowledges receiving the message by sending an ACK as a separate transaction. The response is sent using the value of the **ackProtocol** parameter in the TPA.
- **Sync** HL7 Module acknowledges receiving the message by sending the ACK in the content part of the HTTP response. The HTTP response status code is `200`.

For more information about how to create a TPA and modify the default values in the TPA, see "Defining HL7 Trading Partner Agreements" on page 71 and the *webMethods Trading Networks Administrator's Guide* for your release.

Format of the Generated Acknowledgment Message

The acknowledgment message, generated by HL7 Module, contains the MSH and MSA segment in the HL7 message. The following table lists the values of the fields in the generated acknowledgment message:

Segment	Segment Field	Value
MSH	MSH-1	|
	MSH-2	^~\&
	MSH-3	Receiving application details from the received message.
	MSH-4	Receiving facility details from the received message.
	MSH-5	Sending application details from the received message.
	MSH-6	Sending facility details from the received message.
	MSH-7	Date and time when the acknowledgement message was created.
	MSH-9	*ACK^trigger event* specified in the received *message^ACK.*
	MSH-10	Unique message ID.

Segment	Segment Field	Value
	MSH-11	Processing ID details from the received message.
	MSH-12	The value matches the version of the received message.
	MSH-15	`NE`
MSA	MSA-1	■ `CA` when the message is accepted. or ■ `CR` when the message is rejected. **Note:** When HL7 Module uses the Original acknowledgment mode for sending the acknowledgement, the values are `AA` (when the message is accepted) and `AR` (when the message is rejected). For a full description of the values, see "Original and Enhanced Acknowledgement Mode" on page 62.
	MSA-2	The message control ID of the received message.

5 Defining Trading Networks Information

Overview

To use Trading Networks as a gateway for HL7 message exchange with your trading partners, you define the following Trading Networks information:

- Trading partner profiles help define how you and your trading partners exchange HL7 messages.
- A Trading Partner Agreement (TPA) is a set of parameters that you can use to govern how business documents are exchanged between two trading partners.

In addition, you register the following HL7-specific items with Trading Networks to define how Trading Networks processes message transactions:

- TN document types for HL7 version 2.x messages
- Custom Trading Networks Attributes for HL7 version 2.x messages
- Processing rules for HL7 version 2.x messages

Defining Trading Networks Profiles

For webMethods HL7 Module, you define a single trading partner profile for your organization (**My Enterprise**), and then you define a trading partner profile for each trading partner with whom you want to exchange HL7 messages using Trading Networks.

Defining Your Enterprise Profile

Before you define your trading partner profiles in Trading Networks and exchange business documents with your trading partners, you must first define your Enterprise profile. You define your enterprise profile by completing the fields on the Partner Profiles page in My webMethods.

The following section specifies the required fields you must complete when defining your enterprise profile for use with HL7 Module.

Required Enterprise Profile Fields

Profile information is displayed on the **Applications > Administration > Integration > B2B > Partner Profiles** tab in My webMethods. The following table lists and describes the required fields you must set when defining your enterprise profile (**My Enterprise**).

Required Profile Field for Enterprise	Description
Corporation Name	The name of your enterprise. For information about identifying a trading partner from the fields in the message header, see "Identifying a Trading Partner" on page 167.
Unit Name	Optional. Specify the name of your unit.
External IDs > Add ID	The value for the external ID type that your enterprise uses within documents. When exchanging documents, partners typically identify themselves within a document using some well-known ID scheme, such as a D-U-N-S number. You must select one of the ID Types available in the Trading Networks database, and then specify the value that identifies your enterprise for that ID type.
Delivery Settings > Add Delivery Method	Specify the delivery method that you want Trading Networks to use to send HL7 messages to your partner's system. You must specify at least one delivery method as your preferred method by selecting the **Use as preferred protocol** check box. If you do not specify a preferred delivery method in Trading Networks, HL7 Module will not be able to determine how to send and receive the HL7 messages. You can specify any of the transport protocols that HL7 Module supports. For more information about the supported protocols, see "Configuring Communication Protocols" on page 42.

For procedural information about defining your enterprise profile, as well as descriptions of the fields you must complete when defining your enterprise profile, see the *webMethods Trading Networks Administrator's Guide* for your release.

Activating Your Enterprise Profile

You must activate (or enable) your enterprise profile before you can exchange documents with trading partners. For instructions, see the *webMethods Trading Networks Administrator's Guide* for your release.

Defining Your Trading Partners' Profiles

Each trading partner with whom you want to exchange business documents must have a trading partner profile in Trading Networks. After you have defined your enterprise profile, you are ready to define your trading partners' profiles. You define a trading partner profile by completing the required fields in My webMethods.

The following section specifies the required fields you must complete to define a trading partner profile.

Required Profile Fields

In My webMethods, go to **Application > Administration > Integration > B2B > Partner Profiles > Add Profile** and set the following required fields to define your partner's profile.

Required Profile Field for Trading Partner	Description
Corporation Name	The name of your partner's enterprise. For information about identifying a trading partner from the fields in the message header, see "Identifying a Trading Partner" on page 167.
Unit Name	Optional. Specify the name of your unit.
External IDs > Add ID	The value for the external ID type that your partner uses within documents. When exchanging documents, partners typically identify themselves within a document using some well-known ID scheme, such as a D-U-N-S number. You must select one of the ID Types available in the Trading Networks database, and then specify the value that identifies your partner's enterprise for that ID type.
Delivery Settings > Add Delivery Method	Specify the delivery method that you want Trading Networks to use to send HL7 messages to your partner's system. You must specify at least one delivery method as your preferred method by selecting the **Use as preferred protocol** check box. If you do not specify a preferred delivery method in Trading Networks, HL7 Module will not be able to determine how to send and receive the HL7 messages. You can specify any of the transport protocols that HL7 Module supports. For more information about the supported protocols, see "Configuring Communication Protocols" on page 42.

For procedural information about defining a trading partner profile, as well as descriptions of the fields you must complete when defining a trading partner profile, see the *webMethods Trading Networks Administrator's Guide* for your release.

Activating Your Trading Partners' Profiles

You must activate (or enable) your trading partner profile before you can exchange documents with your trading partners. For instructions, see the *webMethods Trading Networks Administrator's Guide* for your release.

Defining HL7 Trading Partner Agreements

Every TPA is uniquely identified by a Sender, Receiver, and an Agreement ID. During a transaction between trading partners, HL7 Module uses this information to retrieve the TPAs for the initiator/sender and fulfiller/receiver in the transaction and to process the business documents exchanged. Every message that is exchanged in HL7 Module is associated with a TPA.

You define and view TPAs in My webMethods on the **Administration > Integration > B2B > Trading Partner Agreements** page. For detailed information about working with TPAs in My webMethods, see the *webMethods Trading Networks Administrator's Guide*.

Note:
If you are using HL7 Module with Integration Server version 8.0.x or 7.1.x, you define TPAs using the Trading Networks Console. For information about how to use the Trading Networks Console, see the *webMethods Trading Networks Administrator's Guide* for your release.

Note:
You must configure at least one TPA (or use the default HL7 Module TPA) to be able to execute message transactions.

With HL7 Module, you can create a TPA in one of the following ways:

- **Create a default TPA.** HL7 Module automatically creates a default TPA when you run the module for the first time. Use the default TPA to meet the requirements of the majority of your trading partners.

  The default TPA is created with the following values:

Parameter	Value
Agreement ID	`HL7TPA`
Sender	`Unknown`
Receiver	`Unknown`

  During message processing, HL7 Module first searches for a TPA with unique values for the Sender, Receiver, and the Agreement ID, and a Status value of **agreed**. If that TPA exists, HL7 Module uses it. If no such TPA exists, the module uses the default TPA.

- **Create a partner-specific TPA.** Use this option when you must define partner-specific values in the TPA fields that are different from the default values in the HL7 Module default TPA. When creating a partner-specific TPA, you have to specify only the information that is different from the defaults.

Using the Default TPA

You can automatically use the HL7 Module default TPA, for example, when you want to prevent multiple Trading Networks queries to the database to assess what TPA should be used. To use the default TPA regardless of the sender-receiver pair values, you must configure the watt.hl7.tpa.alwaysUse Integration Server configuration parameter.

You configure the watt.hl7.tpa.alwaysUse parameter in **Integration Server Administrator > Settings > Extended**, as follows:

```
watt.hl7.tpa.alwaysUse=value
```

where *value* is `true` or `false`.

- Set the parameter to true when you want HL7 Module to automatically use the default TPA to process the HL7 message.
- Set the parameter to false when you want HL7 Module to search for the TPA to use to process the message. This is the default.

For more information about working with extended configuration settings, see the *webMethods Integration Server Administrator's Guide* for your release.

Creating a Partner-Specific TPA

You create a partner-specific TPA in one of the following ways:

- Using the HL7 Module wm.ip.hl7.tn.tpa:createHL7TPA service as described in "Using the HL7 Module wm.ip.hl7.tn.tpa:createHL7TPA Service" on page 72.
- Duplicate the default TPA and modify the values in the partner-specific fields. For information on duplicating a TPA, see the *webMethods Trading Networks Administrator's Guide* for your release.
- Create a new TPA manually as described in "Manually Creating a TPA" on page 73.

Using the HL7 Module wm.ip.hl7.tn.tpa:createHL7TPA Service

HL7 Module provides the wm.ip.hl7.tn.tpa:createHL7TPA service that you use to create a partner specific TPA and associate it with HL7-specific parameters and values.

Note:
You can also use the createHL7TPA service to create the HL7 Module default TPA. For example, when the default TPA is deleted instead of reloading the WmHL7 package to generate the default TPA, you can create a default TPA by running the createHL7TPA without providing any input values.

To create a partner specific TPA using the wm.ip.hl7.tn.tpa:createHL7TPA service

1. Start Integration Server and Designer, if they are not already running.

2. Make sure Designer is using the Service Development perspective. If not, switch to it by selecting **Window > Open Perspective > Service Development**.

3. To connect to Integration Server:

   a. In Package Navigator view, select the Integration Server to which you want to connect.

   b. Right click, and select **Connect to server**.

      For more information about how to work with Integration Server in Designer, see the *webMethods Service Development Help*.

4. In Package Navigator view, expand the WmHL7 package and locate the wm.ip.hl7.tn.tpa:createDefaultTPA service.

5. In Designer, select **Run > Run As > Run Service**.

6. In the **Enter Input for 'createDefaultTPA'** dialog box, specify input values for the service.

   For information about the service parameters, see the wm.ip.hl7.tn.tpa:createHL7TPA service.

7. Click **OK**. Designer runs the service and displays the results in the Service Result view.

Manually Creating a TPA

You can manually create a TPA by using the Trading Partner Agreements page in My webMethods.

To create a new TPA manually

1. In My webMethods, go to **Administration > Integration > B2B > Trading Partner Agreements > Add TPA**.

2. Specify the following values for the TPA fields on the Trading Partner Agreements page:

TPA Field	Value
Agreement ID	`HL7TPA`
Sender	The name of the trading partner that has the sender role in the TPA.
Receiver	The name of the trading partner that has the receiver role in the TPA.
Initialization Service	`wm.ip.hl7.tn.tpa:initTPA` service, which sets default values for the wm.ip.hl7.tn.tpa.rec:HL7TPA IS document type.
Export Service	Leave this field blank.
IS Document Type	`wm.ip.hl7.tn.tpa.rec:HL7TPA`

3. Do one of the following:

- Click **Save** to save the changes you have made and continue working on the TPA.
- Click **Save and Close** to save the changes and close the TPA.

4. In Designer or Developer, run the wm.ip.hl7.tn.tpa:initTPA service to populate the TPA with default values for the specified **IS Document Type**.

   You can keep the values that you want to use and modify the values that do not suit your requirements. For information about default values, see wm.ip.hl7.tn.tpa.rec:HL7TPA.

For more information about creating a TPA and modifying the default values in the TPA, see the *webMethods Trading Networks Administrator's Guide* for your release.

TN Document Types for HL7 Version 2.x Messages

HL7 Module provides the following default TN document types:

- HL7 Default
- HL7 XML Default
- HL7 Acknowledgment
- HL7 XML Acknowledgment

You can view the default TN document types for HL7 version 2.x messages using the Document Types page in My webMethods.

Note:
If you are using HL7 Module with Integration Server version 8.0.x or 7.1.x, you view the TN document types using the Trading Networks Console. For information about how to use the Trading Networks Console, see the *webMethods Trading Networks Administrator's Guide* for your release.

To view TN document types for HL7 version 2.x messages

1. In My webMethods, go to **Administration > Integration > B2B > Document Types Administration >Document Types**.

   Trading Networks displays all enabled TN document types.

2. You can search for the HL7 Module TN document types. For information about how to perform a TN document type search in My webMethods, see the *webMethods Trading Networks Administrator's Guide*.

For more information about TN document types, see the *webMethods Trading Networks Administrator's Guide* for your release.

Custom Trading Networks Attributes for HL7 Version 2.x Messages

The custom attributes for HL7 version 2.x messages are added to Trading Networks when you install HL7 Module. The module checks if all the HL7 Trading Networks attributes are already present during startup. If they are not present, the attributes are registered with Trading Networks. For information about how to view and manage document attributes, see the *webMethods Trading Networks Administrator's Guide* for your release.

Note:HL7 Module extracts the values for these attributes from the corresponding fields of the MSH segment of the HL7 message.

Note:
Not all attributes listed in the following tables qualify for a specific HL7 2.x version. The tables describe a super-set of all attributes across all 2.x versions supported by HL7 Module. For information about the elements defined in a specific HL7 2.x standard, see the HL7 documentation for that standard. When a Trading Networks attribute does not qualify for a specific HL7 2.x version, the generated BizDocEnvelope has an empty value for that particular attribute.

In My webMethods, you can view the following custom attributes associated with the HL7 Default and HL7 XML Default TN document types:

Custom Attribute	MSH Segment Field	Description
HL7 Field Separator	MSH-1	Defines the character to be used as a field separator in the message. Recommended value is `|`
HL7 Encoding Characters	MSH-2	Contains two to four special characters used in constructing the message in the following order: the component separator, repetition separator, escape character, and subcomponent separator. Recommended values are `^~\&`
HL7 Sending Application	MSH-3	Identifies the sending application among all applications that participate in the HL7 message exchange within the enterprise. The HL7 Module createBizDoc service uses the value of this attribute for the sender profile ID when the input *senderID* information is missing.
HL7 Sending Facility	MSH-4	Identifies the sending application among multiple identical instances of the application running on behalf of different organizations.
HL7 Receiving Application	MSH-5	Identifies the receiving application among all applications that participate in the HL7 message exchange within the enterprise. The HL7 Module createBizDoc service uses the value of this attribute for

Custom Attribute	MSH Segment Field	Description
		the receiver profile ID when the input *receiverID* information is missing.
HL7 Receiving Facility	MSH-6	Identifies the receiving application among multiple identical instances of the application running on behalf of different organizations.
HL7 Datetime Of Message	MSH-7	Contains the date and time when the sending system created the message.
HL7 Security	MSH-8	Contains security information for the HL7 message.
HL7 Message Type	MSH-9	Indicates the message type of the HL7 message. For example, `ADT_A01`.
HL7 Message Control Id	MSH-10	Unique identifier for the message, such as a number. The response is related to the original message based on the message control Id.
HL7 Processing Id	MSH-11-1	Indicates whether to process the message as part of a production, training, or debugging system.
HL7 Processing Mode	MSH-11-2	Indicates whether the message is part of an archival process or an initial load.
HL7 Version Id	MSH-12	Identifies the HL7 message version. For example, `2.3`.
HL7 Sequence Number	MSH-13	Indicates whether the sequence number protocol is in use. Valid values are: ■ `null` - The sequence number protocol is not in use. ■ `non-null` - The sequence number protocol is in use.
HL7 Continuation Pointer	MSH-14	When a single logical HL7 message is fragmented and sent as several actual HL7 messages, this attribute contains a unique value that is used to match a subsequent message with this specific value.
HL7 Accept Acknowledgment Type	MSH-15	Indicates whether an accept acknowledgment is required in response to the message. This attribute has the following valid values: ■ `AL`- Always. HL7 Module always sends an accept acknowledgment. ■ `NE` -Never. HL7 Module never sends an accept acknowledgment.

Custom Attribute	MSH Segment Field	Description
		■ `ER` –Error conditions only. HL7 Module sends an accept acknowledgment only when it encounters errors. ■ `SU` –Successful completion only. HL7 Module sends an accept acknowledgment only when it receives the message successfully.
HL7 Application Acknowledgment Type	MSH-16	Indicates whether an application acknowledgment is required in response to the message. HL7 Module does not process the MSH-16 field. The Application Acknowledgment is processed and generated by the receiving partner as required.
HL7 Country Code	MSH-17	Contains the country of origin of the message specified in three-character (alphabetic) form as defined by ISO 3166.
HL7 Character Set	MSH-18	Contains the character set for the entire message. The default value is **ASCII**.
HL7 Alternate Characterset Handling Scheme	MSH-20	Specifies the scheme to use for alternative character sets. The default value for this attribute is **null**. It indicates that no alternate character sets occur in this message.
HL7 Message Profile Identifier	MSH-21	Refers to or indicates adherence to a message profile. Message profiles contain grammar, syntax, and usage details for a message or set of messages.
HL7 Sending Responsible Organization	MSH-22	Identifies the business organization that is the originator of the message and is legally responsible for the contents of the message. The value for this attribute contains the name and ID number of the organization.
HL7 Receiving Responsible Organization	MSH-23	Identifies the business organization that is the intended receiver of the message and is legally responsible to act on the information that the message contains. The value for this attribute contains the name and ID number of the organization.
HL7 Receiving Network Address	MSH-25	Identifies the network location (such as the URL) to which the message was sent.

The HL7 Acknowledgment and HL7 XML Acknowledgment TN document types are associated with all of the above Trading Networks attributes and a few additional ones, described in the following table:

Custom Attribute	MSA Segment Field	Description
HL7 Code	MSA-1	Indicates the HL7 message coding system. This attribute contains values, such as AA, AL, and AR.
HL7 Control Id	MSA-2	This field contains the message control ID of the message sent by the sending system. The sending system uses this ID to associate this response with the message for which it is intended.
HL7 Text	MSA-3	Contains text data to display on a screen or printer to describe an error condition.
HL7 Expected Sequence Number	MSA-4	This attribute is used in the sequence number protocol.
HL7 Waiting Number	MSA-7	Indicates the number of messages the acknowledging application has waiting in a queue.
HL7 Waiting Priority	MSA-8	Indicates the priority of the most important message in the sending application. Valid values are: ■ `H` - High. There is at least one message with priority "high" waiting for an acknowledgment. ■ `M` - Medium. There is at least one message with priority "medium" and no messages with higher priority waiting for an acknowledgment. ■ `L` - Low. The highest priority of any message waiting is "low".

Processing Rules to Process HL7 Version 2.x Messages

HL7 Module registers four processing rules with Trading Networks that correspond to the TN document type with a similar name provided with HL7 Module. Based on the format of the HL7 version 2.x message to process, HL7 Module invokes one of the processing rules specified in the following table:

HL7 Message Format	Processing Rules	TN document type
ER7	HL7 2.x Message Default	HL7 Default

HL7 Message Format	Processing Rules	TN document type
	HL7 2.x Message Acknowledgment	HL7 Acknowledgment
XML	HL7 2.x XML Message Default	HL7 XML Default
	HL7 2.x XML Message Acknowledgment	HL7 Acknowledgment

HL7 Module verifies if the processing rules are present during startup. If they are not present, HL7 Module registers the processing rules with Trading Networks. These processing rules contain the criteria that you want Trading Networks to use to select the processing rule, and the pre-processing and processing actions that you want Trading Networks to perform against the document. For example, for an inbound HL7 version 2.x message, you might use the **Execute a Service** action to invoke a service that you create; for an outbound HL7 version 2.x message, you might use the **Deliver Document By** action to have Trading Networks deliver the outbound HL7 version 2.x message.

HL7 Module processing rules are executed when sending or receiving HL7 version 2.x messages. HL7 Module invokes the processing rules after persisting the message in Trading Networks. The processing rule invoked is specific to the BizDocEnvelope generated during the send or receive operation. The HL7 2.x Message Acknowledgment and HL7 2.x XML Message Acknowledgment processing rules are executed for ACK messages. The HL7 2.x Message Default and HL7 2.x XML Message Default processing rules are executed for the remaining message types.

Defining a Custom Processing Rule

You manage processing rules using My webMethods. You can use the default processing rules provided with HL7 Module as a template for defining your custom processing rules.

Note:
If you are using HL7 Module with Integration Server version 8.0.x or 7.1.x, you define and manage processing rules using the Trading Networks Console. For information about how to use the Trading Networks Console, see the *webMethods Trading Networks Administrator's Guide* for your release.

Note:
To define a custom processing rule, add a new processing rule that is a duplicate of the default one. Trading Networks adds the duplicate rule to the bottom of the processing rules list. Using My webMethods, you can change the default values that Trading Networks has copied from the original processing rules with custom values. For the steps to define and manage a processing rule, see the *webMethods Trading Networks Administrator's Guide* for your release.

6 Processing HL7 Messages

Overview

webMethods HL7 Module provides built-in services that you use to:

- Parse incoming HL7 ER7 or XML documents into the IS document format supported by Integration Server (IData objects).
- Generate HL7 version 2.x messages from the internal representation supported by Integration Server (IS document types).

For more information on IData objects and IS document types, see "Architecture and Components" on page 12 and the *webMethods Service Development Help*.

Important:
To process HL7 version 2.x messages with HL7 Module correctly, you must verify that the MSH segment of the HL7 message contains all required fields as described in "Verifying The Message Header Segment of HL7 Messages" on page 165.

Note:HL7 Module expects an HL7 ER7-encoded message to contain '\r', '\n', or '\r\n' as a segment separator. When parsing incoming HL7 version 2.x messages, that is converting an HL7 ER7 message to IData, HL7 Module accepts the first of these characters that it encounters as the segment separator. However, when converting IData to an ER7 message, the segment separator used is always '\r' (carriage return <cr> or a hexadecimal 0D).

Parsing Incoming HL7 Version 2.x Messages into IData Objects

When converting an incoming HL7 version 2.x message into an IData object, for example to send it to Integration Server for processing, you use the wm.ip.hl7.service:convertHL7ToIData service. To run the service, use either Developer or Designer. The following procedure describes how to run the service in Designer.

To convert HL7 version 2.x messages into IData objects

1. Start Integration Server and Designer, if they are not already running.
2. Make sure Designer is using the Service Development perspective. If not, switch to it by selecting **Window > Open Perspective > Service Development**.
3. To connect to Integration Server:

   a. In Package Navigator view, select the Integration Server to which you want to connect.

   b. Right click, and select **Connect to server**.

   For more information about how to work with Integration Server in Designer, see the *webMethods Service Development Help*.

4. In Package Navigator view, expand the WmHL7 package and locate the wm.ip.hl7.service:convertHL7ToIData service.

5. In Designer, select **Run > Run As > Run Service**.

6. In the **Enter Input for 'convertHL7ToIData'** dialog box, specify input values for the service.

   For information about the service parameters, see the wm.ip.hl7.service:convertHL7ToIData service.

7. Click **OK**. Designer runs the service and displays the results in the Service Result view.

Creating Outgoing HL7 Version 2.x Messages

HL7 Module provides the wm.ip.hl7.service:convertIDataToHL7 service that you use to convert an IS document type of a specific HL7 message version into the respective HL7 version 2.x message format. Examples of each message format follow the procedure.

To create HL7 version 2.x messages

1. Start Integration Server and Designer, if they are not already running.

2. Make sure Designer is using the Service Development perspective. If not, switch to it by selecting **Window > Open Perspective > Service Development**.

3. To connect to Integration Server:

   a. In Package Navigator view, select the Integration Server to which you want to connect.

   b. Right click, and select **Connect to server**.

   For more information about how to work with Integration Server in Designer, see the *webMethods Service Development Help*.

4. In Package Navigator view, expand the WmHL7 package and locate the wm.ip.hl7.service:convertIDataToHL7 service.

5. In Designer, select **Run > Run As > Run Service**.

6. In the **Enter Input for 'convertIDataToHL7'** dialog box, specify input values for the service.

For information about the service parameters, see the wm.ip.hl7.service:convertIDataToHL7 service.

7. Click **OK**. Designer runs the service and displays the results in the Service Result view.

Example: HL7 ER7-Encoded Message

The example below shows an HL7 ER7-encoded message created for the ADT_A01 message type, version 2.4:

```
MSH|^~\&|ADT1^234567891^DUNS|MCM|LABADT^876543219^DUNS|MCM|1988081
81126|SECURITY|ADT^A01|MSG00001|P|2.4|||AL
EVN|A01|198808181123||
PID|1||PATID1234^5^M11^ADT1^MR^MCM~123456789^^^USSSA^SS||JONES^WILLIAM^A^III||1
9610615|M|RON|2028-9|1200 N ELM STREET^^GREENSBORO^NC^27401-1020|GL|(91-9)379-
1212|(919)271-3434||S||PATID12345001^2^M10^ADT1^AN^A|123456789|987654^NC|
NK1|1|JONES^BARBARA^K|SPO^WIFE||||N^NEXT OF KIN
PV1|1|I|2000^2012^01||||004777^LEBAUER^SIDNEY^J.|||SUR||||3|A0|
```

Example: HL7 XML-Encoded Message

The example below shows an HL7 XML-encoded message created for the ADT_A01 message type, version 2.4:

```
<?xml version="1.0" encoding="UTF-8"?>
<ADT_A01>
  <MSH>
    <MSH.1>|</MSH.1>
    <MSH.2>^~\&amp;</MSH.2>
    <MSH.3>
      <HD.1>ADT1</HD.1>
      <HD.2>234567891</HD.2>
      <HD.3>DUNS</HD.3>
    </MSH.3>
    <MSH.4>
      <HD.1>MCM</HD.1>
    </MSH.4>
    <MSH.5>
      <HD.1>LABADT</HD.1>
      <HD.2>876543219</HD.2>
      <HD.3>DUNS</HD.3>
    </MSH.5>
    <MSH.6>
      <HD.1>MCM</HD.1>
    </MSH.6>
    <MSH.7>
      <TS.1>198808181126</TS.1>
    </MSH.7>
    <MSH.8>SECURITY</MSH.8>
    <MSH.9>
      <MSG.1>ADT</MSG.1>
      <MSG.2>A01</MSG.2>
    </MSH.9>
    <MSH.10>MSG00001</MSH.10>
    <MSH.11>
      <PT.1>P</PT.1>
    </MSH.11>
    <MSH.12>
      <VID.1>2.4</VID.1>
    </MSH.12>
    <MSH.15>AL</MSH.15>
  </MSH>
  <EVN>
    <EVN.1>A01</EVN.1>
    <EVN.2>
      <TS.1>198808181123</TS.1>
    </EVN.2>
  </EVN>
  <PID>
    <PID.1>1</PID.1>
    <PID.3>
      <CX.1>PATID1234</CX.1>
      <CX.2>5</CX.2>
      <CX.3>M11</CX.3>
      <CX.4>
        <HD.1>ADT1</HD.1>
      </CX.4>
      <CX.5>MR</CX.5>
      <CX.6>
        <HD.1>MCM</HD.1>
      </CX.6>
    </PID.3>
    <PID.3>
      <CX.1>123456789</CX.1>
      <CX.4>
        <HD.1>USSSA</HD.1>
      </CX.4>
      <CX.5>SS</CX.5>
    </PID.3>
    <PID.5>
      <XPN.1>
        <FN.1>JONES</FN.1>
      </XPN.1>
      <XPN.2>WILLIAM</XPN.2>
      <XPN.3>A</XPN.3>
      <XPN.4>III</XPN.4>
    </PID.5>
    <PID.7>
      <TS.1>19610615</TS.1>
    </PID.7>
    <PID.8>M</PID.8>
    <PID.9>
      <XPN.1>
        <FN.1>RON</FN.1>
      </XPN.1>
    </PID.9>
    <PID.10>
      <CE.1>2028-9</CE.1>
    </PID.10>
    <PID.11>
      <XAD.1>
        <SAD.1>1200 N ELM STREET</SAD.1>
      </XAD.1>
      <XAD.3>GREENSBORO</XAD.3>
      <XAD.4>NC</XAD.4>
      <XAD.5>27401-1020</XAD.5>
    </PID.11>
    <PID.12>GL</PID.12>
    <PID.13>
      <XTN.1>(91-9)379-1212</XTN.1>
    </PID.13>
    <PID.14>
      <XTN.1>(919)271-3434</XTN.1>
    </PID.14>
    <PID.16>
      <CE.1>S</CE.1>
    </PID.16>
    <PID.18>
      <CX.1>PATID12345001</CX.1>
      <CX.2>2</CX.2>
      <CX.3>M10</CX.3>
      <CX.4>
        <HD.1>ADT1</HD.1>
      </CX.4>
      <CX.5>AN</CX.5>
      <CX.6>
        <HD.1>A</HD.1>
      </CX.6>
    </PID.18>
    <PID.19>123456789</PID.19>
    <PID.20>
      <DLN.1>987654</DLN.1>
      <DLN.2>NC</DLN.2>
    </PID.20>
  </PID>
  <NK1>
    <NK1.1>1</NK1.1>
    <NK1.2>
      <XPN.1>
        <FN.1>JONES</FN.1>
      </XPN.1>
      <XPN.2>BARBARA</XPN.2>
      <XPN.3>K</XPN.3>
    </NK1.2>
    <NK1.3>
      <CE.1>SPO</CE.1>
      <CE.2>WIFE</CE.2>
    </NK1.3>
    <NK1.7>
      <CE.1>N</CE.1>
      <CE.2>NEXT OF KIN</CE.2>
    </NK1.7>
  </NK1>
  <PV1>
    <PV1.1>1</PV1.1>
    <PV1.2>I</PV1.2>
    <PV1.3>
      <PL.1>2000</PL.1>
      <PL.2>2012</PL.2>
      <PL.3>01</PL.3>
    </PV1.3>
    <PV1.7>
      <XCN.1>004777</XCN.1>
      <XCN.2>
        <FN.1>LEBAUER</FN.1>
      </XCN.2>
      <XCN.3>SIDNEY</XCN.3>
      <XCN.4>J.</XCN.4>
    </PV1.7>
    <PV1.10>SUR</PV1.10>
    <PV1.14>3</PV1.14>
    <PV1.15>A0</PV1.15>
  </PV1>
</ADT_A01>
```

Validating HL7 Version 2.x Messages

When parsing ER7 or XML-formatted HL7 version 2.x messages, you use the wm.ip.hl7.service:validate service to validate HL7 messages. You can invoke the validation service at any time during the processing of both inbound and outbound HL7 version 2.x messages. Messages can be validated with or without Trading Networks. For information about how the HL7 Module validate service handles errors during the validation process, see "Handling Validation Errors" on page 108.

Validating HL7 Version 2.x Messages without Trading Networks

To validate HL7 version 2.x messages without using Trading Networks, run the HL7 Module validation service in Developer or Designer.

To validate HL7 version 2.x messages using the validation service in Designer

1. Start Integration Server and Designer, if they are not already running.

2. Make sure Designer is using the Service Development perspective. If not, switch to it by selecting **Window > Open Perspective > Service Development**.

3. To connect to Integration Server:

   a. In Package Navigator view, select the Integration Server to which you want to connect.

   b. Right click, and select **Connect to server**.

   For more information about how to work with Integration Server in Designer, see the *webMethods Service Development Help*.

4. In Package Navigator view, expand the WmHL7 package and locate the wm.ip.hl7.service:validate service.

5. In Designer, select **Run > Run As > Run Service**.

6. In the **Enter Input for 'validate'** dialog box, specify input values for the service.

   For information about the service parameters, see the wm.ip.hl7.service:validate service.

7. Click **OK**. Designer runs the service and displays the results in the Service Result view.

Validating HL7 Version 2.x Messages with Trading Networks

The following procedure lists the basic steps to use the HL7 Module validate service to validate HL7 version 2.x messages using Trading Networks:

To validate HL7 version 2.x messages using Trading Networks

1. Create a flow service with the following logic:
   - Extract the data from the BizDocEnvelope generated for the HL7 message.
   - Invoke the HL7 Module validate service.
2. When you define the processing rule for the HL7 message, specify the **Execute a service** action as the processing action and the user-defined flow service as the service to execute during that processing action.

Code Table Validation Using Partner-Specific Code Tables

By default, HL7 Module performs code table validation using the default set of code tables. To enable HL7 Module to use partner specific code tables for code table validation, ensure that:

1. You have created a profile for the partner in Trading Networks. For information about creating partner profiles, see “Defining Trading Networks Profiles” on page 68.
2. You have enabled code table customizing for that partner. For information about enabling code table customizing for a trading partner, see “Enabling Customizing for a Trading Partner” on page 101.
3. You have enabled the customized code tables for that partner for validation. For information about enabling code tables, see “Enabling Code Tables” on page 97.
4. The Sender application and Receiver application fields of the message are filled in as described in “Format of the Sender Application and Receiver Application Fields” on page 88. HL7 Module uses the Universal ID and Universal ID Type from those fields in the message header to retrieve the partner details from the Trading Networks partner profile.

Format of the Sender Application and Receiver Application Fields

When HL7 Module performs code table validation for a message, it determines whether to use partner specific code tables based on the value of the fields in the message header. During the execution of the convertHL7ToIData service, HL7 Module obtains the validation information from the Sender Application field (MSH.3). During the execution of the convertIDataToHL7 service, the module uses the Receiver Application field (MSH.5).

The MSH.3 and the MSH.5 fields contain the following components:

Component	HD Type	Description
namespace ID	HD.1	The application namespace value. In the MSH.3 and MSH.5 fields, the namespace ID corresponds to the **Corporation Name** in a Trading Networks partner profile. In the MSH.4 and MSH.6 fields, if present, the namespace ID can optionally correspond to the **Unit Name** for the sending and receiving Trading Networks partner respectively. For more information about how HL7 Module uses this information to identify a trading partner, see "Identifying a Trading Partner" on page 167. **Note:** Ensure that the application namespace does not contain an underscore character ('_').
universal ID	HD.2	Corresponds to the External ID of a Trading Networks partner.
universal ID Type	HD.3	Corresponds to the External ID Type of a Trading Networks partner.

The format of the data in the MSH.3 and MSH.5 fields depends on the HL7 message version:

- For HL7 message versions 2.1 and 2.2, the MSH.3 and MSH.5 fields in the message header of both ER7 and XML formatted messages must be specified as a string type ST and separated by underscore "_" as follows:

  namespace ID (IS)_universal ID (ST)_universal ID type (ID)

  For example: `wmIS_1011_DUNS`

- For all other HL7 2.x message versions, the format differs for ER7 and XML messages as follows:

  - For ER7 messages, the schema contains the *namespace ID*, *universal ID*, and *universal ID type* as different components that are separated by the component separator in the ER7 message.

    For example: `wmIS^1011^DUNS`, where ^ is the message component separator.

  - For XML messages, the components are in separate tags in the XML message.

Retaining Empty Fields and Component Separators in HL7 Messages

When serializing incoming HL7 messages, you can control whether HL7 Module retains the empty field and component separators from the incoming HL7 message by configuring the watt.hl7.parser.trimDelimiters Integration Server configuration parameter. When deserializing the HL7 message, it will have empty field or component separators if they are present in the original HL7 message.

You configure the watt.hl7.parser.trimDelimiters Integration Server configuration parameter in **Integration Server Administrator > Settings > Extended**, as follows:

```
watt.hl7.parser.trimDelimiters=value
```

where *value* is `true` or `false`.

- Set the parameter to true if you want HL7 Module to trim (crop) empty field and component separators.
- Set the parameter to false (or delete it from the Extended Settings list) if you want HL7 Module to retain empty field and component separators. This is the default.

Note:
The parameter name is case sensitive.

For more information about the server configuration file and working with extended configuration settings, see the *webMethods Integration Server Administrator's Guide*.

7 Viewing Information about HL7 Messages

- "Overview" on page 92
- "Viewing HL7 Version 2.x Messages" on page 92

Overview

When you use webMethods HL7 Module with webMethods Trading Networks, you can view the HL7 version 2.x messages that have passed through your system and have been saved to the Trading Networks database. You view documents (transactions) for every HL7 version 2.x message that is sent or received via HL7 Module from My webMethods. For more information about viewing documents, see the *webMethods Trading Networks User's Guide* for your release.

Viewing HL7 Version 2.x Messages

You can view the following types of information about your HL7 messages on the Transactions page in My webMethods:

- Sender and receiver of the HL7 message.
- Date the document was received.
- Processing status of the HL7 message.
- User status.
- Document type.
- Related documents (for example, the accept acknowledgment document).
- On the **Content** tab, you view the HL7 ER7 or XML Data content part that contains the actual HL7 message.
- On the **Activity Log** tab, you view the steps that the document goes through during inbound or outbound processing.

For steps to view documents, see the *webMethods Trading Networks User's Guide* for your release.

Trading Networks Processing Status and HL7 Version 2.x Messages

The following tables list the processing statuses that Trading Networks sets while processing HL7 version 2.x messages, along with their meanings. You can use these statuses as criteria for selecting the HL7 version 2.x messages that you want to view.

Status Values When Sending HL7 Messages

Processing Status	User Status	Meaning
IN PROGRESS	Persisted	Message is ready for transmission and is stored in the Trading Networks database.
DONE	SendMessage:sent	Message sent to the receiver successfully.
DONE	SendMessage:ack:ASync	The acknowledgement message is delivered to the specified host in Asynchronous mode.
DONE	SendMessage:ack:Sync	An acknowledgment message has been generated and sent to the MLLP server. The MLLP server in turn sends the message to the remote host from which the MLLP server received the message.
DONE W/ ERRORS	SendMessage:error	Message could not be send to the receiver. An error message is logged into the Trading Networks Activity Log stating the exact reason for the failure.
DONE W/ ERRORS	SendMessage:sent: Duplicate	Message sent to the receiver successfully but the sent message is a duplicate. Two messages are duplicated if they have the same Sender, Receiver, and Message Control ID. In this case HL7 Module creates a relationship between the message and all duplicate messages with a relationship `name = Duplicate`

Status Values When Receiving HL7 Messages

Processing Status	User Status	Meaning
IN PROGRESS	Persisted	Message has been received successfully by HL7 Module and is stored in the Trading Networks database.
DONE	ReceiveMessage:ack	If the message type is Acknowledgment, processing of the HL7 message is successful. HL7 Module relates the acknowledgment with the original message sent. If the original message is a duplicate, the module relates the acknowledgment with all messages duplicating the original and logs in a warning message in the Activity Log.

Processing Status	User Status	Meaning
DONE	ReceiveMessage:default	Message received and processed successfully.
DONE	ReceiveMessage: ErrorInMessage	HL7 Module received Acknowledgment message for a message not sent by HL7 Module.
DONE W/ ERRORS	ReceiveMessage: error	Message received but processing failed. An error message is logged into the Trading Networks Activity Log stating the exact reason for the failure.
DONE W/ ERRORS	ReceiveMessage: default:Duplicate	Message received and processed successfully, but the received message is a duplicate. Two messages are duplicated if they have the same Sender, Receiver, and Message Control ID. In this case HL7 Module creates a relationship between the message and all duplicate messages with a relationship `name = Duplicate`

8 Managing HL7 Code Tables

Overview

This section contains instructions about how to manage and use code tables. It also explains how to format the code table and code table values files.

Before You Manage Code Tables

You must configure the HL7 Module database before you can manage or use code tables. For more information about the configuration tasks, see "Configuring the HL7 Module Database" on page 32.

Managing Standard Code Tables

The code table and code table values files provided with HL7 Module contain the standard HL7 code tables and code table values that you use to populate the standard set of HL7 Module database tables. For more information about the default code table and code table values files, see "HL7 Code Tables" on page 15 and "Formatting the Code Tables and the Code Table Values Files" on page 104. The standard set of code tables is associated with the HL7 Module [Default] trading partner.

Selecting the Default Trading Partner

If you use only the standard set of code tables, you manage the standard code tables and code table values directly from the Manage Code Tables page in the HL7 Module user interface in Integration Server Administrator. The [Default] partner is the only available partner when customizing for Trading Networks partners is disabled. For information about how to enable customizing for trading partners, see "Customizing for a Trading Partner" on page 100.

If you have enabled customizing for one or more Trading Networks trading partners, you must first select the HL7 Module [Default] trading partner before you can manage the standard set of code tables. The [Default] trading partner is automatically selected when you click the left-hand navigation link to the Manage Code Tables page. You can also select the [Default] partner from the **Select Partner** field that becomes available after you enable customizing for Trading Networks partners for at least one trading partner.

To select the Default trading partner

1. Start Integration Server and Integration Server Administrator if they are not already running.

2. In Integration Server Administrator, go to **Solutions > HL7 Module > Manage Code Tables**.

3. In the **Select Partner** field on the HL7 Module Code Tables page, select **[Default]**.

   HL7 Module displays a table that lists the standard code tables available in the HL7 Module database.

Enabling Code Tables

You must enable a code table to perform code table validation. By default, all code tables are disabled.

To enable a code table

1. Start Integration Server and Integration Server Administrator if they are not already running.

2. In Integration Server Administrator, go to **Solutions > HL7 Module > Manage Code Tables**.

3. In the **Enable** column of the **Code Tables for Partner** ***Partner_Name*** table, click **No**for the code table that you want to enable for validation.

   The value changes to **Yes** to indicate that the code table has been enabled for validation.

Tip:
To enable validation for all code tables for a selected partner, click **Enable All Tables for Partner** ***Partner_Name*** on the HL7 Module Manage Code Tables page.

Disabling Code Tables

When you do not want to perform code table validation for a code table, you disable the code table.

To disable a code table

1. Start Integration Server and Integration Server Administrator if they are not already running.

2. In Integration Server Administrator, go to **Solutions > HL7 Module > Manage Code Tables**.

3. In the **Enable** column of the **Code Tables for Partner** ***Partner_Name*** table, click **Yes**for the code table that you want to disable.

   The value changes to **No** to indicate that the code table has been disabled.

Tip:
To disable for validation all tables for a selected partner, click **Disable All Tables for Partner** ***Partner_Name*** on the HL7 Module Manage Code Tables page.

Viewing Code Tables

You can view the code table values for a code table.

To view the values for a code table

1. Start Integration Server and Integration Server Administrator if they are not already running.
2. In Integration Server Administrator, go to **Solutions > HL7 Module > Manage Code Tables**.
3. In the **Code Tables for Partner *Partner_Name*** table, click the **View** icon for the code table that you want to view.

   HL7 Module displays the code table values in the code table for the selected trading partner.

Editing Code Tables

You can add or delete values from code tables.

Adding a Value

To add a value

1. Start Integration Server and Integration Server Administrator if they are not already running.
2. In Integration Server Administrator, go to **Solutions > HL7 Module > Manage Code Tables**.
3. In the **Code Tables for Partner *Partner_Name*** table, click the **Edit** icon for the code table that you want to edit.
4. On the Edit Table page, specify the following values:

Field	Description
Value	The code set value that you want to add.
Description	Optional. The description for the value.

5. Click **Add Value**.

   HL7 Module adds the specified value in the code table for the selected trading partner.

Deleting a Value

To delete a value

1. Start Integration Server and Integration Server Administrator if they are not already running.
2. In Integration Server Administrator, go to **Solutions > HL7 Module > Manage Code Tables**.
3. In the **Code Tables for Partner *Partner_Name*** table, click the **Edit** icon for the code table that you want to edit.
4. On the Edit Table page, click the **Delete** icon for the code set value that you want to delete.

   HL7 Module deletes the specified value from the code table for the selected trading partner.

Tip:
To delete all values in a code table, click **Delete All Values** on the Edit Table page.

Adding a Code Table

HL7 Module supports adding a code table to a single or multiple partner profiles. When you add a code table to a set of trading partners, HL7 Module:

- Adds the table in the code table set of each partner and displays a message with the number of code tables added for the number of selected partners.
- If the table already exists in one of the selected partners code table set, HL7 Module ignores the add operation for that partner and logs a warning message in the Integration Server log. In this case HL7 Module adds the table in the code table sets of the remaining selected partners.
- In all other cases, when HL7 Module fails to add a table in the code table set of one partner, it rolls back the add operation for all selected partners.

To add a code table

1. Start Integration Server and Integration Server Administrator if they are not already running.
2. In Integration Server Administrator, go to **Solutions > HL7 Module > Manage Code Tables**.
3. On the HL7 Module Manage Code Tables page, click **Add Code Table**.
4. On the Add Code Table page, specify the following values:

Field	Description
Table ID	The table ID number for the table that you want to add. For example, `0323`.
Table Name	The name of the table. For example, `Action Code`.

Field	Description
Enable	Enable code table validation for the table. Valid values: ■ **true** Enables validation. ■ **false** Disables validation. This is the default value.

5. In the **Available partners** box, select the names of the partners that you require. You must select at least one partner.

6. Click the right arrow under the **Available partners** box to move the selected partners into the **Selected partners** box.

   Note:
   If you want to remove a partner from the **Selected partners** box, select the partner name and click the left arrow under the **Selected partners** box. The partner will be moved back to the **Available partners** box.

7. Click **Add Table**.

   HL7 Module adds the code table to the selected trading partner code table set.

Deleting a Code Table

To delete a code table

1. Start Integration Server and Integration Server Administrator if they are not already running.

2. In Integration Server Administrator, go to **Solutions > HL7 Module > Manage Code Tables**.

3. On the HL7 Module Manage Code Tables page, click the **Delete** icon for the code table that you want to delete.

   HL7 Module deletes the code table from the selected trading partner code table set.

Managing Customized Code Tables

Customizing for a Trading Partner

To use customized code tables for a specific trading partner, you must enable customizing for that partner. You can also disable customizing for a specific trading partner.

Important:

When you disable customizing for a trading partner, all customized tables in the code table set for that trading partner revert to the standard set of code tables and you will lose all custom information in the code tables for that partner.

Enabling Customizing for a Trading Partner

To enable customizing for a trading partner

1. Start Integration Server and Integration Server Administrator if they are not already running.
2. In Integration Server Administrator, go to **Solutions > HL7 Module > Customize Code Tables**.
3. In the **Customize** column of the **List of partners from Trading Networks** table, click **No** for the trading partner for which you want to enable code table customizing.

   The value changes to **Yes** to indicate that code table customizing has been enabled for the trading partner.
4. To customize and manage the code tables for this specific trading partner, click the **Edit** icon.

   The Manage Code Tables page opens and the trading partner is automatically selected in the **Select Partner** field. You can customize partner specific code tables from the Manage Code Tables page as described in "Customizing Code Tables" on page 102.

   Tip:
   You can also use the left-hand navigation to navigate to the Manage Code Tables page, but in that case you must select the trading partner for which you want to customize code tables in the **Select Partner** field.

Note:
After you enable customizing for a trading partner, the values in each code table in that partner's code table set will follow the default until the respective code table is customized.

Disabling Customizing for a Trading Partner

To disable customizing for a trading partner

1. Start Integration Server and Integration Server Administrator if they are not already running.
2. In Integration Server Administrator, go to **Solutions > HL7 Module > Customize Code Tables**.
3. In the **Customize** column of the **List of partners from Trading Networks** table, click **Yes** for the trading partner for which you want to disable code table customizing.

   The value changes to **No** to indicate that code table customizing has been disabled for the trading partner.

Note:
After you disable customizing for a trading partner, you can no longer edit or customize the code tables for that partner. The trading partner's name will not be available in the list of available partners in the **Select Partner** field on the Manage Code Tables page.

Customizing Code Tables

Before you can customize code tables, you must enable customizing for the Trading Networks partner that you require. For information about how to enable customizing for a trading partner, see "Customizing for a Trading Partner" on page 100.

Customizing a Partner Specific Code Table

To customize partner specific code tables

1. Start Integration Server and Integration Server Administrator if they are not already running.

2. In Integration Server Administrator, go to **Solutions > HL7 Module > Manage Code Tables**.

3. In the **Select Partner** field on the HL7 Module Manage Code Tables page, select the name of the partner for which you want to customize code tables.

4. In the **Customize** column, click **No** for the code table that you want to customize.

   The value changes to **Yes** to indicate that you can customize the code table and use it as described in "Using Customized Tables" on page 103 on "Using Customized Tables" on page 103.

Restoring a Customized Code Table to Default Values

Important:
When you restore a customized code table to the standard set of code table values, all customized values in the table are deleted.

To restore a customized partner specific code table to default values

1. Start Integration Server and Integration Server Administrator if they are not already running.

2. In Integration Server Administrator, go to **Solutions > HL7 Module > Manage Code Tables**.

3. In the **Select Partner** field on the HL7 Module Manage Code Tables page, select the name of the partner for which you want to restore a customized code table to the default.

4. In the **Customize** column, click **Yes** for the code table that you want to restore.

The value changes to **No** to indicate that the customized code table is restored to the default values.

Important:
You cannot edit the code table when the value in the Customize column is set to **No**.

Using Customized Tables

You manage customized code tables for a specific trading partner in the same way you manage the standard code tables for the HL7 default partner. The following table lists the operations you can perform with customized partner specific code tables and where to find the procedure steps for each operation:

For the steps to...	See...
Enable for code table validation	"Enabling Code Tables" on page 97
Disable for code table validation	"Disabling Code Tables" on page 97
View the code table values	"Viewing Code Tables" on page 98
Add or delete code table values	"Editing Code Tables" on page 98
Add a code table	"Adding a Code Table" on page 99
Delete a code table	"Deleting a Code Table" on page 100

Uploading Code Table Values

HL7 Module allows you to load code table values for one or more Trading Networks partners directly from a file.

Important:
Before you load code tables values from a file, you must ensure that the file is located in the *Integration Server_directory* \WmHL7\data\codetables\upload directory and is in tab-separated value (TSV) format. For more information about the file requirements, see "Formatting the Code Tables and the Code Table Values Files" on page 104.

To upload code table values from a file

1. Start Integration Server and Integration Server Administrator if they are not already running.

2. In Integration Server Administrator, go to **Solutions > HL7 Module > Manage Code Tables**.

3. On the Manage Code Tables page, click**Upload Code Table Values**.

4. In the **Input file name** field on the Upload Code Table Values page, select the name of the input file from the list. For example, **sample.tsv**.

If the file list is empty, ensure that the input file is located in the correct directory. See the important note before this procedure for details.

5. In the **Available partners** box, select the names of the partners that you require. You must select at least one partner.

   Note:
   The **Available partners** box contains only the names of trading partners for which you have enabled customizing.

6. Click the right arrow under the **Available partners** box to move the selected partners into the **Selected partners** box.

   Note:
   If you want to remove a partner from the **Selected partners** box, select the partner name and click the left arrow under the **Selected partners** box. The partner will be moved back to the **Available partners** box.

7. Click **Load**.

   HL7 Module loads the code table values and displays the results from the load operation in the **Results** table.

The following table lists the errors that might occur during the load operation and how HL7 Module responds to the error:

Error occurs when...	How HL7 Module Responds to the Error
A code table value specified for a code table for a selected trading partner is already present.	Logs a warning message and ignores the value.
A table specified for a selected partner does not exist or is not customized.	Logs a warning message and ignores the value.
Any other error related to partner specific code tables occurs.	The whole operation rolls back. No partial updates performed.

Formatting the Code Tables and the Code Table Values Files

HL7 Module can load values only from code table and code table value files that are in tab-separated value (TSV) format. If you have code table value files in a different format, such as HTML and pdf files, you must convert each code table value file into a TSV file so that the code values can be added to the HL7 Module database. In addition to the format requirement, the code tables and the code table values file must be located in the *Integration Server_directory* \packages\WmHL7\data\codetables\upload directory.

Note:

The code table and code table values TSV files provided with HL7 Module contain a list of suggested values based on those specified by the HL7 standard. You can add or delete code tables and code table values from those two files as needed.

Formatting the Code Tables File

The code tables file has two tab-separated columns:

- The first column indicates the table ID number, for example **0002**.
- The second column gives the name of the table, for example **Marital Status**.

Each code table appears on a subsequent line. Following is an example of the required format for a code table file:

```
0000                    no table
0001                    Administrative Sex
0002                    Marital Status
0003                    Event type
0004                    Patient Class
0005                    Race
0006                    Religion
0007                    Admission Type
0008                    Acknowledgment code
0009                    Ambulatory Status
0010                    Physician ID
0017                    Transaction Type
0018                    Patient Type
0019                    Anesthesia Code
0021                    Bad Debt Agency Code
0022                    Billing Status
```

Formatting the Code Table Values File

The code table values file also has three tab-separated columns:

- The first column indicates the table ID number, for example `0002`.

- The second column has the code set value, for example `A`.
- The third column gives the description of the value, for example `Separated`.

Each code table value appears on a subsequent line. Following is an example of the required format for a code table values file:

0001	A	Ambiguous
0001	F	Female
0001	M	Male
0001	N	Not Applicable
0001	O	Other
0001	U	Unknown
0002	A	Separated
0002	B	Unmarried
0002	C	Common Law
0002	D	Divorced
0002	E	Legally Separated
0002	G	Living together
0002	I	Interlocutory
0002	M	Annulled
0002	N	Other
0002	O	Domestic Partner

9 Logging and Error Handling

Overview

The following sections describe webMethods HL7 Module error and exception handling, and message logging. A list of error codes and supporting information appears at the end of this chapter.

Handling Validation Errors

webMethods HL7 Module provides the wm.ip.hl7.service:validate service that verifies the HL7 version 2.x message as follows:

- **Syntax validation**. Validates the structure of the message including complex data types.
- **Semantic validation**. Validates the values of simple data types.
- **Code table validation.** Validates the codes in the messages against the codes table values.

  Note:
  If you do not want to perform code table validation of HL7 version 2.x messages, disable all code tables from the HL7 Module user interface in Integration Server Administrator. For information about how to do that, see "Managing HL7 Code Tables" on page 95.

The following table lists the type of validation that the validate service performs, the type of error it may encounter, and the response of HL7 Module to the error.

Note:
When any unexpected (that is, not listed in the following table) error occurs, HL7 Module stops processing the message and reports an error.

Type of validation	Error occurs when...	How HL7 Module Responds to the Error
Syntax	The required message header is invalid.	Logs an error message and stops processing the HL7 message.
	An invalid segment is found.	Logs an error message and stops processing the HL7 message.
	A required segment is absent.	Logs an error message and stops processing the HL7 message.

Type of validation	Error occurs when...	How HL7 Module Responds to the Error
	In the structure of a list/group, either an invalid segment is found, or a required segment is absent.	Logs an error message and stops processing the HL7 message.
	An extra segment is found at the end of the message.	Logs a warning message.
	An extra field is found at the end of the message.	Logs a warning message.
	An extra component is found at the end of the message.	Logs a warning message.
	An extra subcomponent is found at the end of the message.	Logs a warning message.
	A required field is absent.	Logs an error message, ignores the segment, and goes to the next segment.
Semantic	The pattern of simple data types DT, DTM, and TM is invalid.	Logs an error message, ignores the field, and goes to the next field.
	The length of the message in simple data types FT, GTS, ST, and TX is longer than the limit.	Logs a warning message and truncates the data.
	Verifies if the NM simple data type is numeric with a length of 16 digits.	Logs an error message, ignores the field, and goes to the next field.
	Verifies if the SI simple data type is a number between 0 and 9999.	Logs an error message, ignores the field, and goes to the next field.
Code Table	The code tables are not configured.	Logs a warning message.
	The required code table is disabled.	Logs a warning message.
	The required code table does not have any values.	Logs a warning message.
	The code tables values do not match.	Logs an error message, ignores the field, and goes to the next field.

HL7 Module Message Logging

HL7 Module uses the Integration Server logging mechanism to log messages. HL7 Module logs informational, warning, and error messages to the server log of the Integration Server. To view the server log, use the Integration Server Administrator. For steps to view and configure the server log, see the *webMethods Integration Server Administrator's Guide* for your release.

The messages that HL7 Module adds to the server log appear in the following format: WHL.000*n*.*nnnn*, where:

- WHL is the product code that indicates this message is issued by HL7 Module.
- 000*n*is the HL7 Module major error code, where *n* can take the following values:
  - 1 indicates that the error, warning or informational message is generated by HL7 Module to log a general error.
  - 2 indicates that the error, warning or informational message is generated during the serialization of the message; that is, when using the convertIDataToHL7 service to convert the HL7 data from the IData format into the HL7 ER7 or XML representation.
  - 3 indicates that the error, warning or informational message is generated during the deserialization of the message; that is, when using the convertHL7ToIData service to convert the HL7 representation into IData format.
  - 4 indicates that the error, warning or informational message is generated during the processing of the message in Trading Networks.
  - 5 indicates that the error, warning or informational message is generated during a code table operation.
- *nnnn* represents the error's minor code. For detailed descriptions of the HL7 Module error codes, see " HL7 Module Error Codes" on page 110.

The messages that HL7 Module adds to the server log for MLLP connections appear in the following format: ADA.0515.*nnnn*, where:

- ADA is the product code for the MLLP Adapter that manages MLLP connections.
- 515 indicates the MLLP Adapter major code.
- *nnnn* represents the error's minor code.

HL7 Module Error Codes

The following section lists the major and minor HL7 Module error codes and provides information on the error message, reason, and possible action for each error.

0001.1003 Invalid value found for parameter(s): parameter_name. Value: parameter_value

Explanation: Error. When interpreting the HL7 message separators, HL7 Module found an invalid value for the specified *parameter_name*.

Action: Ensure that valid values are specified for the input parameters. If all input parameters are valid, contact Software AG Global Support.

0001.1004 Input value *parameter_name:parameter_input_value* does not match with the value implied in the input message *implied_value*

Explanation: Error. The value of the *parameter_name* input parameter to the service and the value implied from the input message do not match. The error occurs either because the value of *parameter_name*, or the input HL7 version 2.x message are incorrect.

Action: Ensure that the input value for *parameter_name* is correct and that the input HL7 version 2.x message is appropriate.

0001.1005 Input value *parameter_name:parameter_input_value* does not match with input *messageValues*

Explanation: Error. The value of the *parameter_name* input parameter to the service and the input *messageValues* do not match. The error occurs either because the value of *parameter_name*, or the input document in *messageValues* are incorrect.

Action: Ensure that the input value for *parameter_name* is correct and that the input document in *messageValues* is appropriate.

0001.2018 Unsupported HL7 version: *version_number*

Explanation: Error. HL7 Module failed to run a service because an invalid value or an unsupported version was specified in the input parameters.

Action: Specify a valid value for the *messageVersion* parameter of the service. For complete information about the input parameters for each service, see " webMethods HL7 Module Services" on page 139.

0001.2019 Unknown HL7 message type [*message_type*] for: *scheme_ID*

Explanation: Error. The message type specified in the error message is not valid for a scheme with *scheme_ID* and *message_version*.

Action: Check if the message definition for the message type [*message_type*] with the specified scheme information is present. Check if an IS document type is generated for the message type [*message_type*] with the specified scheme information.

0001.2021 No JDBC pool configured for HL7 functional alias.

Explanation: Warning. The HL7 functional alias does not have an associated JDBC pool pointing to the HL7 datastore.

Action: Configure the HL7 Module database as described in "Configuring the HL7 Module Database" on page 32.

0001.2022 Error during reading or creating the HL7 functional alias.

Explanation: Error. HL7 Module could not create the functional alias required for managing code tables.

Action: Contact Software AG Global Support.

0001.2032 Field: *field_name* exceeded the maximum length for *field_type*. Data found: *field_value*

Explanation: Warning. The data for the FT, GTS, ST, or TX text fields of the HL7 version 2.x message exceeds the maximum length limit.

Action: Correct the HL7 version 2.x message if it is generated by HL7 Module. In all other cases, ignore the warning message.

0001.2033 Field: *field_name* did not match the expected format of the date field type *field_type*.Data found: *field_value*

Explanation: Error. The data for the DT, TM, and DTM date fields of the HL7 version 2.x message does not match the expected format.

Action: Ensure that the data in the DT, TM, and DTM fields is in the required format. For information about the required format for the values of those fields, see the HL7 standards documentation.

0001.2034 Field *field_name* is of type*field_type*, expecting a numeric. Data found: *field_value*

Explanation: Error. The data for the NM field of the HL7 version 2.x message is not in the expected format.

Action: Ensure that the value in the NM field of the HL7 version 2.x message is a valid decimal with a maximum length of 16 digits.

0001.2035 Field: *field_name* is of type *field_type*, expecting a non negative integer. Data found: *field_value*

Explanation: Error. The data for the SI field of the HL7 version 2.x message is not in the expected format.

Action: Ensure that the value in the SI field of the HL7 version 2.x message is between 1 and 9999.

0001.2036 Field: *field_name* is of type *field_type*, expecting a non negative integer with maximum length 4. Data found: *field_value*

Explanation: Error. The data for the SI field of the HL7 version 2.x message is not in the expected format.

Action: Ensure that the value in the SI field of the HL7 version 2.x message is with a maximum length of 4.

0001.2040 Error converting Document to XML. Document type name: *document_type_name*

Explanation: Error. When running the convertIDataToHL7 service, this error occurs during the conversion of an IS document into an HL7 XML message.

Action: For more information on this error, see the Integration Server Error log.

0001.2042 Error converting XML to Document. Document type name: *document_type_name*

Explanation: Error. When running the convertHL7ToIData service, this error occurs during the conversion of an HL7 XML message into an IS document.

Action: For more information on this error, see the Integration Server Error log.

0001.2043 Error occurred while trying to delete the node*node_name* under package *package_name*.Error:*error_text*

Explanation: Error. An attempt to delete an HL7 document type of a particular HL7 message version has failed.

Action: Ensure that you have all requisite access rights for the WmHL7DocTypes package. For information on how to control access to packages, see the *webMethods Integration Server Administrator's Guide* for your release.

0001.2051 Error during conversion of IData to HL7 for message with version: *version_number* type: *message_type*.Error:*error_text*

Explanation: Error. When running the convertIDataToHL7 service, this error occurs during the serialization of the specified HL7 version 2.x message.

Action: See the stack trace for this error message in the Integration Server Error log to establish the exact cause for the error.

0001.2054 Error during conversion of HL7 to IData for message with version: *version_number* type: *message_type*.Error: *error_text*

Explanation: Error. When running the convertHL7ToIData service, this error occurs during the deserialization of the specified HL7 version 2.x message.

Action: See the stack trace for this error message in the Integration Server Error log to establish the exact cause for the error.

0001.2055 Unrecognized input format for HL7 message of type message_type, version *version_number*. Expected: string, byte array, or InputStream. Found: *qualified_class_name_representing_object_type*.

Explanation: Error. The format of the message passed to the HL7 convertHL7ToIData service is not compatible with the required format.

Action: Pass the HL7 version 2.x message to the convertHL7ToIData service as either string, byte array, or InputStream.

0001.2056 Unrecognized input format for HL7 message. Expected: string, byte array, or InputStream but found: *message_format*

Explanation: Error. The value of the *message* input parameter of the HL7 Module validate service did not match the expected data types: string, byte array, or InputStream.

Action: Enter the correct value for the *message* parameter and retry. For details about the HL7 Module validate service, see wm.ip.hl7.service:validate.

0001.2057 Unsupported java charset encoding "*charset_encoding*".

Explanation: Error. The user-specified charset encoding is invalid.

Action: Ensure that valid java charset encoding is specified.

0001.5001 Error occurred while generating the Document types for version: *message_version*. *error_message_text*

Explanation: Error. Occurs when the wm.ip.hl7.utils:generateMessageScheme service fails to generate the IS document type for the HL7 message version specified in the *messageVersion*parameter.

Action: For more information on this error, see the Integration Server Error log.

0001.5003 Unable to create the required document interface (folder) structure for populating the IS Document types at *folder_path*.

Explanation: Error. HL7 Module failed to create the required folder structure for the specified IS Document type, using one of the Integration Server built-in services.

Action: Ensure that the Integration Server user account has the required credentials to create the document interface. If correcting the credentials does not resolve the error, contact Software AG Global Support.

0001.5005 Error occurred while creating the interface *namespace_folder*. Exception: *error_message*

Explanation: Error. HL7 Module failed to create the interface namespace folder in the WmHL7DocTypes package when generating the message scheme.

Action: Check if the user has sufficient rights to create nodes under the WmHL7DocTypes package.

0001.5008 Error occurred while retrieving partner details from message.

Explanation: Warning. The sender/receiver application field of the message header is not populated correctly or the configuration of the partner profile in Trading Networks is incorrect. If this error occurs, HL7 Module uses the default code tables for validation.

Action: Correct the fields and ensure that the partner profile in Trading Networks is configured correctly. Run the service again.

0001.5010 Unable to allocate memory for the received message.

Explanation: Error. HL7 Module tried to allocate a byte[] for the given input stream containing the HL7 message.

Action: Increase the heap space of the Integration Server on which HL7 Module is installed and restart Integration Server.

0001.6100 File messages.properties not found at message: *folder_location*.

Explanation: Error. The required HL7 messages.properties file was not found at the location specified in *folder_location*.

Action: Ensure that the messages.properties file exists at the location specified in *folder_location*. For more information, see "Generating Message Schemes" on page 33.

0001.6114 Error while generating message scheme for HL7 Message Scheme ID: [*scheme_ID*] HL7 Message Version:[*message_version*]. Exception: *exception_text*

Explanation: Error. HL7 Module encountered an error when generating the message scheme for the specified HL7 Message Scheme ID. More details about the error are specified in *exception_text*.

Action: For more information on this error, see the Integration Server Error log.

0001.6115 Input HL7 Message Scheme ID: *scheme_ID* is invalid. Error: *error_text*

Explanation: Error. The specified HL7 Scheme ID is not valid. More details about the error are specified in *error_text*.

Action: For more information on this error, see the Integration Server Error log.

0001.6119 Error while reading messages.properties file.

Explanation: Error. The error occurs when HL7 Module attempts to read the messages.properties file.

Action: Ensure that the file is accessible for reading.

0001.6121 Error retrieving scheme info for TPA *TPA_ID*. Exception: *error_text*

Explanation: Error. Error while loading the specified TPA with the specified inputs. More details about the error are specified in *error_text*.

Action: Ensure that the specified TPA exists, the TPA-related values are specified correctly, and the TPA state is set to 'agreed'. For more information on this error, see the Integration Server Error log.

0001.6122 Unable to delete the temporary file: *file_name*.

Explanation: HL7 Module creates temporary files when you invoke the wm.ip.hl7.utils:generateMessageScheme service to generate IS document types.

Action: After the schema has been generated, you can manually delete these temporary files.

0002.2000 Error during serialization process.

Explanation: Error. HL7 Module failed to serialize the HL7 version 2.x message.

Action: See the stack trace for this error message in the Integration Server Error log to establish the exact cause for the error.

0002.2010 Unable to serialize message for node *node_name* (*node_type*) at *node_path*. *error_reason*.

Explanation: Error. HL7 Module was not able to parse the message, because of validation issue at the specified *node_path*.

Action: Check the *error_reason* and correct the issue for the specified *node_name*.

0002.2015 Unsupported message part *message_part_name* found. Ignoring the data for the message part.

Explanation: Warning. HL7 Module found a message part that is not supported.

Action: Ensure that the message contains a valid message part.

0002.2028 Error creating HL7 XML writer. Error: *error_text*

Explanation: Error. Occurs when creating an instance of the XML Event Writer.

Action: See the Integration Server Error log for details. For Integration Server 7.1.3, ensure that the STAX implementation library (wstx-asl.jar) is located in the following directory: *IntegrationServer_directory*\packages\WmHL7\code\jars. For Integration Server 8.x (that ships with the WoodSToX XML-processor), ensure that the STAX implementation library is part of the Integration Server classpath.

0002.2029 Error processing XML Stream at location: *location_path*.

Explanation: Error. HL7 Module failed to parse the HL7 version 2.x message at the specified location.

Action: Correct the HL7 version 2.x message. See the Integration Server Error log for details.

0002.2030 I/O error during message serialization.

Explanation: Error. An I/O error occurs during the serialization of the XML encoded HL7 message.

Action: See the Integration Server Error log for more details.

0002.2031 Mismatch between expected message type:*message_type* and found message type: *message_type*

Explanation: Error. The message type in the HL7 version 2.x message does not match the message type of the *messageType* parameter of the parsing wm.ip.hl7.service:convertIDataToHL7 service.

Action: Ensure that the message type in the parsing service matches the value specified in the *messageType* input parameter. Retry the service.

0003.2000 Error during deserialization process.

Explanation: Error. HL7 Module failed to process the HL7 version 2.x message.

Action: See the stack trace for this error message in the Integration Server Error log to establish the exact cause for the error.

0003.2010 Unable to deserialize message for node *node_name* (*node_type*) at *node_path*. *error_reason*.

Explanation: Error. HL7 Module was not able to parse the message, due to validation issue at the specified *node_path*.

Action: Check the *error_reason* and correct the issue for the specified *node_name*.

0003.2014 Unable to determine the HL7 encoding. An HL7 message should start with MSH or <.

Explanation: Error. The input HL7 version 2.x message is incorrect, either because the MSH segment is missing, or the XML format of the HL7 message is not correct.

Action: Correct the HL7 version 2.x message and retry.

0003.2015 No '*body' element found in field: *field_name*. Ignoring data: *field_data*

Explanation: Error. The record *field_name* does not contain the *body element, required to set the data.

Action: Review the document type definition for the message, and ensure that the *field_name* is a leaf record and contains the element *body.

0003.2019 HL7 Segment separator not found. HL7 message will not be parsed.

Explanation: Error. HL7 Module cannot parse the HL7 version 2.x message, because the message does not contain any of the \r, \n, or \r\n segment separators.

Action: Correct the HL7 version 2.x message and retry.

0003.2021 Error during MSH segment processing.

Explanation: Error. HL7 Module failed to parse the HL7 version 2.x message because of an error in the MSH segment.

Action: See the Integration Server Error log for more details.

0003.2023 Mismatch between expected message type: *expected_message_type* and found message type: *actual_message_type*

Explanation: Error. The message type of the incoming HL7 version 2.x message does not match the user-specified version for the **convertHL7ToIData** service.

Action: Specify the correct message type in the **convertHL7ToIData** service input parameters.

0003.2024 Additional segment data found for message *message_type*. Unparsed segments: *unrecognized_segments_number*

Explanation: Warning. The HL7 version 2.x message contains one or more additional segments at the end of the message.

Action: HL7 Module ignores the additional data. Correct the HL7 version 2.x message where possible.

0003.2025 Additional subcomponent data found for component *field_name*. Data count: *unrecognized_subcomponents_number*

Explanation: Warning. The HL7 version 2.x message contains one or more additional subcomponents for the specified field at the end of the specified component.

Action: HL7 Module ignores the additional data. Correct the HL7 version 2.x message where possible.

0003.2026 Additional field data found for segment *segment_name*and will be ignored. Data count: *unrecognized_fields_number*

Explanation: Warning. Additional field data was found at the end of the specified segment.

Action: HL7 Module ignores the additional data. Correct the HL7 version 2.x message where possible.

0003.2027 Additional component data found for field *field_name*and will be ignored. Data count: *unrecognized_components_number*

Explanation: Warning. The HL7 version 2.x message contains one or more additional components at the end of the specified field.

Action: HL7 Module ignores the additional data. Correct the HL7 version 2.x message where possible.

0003.2028 Unable to find MSH segment in the message.

Explanation: Error. HL7 Module could not identify the Message Header in the message. The MSH segment was not found.

Action: Ensure that the Message Header is formed correctly as required in the HL7 standards.

0003.2029 Error processing XML Stream at location: *location_path*

Explanation: Error. HL7 Module failed to parse the HL7 version 2.x message at the specified location.

Action: Correct the HL7 version 2.x message. See the Integration Server Error log for details.

0003.2030 I/O error during message deserialization.

Explanation: Error. An I/O error occurs during the deserialization of the XML encoded HL7 message.

Action: See the Integration Server Error log for more details.

0003.2044 Error creating HL7 XML writer. Error: *error_text*

Explanation: Error. Occurs while creating an instance of the XML Event Writer.

Action: See the Integration Server Error log for more details. Ensure that you have STAX Implementation jar files (for example, sjsxp.jar) located in the *IntegrationServer_directory*\packages\WmHL7\code\jars directory.

0003.2046 Expected *part_name* Segment/Field not found. Remaining data in the message will not be parsed.

Explanation: Error. The required *part_name* field or segment is missing.

Action: Correct the HL7 version 2.x message and reprocess.

0003.2051 Error deserializing field: *field_name*. Error: *error_text*

Explanation: Error. HL7 Module failed to parse the specified *field_name* in the HL7 version 2.x message.

Action: Correct the HL7 version 2.x message. See the Integration Server Error log for details.

0003.2052 Unable to find message definition for deserializing Z segment: *segment_name*. Ignoring the segment: *segment_data*

Explanation: Error. A Z segment is present in the input HL7 version 2.x message but there is no corresponding definition for *segment_name* in the message scheme. HL7 Module ignores the segment data: *segment_data*.

Action: Add the definition for the Z segment with segment_name to the message scheme and retry after regenerating the message schemes in HL7 Module.

0003.2053 Missing data for mandatory field: *field_name*

Explanation: Error. HL7 Module could not find the specified mandatory field.

Action: Ensure that the mandatory field is specified.

0004.1003 NULL/empty value found for parameter: *parameter_name*

Explanation: Error. When sending the HL7 message, HL7 Module found an invalid value for the specified parameter.

Action: Ensure that valid values are specified for the input parameters. If all input parameters are valid, contact Software AG Global Support.

0004.1004 Invalid input found for parameter *parameter_name*. Value: *parameter_value*

Explanation: Error. The value of the input parameter to the service and the value implied from the input message do not match.

Action: Ensure that the input value for the specified parameter is correct.

0004.1012 Error occurred while parsing the pipeline content. Exception: *error_message*

Explanation: Error. An error occurred when parsing an HL7 version 2.x message received as a document by the HL7 Module receive service.

Action: Verify the input node that is passed in the pipeline.

0004.1014 Error extracting request data. Exception: *error_text*

Explanation: Error. HL7 Module has encountered an error during the processing of an incoming request.

Action: See the Integration Server Error log for more details.

0004.1016 No HL7 message found in E-mail.

Explanation: Error. HL7 Module did not find an HL7 version 2.x message in an e-mail message received using the E-mail transport.

Action: Ensure that the HL7 version 2.x message is sent as an attachment and the name of the attached file contains `hl7data`.

0004.1020 Error sending response code 'Internal Error'. Exception: *error_text*

Explanation: Error. HL7 Module encountered an error while sending the HTTP response code 500.

Action: See the Integration Server Error log for more details.

0004.1021 Protocol "*transport_protocol*" specified by the user/auto selected by system is invalid. Set valid protocol explicitly in input or configure preferred protocol in the receivers TN profile.

Explanation: Error. HL7 Module found that the specified transport protocol is incorrect or the preferred protocol for the receiver is either not set or not supported.

Action: Ensure that the protocol name passed to the service is valid and supported by HL7 Module. If the protocol is not passed to the service, ensure that the preferred protocol in the receiver Trading Networks profile is set and is one of the transports supported by HL7 Module.

0004.1031 Error retrieving internal IDs for ID: *external_ID* IDType: *ID_Type*. Exception: *error_text*

Explanation: Error. HL7 Module can not find the internal Trading Networks profile IDs for the specified external ID.

Action: See the Integration Server Error log for more details about the error. Ensure that you have set up trading partner profiles for the specified external ID as described in "Defining Trading Networks Profiles" on page 68.

0004.1032 Internal ID not found for *party* message ID: *message_ID*

Explanation: Error. The incoming HL7 version 2.x message with *message_ID* for *party*, that is Sender or Receiver, that does not have corresponding trading partner profiles in Trading Networks.

Action: Set up trading partner profiles for the sender and receiver involved in the message exchange.

0004.1033 No partner profile found for corporation name: *corporation_name* and organizational unit: *unit_name*, and will treat the partner as Unknown.

Explanation: Warning. HL7 Module did not find a partner profile for the specified *corporation_name* and *unit_name* combination and will use the default Unknown profile.

Action: Ensure that you have a unique trading partner represented by the *corporation_name* and *unit_name* combination.

0004.1034 Multiple partner profiles found for corporation name: *corporation_name* and organizational unit: *unit_name*, will use partner ID *partner_ID*

Explanation: Warning. HL7 Module found multiple partner profiles for the specified *corporation_name* and *unit_name* combination and will use the first found *partner_ID*.

Action: Ensure that you have a unique trading partner represented by the *corporation_name* and *unit_name* combination.

0004.1050 Empty HL7 content data found in BizDocEnvelope with Internal ID: *BizDocEnvelope_Internal_ID* (Document ID: *BizDocEnvelope_Document_ID*)

Explanation: Error. HL7 Module was expecting a BizDocEnvelope with content part named "HL7 ER7 Data", but the content part is missing.

Action: Ensure that the specified BizDocEnvelope is valid and it has a content part named "HL7 ER7 Data" containing the HL7 message.

0004.1051 Error occurred during MLLP send operation. Exception: *error_text*

Explanation: Error. HL7 Module encountered an error when sending an HL7 version 2.x message via the MLLP protocol. The error occurred due to a network failure or an I/O Error because either the server, or the machine went down.

Action: Ensure that the destination host is reachable and there is no network outage.

0004.1052 Error occurred during MLLP client socket operation. Exception: *error_message*

Explanation: Error. A network error occurred while sending an HL7 version 2.x message via the MLLP port. The error occurred due to a network failure or an I/O Error because either the server, or the machine went down.

Action: Check if the network is enabled. Check the firewall settings. Ensure that the destination host is reachable and there is no network outage.

0004.1053 HL7 MLLP Client unable to get end-point information for profile: *profile_name*

Explanation: Error. HL7 Module can not find the details required to establish connection to the MLLP listener in the trading partner profile.

Action: Ensure that the trading partner profile is set up correctly as described in "Defining Trading Networks Profiles" on page 68. See the Integration Server Error log for more details.

0004.1054 HL7 MLLP Client unable to connect to *hostname@portnumber*. Timed out after *timeout_period* milliseconds.

Explanation: Error. The error occurred while connecting to *hostname@portnumber*.

Action: Check if *hostname@portnumber* is available. Check the firewall settings.

0004.1060 Found invalid charset encoding *charset_encoding* in the incoming protocol header. Using encoding from TPA.

Explanation: Warning. The charset encoding specified in the incoming message header is not a valid Java charset encoding.

Action: Ensure that the client/HL7 partner who is sending a message has set the correct charset encoding.

0004.1100 The specified encoding in the input is invalid or not supported in Java. HL7 Module will use the default character encoding as "*character_encoding*"

Explanation: Warning. The charset encoding specified in the default "HL7TPA" HL7 TPA is invalid or not available.

Action: Ensure that a TPA named "HL7TPA" is created for Sender and Receiver "Unknown", and that the state of that TPA is set to "agreed". Also ensure that the "charsetEncoding" property is set in the HL7Configuration parameter in the TPA document.

0004.2051 Error while registering default HL7 TN document types. Exception: *error_text*

Explanation: Error. HL7 Module encountered an error when registering the default HL7 BizDocEnvelope types with Trading Networks during startup.

Action: See the Integration Server Error log for more details.

0004.2052 Error while registering default HL7 TN processing rules. Exception: *error_text*

Explanation: Error. HL7 Module encountered an error when registering the default HL7 processing rules with Trading Networks during startup.

Action: See the Integration Server Error log for more details.

0004.2053 Error while registering default HL7 BizDocEnvelope attributes. Exception: *error_text*

Explanation: Error. HL7 Module encountered an error when registering the default HL7 document attributes with Trading Networks during startup.

Action: See the Integration Server Error log for more details.

0004.2054 Error while registering MLLP transport service with Trading Networks. Exception: *error_text*

Explanation: Error. HL7 Module encountered an error while registering the HL7 MLLP delivery service during startup.

Action: See the Integration Server Error log for more details.

0004.4001 Error occurred while persisting the BizDocEnvelope. Error: *error_text*

Explanation: Error. Saving the BizDocEnvelope for an HL7 version 2.x message in the Trading Networks database has failed.

Action: See the Integration Server Error log for more details. Ensure that the Trading Networks database is set up and the connection details properly configured. For information about configuring the Trading Networks database, see the *webMethods Trading Networks Administrator's Guide* for your release.

0004.5001 Cannot process the incoming HL7 message. Received message is invalid HL7 message.

Explanation: Error. The received HL7 message is not valid as per the HL7 specification.

Action: Ensure that all required fields and components are present in the HL7 message with valid data.

0004.5003 Error generating acknowledgment receipt for message *message_ID*. Error: *error_text*

Explanation: Error. HL7 Module has encountered an error while generating an Acknowledgment message for an HL7 version 2.x message with *message_ID*.

Action: See the Integration Server Error log for more details.

0004.5005 Protocol to send Acknowledgment messages not found. Acknowledgment will not be sent for message *message_ID*

Explanation: Error. HL7 Module could not find a valid Trading Networks delivery method to send an Acknowledgement message.

Action: Specify a valid transport protocol in the TPA for the message exchange. Alternatively, set a Preferred Protocol in the Trading Networks partner profile as described in "Configuring Communication Protocols" on page 42.

0004.5007 Error dispatching acknowledgment for message *message_ID*. Error: *error_message*

Explanation: Error. The error occurred while sending the acknowledgement message.

Action: See the Integration Server Error log for more details.

0004.5008 Could not send the message.

Explanation: Error. Occurred while sending the HL7 message to the specified recipient.

Action: Ensure that the HL7 message is valid and the transport protocol is configured in the receiver partner profile, set explicitly in the send service, or a preferred valid delivery protocol is set in the TPA.

0004.5010 Error occurred while setting an HTTP response code.

Explanation: Error. HL7 Module uses the pub.flow:setResponseCode service to set the response code for a received HTTP request after processing the request. Occurs when there is an error setting the HTTP response code.

Action: Contact Software AG Global Support.

0004.5012 "ACK" message generation failed.

Explanation: Error. Occurs when HL7 Module fails to generate the Acknowledgement message because an IS document is missing or any runtime error occurs while constructing an acknowledgement message.

Action: Ensure that an HL7 version specific IS document is generated and available.

0004.5013 Preferred protocol is not set for the receiver profile "*Receiver_ID*".

Explanation: Error. The preferred transport protocol is not set in the Trading Networks receiver partner profile.

Action: Ensure that a valid preferred protocol value is set for all receiver's Trading Networks profiles.

0004.5015 Invalid receiver ID: *receiver_ID* type: *receiver_ID_type*. No such partner profile exists in trading network.

Explanation: Error. This exception is thrown when trying to generate a BizDoc but the Receiver details specified in the enclosed HL7 message are invalid, that is they do not exist in the trading network profiles.

Action: Ensure that a partner profile entry exists in Trading Networks for the Receiver ID and Receiver ID type combination.

0004.6001 HL7 MLLP listener thread started up at *port_number* , but listener has invalid state. Listener STATE: *status_value*. Terminating listener thread.

Explanation: Warning. The specified MLLP listener failed to initialize.

Action: See the Integration Server Error log for more details. Contact Software AG Global Support.

0004.6003 Error starting to listen: *port_number*

Explanation: Error. The specified port number is already in use.

Action: Specify a different port number and retry.

0004.6004 HL7 MLLP listener failed to open non-blocking listener port *port_number*. Listener stopped.

Explanation: Error. The MLLP listener failed to start.

Action: See the Integration Server Error log for more details.

0004.6007 I/O error during HL7 MLLP read/write operation at *listener_name*. Connection: *client_information*

Explanation: Error. An input/output error occurred while reading or writing data from/to the client connection.

Action: See the Integration Server Error log for more details. Check for network failure. Contact Software AG Global Support.

0004.6011 I/O error occurred while trying to use the channel for connection: *socket_connection*

Explanation: Error. An input/output error occurred while writing data from the MLLP client connection.

Action: See the Integration Server Error log for more details.

0004.6012 Exception for connection: *client_information*

Explanation: Error. An exception occurred while the MLLP listener was receiving data from a client.

Action: See the Integration Server Error log for more details. Check for network failure. Contact Software AG Global Support.

0004.6013 Error during HL7 MLLP listener (*listener_name*) execution: *error_text*

Explanation: Error. An exception occurred while the specified MLLP listener was running.

Action: See the Integration Server Error log for more details about the exact nature of the error.

0004.6014 Error during HL7 MLLP listener (*listener_name*) cleanup. Exception: *exception_text*

Explanation: Error. Occurred while shutting down or disabling the MLLP listener when input and output operations are in progress.

Action: Before disabling the MLLP listener, make sure that no I/O operations are in progress by running the HL7 Module MLLP listener status service. For more information about the service, see the webMethods HL7 Module documentation.

0004.6019 HL7 MLLP listener startup request at port *port_number* ignored. Listener status: *listener_status*

Explanation: Warning. You attempted to start an MLLP listener that is not stopped.

Action: Check the status of the listener.

0004.6020 HL7 MLLP listener (*listener_details*) shutdown request ignored. Listener status: *listener_status*

Explanation: Warning. The listener is not running.

Action: Check the status of the listener.

0004.6021 HL7 MLLP listener ignored message from connection: *client_name*. Invalid protocol data found.

Explanation: Warning. The MLLP listener ignored an HL7 version 2.x message because it contained data that did not conform to the MLLP standard.

Action: Ensure that the HL7 version 2.x message send via the MLLP transport is a valid message.

0004.6029 HL7 MLLP listener task: *task_name* execution encountered error: *error_text*

Explanation: Error. An error occurred during the execution of the HL7 Module receive service in the *task_name* listener task.

Action: See the Integration Server Error log for more details.

0004.6031 HL7 MLLP listener cannot be added in package *package_name*. Set the package dependency of *package_name* to "WmHL7" package and retry.

Explanation: Error. The MLLP listener is created in a package with *package_name* that is not dependent on the WmHL7 package.

Action: Set the package dependency of the *package_name* package to the WmHL7 package.

0004.6033 Could not initialize MLLP listener. Invalid port specified: ***port_number***

Explanation: Error. The specified port number is incorrect.

Action: Specify a valid value for the port number. Valid values for port numbers range from 1 to 65536. Retry.

0004.7000 Could not find TPA for message ***message_ID*** **Receiver ID:** ***receiver_ID*** **Sender ID:** ***sender_ID*** **Agreement ID:** ***agreement_ID***

Explanation: Error. HL7 Module could not find a trading partner agreement associated with the HL7 version 2.x message with the specified ID.

Action: Create a TPA between the trading partners that take part in the message exchange and set the appropriate Agreement ID for the TPA, as described in "Defining HL7 Trading Partner Agreements" on page 71.

0004.7001 Error while creating a TPA for Sender ID: ***sender_ID*** **Receiver ID:** ***receiver_ID*** **Agreement ID:** ***agreement_ID***

Explanation: Error. HL7 Module encountered an error when creating the trading partner agreement between trading partners ***sender_ID*** **and** ***receiver_ID*****.**

Action: See the Integration Server Error log for more details.

0004.7002 Error while getting Delivery Method of the Sender. Error: ***error_text***

Explanation: Error. HL7 Module could not find the Trading Networks delivery method of the sending partner when replying to an incoming message on the receiver side.

Action: Set up the required Trading Networks delivery method details of the sending trading partner as described in "Configuring Communication Protocols" on page 42. For more details about the error, see the Integration Server Error log.

0004.7003 Invalid TPA state found for sender ID: ***sender_ID*****, receiver ID:** ***receiver_ID*****, agreement ID:** ***agreement_ID*****. Expected: agreed Found:** ***agreement_status***

Explanation: Error. HL7 Module found the required trading partner agreement but its status is not `Agreed`.

Action: Set the TPA status to `Agreed`.

0004.7004 Error occurred while loading a TPA for Sender ID: ***sender_ID*** **and Receiver ID:** ***receiver_ID*****, using charset encoding as** ***charset_encoding***

Explanation: Warning. Unable to load the TPA or a TPA exists but it is not in agreed state.

Action: Ensure that an HL7 Module TPA exists between the specified Sender and Receiver and a valid charset encoding is set. If a partner specific HL7 Module TPA does not exist, the module uses the default HL7TPA.

0004.7006 Invalid encoding "***charset_encoding*****" found for sender ID:** ***sender_ID*****, receiver ID:** ***receiver_ID*****, agreement ID:** ***agreement_ID*****. Using charset encoding** ***charset_encoding***

Explanation: Warning. The charset encoding specified in the TPA is invalid.

Action: Ensure that the charsetEncoding parameter in the TPA is set and the encoding specified is a valid charset encoding

0005.1004 Invalid value found for parameter in method: ***method_name***

Explanation: Error. The expected input for the specified methods is not present.

Action: See the Integration Server Error log for more details and provide the required inputs.

0005.2000 Error in getting connection.

Explanation: Error. HL7 Module could not get a connection from the datastore.

Action: Ensure that the JDBC connection pool is set up properly. See the Integration Server Error log for more details.

0005.2001 Error in closing statement.

Explanation: Error. HL7 Module could not close an SQL statement after execution.

Action: See the Integration Server Error log for more details.

0005.2002 Error in closing connection.

Explanation: Error. HL7 Module could not close a connection after execution.

Action: See the Integration Server Error log for more details.

0005.3000 Line does not match expected format in code table data file and will be ignored: ***line_text***

Explanation: Warning. A line in the code table TSV file does not match the following format: ***table_id*****TAB*****table_value*****TAB*****table_value_description*****. HL7 Module ignores the line in the code table file.**

Action: Re-enter the values in the line in the proper format. For information about the format of the TSV file, see "Formatting the Code Tables File" on page 105.

0005.3001 Error reading from code table file.

Explanation: Error. HL7 Module encountered an input/output error when reading from the code table file.

Action: Ensure that the file is present at the required location when running the loadData service. For information about the service and the file location, see wm.ip.hl7.codetables:loadData.

0005.3002 Line does not match expected format in code table values data file and will be ignored: ***line_text***

Explanation: Warning. A line in the code table values TSV file does not match the following format: ***table_id*****TAB*****table_value*****TAB*****table_value_description*****. HL7 Module ignores the line in the code table values file.**

Action: Re-enter the values in the line in the proper format. For information about the format of the TSV file, see "Formatting the Code Tables File" on page 105.

0005.3003 Error reading from code table values file.

Explanation: Error. HL7 Module encountered an input/output error when reading from the code table values file.

Action: Ensure that the file is present at the required location when running the **loadData** service. For information about the service and the file location, see wm.ip.hl7.codetables:loadData.

0005.3004 Configuration file *file_name* not found.

Explanation: Error. HL7 Module can not find the code tables TSV file.

Action: Ensure that the file is present in the *Integration Server_directory* \packages\HL7\data\codetables directory.

0005.3005 Unable to close configuration file.

Explanation: Error. HL7 Module could not close the TSV file after processing.

Action: See the Integration Server Error log for more details.

0005.3006 Error reading line from database script file.

Explanation: Error. During execution of the **wm.ip.hl7.codetables:runScripts** service, HL7 Module encountered an input/output error when reading from the database script file.

Action: Ensure that the file is present at the required location when running the **runScripts** service. For information about the service and the file location, see wm.ip.hl7.codetables:runScripts.

0005.3007 Invalid Table ID: *table_ID*. Code table ID should be an integer between 1 and 9999.

Explanation: Error. The code table ID is invalid for the required code table operation.

Action: Enter the correct table ID and retry.

0005.3008 Code table name cannot be null.

Explanation: Error. Inserting a code table has failed because the value for the code table name was null.

Action: Enter the correct value for the code table name and retry.

0005.3011 Code table value cannot be null.

Explanation: Error. When inserting a code table, the code table value was null.

Action: Enter the correct value for the code table name and retry.

0005.3011 Code table value cannot be null.

Action: Enter the correct code table value and retry.

0005.3014 Code table ID cannot be null.

Explanation: Error. Inserting a code table has failed because the value for the code table ID was null.

Action: Enter the correct code table ID and retry.

0005.4000 Functional alias has not yet been set up.

Explanation: Error. The HL7 functional alias has not been configured.

Action: Configure the HL7 functional alias as described in "Configuring the HL7 Module Database" on page 32.

0005.4001 SQL error while retrieving the code tables.

Explanation: Error. HL7 Module encountered an SQL error when retrieving the code tables from the HL7 Module database.

Action: See the Integration Server Error log for more details.

0005.4002 SQL error while retrieving the state of the HL7 Module database.

Explanation: Error. HL7 Module encountered an SQL error when attempting to retrieve the current state of the HL7 Module database.

Action: See the Integration Server Error log for more details.

0005.4003 SQL error while updating the state of the HL7 Module database.

Explanation: Error. HL7 Module encountered an SQL error when attempting to update the state of the HL7 Module database.

Action: See the Integration Server Error log for more details.

0005.4004 SQL error while retrieving the code table values.

Explanation: Error. HL7 Module encountered an SQL error when attempting to retrieve the code table values from the HL7 Module database.

Action: See the Integration Server Error log for more details.

0005.4005 SQL error while inserting code tables.

Explanation: Error. HL7 Module encountered an SQL error when attempting to insert the code tables in the HL7 Module database.

Action: See the Integration Server Error log for more details.

0005.4006 SQL error while deleting code tables.

Explanation: Error. HL7 Module encountered an SQL error when attempting to delete the code tables in the HL7 Module database.

Action: See the Integration Server Error log for more details.

0005.4007 SQL error while inserting code table values.

Explanation: Error. HL7 Module encountered an SQL error when attempting to insert the code table values in the HL7 Module database.

Action: See the Integration Server Error log for more details.

0005.4008 SQL error while deleting code table values.

Explanation: Error. HL7 Module encountered an SQL error when attempting to delete the code table values in the HL7 Module database.

Action: See the Integration Server Error log for more details.

0005.4009 SQL error while performing rollback.

Explanation: Error. HL7 Module encountered an SQL error when attempting to perform a rollback of the HL7 Module database.

Action: See the Integration Server Error log for more details.

0005.4010 Unable to rollback because the connection was null or was closed.

Explanation: Error. HL7 Module encountered an error when attempting to perform a rollback of the HL7 Module database because the connection was either null, or closed.

Action: See the Integration Server Error log for more details.

0005.4011 SQL error while running the database scripts.

Explanation: Error. HL7 Module encountered an SQL error when running the database scripts.

Action: See the Integration Server Error log for more details.

0005.4012 Data error in inserting values for code tables.

Explanation: Error. HL7 Module failed to insert the data for code tables in the HL7 Module database.

Action: Retry and if the error persists, contact Software AG Global Support.

0005.4013 Data error in deleting code tables.

Explanation: Error. HL7 Module failed to delete the data for code tables in the HL7 Module database.

Action: Retry and if the error persists, contact Software AG Global Support.

0005.4015 Data error in deleting code table values.

Explanation: Error. HL7 Module failed to delete the data for code table values in the HL7 Module database.

Action: Retry and if the error persists, contact Software AG Global Support.

0005.4016 Error encountered when updating the database state.

Explanation: Error. HL7 Module encountered an error when attempting to update the HL7 Module database, because the code tables and code table values data in the HL7 Module database is corrupted.

Action: Run the runScripts service in `drop` mode to clean up the database tables. After cleaning the database tables, run the runScripts service in `create` mode to re-create the database tables required by the HL7 Module database. For details about the service, see wm.ip.hl7.codetables:runScripts.

0005.4017 SQL error while trying to set the enabled/disabled flag for code tables.

Explanation: Error. HL7 Module encountered an SQL error when attempting to set the enabled or disabled flag for code tables.

Action: See the Integration Server Error log for more details.

0005.4018 Data error in enabling or disabling code tables.

Explanation: Error. HL7 Module failed to update the code tables data in the HL7 Module database.

Action: Retry and if the error persists, contact Software AG Global Support.

0005.4020 SQL error while retrieving partner details for the partner profile *partner_name.*

Explanation: Error. HL7 Module encountered an SQL error when attempting to retrieve partner details from the specified trading partner profile.

Action: See the Integration Server Error log for more details.

0005.4021 SQL error while customizing partner profile *partner_name.*

Explanation: Error. HL7 Module encountered an SQL error when attempting to customize a trading partner profile.

Action: See the Integration Server Error log for more details.

0005.4022 SQL error while customizing the code table: *code_table_ID* for the internal partner identifier *internal_partner_ID.*

Explanation: Error. HL7 Module encountered an SQL error when attempting to customize a code table for a specific trading partner.

Action: See the Integration Server Error log for more details.

0005.4023 SQL error while removing customizing for the internal partner identifier *internal_partner_ID.*

Explanation: Error. HL7 Module encountered an SQL error when attempting to remove customizing of a partner.

Action: See the Integration Server Error log for more details.

0005.4024 SQL error while removing the custom table: *code_table_ID* for the internal partner identifier *internal_partner_ID*.

Explanation: Error. HL7 Module encountered an SQL error when attempting to remove a customized table for the specified trading partner.

Action: See the Integration Server Error log for more details.

0005.4025 SQL error while retrieving the list of customized partners.

Explanation: Error. HL7 Module encountered an SQL error when attempting to retrieve the list of customized partners.

Action: See the Integration Server Error log for more details.

0005.4026 SQL error while retrieving the list of customized tables for partner profile *partner_name*.

Explanation: Error. HL7 Module encountered an SQL error when attempting to retrieve the list of customized tables for the specified trading partner profile.

Action: See the Integration Server Error log for more details.

0005.4027 SQL error while checking if the partner/table is customized.

Explanation: Error. HL7 Module encountered an SQL error when attempting to verify if customizing is enabled for a partner or a table is customized.

Action: See the Integration Server Error log for more details.

0005.5000 Database parameter value was not db2, oracle or sql.

Explanation: Error. The value for the *database* parameter of the runScripts service was other than `db2`, `oracle`, or `sql`.

Action: Enter the correct value for the *database* parameter of the runScripts service and retry. For details about the service, see wm.ip.hl7.codetables:runScripts.

0005.5001 Mode parameter value was not create, drop or cleanup.

Explanation: Error. The value for the *mode* parameter of the runScripts service was other than `create`, `drop`, or `cleanup`.

Action: Enter the correct value for the *mode* parameter of the runScripts service and retry. For details about the service, see wm.ip.hl7.codetables:runScripts.

0005.5002 Could not insert [*table_ID*] for partner: *partner_name* because the code table is already present.

Explanation: Warning. HL7 Module did not add a table to the *partner_name* partner because the table already exists in the specified partner code table set.

Action: None.

0005.5003 Could not insert [*table_ID*, *value*] for partner: *partner_name* because the code table is not present or it is not customized.

Explanation: Warning. HL7 Module failed to insert a code table value in the specified table for the selected *partner_name*. The error occurred either because the code table is not present in the HL7 Module database component, or the table is not enabled for customizing.

Action: Ensure that the code table is present in the HL7 Module database component and that the table is enabled for customizing.

0005.5004 Could not insert [*table_ID*, *value*] for partner: *partner_name* because the code table value is already present.

Explanation: Warning. HL7 Module did not insert the code table value in the specified table for the selected *partner_name* because the code table value is already present.

Action: None.

0005.6000 Code tables not configured. No validation was performed for Field at*field_path*

Explanation: Warning. HL7 Module is not configured to manage code tables and it does not perform code table validation.

Action: If you have not configured HL7 Module to manage code tables deliberately, ignore this message. In all other cases, configure HL7 Module to manage tables as described in "Configuring the HL7 Module Database" on page 32 and "Managing HL7 Code Tables" on page 95.

0005.6001 Code table *code_table_ID* was disabled. No validation was performed for Field at*field_path*

Explanation: Warning. The code table with the specified *code_table_ID* is disabled and HL7 Module does not perform code table validation.

Action: If you disabled the code table with *code_table_ID*deliberately, ignore this message. In all other cases, enable the code table as described in "Enabling Code Tables" on page 97.

0005.6002 Values for code table *code_table_ID* were empty. No validation was performed for Field at*field_path*

Explanation: Warning. The code table with the specified *code_table_ID* did not have any code table values and HL7 Module could not perform code table validation.

Action: If you left the code table with *code_table_ID* empty deliberately, ignore this message. In all other cases, insert code table values in the code table with *code_table_ID* as described in "Configuring the HL7 Module Database" on page 32 and "Managing HL7 Code Tables" on page 95.

0005.6003 Code table validation failed for Field:*field_name*.Data found: *field_value*.Table ID: *table_ID*

Explanation: Error. The value in the field with *field_name* does not match one of the predefined values in the code tables.

Action: If the field value in *field_name* is wrong, enter the correct value in the specified field. If the field value in *field_name* is correct, add it to the code table values.

0005.6500 Cannot upload code table values because the HL7 Module database is not configured.

Explanation: Error. HL7 Module failed to upload the code table values because the HL7 Module database is not configured.

Action: See "Configuring the HL7 Module Database" on page 32 for information about how to configure the HL7 Module database.

10 Using HL7 Module In a Clustered Environment

- "What is webMethods Integration Server Clustering?" on page 136
- "Requirements for Each Integration Server in a Cluster" on page 136
- "Considerations When Installing HL7 Module Packages" on page 137

What is webMethods Integration Server Clustering?

Clustering is an advanced feature of the webMethods product suite that substantially extends the reliability, availability, and scalability of webMethods Integration Server. Clustering accomplishes this by providing the infrastructure and tools to deploy multiple Integration Servers as if they were a single virtual server and to deliver applications that leverage that architecture. Because this activity is transparent to the client, clustering makes multiple servers look and behave as one. For details on webMethods Integration Server clustering, see the *webMethods Integration Server Clustering Guide* for your release.

Integration Server 8.2 SP2 and higher supports the caching and clustering functionality provided by Terracotta. Caching and clustering are configured at the Integration Server level and HL7 Module uses the caching mechanism that is enabled on Integration Server. When Integration Server uses Terracotta as its caching provider, HL7 Module does not explicitly implement any clustering or caching beyond what is already provided by Integration Server.

Note:webMethods Integration Server clustering redirects HTTP and HTTPS requests, but does not redirect SMTP requests.

Requirements for Each Integration Server in a Cluster

The following table describes the requirements of each Integration Server in a given cluster:

All Integration Servers in a given cluster must have identical...	For example...
Integration Server versions	One Integration Server in the cluster cannot be version 7.1 and another Integration Server in the cluster be version 7.1.2 - all servers must be the same version, with the same fixes (updates) and service packs applied.
HL7 Module packages	All HL7 Module packages on one Integration Server should be replicated to all other Integration Servers in the cluster.
HL7 Module versions	All HL7 Modules must be the same version, with the same fixes (updates) and service packs applied.
HL7 Module services	If you configure a specific HL7 Module service, this same service must appear on all servers in the cluster so that any Integration Server in the cluster can handle the request identically. If you allow different Integration Servers to contain different services, you might not derive the full benefits of clustering. For example, if a client requests a service that resides on only one server, and that server

All Integration Servers in a given cluster must have identical...	For example...
	is unavailable, the request cannot be successfully redirected to another server.

Considerations When Installing HL7 Module Packages

For each Integration Server in the cluster, use the standard HL7 Module installation procedures for each machine, as described in "Installing webMethods HL7 Module 7.1 SP1" on page 24.

A webMethods HL7 Module Services

Overview

This appendix describes the built-in services and other elements, provided with the WmHL7 and WmHL7DocTypes packages of webMethods HL7 Module.

WmHL7 Package

The WmHL7 package contains public services used to implement and support the HL7-compliant functionality of HL7 Module. The following elements are available in this package:

Folder	Description
wm.ip.hl7.codetables	Contains services that you use to manage code tables.
wm.ip.hl7.service	Contains services that construct, parse and validate HL7 version 2.x messages.
wm.ip.hl7.tn	Contains built-in services that enhance Trading Networks to function as a gateway for HL7 message exchange.
wm.ip.hl7.utils	Contains utility services to generate and delete message schemes and IS document types.

wm.ip.hl7.codetables

Contains services that you use to manage code tables.

wm.ip.hl7.codetables:loadData

Loads the code tables and their values from the tab-separated value (TSV) files and enters them into the HL7 Module database. The code tables and code table values files are located in the *Integration Server_directory* \packages\WmHL7\data\codetables directory.

Input Parameters

None.

Output Parameters

isSuccessful	**String** Indicates whether the code tables have been loaded successfully. Valid values are: ■ `true` - The load operation is successful. ■ `false` - The load operation has failed.

wm.ip.hl7.codetables:runScripts

Executes database scripts required to initialize and set up the HL7 Module database. The HL7 Module database should refer to the JDBC pool corresponding to the HL7 functional alias. The database scripts are located in the *Integration Server_directory* \packages\WmHL7\config\dbscripts folder.

Note:
For a list of supported databases, see *System Requirements for Software AG Products*.

Input Parameters

database	**String** The database on which the scripts should be run. Valid values are: ■ `oracle` - The database configured in the JDBC pool is an Oracle database. ■ `sql` - The database configured in the JDBC pool is an SQL database. ■ `db2` - The database configured in the JDBC pool is an DB2 database. ■ `mysql` - The database configured in the JDBC pool is an MYSQL database.
mode	**String** The mode of operation. Valid values are: ■ `create` - Creates the tables on the database. ■ `drop` - Drops the tables from the database. ■ `cleanup` - Deletes the code tables and code table values from the database.

Output Parameters

isSuccessful	**String** Indicates whether the scripts are executed successfully. Valid values are: ■ `true` - The scripts were run successfully.

- `false` - The scripts failed to execute.

wm.ip.hl7.service

Contains services that construct, parse, and validate HL7 version 2.x messages.

wm.ip.hl7.service:convertHL7ToIData

Converts an ER7 or XML-encoded HL7 version 2.x message into an IS document (IData object) that contains the internal document format.

Note:
When parsing XML-encoded messages using the HL7 Module convertHL7ToIData service, the service does not generate correct IData when the XML-encoding does not conform to the schema (for example, required segments are missing).

Input Parameters

message	**Document** The HL7 message to be converted into an IData object.
messageType	**String** Optional. The name of the message schema definition. For example, `ADT_A01`. **Important:** The *messageType* parameter is case sensitive. By default all HL7 message schema definitions use the uppercase.
messageVersion	**String** Optional. The HL7 message version. Valid values: `2.1, 2.2, 2.3, 2.3.1,2.4,2.5,2.5.1, 2.6,`and `2.7`.
encoding	**String** Optional. The encoding type of the HL7 message. Valid values are: - `ER7` - The input message is in HL7 ER7 format. - `XML` - The input message is in HL7 XML format.
charsetEncoding	**String** Optional. Used to determine the encoding when reading the HL7 message. If no value is specified, the *charsetEncoding* to be used is determined based on the information in the TPA. For more information, see the flow charts in "Determining the Encoding of HL7 Messages" on page 171. Valid values for this parameter are the Java charset encoding values supported by the HL7 standard. For information about the charset encoding supported by the HL7 standard, see Code Table: 0211 (Alternate Character Sets).

> **Note:**
> The *charsetEncoding* input parameter is different from the encoding mentioned in the HL7 message MSH.18 field. HL7 Module does not use the MSH.18 field encoding to parse or process the HL7 message.

validate

String Optional. Specifies whether to perform syntactic, semantic, and code table validation during the conversion process.

When validate is set to `true`:

- If the convertHL7ToIData service expects a required segment, but gets an invalid segment, the service throws an exception.
- If the convertHL7ToIData service does not expect a required segment and gets an invalid segment, the service does not parse the remaining message and logs a warning for the invalid segment.

> **Note:**
> If code tables are disabled, no code table validation is performed even if *validate* is set to `true`.

> **Note:**HL7 Module performs mandatory field validation in the MSH segment even when *validate* is set to `false`.

> **Note:**
> Although HL7 Module does not restrict the length of the field data, the module logs a warning when length constraints are violated.

When the value is set to `true`, the following operations will be performed:

- Check if all required elements are present in the input data.
- Validate code tables.
- Validate field data. This is the default. value.

When the value is set to `false`, validation is disabled.

> **Note:**
> When validate is set to false and a required segment or a required segment in a segment group is missing in the message, the generated IData structure does not contain the missing segment or segment group reference.

messageScheme

Document Optional. Specifies settings related to the message scheme. Parameters are:

- *schemeID* **String** Optional. The unique message scheme ID to use for conversion.
    - To use custom message definitions, do one of the following:

    - Specify a value for the *schemeID*parameter. HL7 Module uses the message schemas that correspond to the *schemeID*value. In this case, the values that you specify for some or all of the other *messageScheme* parameters will be ignored.
    - Do not specify a value for the *schemeID* parameter, but provide values for the remaining *messageScheme* parameters. In this case, HL7 Module extracts the *schemeID* from the TPA. The module uses the message schemas that correspond to the *schemeID* extracted from the TPA.
  - To use the default message definitions provided with HL7 Module, do one of the following:
    - Specify `wm.ip.h.l7.docType` (the *schemeID* for the default XML schemas provided with HL7 Module).
    - Do not specify values for any of the *messageScheme* parameters.
- *senderID* **String** Optional. The external sender ID for the input HL7 message.
- *senderIDType* **String** Optional. The external sender ID type for the input HL7 message.
- *receiverID* **String** Optional. The external receiver ID for the input HL7 message.
- *receiverIDType* **String** Optional. The external receiver ID type for the input HL7 message.

Output Parameters

messageValues	**Document** The IData representation of the HL7 message for the specified message.
errors	**Document** Specifies whether any parsing or invalid message errors occurred.
warnings	**Document List** When the validation flag is turned on, lists the validation warnings (if any).

wm.ip.hl7.service:convertIDataToHL7

Converts an IS document type of a specific HL7 version into its respective HL7 version 2.x message format representation with the specified encoding type.

Input Parameters

messageValues **Document** The IS document to be converted into HL7 message format.

messageType **String** The name of the message schema definition. For example, `ADT_A01`.

> **Important:**
> The *messageType* parameter is case sensitive. By default all HL7 message schema definitions use the uppercase.

messageVersion **String** The HL7 message version. Valid values: `2.1, 2.2. 2.3, 2.3.1,2.4,2.5,2.5.1, 2.6,`and `2.7`.

encoding **String** Optional. The encoding type of the HL7 message. Valid values are:

- `ER7` - The output message must be in HL7 ER7 format. This is the default.
- `XML` - The output message must be in HL7 XML format.

returnAsBytes **String** Optional. Specifies the return format of the HL7 message. Valid values are:

- `true` - Returns the string representation of the HL7 message.
- `false` - Returns the bytes representation of the HL7 message. This is the default.

validate **String** Optional. Specifies whether to perform syntactic, semantic and code table validation during the conversion process.

> **Note:**
> If code tables are disabled, no code table validation will be performed even if *validate* is set to `true`.

> **Note:**HL7 Module performs mandatory field validation in the MSH segment even when *validate* is set to `false`.

Valid values are:

- `true` - The following operations will be performed:
  - Check if all required elements are present in the input data.
  - Validate code tables.
  - Validate field data. This is the default.
- `false` - Disables validation.

formatXML	**String** Optional. When the *encoding* parameter is set to XML, the *formatXML* parameter specifies if the output HL7 data must be in properly indented XML layout or not. Valid values are: ■ `true` - The XML format will be indented consistently, based on the logical nesting depth of the tags. ■ `false` - No XML format will be applied. The data will be presented as a single text entity. This is the default.
messageScheme	**Document** Optional. Specifies settings related to the message scheme. Parameters are: ■ *schemeID* **String** Optional. The unique message scheme ID to use for conversion. ■ To use custom message definitions do one of the following: ■ Specify a value for the *schemeID* parameter. HL7 Module uses the message schemas that correspond to the *schemeID*value. In this case, the values that you specify for some or all of the other *messageScheme* parameters will be ignored. ■ Do not specify a value for the *schemeID* parameter, but provide values for the remaining *messageScheme* parameters. In this case, HL7 Module extracts the *schemeID* from the TPA. The module uses the message schemas that correspond to the *schemeID* extracted from the TPA. ■ To use the default message definitions provided with HL7 Module do one of the following: ■ Specify `wm.ip.hl7.docType` (the *schemeID* for the default XML schemas provided with HL7 Module). ■ Do not specify values for any of the *messageScheme* parameters. ■ *senderID* **String** Optional. The external sender ID for the input HL7 message. ■ *senderIDType* **String** Optional. The external sender ID type for the input HL7 message. ■ *receiverID* **String** Optional. The external receiver ID for the input HL7 message. ■ *receiverIDType* **String** Optional. The external receiver ID type for the input HL7 message.

Output Parameters

messageString	**String** Conditional. The string representation of the HL7 message for the specified *messageType* and *messageVersion*. The format of the output is either ER7 or XML-encoded, based on the value you have specified for the *encoding* parameter.
messageBytes	**Object** Conditional. The bytes representation of the HL7 message for the specified *messageType* and *messageVersion*. The format of the output is either ER7 or XML-encoded, based on the value you have specified for the *encoding* parameter.
errors	**Document** Conditional. Specifies whether any parsing or invalid message errors occurred.
warnings	**Document List** When the validation flag is turned on, lists the validation warnings (if any).

wm.ip.hl7.service:generateACK

Generates an Acknowledgement message for a given HL7 message.

Input Parameters

message	**Object** The HL7 message for which to generate an acknowledgement.
mode	**String** Optional. Determines the way in which the acknowledgement is generated. Valid values are: ■ `Enhanced` - HL7 Module will use the value in the MSH-15 field to determine the way in which the acknowledgement is generated, as per the HL7 standard. This is the default. ■ `Original` - HL7 Module will always generate an acknowledgement for the message it receives, regardless of whether the MSH-15 field is present or what value it has, and without depending on any other constraints from the message.
schemeID	**String** Optional. The unique message scheme ID. If you do not specify a message scheme ID, HL7 Module uses the default message scheme, wm.ip.hl7.docType.
charsetEncoding	**String** Optional. Used to determine the encoding when reading the HL7 message. Valid values for this parameter are the Java charset encoding values supported by the HL7 standard. The default value is `UTF-8`.

generateAs — **String** Optional. The format in which you want to create the ACK message. Select either **bytes**, **string**, or **document**. The default value is `string`.

Output Parameters

ack — **Document** The ACK message in either bytes, string, or document data type, as specified in the *generateAs* input parameter.

wm.ip.hl7.service:validate

Validates the HL7 version 2.x message and reports a list of errors and warnings, if any. The service performs syntactic, semantic, and code table validation of the message.

The validate service parses the input message and if it encounters an error, it tries to ignore the section where the error was encountered and continue parsing. When the service completes parsing the HL7 version 2.x message, it returns a list of all the errors that it has encountered during the process. When the service can not continue parsing the message, it returns a single error.

Input Parameters

message — **Object** The HL7 message as a string, byte array, or InputStream.

charsetEncoding — **String** Optional. Used to determine the encoding when reading the HL7 message. If no value is specified, the *charsetEncoding* to be used is determined based on the information in the TPA. For more information, see the flow charts in "Determining the Encoding of HL7 Messages" on page 171.

Valid values for this parameter are the Java charset encoding values supported by the HL7 standard.

For information about the charset encoding supported by the HL7 standard, see Code Table: 0211 (Alternate Character Sets).

> **Note:**
> The *charsetEncoding* input parameter is different from the encoding mentioned in the HL7 message MSH.18 field. HL7 Module does not use the MSH.18 field encoding to parse or process the HL7 message.

messageScheme — **Document** Optional. Specifies settings related to the message scheme. Parameters are:

- *schemeID* **String** Optional. The unique message scheme ID to use for validation.
    - To use custom message definitions do one of the following:

        - Specify a value for the *schemeID* parameter. HL7 Module uses the message schemas that correspond to the *schemeID*value. In this case, the values that you specify for some or all of the other *messageScheme* parameters will be ignored.
        - Do not specify a value for the *schemeID* parameter, but provide values for the remaining *messageScheme* parameters. In this case, HL7 Module extracts the *schemeID* from the TPA. The module uses the message schemas that correspond to the *schemeID* extracted from the TPA.
    - To use the default message definitions provided with HL7 Module, do one of the following:
        - Specify `wm.ip.hl7.docType` (the *schemeID* for the default XML schemas provided with HL7 Module)
        - Do not specify values for any of the *messageScheme* parameters.
- *senderID* **String** Optional. The external sender ID for the input HL7 message.
- *senderIDType* **String** Optional. The external sender ID type for the input HL7 message.
- *receiverID* **String** Optional. The external receiver ID for the input HL7 message.
- *receiverIDType* **String** Optional. The external receiver ID type for the HL7 input message.

Output Parameters

isValid	**String** Specifies whether the input message is valid. The value is derived from the *errors* parameter. Valid values are: ■ `true` - The message is valid. ■ `false` - The message is invalid.
errors	**Document List** Contains an accumulated list of errors that occurred while parsing the document. Parameters are: ■ *errorMessage* **String** The text of the error message.
warnings	**Document List** Contains an accumulated list of warnings that occurred while parsing the document. Parameters are: ■ *warningMessage* **String** The text of the warning message.

wm.ip.hl7.tn

Contains built-in services that enhance Trading Networks to function as a gateway for HL7 message exchange. The following elements are available in the wm.ip.hl7.tn folder:

Folder	Description
wm.ip.hl7.service	Contains built-in services to send and receive HL7 version 2.x messages using Trading Networks.
wm.ip.hl7.tn.tpa	Contains a service to create a default TPA and an IS document to associate with the generated TPA.
wm.ip.hl7.tn.transport	Contains services and IS document types to support the MLLP transport.
wm.ip.hl7.tn.utils	Contains a utility service that forms a BizDocEnvelope for inbound HL7 version 2.x messages during their processing in Trading Networks.

wm.ip.hl7.tn.service

Contains built-in services to send and receive HL7 version 2.x messages using Trading Networks.

wm.ip.hl7.tn.service:receive

Receives HL7 version 2.x messages sent by a trading partner using the specified Trading Networks transport protocol.

Usage Notes

The receive service reads the input stream and converts it into an internal Java object that it stores in the Trading Networks database. When reading the input stream content, the service determines which charset encoding to use as described in the following steps:

1. Searches for the *Content-Type* property in the header of the message. The value of *Content-Type* can contain the charset information. If the charset information is provided in *Content-Type*, the service reads the input stream message with this charset and ignores the charset encoding provided in the TPA. For example, when the value of *Content-Type* in the header is `Content-Type=text/plain; charset=utf-8`, the service will use `utf-8` as the charset encoding.
2. If no *Content-Type* property is found in the header of the message, the service reads the input stream with the system default charset encoding and then constructs the HL7 message object.
3. Obtains the Sender and Receiver ID from the HL7 message and retrieves the TPA for this Sender and Receiver ID.
4. Obtains the charset encoding from the retrieved TPA. If it is not present in the retrieved TPA, the service uses the charset encoding from the default TPA.

5. If the charset encoding specified in the TPA is different from the system default encoding, the service reads the input stream again with the encoding specified in the TPA.

Input Parameters

ffdata	**Object** Optional. The incoming HL7 message in the form of a stream. **Note:** This parameter is filled in internally with the HL7 message when it is received via HTTP.
contentStream	**Object**Optional. The incoming HL7 message in the form of a stream. **Note:** This parameter is filled in internally with the HL7 message when it is received via FTP.

Output Parameters

response	**byte [ ]** When you use the synchronous delivery mode, the value of the *response* parameter is the payload of the acknowledgement message. The value of the *response* parameter is sent as a response to the Sender.

wm.ip.hl7.tn.service:send

Sends HL7 version 2.x messages to a trading partner using the specified Trading Networks transport protocol. Based on the transport protocol you specify, the service invokes the corresponding Trading Networks delivery service located in the wm.tn.transport folder of the WmTN package. For more information about Trading Networks delivery services, see the *webMethods Trading Networks Built-In Services Reference* for your release.

Note:
If the *bizDoc* value is present (that is, the message is in BizDocEnvelope format), the service ignores all the input parameter values, except for *tnProtocol* and extracts the sender and receiver details from the information already presented in the BizDocEnvelope.

Usage Notes

- If the *senderID* or *senderIDType* is not present, the service obtains the *senderID* or *senderIDType* from the 'MSH-3' (Sending Application) field of the MSH segment of the message. The service uses the value of the 'HD-2' (Universal ID) component as the *senderID* and the value of the 'HD-3' (Universal ID Type) component as the *senderIDType*.
- If the *receiverID* or *receiverIDType* is not present, the service obtains the *receiverID* or *receiverIDType* from the 'MSH-5' (Receiving Application) field of the MSH segment of the

message. The service uses the value of the 'HD-2' (Universal ID) component as the *receiverID* and the value of the 'HD-3' (Universal ID Type) component as the *receiverIDType*.

For more information about the MSH-3 and MSH-5 fields, see "Format of the Sender Application and Receiver Application Fields" on page 88.

- The default time out for the send service is 0 (wait infinitely). You can change the default time out by setting the following watt property: *watt.net.timeout*. For information about how to set the watt property, see the *webMethods Integration Server Administrator's Guide* for your release.
- When you set the *tnProtocol* parameter to MLLP, the value of the **Ack Timeout** parameter of the MLLP connection configured in the receiver's profile determines whether to read the acknowledgement message.
  - When the value of *ackTimeout* is positive or zero, the send service will wait during the specified timeout to read the response. The status will be `success` only when the `send` service sends the response to the remote host and receives a response in return. If the send service does not receive the response from the remote host, the status will be `fail`.
  - When the value of *ackTimeout* is negative, the status will be `success` when the send service sends the response message to the remote host.

Input Parameters

message	**Object** Optional. The incoming HL7 message in the form of a string, stream, or bytes. When the input HL7 message object is a string, you should initialize the string object with the desired charset encoding.
bizDoc	**Document** Optional. The Trading Networks document representation of an HL7 message.
charsetEncoding	**String** Optional. Used to determine the encoding when reading the HL7 message. If no value is specified, the *charsetEncoding* to be used is determined based on the information in the TPA. For more information, see the flow charts in "Determining the Encoding of HL7 Messages" on page 171. Valid values for this parameter are the Java charset encoding values supported by the HL7 standard. For information about the charset encoding supported by the HL7 standard, see Code Table: 0211 (Alternate Character Sets). **Note:** The *charsetEncoding* input parameter is different from the encoding mentioned in the HL7 message MSH.18 field. HL7 Module does not use the MSH.18 field encoding to parse or process the HL7 message.
tnProtocol	**String** Optional. The Trading Networks transport protocol to use for sending the message. Valid values are:

- For HTTP, set to
  - `Primary HTTP`
  - `Secondary HTTP`
- For HTTPS, set to
  - `Primary HTTPS`
  - `Secondary HTTPS`
- For E-mail, set to
  - `Primary E-mail`
  - `Secondary E-mail`
- For FTP, set to
  - `Primary FTP`
  - `Secondary FTP`
- For MLLP, set to
  - `MLLP`

If you provide a value for *tnProtocol*, it will take precedence over the preferred protocol set for the trading partner in his Trading Networks profile.

senderID	**String** Optional. External ID of the sender.
senderIDType	**String**Optional. External ID type of the sender.
receiverID	**String**Optional.External ID of the receiver.
receiverIDType	**String**Optional. External ID type of the receiver.

Output Parameters

sendStatus **Document** Contains the outcome of the send operation. The output of the delivery is determined by the Trading Networks wm.tn.rec:DeliveryServiceOutput document record type. For more information about this document record type, see the *webMethods Trading Networks Built-In Services Reference* for your release. The document typically has the following parameters:

- *status* **String** The status the delivery service returns. Valid values are:
  - `success` - The send operation is successful.

    - `fail` - The send operation has failed.

- *status Message* **String** The delivery-specific message that the delivery service returns along with the status.

- *transport Time* **String** The total time (in milliseconds) that the delivery service used to deliver the document.

- *bizDocID* **String** The BizDocEnvelope ID of the sent document.

- *messageID* **String** The HL7 message Control ID of the sent message.

- *output* **Document** The *serviceOutput*output parameter returned by the TN transport protocol specified in the *tnProtocol* input parameter. For more information about the structure of the *serviceOutput* parameter, see the *webMethods Trading Networks Built-In Services Reference*.

wm.ip.hl7.tn.tpa

Contains a service to create a default TPA and an IS document to associate with the generated TPA.

wm.ip.hl7.tn.tpa.rec:HL7TPA

An IS document that contains a set of variables used for processing HL7 version 2.x messages. This document is associated with the TPA generated by the wm.ip.hl7.tn.tpa:createHL7TPA service.

Parameters

hl7ProtocolAgreement	Settings related to the transport protocol.

- *Acknowledgement* Settings related to the Acknowledgement.

    - *ackProtocol* Optional. The transport protocol to use when sending the Acknowledgment to the sender. The default value is `blank`.

        - For HTTP, set to `Primary HTTP` or `Secondary HTTP`.

        - For HTTPS, set to `Primary HTTPS` or `Secondary HTTPS`.

        - For E-mail, set to `Primary SMTP` or `Secondary SMTP`.

        - For FTP, set to `Primary FTP` or `Secondary FTP`.

        - For MLLP, set to `MLLP`.

    - *sendACK* Optional. Specifies whether to send an acknowledgement for the received message, regardless of the value in the MSH-15 field, or if the MSH-15 field is not provided. Valid values:

- `Default` - HL7 Module will use the value in the MSH-15 field to determine whether to send an acknowledgment, as per the HL7 standard. This is the default.
- `Always` - HL7 Module will always send an acknowledgment for the message it received, regardless of whether the MSH-15 field is present or what value it has, and without depending on any other constraints from the message.
- `Never` - HL7 Module will never send an acknowledgment, regardless of whether the MSH-15 field is present or what value it has.

- *ackMode* **String** When you select HTTP as the transport protocol in the *ackProtocol* parameter in the HL7TPA document, the service sends an Accept Acknowledgment to the Sender using either a synchronous, or an asynchronous mode based on the value you specify. Valid values:
  - `Async` - Acknowledges receiving the message by sending an ACK as a separate transaction. This is the default.
  - `Sync` - Acknowledges receiving the message by sending the ACK in the content part of the HTTP response. The HTTP response status code is 200.

hl7MessageScheme Optional. Settings related to the message scheme.

- *schemeID* Optional. The message scheme ID to use for an HL7 message associated with this TPA. For example, `com.example.custom.scheme`

hl7Properties Optional. Settings related to the configuration properties.

Note:
The default HL7 Module TPA has only one sub-parameter for the *hl7Properties*parameter, *charsetEncoding*. The default value for the *charsetEncoding* parameter is `UTF-8`.

- *name* The property name used for configuring a *charsetEncoding*. The value for this parameter is *charsetEncoding*.
- *value* Any Java character encoding supported by the HL7 standard.

wm.ip.hl7.tn.tpa:createHL7TPA

Use this service to:

- Create a partner specific TPA and associate it with a blue print of the TPA that establishes the TPA parameters and values.
- Associate the TPA with a fully-qualified service name of a service that fills in the *tpaData* parameter with default values.
- Set the Status of the TPA to `agreed`.

When the specified *senderID*, *receiverID*, and *agreementID* are not unique, the service reports an error. If the service fails to create a TPA for any other reason, it throws an exception.

Input Parameters

senderID	**String** External ID of the sender. The default value is `Unknown`.
senderIDType	**String**External ID type of the sender.
receiverID	**String**External ID of the receiver. The default value is `Unknown`.
receiverIDType	**String**External ID type of the receiver.
tpaData	**Document** The TPA values. For details about this IS document, see wm.ip.hl7.tn.tpa.rec:HL7TPA.

Output Parameters

tpa	**Document** The TPA object.
error	**Document** Optional. The error reported by the service.

wm.ip.hl7.tn.transport

Contains services and IS document types to support the MLLP transport.

wm.ip.hl7.tn.transport.rec:Listener

An IS document type that contains information about the MLLP listeners.

Parameters

name	**String** The name of the listener.
package	**String** The package that the listener is associated with.
enabled	**String** The enabled status of the listener. Valid values are: - `yes` - The listener is enabled.

	■ `no` - The listener is disabled.
serverStatus	**String**The status of the server. Valid values are: ■ `Initialized` - The server has been initialized. ■ `Starting` - The server is starting. ■ `Running` - The server is running. ■ `Stopping` - The server is closing down existing sessions. ■ `Stopped` - The server has stopped.
serverInfo	**Document** The server information provided as an IS document. For details about this IS document, see wm.ip.hl7.tn.transport.rec:ServerInfo.

wm.ip.hl7.tn.transport.rec:ServerInfo

An IS document type that contains information about the MLLP listener.

Parameters

port	**String** The port at which the server is running. When the server is not running, the value is `-1`.
serverSocket	**String** Details about the server socket.
status	**String**The status of the server. The value is one of the *serverStatus* parameter values described in the wm.ip.hl7.tn.transport.rec:Listener service.
thread	**String**Details about the server thread.
connectionCount	**String**The number of active connections currently handled by the server.
connections	**Document List**Details about the connections. Parameters are: ■ *socket* Details about the client socket.
readCount	**String** The count of current partial read operations.
reads	**Document List**Details about the partial read operations. Parameters are: ■ *socket* **String** Details about the socket for which a read operation is in progress. ■ *size* **String** Length of the data read so far.

wm.ip.hl7.tn.transport:mllp

Delivers HL7 version 2.x messages via the MLLP transport. The service is either invoked directly, or it is invoked internally by the wm.ip.hl7.tn.service:send service when the *tnProtocol* parameter is set to `MLLP`. The service obtains the values for *hostname*, *port*, and *timeout* from the receiver Trading Networks partner profile.

Note:
If the input to the service is *message*, the mllp service uses the value specified for *connectionAlias* and this parameter is mandatory. If the input to the service is *bizdoc*, the details of the connection alias are taken from the trading partner profile.

Input Parameters

bizDoc	**Object** The BizDocEnvelope of the message.
message	**Object** Optional. The incoming HL7 message as a string, stream, or bytes. When the input HL7 message object is a string, you should initialize the string object with the desired charset encoding.
connectionAlias	**String** The qualified namespace of the MLLP connection to the MLLP server, in the format *folder:connectionName*. For example, `connections:mllp_v2_host`
charsetEncoding	**String** Optional. Used to determine the encoding when reading the HL7 message. If no value is specified, the *charsetEncoding* to be used is determined based on the information in the TPA. For more information, see the flow charts in "Determining the Encoding of HL7 Messages" on page 171. Valid values for this parameter are the Java charset encoding values supported by the HL7 standard. For information about the charset encoding supported by the HL7 standard, see Code Table: 0211 (Alternate Character Sets). **Note:** The *charsetEncoding* input parameter is different from the encoding mentioned in the HL7 message MSH.18 field. HL7 Module does not use the MSH.18 field encoding to parse or process the HL7 message.

Output Parameters

serviceOutput **String** The output of the delivery service. Parameters are:

- *status* **String** The outcome of the delivery. Valid values are:
  - `success` - The delivery is successful.

- `fail` - The delivery has failed.

- *statusMessage* **String** The status message from the last attempt to deliver the document.
- *transportTime* **String** The total time (in milliseconds) it took to deliver the document.

output	**Document** Contains the **response** parameter. **response** String When the send service sends an ACK message, this parameter contains the acknowledgement sent in response to the ACK message.

wm.ip.hl7.tn.transport:mllpListenerStatus

Retrieves current listener status and server information.

Input Parameters

None.

Output Parameters

serviceOutput	**Document** The execution status of the listeners. Parameters are: - *listeners* **Document List** The listeners information provided as an IS document. For details about this IS document, see wm.ip.hl7.tn.transport.rec:Listener.

wm.ip.hl7.tn.utils

Contains a utility service that forms a BizDocEnvelope for HL7 version 2.x messages so that Trading Networks can process them.

wm.ip.hl7.tn.utils:createBizDoc

A utility service that converts an HL7 version 2.x message into BizDocEnvelope format. The BizDocEnvelope that the service creates takes one of the following TN document types:

- When the HL7 message type of the inbound HL7 message is ACK:
  - HL7 Acknowledgment (for HL7 ER7 messages)
  - HL7 XML Acknowledgment (for HL7 XML messages)
- For all other HL7 message types.

- HL7 Default (for HL7 ER7 messages)
- HL7 XML Default (for HL7 XML messages)

The generated BizDocEnvelope contains all the attributes for which the message has specified values. Based on the HL7 message format, the service:

- Adds an HL7 ER7-encoded message to the BizDocEnvelope as a content part with `HL7 ER7 Data` as the content part name.
- Adds an HL7 XML-encoded message to the BizDocEnvelope as a content part with `HL7 XML Data` as the content part name.

Usage Notes:

- If the *senderID* or *senderIDType* is not present, the service obtains the*senderID* or *senderIDType* from the 'MSH-3' (Sending Application) field of the MSH segment of the message. The service uses the value of the 'HD-2' (Universal ID) component as the *senderID* and the value of the 'HD-3' (Universal ID Type) component as the*senderIDType*.
- If the *receiverID* or *receiverIDType* is not present, the service obtains the *receiverID* or *receiverIDType*from the 'MSH-5' (Receiving Application) field of the MSH segment of the message. The service uses the value of the 'HD-2' (Universal ID) component as the *receiverID* and the value of the 'HD-3' (Universal ID Type) component as the *receiverIDType*.

For more information about the MSH-3 and MSH-5 fields, see "Format of the Sender Application and Receiver Application Fields" on page 88.

Input Parameters

senderID	**String** Optional. External ID of the sender.
senderIDType	**String**Optional. External ID type of the sender.
receiverID	**String**Optional.External ID of the receiver.
receiverIDType	**String**Optional. External ID type of the receiver.
message	**Object** The incoming HL7 message in the form of a string, stream, or bytes.
charsetEncoding	**String** Optional. Used to determine the encoding when reading the HL7 message. If no value is specified, the *charsetEncoding* to be used is determined based on the information in the TPA. For more information, see the flow charts in "Determining the Encoding of HL7 Messages" on page 171. Valid values for this parameter are the Java charset encoding values supported by the HL7 standard. For information about the charset encoding supported by the HL7 standard, see Code Table: 0211 (Alternate Character Sets).

Note:
The *charsetEncoding* input parameter is different from the encoding mentioned in the HL7 message MSH.18 field. HL7 Module does not use the MSH.18 field encoding to parse or process the HL7 message.

Output Parameters

bizDoc	**Document** Trading Networks document representation of the incoming HL7 message.

wm.ip.hl7.utils

Contains utility services to generate and delete message schemes and IS document types.

wm.ip.hl7.utils:deleteMessageScheme

Deletes the IS document types for the specified HL7 message version and message scheme ID.

The deleteMessageScheme service does not check for dependencies or references before deleting the generated IS document types. Ensure that there are no dependencies or references in the IS documents before you run the service. For information about how to configure dependency checking for elements, see the *webmethods Developer User's Guide* or the *webMethods Service Development Help* for your release.

Input Parameters

messageVersion

String The HL7 version for which the IS document types will be deleted.

Valid values are:

`2.1, 2.2, 2.3`, `2.3.1`, `2.4`, `2.5`, `2.5.1`, `2.6`, and `2.7`.

schemeID

String Optional. The unique message scheme ID. If no value for *schemeID* is specified, HL7 Module uses the default message scheme, `wm.ip.hl7.docType`.

quickDelete

String Optional. Indicates whether to check for any dependencies before deleting the document types nodes.

Important:
Note that when *quickDelete* is set to **true**, the service deletes the documents types quickly, but without any reference check.

Valid values are:

- `true` - HL7 Module does not check for any dependencies and deletes all node definitions without any verification.
- `false` - HL7 Module checks for dependencies. This is the default.

reloadPackage

String Optional. Indicates whether to reload the WmHL7DocTypes package after deleting the document types nodes.

Important:
You must reload the package after making any changes to the nodes in the WmHL7DocTypes package. Otherwise, you can select not to reload the package, and do it at a later point in time.

Valid values are:

- `true` - HL7 Module reloads the WmHL7DocTypes package.
- `false` - HL7 Module does not reload the WmHL7DocTypes package. This is the default.

Output Parameters

isSuccessful

String Specifies whether the IS document types were deleted successfully.

- `true` - The delete operation is successful.
- `false` - The delete operation has failed.

wm.ip.hl7.utils:generateMessageScheme

Generates related IS document types for the specified supported HL7 2.x message version, message types, and message scheme ID. The service performs the following operations:

Note:HL7 Module provides default XML schemas for the HL7 message definitions, supplied by the HL7 organization, for all supported HL7 message versions. The files are located in the *Integration Server_directory* \packages\WmHL7\data\xsd\wm.ip.hl7.docType\ v*messageVersion* directory. For example, the default XML schemas for HL7 messages version 2.1 are located in the *Integration Server_directory* \packages\WmHL7\data\xsd\wm.ip.hl7.docType\ v21 directory.

Note:
If you use custom HL7 XML schemas, they must be located in the *Integration Server_directory* \packages\WmHL7\data\xsd\schemeID\v*messageVersion* directory.

1. Parses the file with the message definitions for the specified HL7 message version and *schemeID*, and generates the IS document types for that version, message types, and message scheme.
2. Creates wrapper IS documents for the specified HL7 message types. Each wrapper IS document directly references the generated IS document corresponding to the selected message type.

Input Parameters

messageVersion	**String** The HL7 version for which the IS document types will be generated. Valid values are: `2.1, 2.2, 2.3`, `2.3.1`, `2.4`, `2.5`, `2.5.1`, `2.6`, and `2.7`.
messageTypes	**String List** Optional. List of valid HL7 messageTypes that belong to the specified *messageVersion* for which the IS document types will be generated. **Note:** If you do not specify a value for the *messageTypes* parameter, HL7 Module generates IS document types for all the HL7 message types valid for the specified *messageVersion*.
schemeID	**String** Optional. The unique message scheme ID. **Note:** If you do not specify a message scheme ID, HL7 Module uses the default message definitions provided with the module.

Output Parameters

isSuccessful	**String** Specifies whether the IS document types were created successfully. Valid values are: ■ `true` - The IS document type generation is successful. ■ `false` - The IS document type generation operation has failed.
errors	**Document List** Specifies whether any errors occurred during the XML Schemas processing.
warnings	**Document List** Specifies whether any warnings occurred during the XML Schemas processing.

WmHL7DocTypes Package

The WmHL7DocTypes package contains the IS document types generated using the utility service provided with HL7 Module.

wm.ip.hl7.docType

When you use the default HL7 Module message scheme to generate IS document types, this folder contains generated IS document types for the supported HL7 v2.x message versions. The generated documents are placed under the wm.ip.hl7.docType.*version*.reference folder for the specified message version. For example, the generated IS document types for an HL7 message version 2.3.1 are located in the wm.ip.hl7.docType.v231.reference folder.

The wm.ip.hl7.docType folder also contains wrapper IS documents that reference the generated IS document type for the corresponding message type. These wrapper IS documents are located under the respective wm.ip.hl7.docType.*version*folder. The namespace for each wrapper IS document has the HL7_ prefix attached to the message type.

When you use a custom message scheme, this folder will be empty.

B Verifying The Message Header Segment of HL7 Messages

Required Fields in the Message Header Segment of HL7 Version 2.x Messages

When you send or receive HL7 messages with Trading Networks using any of the supported transport protocols, Trading Networks extracts the information required to process HL7 version 2.x messages from the Message Header (MSH) segment fields of the HL7 message. To send or receive HL7 version 2.x messages using Trading Networks, verify that the required MSH segment fields are included and set correctly in the message header.

If any of the segment fields listed in the following table are missing, the wm.ip.hl7.tn.service:send service fails to send the HL7 message and logs in an error message in the Integration Server logs. In this case, the transaction is not saved in the Trading Networks database. On the receiving end if any of the required segment fields are missing from the message header, the HL7 message will be saved in the Trading Networks database with the status of DONE W/ Errors and HL7 Module logs an error message in the Trading Networks Activity Log for the persisted transactions. For more information about how to manage information about HL7 message transactions, see "Viewing Information about HL7 Messages" on page 91.

Note:HL7 Module requires and processes only the MSH segment fields listed in the following table.

Required Segment Field	Specifies...
MSH.1 Field Separator	The character to be used as the field separator in the message. Recommended value: |
MSH.2 Encoding Characters	The characters to be used as the component separator, repetition separator, escape character, and subcomponent separator (in that order). Recommended values: `^~\&` (ASCII 94, 126, 92, and 38, respectively).
MSH.3 Sender Application	The identity of the sender when receiving an HL7 message. The field contains the following components: `Namespace ID (IS) ^ Universal ID (ST) ^ Universal ID Type (ID)` For more information about the format of the Sender Application field, see "Format of the Sender Application and Receiver Application Fields" on page 88.

Required Segment Field	**Specifies...**
	HL7 Module requires that either the namespace ID or the combination of universal ID and universal ID Type are specified. For more information about using the namespace ID component to identify a trading partner, see "Identifying a Trading Partner" on page 167.
MSH.5 Receiver Application	The identity of the receiver when receiving an HL7 message. The components in the MSH.5 field are identical to the components of the MSH.3 field. For more information about the format of the Receiver Application field, see "Format of the Sender Application and Receiver Application Fields" on page 88. HL7 Module requires that either the namespace ID or the combination of universal ID and universal ID Type are specified. For more information about using the namespace ID component to identify a trading partner, see "Identifying a Trading Partner" on page 167.
MSH.7 Date/time of message	The date and time when the message was created. The value of this field is in the following format: `YYYYMMDDHHMMSS`
MSH.9 Message Type	The message type of the HL7 message. **Important:** The Message Type segment field is case sensitive. By default all HL7 message schema definitions use the uppercase. The field contains the following components: `Message Code (ID) ^ Trigger Event (ID) ^ Message Structure (ID)` where `<Message Code (ID)>` identifies the message code, for example: `ADT` `<Trigger Event (ID)>` identifies the event type, for example: `A01` `<Message Structure (ID)>` identifies the message structure, for example: `ADT_A01`. HL7 Module does not use this component. The structure of the data in the MSH.9 field depends on the HL7 message version: ■ For HL7 message version 2.1, the structure is: `Message Code (ID)>` ■ For HL7 message versions 2.2 and 2.3, the structure is:

Required Segment Field	Specifies...

	`Message Code (ID) ^ Trigger Event (ID)`
	■ For HL7 message version 2.3.1 to 2.7, the structure is:
	`Message Code (ID) ^ Trigger Event (ID) ^ Message Structure (ID)`

MSH.10 Message Control ID	A unique identifier for the message, for example a number. HL7 Module uses this ID to send back an acknowledgment to the sending system in the Message Acknowledgement Segment (MSA).
MSH.11 Processing ID	Specifies the processing ID or processing mode of the HL7 message. Valid values for the processing ID are: ■ `D` - debugging ■ `P` - production (Default) ■ `T` - training Valid values for the processing mode are: ■ `A` - archive ■ `R` - restore from archive ■ `I` - initial load ■ `not present` - currently processing the message (Default)
MSH.12 Version ID	The HL7 message version. The field contains the following components: `Version ID (ID) ^ Internationalization Code (CWE) ^ International Version ID (SWE)` HL7 Module uses only the *Version ID (ID)* component to identify the HL7 message version of the incoming message. The sending application should set the *Version ID* correctly for HL7 Module to process the message. When sending a message from HL7 Module to other trading partners, you must set the correct message version for the outgoing HL7 message.

Extracting the Sender and the Receiver from the HL7 Message Header

Identifying a Trading Partner

Because Trading Networks uses the combination of **Corporation Name** and **Unit Name** in a trading partner profile to identify a partner, HL7 Module requires that either the namespace ID or the combination of universal ID and universal ID Type are specified in the MSH.3 field (for the sender) and the MSH.5 field (for the receiver). The namespace ID in the MSH field of the message header correspond to the **Corporation Name** and **Unit Name** values specified in the trading partner profile as follows:

- To identify the sender, the namespace ID in the MSH.3 field corresponds to **Corporation Name** and optionally, the namespace ID in the MSH.4 field corresponds to **Unit Name**.
- To identify the receiver, the namespace ID in the MSH.5 field corresponds to **Corporation Name** and optionally, the namespace ID in the MSH.6 field corresponds to **Unit Name**.

When all three components (namespace ID, universal ID, and universal ID Type) are specified in the MSH field, HL7 Module uses the universal ID (HD.2) and the universal ID Type (HD.3) components to identify the trading partner. When one of HD.2 or HD.3 components is invalid or does not exist, HL7 Module uses the default `Unknown` partner profile.

To identify the trading partner when only the namespace ID (HD.1) component is specified, HL7 Module uses the **Corporation Name** specified in the Trading Networks partner profile and matches the HD.1 value from the MSH.4 or MSH.6 fields against the **Unit Name** specified in Trading Networks.

Identifying the Sender and Receiver When Sending an HL7 Version 2.x Message

When using the wm.ip.hl7.tn.service:send service to send HL7 version 2.x messages, the sender and the receiver of the message are determined from the value of the *senderID*/*senderIDType* and *receiverID*/*receiverIDType* parameters respectively. However, these fields are optional. If no values are provided for these parameters, the sender and the receiver will be determined using the values of the MSH.3 Sender Application and MSH.5 Receiver Application fields in the message header of the HL7 message being sent. For more information see the usage notes of the wm.ip.hl7.tn.service:send service.

If there is no partner profile in Trading Networks that corresponds to the sender and the receiver extracted from the MSH.3 and MSH.5 fields, the sender and the receiver of the message take the value of `Unknown`.

Important:
When sending a message, HL7 Module uses the Delivery Settings of the receiver. If the receiver is `Unknown`, the message delivery will fail. Create a Trading Networks partner profile for the sender and the receiver that you extracted from the HL7 message header.

Identifying the Sender and Receiver When Receiving an HL7 Version 2.x Message

On the receiving end, the sender and the receiver will be determined using the values of the MSH.3 Sender Application and MSH.5 Receiver Application fields in the HL7 message being sent. The second subfield of the MSH.3 and MSH.5 segment fields is *sender ID* and the third subfield is *sender ID Type*.

If there is no partner profile in Trading Networks that corresponds to the sender and the receiver extracted from the MSH.3 and MSH.5 fields, the sender and the receiver of the message take the value of `Unknown`.

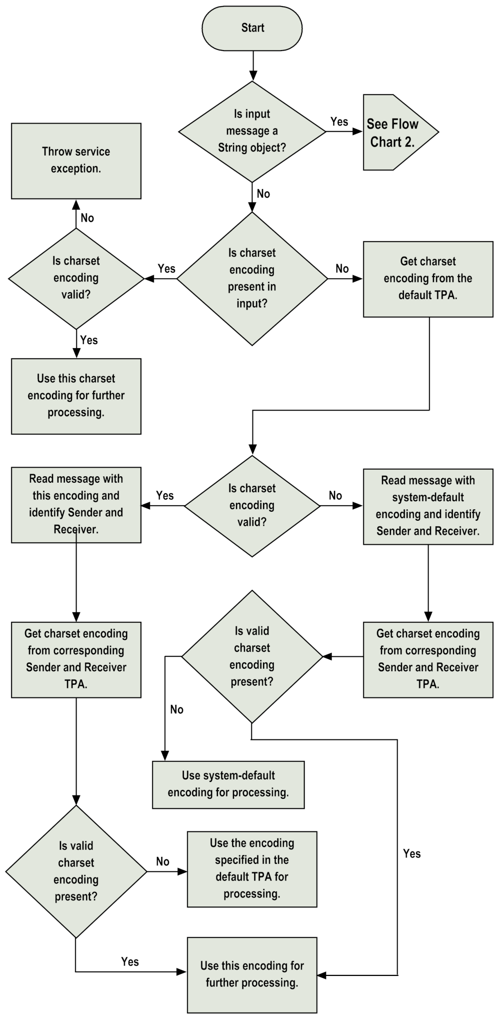
Important:
If the sender of a message received is `Unknown`, HL7 Module cannot return an acknowledgment (if requested), because the module uses the Delivery Settings of the sender. Create a Trading Networks partner profile for the sender and the receiver that you extracted from the HL7 message header.

C Determining the Encoding of HL7 Messages

Determining the Encoding for wm.ip.hl7.tn.service:send

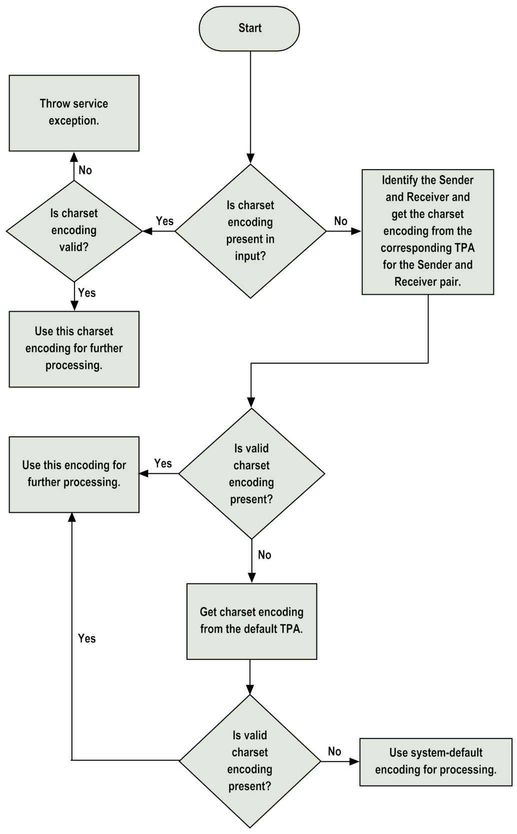
The following flow charts show how HL7 Module determines the character set encoding to use when reading an HL7 message for the wm.ip.hl7.tn.service:send service.

Flow Chart 1



Start
Is input message a String object?
Yes
See Flow Chart 2.
No
Throw service exception.
No
Is charset encoding valid?
Yes
Is charset encoding present in input?
No
Get charset encoding from the default TPA.
Yes
Use this charset encoding for further processing.
Read message with this encoding and identify Sender and Receiver.
Yes
Is charset encoding valid?
No
Read message with system-default encoding and identify Sender and Receiver.
Get charset encoding from corresponding Sender and Receiver TPA.
Is valid charset encoding present?
Get charset encoding from corresponding Sender and Receiver TPA.
No
Use system-default encoding for processing.
Yes
Is valid charset encoding present?
No
Use the encoding specified in the default TPA for processing.
Yes
Use this encoding for further processing.

Flow Chart 2


Start
Is charset encoding present in input?
Yes
No
Is charset encoding valid?
No
Throw service exception.
Yes
Use this charset encoding for further processing.
Identify the Sender and Receiver and get the charset encoding from the corresponding TPA for the Sender and Receiver pair.
Is valid charset encoding present?
Yes
Use this encoding for further processing.
No
Get charset encoding from the default TPA.
Is valid charset encoding present?
Yes
No
Use system-default encoding for processing.