



Ensemble HL7 Version 2 Development Guide

Version 2018.1
2024-11-07

InterSystems®, HealthShare Care Community®, HealthShare Unified Care Record®, IntegratedML®, InterSystems Caché®, InterSystems Ensemble®, InterSystems HealthShare®, InterSystems IRIS®, and TrakCare are registered trademarks of InterSystems Corporation. HealthShare® CMS Solution Pack™, HealthShare® Health Connect Cloud™, InterSystems® Data Fabric Studio™, InterSystems IRIS for Health™, InterSystems Supply Chain Orchestrator™, and InterSystems TotalView™ For Asset Management are trademarks of InterSystems Corporation. TrakCare is a registered trademark in Australia and the European Union.

All other brand or product names used herein are trademarks or registered trademarks of their respective companies or organizations.

This document contains trade secret and confidential information which is the property of InterSystems Corporation, One Memorial Drive, Cambridge, MA 02142, or its affiliates, and is furnished for the sole purpose of the operation and maintenance of the products of InterSystems Corporation. No part of this publication is to be used for any other purpose, and this publication is not to be reproduced, copied, disclosed, transmitted, stored in a retrieval system or translated into any human or computer language, in any form, by any means, in whole or in part, without the express prior written consent of InterSystems Corporation.

The copying, use and disposition of this document and the software programs described herein is prohibited except to the limited extent set forth in the standard software license agreement(s) of InterSystems Corporation covering such programs and related documentation. InterSystems Corporation makes no representations and warranties concerning such software programs other than those set forth in such standard software license agreement(s). In addition, the liability of InterSystems Corporation for any losses or damages relating to or arising out of the use of such software programs is limited in the manner set forth in such standard software license agreement(s).

THE FOREGOING IS A GENERAL SUMMARY OF THE RESTRICTIONS AND LIMITATIONS IMPOSED BY INTERSYSTEMS CORPORATION ON THE USE OF, AND LIABILITY ARISING FROM, ITS COMPUTER SOFTWARE. FOR COMPLETE INFORMATION REFERENCE SHOULD BE MADE TO THE STANDARD SOFTWARE LICENSE AGREEMENT(S) OF INTERSYSTEMS CORPORATION, COPIES OF WHICH WILL BE MADE AVAILABLE UPON REQUEST.

InterSystems Corporation disclaims responsibility for errors which may appear in this document, and it reserves the right, in its sole discretion and without notice, to make substitutions and modifications in the products and practices described in this document.

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)

Tel: +1-617-621-0700

Tel: +44 (0) 844 854 2917

Email: support@InterSystems.com

Table of Contents

About This Book	1
1 Introduction	3
1.1 Ensemble Support for HL7 Version 2	3
1.2 An HL7 Version 2 Routing Production	3
2 HL7 Schemas and Available Tools	7
2.1 Overview of HL7 Schemas and Messages	7
2.2 Using the HL7 Schema Structures Page	8
2.2.1 Viewing a List of Document Types	9
2.2.2 Viewing a Message Structure	9
2.2.3 Viewing a Segment Structure	11
2.2.4 Viewing a Data Structure	13
2.2.5 Viewing a Code Table	14
2.2.6 Choosing a Different Category	15
2.3 Using the Custom Schema Editor	16
2.3.1 Creating a New Custom Schema	17
2.3.2 Defining a New Segment	18
2.3.3 Defining a New Message Type and Structure Type	20
2.3.4 Editing Data Structures and Code Tables	23
2.4 Using the HL7 Message Viewer Page	23
2.4.1 Selecting Options	23
2.4.2 Parsing the Message	24
2.4.3 Testing a Transformation	26
2.5 Viewing Batch Messages	26
2.6 HL7 Classes	27
2.6.1 Details on the HL7 Message Class	29
3 Configuring the Production	31
3.1 Creating a New HL7 Routing Production	31
3.2 Adding HL7 Business Services	32
3.2.1 Creating an HL7 Business Service	32
3.2.2 Integrating and Configuring an HL7 Business Service	33
3.3 Adding HL7 Routing Processes	33
3.3.1 Creating an HL7 Routing Process	33
3.3.2 Integrating and Configuring an HL7 Routing Process	34
3.4 Adding HL7 Sequence Managers	34
3.4.1 Creating an HL7 Sequence Manager	35
3.4.2 Integrating and Configuring an HL7 Sequence Manager	35
3.4.3 Accessing HL7 Sequence Data Programmatically	35
3.5 Adding HL7 Business Operations	36
3.5.1 Creating an HL7 Business Operation	36
3.5.2 Integrating and Configuring an HL7 Business Operation	37
3.6 Pager and Email Alerts	37
4 Additional Steps	39
4.1 Defining Routing Rule Sets for HL7	39
4.2 Defining DTL Data Transformations for HL7	41
4.2.1 Null Mapping Codes	43

4.2.2 Transforming Long Segment Fields	44
4.3 Defining HL7 Search Tables	45
4.3.1 Properties That Are Indexed by Default	45
4.3.2 Examples	46
5 Important HL7 Scenarios	49
5.1 HL7 Acknowledgment (ACK) Mode	49
5.2 HL7 Dual Acknowledgment Sequences	51
5.2.1 Dual ACK Sequence for Incoming Messages	51
5.2.2 Dual ACK Sequence for Outgoing Messages	52
5.2.3 Configuring a Dual ACK Sequence	53
5.3 HL7 Batch Messages	54
5.3.1 Supported Batch Formats	54
5.3.2 Processing Incoming Batch Documents	55
5.3.3 Sending Batch Messages	56
5.3.4 Batch Modes	56
5.3.5 Custom Outbound Batch Handling	56
Reference	59
Settings for HL7 Business Services	60
Settings for HL7 Routing Processes	66
Settings for HL7 Sequence Managers	71
Settings for HL7 Business Operations	75
HL7 Escape Sequences	79

About This Book

This book is one of a set that describes how to build Ensemble productions that route and transform documents in Electronic Data Interchange (EDI) formats. This book explains how to route HL7 Version 2 messages from one application to another using Ensemble as the routing engine. It contains the following sections:

- [Introduction](#)
- [HL7 Schemas and Available Tools](#)
- [Configuring the Production](#)
- [Additional Steps](#)
- [Important HL7 Scenarios](#)
- [Reference](#)

For a detailed outline, see the [table of contents](#).

The following books provide related information:

- [Ensemble Best Practices](#) describes best practices for organizing and developing Ensemble productions.
- [Developing Ensemble Productions](#) explains how to perform the development tasks related to creating an Ensemble production.
- [Configuring Ensemble Productions](#) describes how to configure Ensemble productions, business hosts, and settings. It also provides reference information on settings not discussed in this book.
- [Ensemble Virtual Documents](#) describes the concept of Ensemble virtual documents and provides generic information on working with them.

For general information, see the *InterSystems Documentation Guide*.

1

Introduction

This chapter introduces Ensemble support for HL7 Version 2. It contains the following sections:

- [Ensemble Support for HL7 Version 2](#)
- [An HL7 Version 2 Routing Production](#)

1.1 Ensemble Support for HL7 Version 2

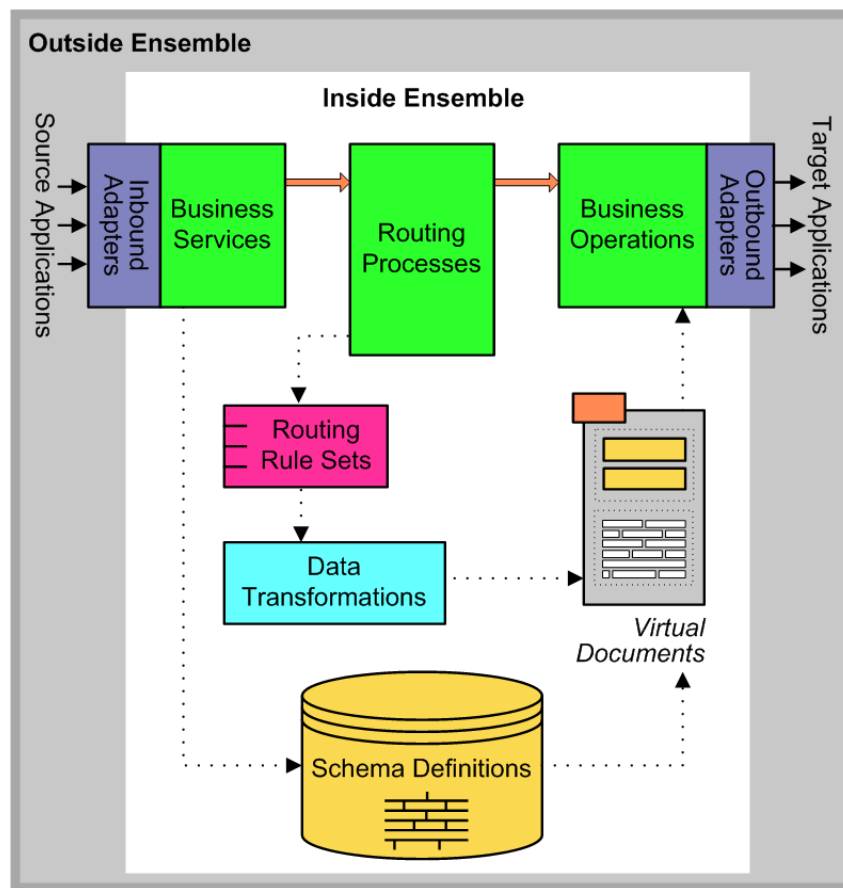
Ensemble supports HL7 Version 2 messages as virtual documents. A *virtual document* is a kind of message that Ensemble parses only partially. This kind of message has the standard Ensemble message header and the standard message properties such as ID, Priority, and SessionId. The data in the message, however, is not available as message properties; instead it is stored directly in an internal-use global, for greater processing speed.

Ensemble provide tools so that you can access values in virtual documents, for use in data transformations, business rules, and searching and filtering messages. For background information, see [Ensemble Virtual Documents](#).

HL7 segment names must be all uppercase.

1.2 An HL7 Version 2 Routing Production

The following figure illustrates the flow of an HL7 Version 2 message through an Ensemble production that works as an HL7 interface routing engine. It shows elements that are referenced by configuration items, but that are not themselves configuration items. These elements include routing rule sets, data transformations, virtual documents, and schema definitions. You create these items when you follow the instructions in this book.



An HL7 Version 2 message flows through a production as follows:

1. An *HL7 business service* receives an incoming message from the specific source application whose messages it is configured to accept.
2. The business service passes the message to a specific *HL7 routing process*. This is an Ensemble business process that prepares incoming messages from a HL7 business service for delivery outside Ensemble via a specific HL7 business operation.
3. The routing process may validate the message against an expected *HL7 schema definition*. This may be a standard HL7 schema or a custom schema.

(Not shown) If validation fails, the HL7 routing process passes the message to its configured *bad message handler*. This is an HL7 business operation that disposes of any incoming HL7 messages that fail validation, usually by saving the messages to a file. It may also enter an error in the Event Log or alert an operator.

4. The HL7 routing process applies a *routing rule set* to the message. The routing rule set chooses one or more target business operations and applies any data transformations that may be needed to prepare the message for the target application.
5. In the typical case, some *data transformation* is required to prepare the message for the target. The routing rule set may invoke transformations that are custom coded, but commonly transformations are created using the Ensemble Data Transformation Language (DTL). DTL can call out to *utility functions* or your own class methods for more complex calculations.
6. When the outgoing message is ready, the routing process passes it to an *HL7 business operation*. The business operation provides the address and framing information required to send HL7 messages to the target application.

(Not shown) By default, all *HL7 messages* that pass through the production are retained in the Ensemble message warehouse for as long as wanted. While in the message warehouse, the contents of HL7 messages are available for

tracking and viewing using Management Portal features such as the [HL7 Message Viewer](#), Message Browser, and Visual Trace, or by issuing an SQL query. You can configure the production to purge old messages automatically or at an administrator's discretion.

2

HL7 Schemas and Available Tools

This chapter provides an overview of the Ensemble tools that you can use to work with HL7 Version 2 schemas and documents. It contains the following sections:

- [Overview of HL7 Schemas](#)
- [Using the HL7 Schema Structures Page](#)
- [Using the Custom Schema Editor](#)
- [Using the HL7 Message Viewer Page](#)
- [Viewing Batch Messages](#)
- [HL7 Classes](#)

2.1 Overview of HL7 Schemas and Messages

Ensemble can process and pass through an HL7 message without using a schema to parse it, but associating a schema with a message allows you to do the following:

- Parse the message and access field values in:
 - Data transformations
 - Routing rules
 - Custom ObjectScript code
- Validate that the message conforms to the schema.

Each HL7 message is identified by a message type, which is specified in the MSH segment MessageType field (MSH:9). Some message types share the same message structure. For example, in HL7 Version 2.3.1, the ADT_A05 message to pre-admit a patient has the same structure as the ADT_A01 admit message. The schema specifies that the ADT_A05 message has the structure type ADT_A01.

In order to parse an HL7 message, Ensemble needs two pieces of information:

- Schema category—this is the HL7 version number, such as 2.3.1 or 2.7, or it may be a category for a custom schema defined in Ensemble. Ensemble gets the schema category from the business service **Message Schema Category** setting or from the Data Transformation settings. Although the HL7 message includes a schema version number in the MSH

segment VersionID field (MSH:12), Ensemble does not use this value because many applications do not set this field consistently.

- Structure type—Ensemble gets the message type from the MSH:9 field and then checks the schema definition to get the structure type for that message.

Ensemble uses the MSH:9.3 subfield to qualify the message type in some cases. The MSH:9.3 subfield is used in HL7 messages in two ways: 1) as a modifier to the message type, or 2) to specify the structure type. If MSH:9.3 modifies the message type, typically as a numeric digit, Ensemble includes it as part of the message type. If MSH:9.3 specifies a structure type, such as ADT_A01, Ensemble ignores it in both determining the message type and setting the Name property. Ensemble does not need the MSH:9.3 subfield to determine the structure type because it gets the structure type from the schema.

When a business service or Data Transformation creates an `EnsLib.HL7.Message` object to store an HL7 message, it combines the schema category and structure type and stores it in the `DocType` property using this syntax:

category:structureType

For example, valid `DocType` values for category 2.3.1 include `2.3.1:ACK`, `2.3.1:ADT_A17`, `2.3.1:BAR_P01`, and `2.3.1:PEX_P07`. The message type, which can be different from the structure type is stored in the `Name` property.

If you create an `EnsLib.HL7.Message` object in ObjectScript code, you should set the `DocType` and `Name` properties based on the value in the MSH:9 field.

The HL7 standard allows local extensions, such as trailing Z segments. These segments are not defined in the base schema categories. If you want to access a field in a custom Z segment in a data transformation, routing rule, or ObjectScript, you need to define a custom schema category that specifies the extended message. See “[Using the Custom Schema Editor](#)” for details on defining a custom schema.

2.2 Using the HL7 Schema Structures Page

The HL7 Schemas page enables you to import and view HL7 Version 2 schema specifications.

To display this page, navigate to **Ensemble > Interoperate > HL7 v2.x > HL7 v2.x Schema Structures** from the Home page. For general information on using this page, see “[Using the Schema Structures Page](#)” in *Ensemble Virtual Documents*.

The HL7 Schemas page provides an additional tab: Message Types. This tab identifies two message structures as a request/response pair.

Ensemble includes the following HL7 schema versions:

- 2.1
- 2.2
- 2.3
- 2.3.1
- 2.4
- 2.5
- 2.5.1
- 2.6
- 2.7
- 2.7.1

For information on creating and editing custom schema categories, see “[Using the Custom Schema Editor](#)”.

Note that HL7 Version 3 is not listed as an HL7 schema category, because it uses entirely different data conventions from HL7 Version 2. For details, see the [Ensemble HL7 Version 3 Development Guide](#).

The following example shows how to use this page in more detail.

2.2.1 Viewing a List of Document Types

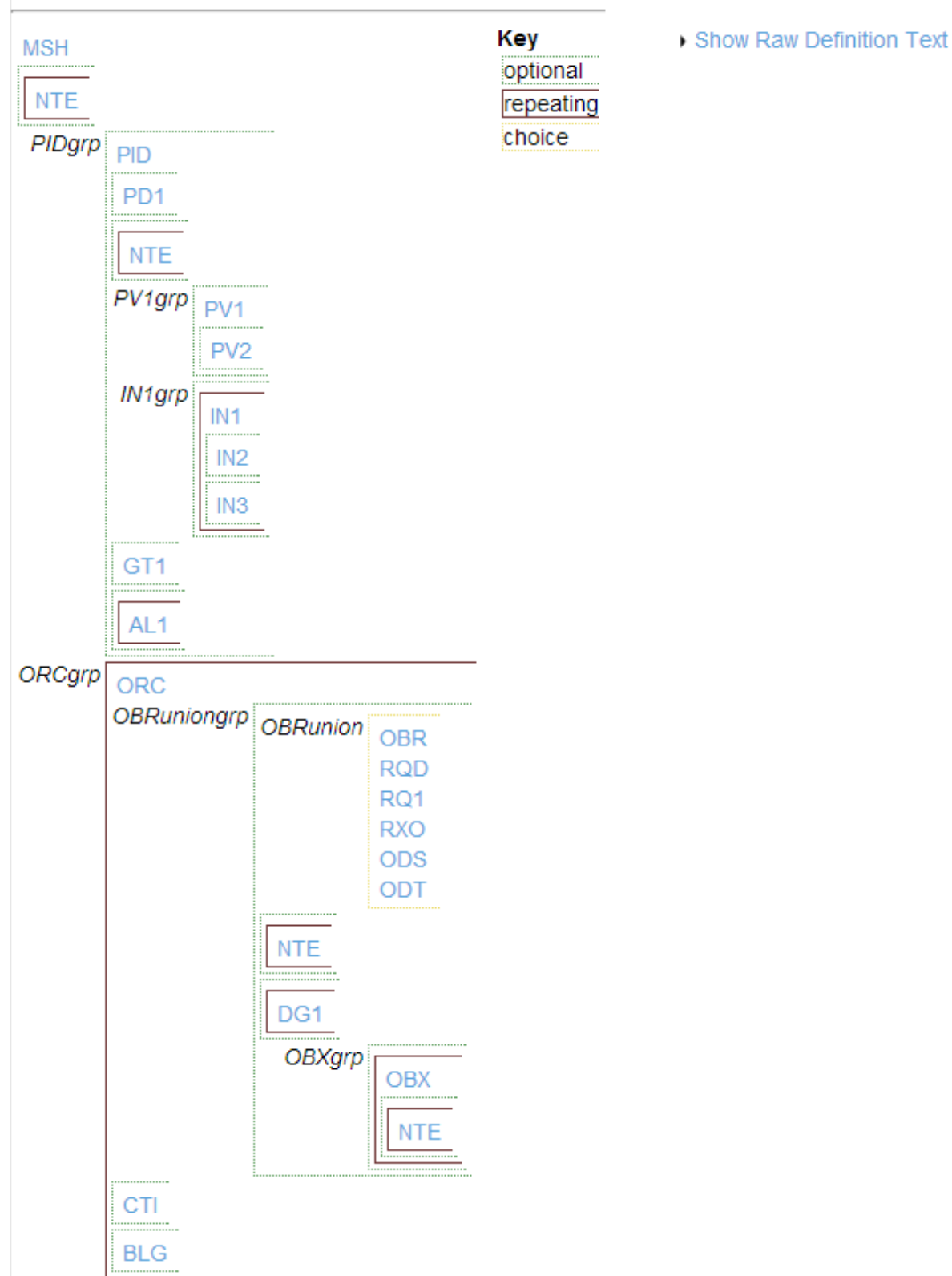
To list all the document type structures in a category, first select the category and then click the **DocType Structures** tab.

2.2.2 Viewing a Message Structure

To view the internal organization of a message structure, click its name from the **DocType Structures** tab on the HL7 Schemas page. Ensemble displays the segment structure of the message using the system of visual cues explained below. This is the HL7 Schema Message Structure page. The following example shows the 2.3.1:ORM_O01 message structure.

Schema Category: **2.3**Message Structure: **ORM_O01**

2.3:ORM_O01 Pharmacy/treatment order message - Order message (also RDE, RDS, RGV, RAS,



The visual conventions on this page are as follows:

- The segments that comprise the message structure are listed in sequential order, from left to right and top to bottom.
- The three-letter name of each message segment is displayed: MSH, NTE, PID, etc. This name indicates the type of segment that exists at this location in the HL7 message structure.
- Segment names must be all uppercase.
- Green dotted lines enclose segments, groups, or fields that are optional.
- Brown solid lines enclose segments, groups, or fields that, if present, may repeat several times.
- Yellow dashed lines enclose a choice: this is a *union* of segments. Only *one* segment from the union can appear at this location within the message structure. It may be any of the segments listed.
- A segment, group, or field may be both repeating and optional (see any NTE above).
- To see the message structure in a raw text format, click **Show Raw Definition Text**.

When you are viewing a segment diagram, if you hover the cursor over a three-letter segment name, a tooltip displays the syntax for referring to this segment in a [virtual property path](#).

The message structure name displayed before the diagram has two parts, separated by a colon:

category:structure

Where:

- *category* is the name of a schema category, such as 2.3.1.
- *structure* is the name of a message within that schema category.

2.2.3 Viewing a Segment Structure

To view the structure of a message segment, click on its name in any page similar to the example shown in the [previous section](#). Ensemble displays a table that lists all the fields in that segment. This is the HL7 Schema Segment Structure page.

For example, if you click the **PR1** segment in the 2.3:ADT_A01 message structure, Ensemble displays the following page.

Schema Category: [2.3](#)
 Message Structure: [ADT_A01](#)
 Segment Structure: **PR1**

Path you followed to get to this Segment Structure: **PR1grp().PR1**

2.3:PR1 Procedures

Field	Description	Property Name	Data Structure	Symbol	Repeat Count	Minimum Length	Maximum Length	Required	Repeating	Code Table	Alternate Description
1	Set ID - Procedure	SetIDProcedure	2.3:SI	! (exactly one required)			4	R	0		
2	Procedure Coding Method	ProcedureCodingMethod	2.3:IS	! (exactly one required)			2	R	0	2.3:89	
3	Procedure Code	ProcedureCode	2.3:CE				80	O	0	2.3:88	
4	Procedure Description	ProcedureDescription	2.3:ST				40	O	0		
5	Procedure Date/Time	ProcedureDateTime	2.3:TS				26		0		
6	Procedure Type	ProcedureType	2.3:ID	! (exactly one required)			2	R	0	2.3:230	
7	Procedure Minutes	ProcedureMinutes	2.3:NM				4	O	0		
8	Anesthesiologist	Anesthesiologist	2.3:XCN	* (zero or more)			120	O	1	2.3:10	
9	Anesthesia Code	AnesthesiaCode	2.3:IS				2	O	0	2.3:19	
10	Anesthesia Minutes	AnesthesiaMinutes	2.3:NM				4	O	0		
11	Surgeon	Surgeon	2.3:XCN	* (zero or more)			120	O	1	2.3:10	
12	Procedure Practitioner	ProcedurePractitioner	2.3:XCN	* (zero or more)			230	O	1	2.3:10	
13	Consent Code	ConsentCode	2.3:CE				60	O	0	2.3:59	
14	Procedure Priority	ProcedurePriority	2.3:NM				2	O	0		
15	Associated Diagnosis Code	AssociatedDiagnosisCode	2.3:CE				80	O	0		

The columns are as follows:

- **Field** — the number to use to access the field within the segment (if you prefer numbers).
- **Description** — a short description of the field.
- **Property Name** — the name to use to access the field within the segment (if you prefer names).
- **Data Structure** — for more complex field values that use a data structure, you need further syntax details before you can complete the *segment:field* virtual property path. You can get this by clicking on the name in this column
- **Symbol** — indicates the syntax rules for the field. The characters in this column indicate whether you can expect this field to be present, absent, or repeated within the message segment. Possible values are:

Symbol	Meaning
!	(1 only) The field is required; it must occur only once.
?	(0 or 1) The field is optional, but if it occurs, it may occur only once.
+	(1 or more) The field may repeat one or more times.
*	(0 or more) The field may repeat zero or more times.
&	The field may be present, and may repeat, but only under certain conditions.
<i>n</i> *	(0 to <i>n</i>) The field repeats a maximum of <i>n</i> times.

- **Repeat Count** — the maximum number of times the field can repeat (if it repeats, and if there is a maximum).

- **Minimum Length** — the minimum number of characters in the field. Each repeat of the field must contain this number of characters.
- **Maximum Length** — the maximum number of characters in the field. Each repeat of the field may contain this number of characters.
- **Required** — displays R for required, O for optional.
- **Repeating** — displays 1 for true, 0 for false.
- **Code Table** — click on an entry to view the valid codes that may be entered in this field.
- **Alternate Description** — a second, longer description of the field.

You can use this information, particularly the **Property Name** column, to build virtual property paths for Ensemble in the format *segment:field*. The following are examples of virtual property paths involving simple *field* values from the **PR1** segment in the 2.3:ADT_A01 message structure. The () shortcut syntax indicates all available instances of a repeating field, whereas (1) indicates the first instance:

```
PR1grp().PR1:ProcedureType
PR1grp().PR1:ProcedureCode()
PR1grp().PR1:ProcedureCode(1)
PR1grp().PR1:ProcedureCode(x)
PR1grp().PR1:ProcedurePriority
```

2.2.4 Viewing a Data Structure

When you click on a name in the **Data Structure** column, Ensemble displays all the fields in that data structure. This is the HL7 Data Structure page. The following columns of the display are most useful:

- The **Component** column lists the numbers you can use to access fields within the segments (if you prefer numbers).
- The **Property Name** column lists the names you can use to access fields within the segments (if you prefer names).
- Click on an entry in the **Data Structure** column (if any) to drill down for detail.
- Click on an entry in the **Code Table** column (if any) to view the valid codes that may be entered in this field.

The following sample page appears when you click the **Data Structure** item called 2.3:XCN in the segment structure page above. The page states that the category 2.3 data structure XCEN describes an “Extended Composite ID number and name” and consists of fourteen fields. Of these, some are simple values, some are data structures, and some are codes.

Schema Category: [2.3](#)

Message Structure: [ADT_A01](#)

Segment Structure: [PR1](#)

Field Number: **8**

Data Structure: **XCN**

Path you followed to get to this Data Structure: **PR1grp().PR1:Anesthesiologist()**

2.3:XCN Extended Composite ID number and name (2.8.46)

Component	Description	Property Name	Data Structure	Minimum Length	Maximum Length	Required	Code Table	Alternate Description
1	ID number (ST)	IDnumberST	2.3:ST					
2	family name	familyname	2.3:ST					
3	given name	givenname	2.3:ST					
4	middle initial or name	middleinitialname	2.3:ST					
5	suffix (e.g., JR or III)	suffix	2.3:ST					
6	prefix (e.g., DR)	prefix	2.3:ST					
7	degree (e.g., MD)	degree	2.3:ST					
8	source table	sourcetable	2.3:ID				2.3:297	
9	assigning authority	assigningauthority	2.3:HD					
10	name type	nametype	2.3:ID				2.3:200	
11	identifier check digit	identifiercheckdigit	2.3:ST					
12	code identifying the check digit scheme employed	codeidentifyingthecheckdigit	2.3:ID				2.3:61	
13	identifier type code	identifiertypecode	2.3:IS				2.3:203	
14	assigning facility ID	assigningfacilityID	2.3:HD					

Given this information, you can create virtual property paths for the complex PR1grp().PR1:Surgeon field in the message structure 2.3:ADT_A01 as follows:

```
PR1grp().PR1:Surgeon.familyname
PR1grp().PR1:Surgeon.degree
```

2.2.5 Viewing a Code Table

When you click on a name in the **Code Table** column, it lists and explains the valid codes for that field. This is the HL7 Code Table page. The following sample page appears when you click the **Code Table** item called 2.3:200 in the data structure page shown in the [previous section](#).

Schema Category: **2.3**

Code Table: **200**

Description: **Name Type**

Type: **2** (HL7)

Code	Meaning
L	Legal Name
O	Other
M	Maiden Name
A	Alias Name
C	Adopted Name
D	Display Name

The example above shows that the category 2.3 code table 200 describes a “Name type” that can have the value L, O, M, A, C, or D.

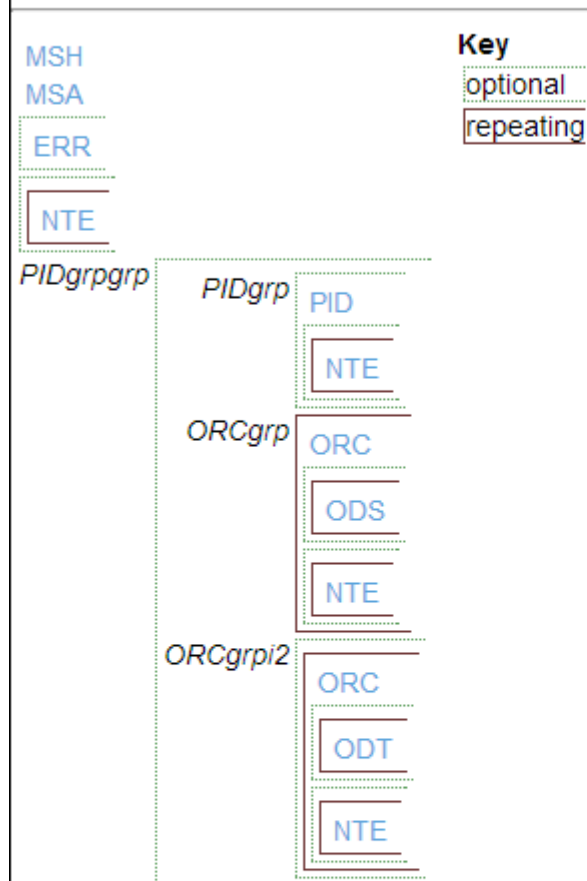
This means that if you have an HL7 message with a DocType of 2.3:ADT_A01, it has an optional virtual property with the path `PR1grp().PR1:Anesthesiologist.nametype` that can contain one of the following values: L, O, M, A, C, or D.

Note: For details, see “[Syntax Guide](#)” in *Ensemble Virtual Documents*.

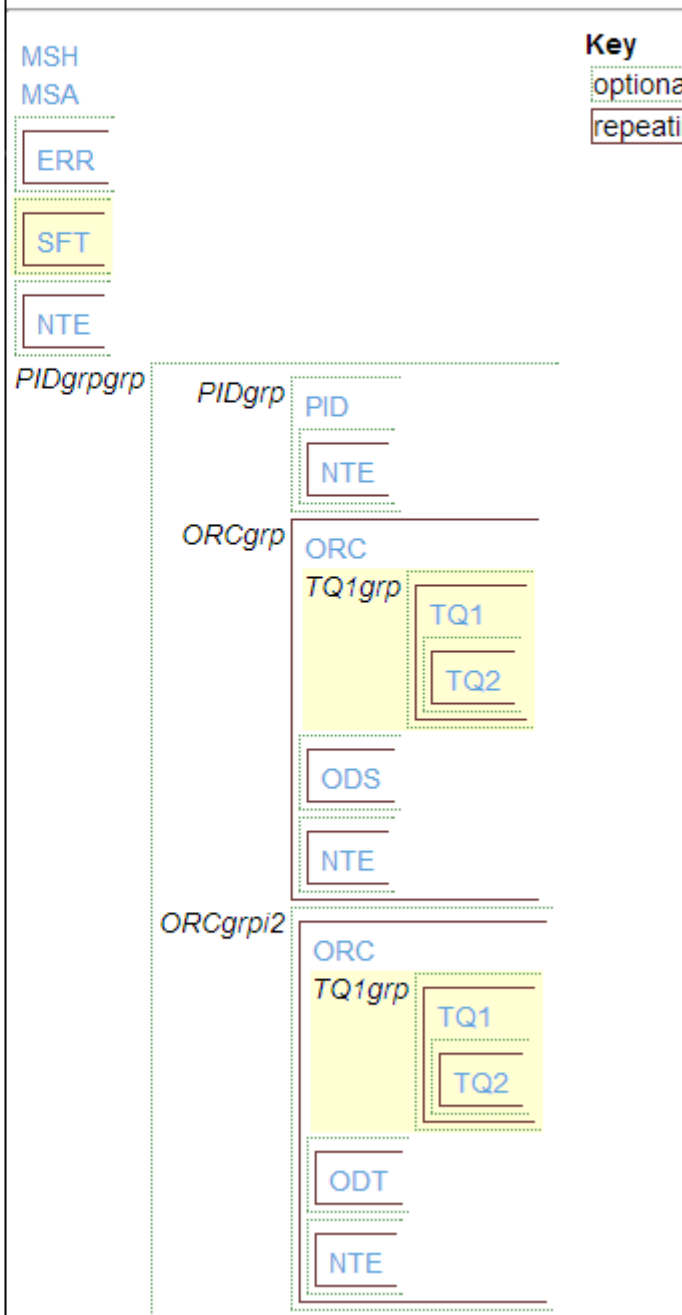
2.2.6 Choosing a Different Category

It is a feature of the HL7 standard that a message structure can differ by HL7 version, even when the structure has the same name and number. For example, both HL7 2.4 and HL7 2.5 define a message structure called ORD_O04, but these definitions contain different segments. Ensemble provides the message structure definitions 2.4:ORD_O04 and 2.5:ORD_O04. The HL7 Message Structure page makes it easy to see the differences between the two definitions, as the following two figures show.

Schema Category: **2.4**
 Message Structure: **ORD_O04**
 2.4:ORD_O04 Dietary order acknowledgment message
 - Diet order acknowledgment ▶ [Show Raw Definition Text](#)



Schema Category: **2.5**
 Message Structure: **ORD_O04**
 2.5:ORD_O04 Dietary order acknowledgment message
 - Diet order acknowledgment ▶ [Show Raw Definition Text](#)



2.3 Using the Custom Schema Editor

The Custom Schema Editor allows you to create a new custom HL7 schema or edit an existing custom HL7 schema. Typically, a custom schema has a base schema, which is a standard schema or another custom schema. When Ensemble is

using the custom schema to parse a message, if the message type, segment, or other element is not defined in the custom schema, it uses the definition in the base schema. Consequently, you only have to define the elements in the custom schema that are not present in the base schema or require a different definition from the one in the base schema. You cannot edit a standard schema.

The most common reason to define a custom schema is to be able to parse HL7 messages with trailing Z segments. Ensemble can handle messages with trailing Z segments that are not defined in the schema, but to do any of the following you need to define a custom schema:

- Access field paths in the trailing Z segments in a routing rule, data transformation, or ObjectScript code.
- Validate the trailing Z segments.

If you have a production that is currently using a standard schema and you need to access trailing Z segment field paths in a data transformation or routing rule, you should do the following:

1. Use the Custom Schema Editor in the Management Portal to create a new HL7 schema. Enter a name for the custom schema and specify the base schema.
2. Define the Z segment that can appear in your message. If your Z segment has similar fields to an existing segment in the base schema, you can copy the definition from the base and then modify it as needed. Otherwise, you can create a new segment. You can add fields, delete fields, or change the order of fields.
3. For each message type that includes the trailing Z segments, create a message type and structure type in the custom schema that is copied from the underlying schema. Add the Z segment to the end of the structure type.
4. Modify the business service in your production to use the new custom schema instead of the base schema.
5. Test the production by supplying new messages with the trailing Z segments to the production's business services. If you view the messages in the message viewer, the Z segments will be shown in blue if they are defined in the schema. Unrecognized segments are shown in black.

Detailed instructions are in the following sections.

2.3.1 Creating a New Custom Schema

To start the Custom Schema Editor from the Management Portal, select **Ensemble**, **Interoperate**, **HL7 v2.x**, and **HL7 v2.x Schema Structures**.

To create a new HL7 schema, click **New**. In the Custom Schema Editor, select the base schema, schema name, and, optionally, a schema description. For example, to define a custom schema based on the Version 2.5 standard schema category, you could enter the following:

Custom Schema Wizard

Standard | Message Types | DocType Structures | Segment Structures

Custom Schema Wizard
Create a new custom schema definition.

Base Schema: 2.5

Schema Name: Custom2.5

Schema Description:

Use this form to create a new custom schema category.

For help with any field in this form, hover the cursor over the field name.

Cancel OK

Once you have created the custom schema, the Custom Schema Editor presents you with an empty schema, and you can define the message types, structure types, segment structures, data structures, and code tables in it. You have to define only the elements that have different definitions than those in the base schema or that are not defined in the base schema.

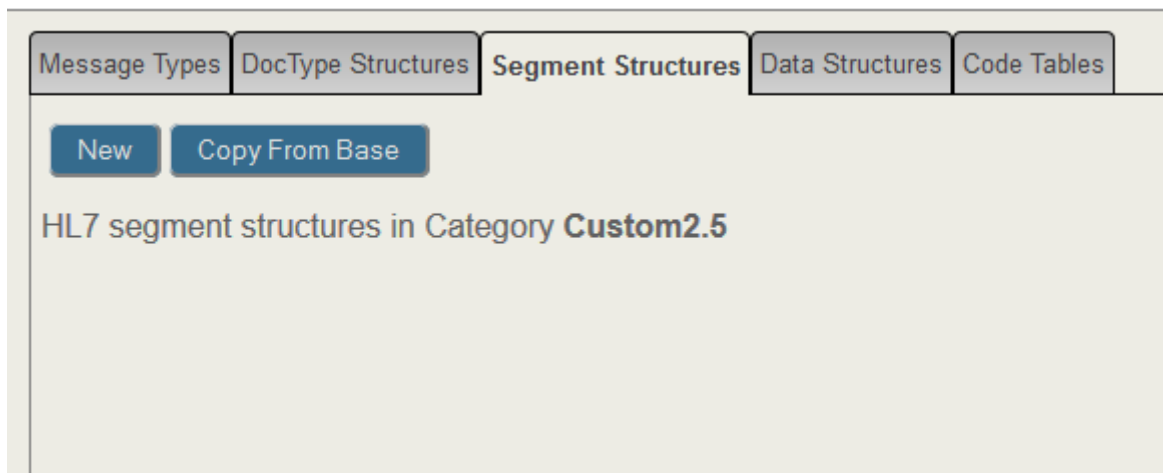
The Custom HL7 Schema Editor has the following tabs:

- **Message Types**
- **DocType Structures**
- **Segment Structures**
- **Data Structures**
- **Code Tables**

On each of these tabs, you can copy the elements from an existing base definition by selecting **Copy From Base**. This allows you to create a message type or other element that is an extension of an existing definition without having to reenter the common definitions.

2.3.2 Defining a New Segment

To define a new Z segment, select your new custom schema in the left panel and then select the **Segment Structures** tab. For custom schemas, the **Segment Structures** tab has the **New** and **Copy From Base** buttons and lists the segments that are currently defined in the custom schema. Since the schema doesn't have any segments defined, the segment structure list is empty. If you are viewing a standard schema, you cannot add new segments and the buttons are not present.



If you want to define a new segment without copying fields from another segment, click the **New** button and the wizard creates an empty segment. Name the segment and click the **Add Field** button. The wizard creates an empty field and you can fill in the form. For example:

Custom Schema Category Custom2.5
Base Schema Category 2.5

New Segment Structure Name
ZZR

Description

Segment Fields

Field	Description	Property Name	Data Structure	Symbol	Count	Min Length	Max Length	Required	Repeating	Code Table	Alternative Description
1								O	No		

Add Field

If you are creating a segment that is very similar to an existing segment in the base schema, you should choose **Copy From Base**, it creates a new segment with the same fields as the specified segment. The Custom Segment Structure Wizard displays the copied fields and has a **New** button after the fields to create a new field. For example, the following segment was copied from the PID segment.

Custom Schema Category Custom2.5
Base Schema Category 2.5

Segment Structure to Copy
PID (2.5)

New Segment Structure Name
ZPI

Description
Patient Identification

Segment Fields

Field	Description	Property Name	Data Structure	Symbol	Count	Min Length	Max Length	Required	Repeating	Code Table	Alternative Description
1	Set ID - PID	SetIDPID	SI				4	O	No		
2	Patient ID	PatientID	CX				20	B	No		
3	Patient Identifier List	PatientIdentifierList	CX	+			250	R	Yes		
4	Alternate Patient ID - PID	AlternatePatientIDPID	CX	*			20	B	Yes		
5	Patient Name	PatientName	XP	+			250	R	Yes		

Once you have creating the segment either as a new segment or as a copy of a base segment, you can add or update fields as follows:

- Click the **Add Field** button to add a field at the end of the segment.
- Update the form text boxes that define the segment field. You cannot edit the **Property Name** text box. The wizard sets the property name based on the field **Description** value after it removes any spaces or special symbols. The property name is set when you click **OK** to end the wizard.
- Change the order of a field by clicking on the up or down arrows.
- Delete a field from the segment by clicking the red X.

When you have completed entering the fields, click **OK** to save the segment.

You can edit any saved segment in a custom schema by clicking on the segment name and then clicking **Edit**.

Once you have defined your Z segments, you should define the message types and structure types that contain the Z segments.


2.3.3 Defining a New Message Type and Structure Type

The Message Type identifies the message and matches the value in the HL7 MSH:9 field. When you define a message type, you specify the sending message structure type, which may be the same as the message type, and the return type. But you specify the segments that can be in the message in the structure type not in the message type. When you create a message type, you can optionally create a structure type at the same time.

To add a Z segment to a message type that is defined in the base schema, copy the message type and structure to the custom schema and then add the trailing Z segment to the structure type. For example, to add the ZPI segment to the ORU_R01 message in a custom schema that has Version 2.5 as the base schema, do the following:

1. Select your custom schema in the left panel, select the **Message Types** tab, and click **Copy From Base**.
2. Select the ORU_R01 message type in the **Message Type to Copy** pull-down. The wizard fills in the new message type name to be the same as the copied message type and sets the **Sending Message Structure** and **Returning Message Type** to match the definition in the base. By checking the box, you automatically create the sending message structure in the custom schema if it is not yet defined. It is created by copying the structure from the base schema.

Custom Message Type Wizard

 **Custom Message Type Wizard**
Create a custom message type definition

Custom Schema Category Custom2.5
Base Schema Category 2.5

Message Type to Copy ORU_R01 (2.5)

New Message Type Name ORU_R01

Message Type Description

Sending Message Structure ORU_R01 (2.5)

☒ Create this message structure in the Custom Schema if it does not exist there already

Returning Message Type ORU_R01 (2.5)

Use this form to create or edit a custom message type.

For help with any field in this form, hover the cursor over the field name.

Cancel OK

After you click **OK**, the ORU_R01 message type and the ORU_R01 structure type are defined in the custom schema.

- Click the **DocType Structures** tab and the ORU_R01 structure type. The custom schema editor displays the graphical representation of the structure type. Click the **Edit** button.

The Custom Message Structure Wizard displays the following:

- Message Structure Name
- Description

- **Raw Definition**—To edit the structure, you edit the raw definition and then click **Save**. The raw definition uses the same convention as Studio to describe the structure. Each element is separated by ~ (tilde character), optional segments are indicated by [] (square brackets) and repeating segments are indicated by <> (angle brackets).
- **All Segments**—This list contains all segments defined in the custom schema and base schema. You can type the segment names listed into the raw definition.
- **Visual Representation**—This graphic description of the structure is updated whenever you save the raw definition.

Update the raw definition by entering ~[~ZPI~] at the end of the definition to indicate that the trailing Z segment is optional.

4. Click **Save** to save the raw definition. The wizard updates the visual representation. Click **OK** to end the wizard.

For example, if you are using the Custom Message Structure Wizard to edit a copy of the ORU_R01 structure type, the wizard displays the following:

The **Undo** button reverts your previous key stroke. The **Save** button saves the raw definition. The **Previously Saved** button undoes all changes and reloads the last saved definition. The **Legend** button displays a help message that explains the raw definition syntax.

When you are extending a message definition from the base schema, you should use the same segment and structure name as specified in the base schema.

Note: Once you have defined a message structure in the custom schema, that definition is used for all message types that share the same structure. For example, if you add the ZPI trailing segment to the ORU_R30 structure, the trailing Z segment is allowed in the ORU_R30, ORU_R31, and ORU_R32 message types because they all share the same ORU_R30 structure. It is not necessary to include the message types in the custom schema. The definition from the base schema will use the structure type from the custom schema.

2.3.4 Editing Data Structures and Code Tables

Data structures provide a mechanism to specify a field that has a structured value rather than a simple data type, and code tables provide a mechanism to define a set of allowed values for a field. Typically, data structures and code tables are defined by the HL7 standard body and are not defined as custom extensions. The Custom HL7 Schema Editor does allow you to define data structures and code tables in your custom schemas in the rare cases where this is needed. The wizard to edit data structures is very similar to the wizard to edit segments. The wizard to edit code tables allows you to define codes and descriptions for the code table. The codes specify the values that can be used in a field.

2.4 Using the HL7 Message Viewer Page

Ensemble provides a Message Viewer page for HL7. You can use this page to display, transform, and export HL7 messages (either external files or messages from the Ensemble message archives).

To access this page:

1. Click **Ensemble**.
2. Click **Interoperate**.
3. Click **HL7 v2.x**.
4. Click **HL7 v2.x Message Viewer** and then click **Go**.

2.4.1 Selecting Options

To specify the document to display:

1. For **Document Source**, select **File**, **Message Header ID**, or **Message Body ID**.
2. Specify the document to display:
 - If you selected **File**, use **Browse** to choose a file. For **Document Number in File**, type the number of the document to display.
 - If you selected **Message Header ID** or **Message Body ID**, type the ID of the message header or message body to display.
3. Specify how to parse the document. To do so, select one of the following options for **Document Structure or Schema**:
 - **As received by a business service** — Use the schema as assigned by a business service. If you select this, select a business service from the drop-down list.
 This option enables you to determine the DocType to which a particular business service would assign this document.
 - **Use a specific Schema Category/Version** — Choose a document category from the drop-down list.
 - **Use a specific DocType** — Enter the name of a document structure (<MessageStructure>) in the format *category: structure*. The parser uses this document structure.
 - **Use content-declared Version:Name** — Use the document structure associated with the document type declared in the document.
 - **Use object's stored DocType** — Use the DocType as declared in the document body object. (This option does not apply to stored documents loaded from a file.)

- **None** — Do not use any DocType to parse the document. Instead, display the raw segments without transforming any of them into links.

This option enables you to try interpreting documents from a particular data source as different schema category types to determine which DocType is the right one to use when handling documents from that source. There are a variety of reasons why you might need to do this. For example, you might find when you update an external application that it has changed the actual version of the documents it sends, but has neglected to update the type declaration that it sends in these documents. It is also useful in determining which of the built-in categories to use as a schema base, when a document uses a custom document structure.

4. Optionally click **Transform Document?** and specify the transformation details. See [Testing a Transformation](#).
5. Click **OK**.

2.4.2 Parsing the Message

The Message Viewer displays the following on the right side of the screen after completing the steps above:

- Summary Report, which contains following basic information about the document:
 - The Data Transformation applied, if applicable
 - The Message ID
 - The DocType
 - The DocType Category
 - The DocType description, if available
 - The number of segments
 - The number of child and parent documents, if applicable
- Message Data, which has one row for each segment in the message structure. Each row contains:
 - Segment number
 - Segment name, such as PID or NTE
 - Field contents and separators, as contained in the message

If the message matches the schema you have selected, segments and elements will appear in blue, as seen below. Clicking on the segments or fields will link to the relevant structure page.

1	MSH ^~& EPIC_EC CCF PHARMACY 2623735 20050126150624 11111 ORM^O01
2	PID _ 16284718^_^-^_^-^AM Z1907 _ JONES^IVAN^S^A^_^-^_^-^ _ 19490518 M
3	ORC NW 244674^EPC _ _ _ _ _ _^-^_^-^200501260000^200503272359^Norm^_ _
4	RXO 0045--04-52-2^PRILOSEC 325 MG PO TABS^NDC 60 _ _ _^-^_ _ _ 2 tabs po q4-6
5	NTE 1 _ _ _
6	ORC NW 244674^EPC _ _ _ _ _ _^-^_^-^200501260000^200503272359^Norm^_ _
7	RXO 0045--04-52-2^WELLBUTRIN 325 MG PO TABS^NDC 60 _ _ _^-^_ _ _ 2 tabs po q
8	NTE 1 _ _ _

To display the segment address, hover the cursor over a segment name in the shaded column. The tooltip displays the following:

- [illegible]

To display the field address, hover the cursor over a field within the message structure. The tooltip displays the following:

- | Symbol | Meaning |
|--------|-----------------------------------------------------------------------------------------------------------------------|
| ! | (1 only) The field is required; it must occur only once. |
| ? | (0 or 1) The field is optional, but if it occurs, it may occur only once. |
| + | (1 or more) The field may repeat one or more times. |
| * | (0 or more) The field may repeat zero or more times. |
| & | The field may be present, and may repeat, but only under certain conditions. |
| n^* | (0 to n) The field repeats a maximum of n times. |
| (m) | m is the maximum number of characters in the field. Each repeat of the field may contain this number of characters. |

^AWELLBUTRIN 325 MG PO TABS^ANDC|60|_|.
 1.2.1 / RequestedGiveCode.text.1 / ?(100)

2.4.2.3 Batch Messages

If a field is enclosed with angle brackets (< , >) it is a link to a sub-document. Click on it to view that document's summary report and message data.

Also see “[Viewing Batch Messages](#),” later in this chapter.

2.4.3 Testing a Transformation

To test a transformation:

1. Click **Transform Document?**.
2. For **Choose Data Transformation**, select a data transformation.
3. For **Choose Display Option**, select one of the following:
 - **Transformation Result Only** — Display only the transformed document.
 - **Original Message and Result Together** — Display both the original document and the transformed document.
4. Now do either or both of the following:
 - Click **OK** to display the transformed document.
 - Click **Save Result To File?** to save the transformed document to a file. In this case, also specify a path and filename.

The default directory is the management directory for the active namespace. For example, if you installed Ensemble into the directory C:\MyCache and your current namespace is ENSDEMO, the file is saved as C:\MyCache\Mgr\ENSDEMO\filename

2.5 Viewing Batch Messages

The HL7 Document page handles a message differently if it is a group of HL7 messages in batch format, rather than a single HL7 message. Specifically, it allows you to walk through the batch message structure one level at a time.

The following display is the result of asking to view a batch message that begins with an FHS segment. Ensemble parses the batch message and finds that it has 3 segments: FHS, FTS, and a block of child documents in between. The block contains two child documents; each beginning with BHS and ending with BTS. The message is a two-level batch message.

The Message Viewer assigns the child documents the identifiers <2> and <33>. It displays the top-level parent document, using links (<2> and <33>) to represent the two child documents. The display is as follows:

This is HL7 Message #1 found in file 'C:\EnsProjectFiles\HL7_practice\HL7safe\XYZ30Batch2'
 DocType '2.3.1:FHS' based on 'Category / doc.Name'

HL7 FHS Message - Id = 1, DocType = '2.3.1:FHS', MessageTypeCategory = '2.3.1'
 'batch message file', 3 Segments, 2 child documents

1	FHS	1 : 18 : 1	
2	BHSDocsRef : 2 : 2	BHS documents : <2> <33>	
3	FTS		

The next display is the result of clicking the child document link <2> in the previous display. This example is a two-level batch message, so the child document <2> has children of its own: child documents <3> through <32>.

This is the HL7 Message object with Id **2**
DocType " based on '**HL7 Message stored DocType property**'

The next display is the result of clicking the child document link <6> in the previous display. Since this is the lowest level of the batch message hierarchy, message <6> is a normal HL7 Version 2 message that begins with an MSH segment.

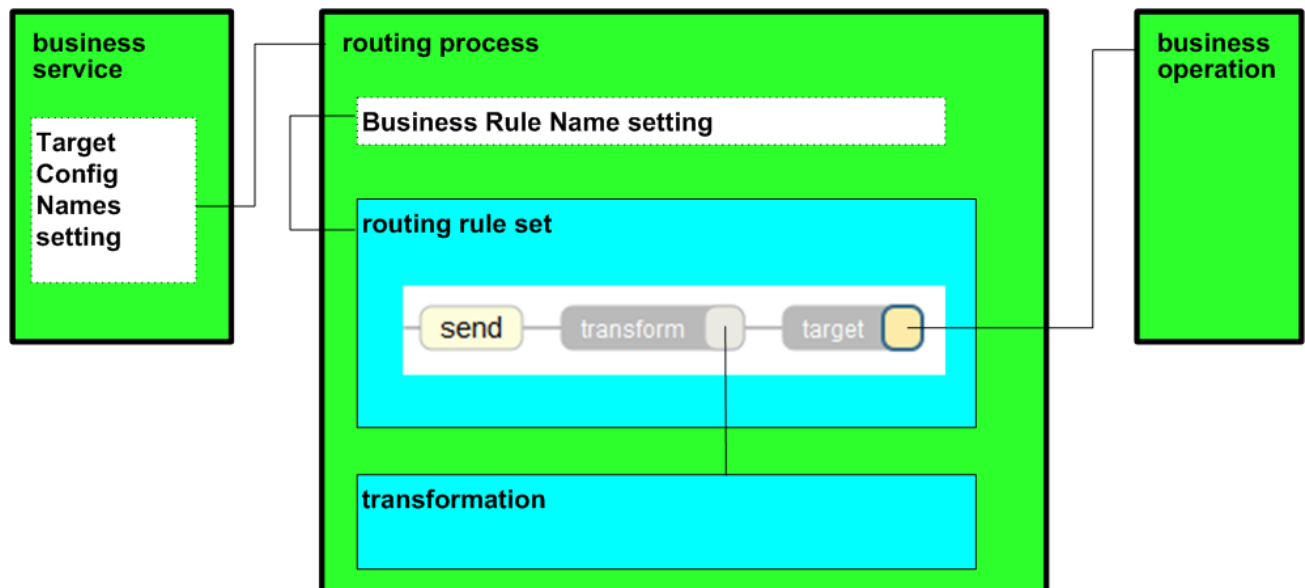
When you are done viewing messages in the batch message hierarchy, you can close all the pop-up browser windows until the top-level parent document remains in the original Message Viewer window. From here, you may return to other Management Portal activities.

For reference, this section lists the classes that Ensemble provides to enable you to work with HL7 Version 2 documents.

Item	Classes	Notes
Business services	<ul style="list-style-type: none"> EnsLib.HL7.Service.FileService EnsLib.HL7.Service.FTPService EnsLib.HL7.Service.HTTPService EnsLib.HL7.Service.SOAPService EnsLib.HL7.Service.TCPService 	Each of these HL7 business service classes uses a different adapter, as indicated by the class name. The HL7 HTTP service can either use the CSP port or a special port.
Business processes	EnsLib.HL7.MsgRouter.RoutingEngine	This class is a specialized version of the standard virtual document routing process.
Business operations	<ul style="list-style-type: none"> EnsLib.HL7.Operation.FileOperation EnsLib.HL7.Operation.FTPOperation EnsLib.HL7.Operation.TCPOperation 	Each of these HL7 business operation classes uses a different adapter, as indicated by the class name.
Messages	EnsLib.HL7.Message	This is a specialized message class to carry HL7 documents as Ensemble virtual documents.
Search tables	EnsLib.HL7.SearchTable	This is a specialized search table class for HL7 documents.

You can also create and use subclasses of these classes.

The business host classes include configurable targets. The following diagram shows some of them:



For information on other configurable targets, see “[Reference](#).”

2.6.1 Details on the HL7 Message Class

Ensemble provides a built-in class for HL7 Version 2 virtual documents. The class is `EnsLib.HL7.Message`. For basic information on virtual document message classes, see “[Virtual Document Classes](#)” in *Ensemble Virtual Documents*. In addition to the basic properties and methods, `EnsLib.HL7.Message` provides the following properties:

TypeCategory

The `TypeCategory` property contains an HL7 category name. Typically, the HL7 business service that receives HL7 data from outside Ensemble instantiates an HL7 message and assigns a `TypeCategory` value to it. Ensemble combines this `TypeCategory` with the message type declared in the MSH segment of the incoming message data; this combination identifies a `<MessageType>` within the HL7 schema definition. This `<MessageType>` has an associated `<MessageStructure>` that Ensemble uses as the `DocType` for the HL7 message, if no other `DocType` is assigned.

Name

The `Name` property is a read-only string that contains the HL7 message structure name (such as `ADT_A08` or `ORM_O01`) that the external data source has provided in the MSH segment. The `Name` can be useful in determining the HL7 message structure that the clinical application *thinks* that it is sending, although this can differ from the actual message contents.

3

Configuring the Production

This chapter describes how to configure a production to include an HL7 routing interface. It also describes how to create a new HL7 routing production, if you do not yet have a production. This chapter includes the following topics:

- [Creating a New HL7 Routing Production](#)
- [Adding HL7 Business Services](#)
- [Adding HL7 Routing Processes](#)
- [Adding HL7 Sequence Managers](#)
- [Adding HL7 Business Operations](#)
- [Pager and Email Alerts](#)

This chapter discusses tasks that you perform on the Production Configuration page. The following chapter describes additional tasks.

3.1 Creating a New HL7 Routing Production

You can create a new HL7 routing production as follows:

1. In the Management Portal, switch to the appropriate namespace.
To do so, click **Switch** in the title bar, click the namespace, and click **OK**.
2. Click **Ensemble**.
3. Click **Configure**.
4. Click **Production** and then click **Go**.
Ensemble then displays the last production you accessed, within the Production Configuration page.
5. Click **New** to invoke the Production Wizard.
6. Enter a **Package Name**, **Production Name**, and **Description**.
7. Choose the **HL7 Messaging** option and click **OK**.

The starter production has one interface, whose elements are:

- **HL7FileService** — A disabled HL7 file service with default settings
- **MsgRouter** — An HL7 routing process with an empty routing rule set

- **HL7FileOperation** — A disabled HL7 file operation with default settings

When you build an HL7 routing production, you create and configure many such interfaces. You can start by enabling these starter elements, copying them, renaming them, and modifying them to suit your needs. As you build an interface, it frequently happens that while configuring one item you must enter the name of another item that you have not yet created. *A clear naming convention is essential to avoid confusion.* For suggestions, see “[Naming Conventions](#)” in *Ensemble Best Practices*. For rules, see “[Configuration Names](#),” in *Configuring Ensemble Productions*.

In addition to interface elements, the starter production provides several elements that do not route HL7 messages, but that provide supporting functionality for the production:

- **Bad Message Handler** — A built-in destination for messages that fail validation.
- **Ens.Alert** — A routing process with a routing rule set that you can configure to [route alert messages](#) to different email business operations depending on the source of the alert (that is, which element is in trouble) and various conditions (such as the time of day) to suit your corporate schedules and procedures.
- **PagerAlert** and **EmailAlert** — Email business operations that can be configured to send a text message (such as an alert) to a pager or email address.

3.2 Adding HL7 Business Services

For your production to receive HL7 messages from outside Ensemble, you must add an HL7 business service to the production configuration. To add HL7 business service to a production, you must create it, integrate it into the production, and configure it as needed. The following subsections provide the details.

3.2.1 Creating an HL7 Business Service

To add an HL7 business service to a production:

1. Display the production in the Production Configuration page of the Management Portal (**Ensemble** > **Configure** > **Production** from the Home page).
2. In the **Services** column, click the Add button (a plus sign).
Ensemble displays a dialog box.
3. Click the **HL7 Input** tab.
4. Click one of the following from the **Input type** list:
 - **TCP**
 - **File**
 - **FTP**
 - **HTTP**
 - **SOAP**

This choice selects the host class for your business service.

5. For **HL7 Service Name**, type the name of this business service. The name should be unique among the business services. Do not use periods or spaces.

The default is the name of the class on which this service is based.

6. For **HL7 Service Target**, select one of the following:
 - **Create New Router** — Ensemble will add a routing process to the production and will configure the business service to use it as a target. You can edit its details later.
 - **None for Now** — No target is specified for this business service.
 - **Choose From List** — In this case, also select an existing business host from the drop-down list.
7. Click **OK**.

In addition to the HL7 business services used here, Ensemble provides two simple business services: `EnsLib.File.PassthroughService` and `EnsLib.FTP.PassthroughService`. You can select either of these if you simply need to pass a file through the production, and do not want to parse or format the file as HL7.

If you want the production to receive data that is not an HL7 message, see “[Defining Business Services](#)” in *Developing Ensemble Productions*. Also see “[Connectivity Options](#)” in *Introducing Ensemble*.

3.2.2 Integrating and Configuring an HL7 Business Service

To integrate a new HL7 business service into an Ensemble production, you must associate it with the routing process or business operation to which it relays messages. Additionally, if you expect the business service to receive nonstandard message structures you will need to create a custom HL7 schema definition to parse and validate these messages. To do this:

1. Complete the instructions for creating any **Target Config Names** or **Message Schema Category** items that your HL7 business service needs. The items might be:
 - An [HL7 routing process](#) (for a routing interface). See the [next section](#).
 - An [HL7 business operation](#) (if your design bypasses a routing process for this interface and simply relays messages from the incoming business service to the outgoing business operation). See “[Adding HL7 Business Operations](#),” later in this chapter.
 - A custom HL7 schema definition, to parse the incoming HL7 messages. For information on creating custom schema categories, see “[Creating Custom Schema Categories](#)” in *Ensemble Virtual Documents*.
2. Return to the diagram on the Production Configuration page. Select the new HL7 business service. If the **Target Config Names** and **Message Schema Category** fields were previously blank, configure them now, and click **Apply**.
3. Configure additional settings of the business service, as needed. For details, see “[Settings of an HL7 Business Service](#)” in the “[Reference](#).”

3.3 Adding HL7 Routing Processes

To add an HL7 routing process to a production, you must create it, integrate it into the production, and configure it as needed. The following subsections provide the details.

3.3.1 Creating an HL7 Routing Process

To add an HL7 routing process to a production:

1. Display the production in the Production Configuration page of the Management Portal (**Ensemble > Configure > Production** from the Home page)

2. In the **Processes** column, click the Add button (a plus sign).
Ensemble displays a dialog box.
3. Click **HL7 Message Router** for the business process option; the router class defaults to `EnsLib.HL7.MsgRouter.RoutingEngine`.
4. For **HL7 Router Name**, type the name of this business process. The name should be unique among the business processes.
Do not use periods or spaces.

The default is the name of the class on which this process is based.
5. For **Routing Rule Name**, do one of the following:
 - Select an existing routing rule from the **Routing Rule Name** drop-down list.
 - Select **Auto-Create Rule** and type a rule name into **Routing Rule Name**. In this case, the wizard creates the routing rule class in the same package as the production.
6. Click **OK**.

3.3.2 Integrating and Configuring an HL7 Routing Process

To integrate a new HL7 routing process into an Ensemble production, you must associate it with the business service that receives its incoming messages, and with the routing rule set that determines its actions based on those messages. To do this:

1. Select the [HL7 business service](#). In the menu to the right of the screen, click the **Settings** tab and open the **Basic Settings** menu. In the **Target Config Names** field, enter the name of the new HL7 routing process.
2. Create a [routing rule set](#), as described in the [next chapter](#). Select the routing process in the configuration diagram. In the **Business Rule Name** field, enter the full name of the new routing rule set.
3. Configure additional settings of the routing process, as needed. For details, see “[Settings for HL7 Routing Processes](#)” in the “[Reference](#).”

3.4 Adding HL7 Sequence Managers

HL7 messages can become out of sequence for various reasons, particularly when multiple processors are handling them. In some cases, it is desirable to ensure that HL7 messages are processed in the correct sequence. In such cases, you can add an HL7 sequence manager to the appropriate part of the production.

An HL7 sequence manager is a business process that accepts incoming HL7 messages (possibly from multiple sources), then forwards the messages to a target configuration item in the order specified by the MSH:13 SequenceNumbers field in the messages.

The sequence manager can detect duplicate messages and timing gaps between messages. It also determines when the timing gaps between sequential messages are sufficiently large to indicate a problem. Its level of sensitivity can be adjusted using its configuration settings.

To build an HL7 sequence manager for use in an HL7 message routing production, you must [create and configure it](#) and then [integrate it](#) into the production. This topic explains each step.

Important: The Ensemble HL7 Sequence Manager is a HL7-compatible store-and-forward application and does not support the HL7 Sequence Number Protocol as defined in Chapter 2, section 2.10.1 of the *HL7 Standard*.

3.4.1 Creating an HL7 Sequence Manager

To add an HL7 sequence manager to a production:

1. Display the production in the Production Configuration page of the Management Portal (**Ensemble > Configure > Production** from the Home page).
2. In the **Processes** column, click the Add button (a plus sign).
Ensemble displays a dialog box.
3. Select `EnsLib.HL7.SequenceManager` from the **ProcessClass** list.
4. For **Name**, type the name of this business process. The name should be unique among the business processes. Do not use periods or spaces.
The default is the name of the class on which this process is based.
5. Click **OK**.

A production can have multiple sequence managers, if needed.

3.4.2 Integrating and Configuring an HL7 Sequence Manager

To integrate a new HL7 sequence manager into an Ensemble production, you must associate it with the business service that receives its incoming messages, and with the target destination for the messages that it sends, once it has sorted out their proper sequence. To do this:

1. Select the [HL7 business service](#). In the **Target Config Names** field, enter the name of the new HL7 sequence manager.
2. Return to configuring the HL7 sequence manager. Provide it with a list of **Output Target Config Names** for successful messages.
3. If you want the sequence manager to check for duplicate messages, set **Enable Duplicated Message Check** to True.
If you want to save the duplicate messages for troubleshooting purposes, create a [HL7 business operation](#) to receive them. In the **Duplicated Message Target** field enter the name of the new HL7 business operation.
4. If you want the sequence manager to check for out-of-sequence messages, set **Perform Sequence Number Check On** to `Sender` or `Receiver` and configure the details of sequence checking by setting values for **Large Gap Size** and **Message Wait Timeout**. Otherwise, set **Perform Sequence Number Check On** to `None`.
If you want to save the out-of-sequence messages for troubleshooting purposes, create a [HL7 business operation](#) to receive them. In the **Out Of Sequence Message Target** field, enter the name of the new HL7 business operation.
5. If you want the sequence manager to transform messages before sending them outside Ensemble, set **Perform Output Transformation On** to `Sender` or `Receiver` and set a value for the **Output Facility Application**. Otherwise, set **Perform Output Transformation On** to `None`.
6. Identify any **Passthrough Message Types** that the sequence manager should simply send to the **Output Target Config Names** without checking or transforming them.
7. Configure additional settings of the sequence manager, as needed. For details, see “[Settings for HL7 Sequence Managers](#)” in the “[Reference](#).”

3.4.3 Accessing HL7 Sequence Data Programmatically

You can access the runtime data of the Sequence Manager via SQL. To do so, execute a query on the table `EnsLib_HL7.SM.RuntimeData.Thread`. This table provides the following string fields:

Application

The name of the sending or receiving application, as taken from the HL7 messages.

Facility

The name of the sending or receiving facility, as taken from the HL7 messages.

Thread

One of the following strings:

- `main`
- `resend`

Type

One of the following strings:

- `Sender`
- `Receiver`

NextSequenceNumber

Identifies the next number in the sequence for the given facility, application, thread, and type.

A sample SQL query might be as follows:

SQL

```
SELECT Application,Thread,Type,NextSequenceNumber FROM EnsLib_HL7.SM.RuntimeData.Thread WHERE Facility = 'mine'
```

3.5 Adding HL7 Business Operations

To send HL7 messages from a production, you must add an HL7 business operation. To add an HL7 business operation to a production, you must create it, integrate it into the production, and configure it as needed. The following subsections provide the details.

3.5.1 Creating an HL7 Business Operation

To add an HL7 business operation to a production:

1. Display the production in the Production Configuration page of the Management Portal (**Ensemble > Configure > Production** from the Home page).
2. In the **Operations** column, click the Add button (a plus sign).
Ensemble displays a dialog box.
3. Click **HL7 Output**.
4. Click one of the following from the **Output type** list:
 - **TCP**
 - **File**

- **FTP**
- **HTTP**
- **SOAP**

This selection determines the host class for this business operation.

5. For **Operation Name**, type the name of this business operation. The name should be unique among the business operations. Do not use periods or spaces.

The default is the name of the class on which this operation is based.

6. Click **OK**.

In addition to the HL7 business operations used here, Ensemble provides two simple business operations: `EnsLib.File.PassthroughOperation` and `EnsLib.FTP.PassthroughOperation..` You can select either of these if you simply need to pass a file through the production, and do not want to parse or format the file as HL7.

If you want the production to send data that is not an HL7 message, see “[Defining Business Operations](#)” in *Developing Ensemble Productions*. Also see “[Connectivity Options](#)” in *Introducing Ensemble*.

3.5.2 Integrating and Configuring an HL7 Business Operation

To integrate a new HL7 business operation into an Ensemble production, simply identify it as a target that receives messages. To do this, you must configure the item that sends messages to the HL7 business operation and enter the configured **Name** of the HL7 business operation into the appropriate field:

- For a routing interface, enter the name in the **Target** field of the [routing rule set](#).
- If your design uses a pass-through interface that simply relays messages from the incoming business service to the outgoing business operation, enter the name in the **Target Config Names** field of the [HL7 business service](#).
- Configure additional settings of the business operation, as needed. For details, see “[Settings for HL7 Routing Operations](#)” in the “[Reference](#).”

3.6 Pager and Email Alerts

Alerts provide the ability to send notification to a user device while an Ensemble production is running. The intention is to alert a system administrator or service technician to the presence of a problem. Alerts may be sent via email, text pager, or another mechanism. For details, see “[Configuring Alerts](#)” in *Configuring Ensemble Productions*.

The starter HL7 routing production provides the following elements to support alerts:

- **Ens.Alert** — The starter production provides a built-in `Ens.Alert` element. This `Ens.Alert` is a routing process with an associated routing rule set called `AlertRule`.

You can configure `AlertRule` to route alert messages to different business operations depending on the source of the alert message (that is, which element is in trouble) and various conditions (such as time of day).

Tip: To view time-of-day function examples, see the sample functions in the class `Demo.HL7.MsgRouter.Functions` in the `ENSDEMO` namespace.

To configure `AlertRule`, follow a similar procedure to the one for [HL7 routing rule sets](#), except:

- For the **Message Class**, choose `Ens.AlertRequest`

- In the **Target** field, choose or type the configured name of an email business operation.
- **PagerAlert** and **EmailAlert** — These are ordinary email operations that can be configured to send a text message (such as the body of an alert) to a pager or email address. **PagerAlert** and **EmailAlert** are initially identical, other than their names. Both are provided to emphasize the fact that alerts have different priorities. Generally the higher-priority alerts go to a pager, lower-priority to email.

You can configure any number of destinations that copy these starter operations. At runtime, your **AlertRule** routing rule set determines which destination is appropriate to use.

4

Additional Steps

This chapter discusses the additional steps needed to add HL7 processing to a production. It includes the following topics:

- [Defining Routing Rule Sets for HL7](#) (required)
- [Defining DTL Data Transformations for HL7](#)
- [Defining HL7 Search Tables](#)

Be sure to perform these tasks in the same namespace that contains your production. When you create rule sets, transformations, and search tables, do not use reserved package names; see “[Reserved Package Names](#)” in *Developing Ensemble Productions*.

Also see “[Overriding the Validation Logic](#)” in *Ensemble Virtual Documents*.

4.1 Defining Routing Rule Sets for HL7

When creating a routing rule set for an HL7 interface, your goal is to tell Ensemble what to do with the source message based on the segments found within it. Sometimes it matters which segments are found; sometimes it matters which values are found within those segments.

In ordinary rule sets, each rule returns a value to the business process that invoked the rule set. In a routing rule set, a rule usually directs an HL7 message to a destination, possibly transforming the HL7 message before sending it.

For information on using the rule editor, see “[Creating and Editing Rules Sets](#)” in *Developing Business Rules*.

1. When you create a new HL7 routing process using the Business Rule Wizard, Ensemble creates a new, empty routing rule set to accompany the new routing process. Its information tables contain the following values:

- **Package Name** — The package that contains the production class. For example, if you used the wizard to add a routing process to a production called `TestRule.MyTest`, the associated routing rule has a **Package Name** of:

`TestRule`

- **Rule Name** — The simple **Routing Rule Name** that you chose in the wizard, such as:

`MyRule`

The combination of the **Package Name** and **Rule Name** identify the rule uniquely within the Ensemble namespace. The full name for any rule definition is the **Package Name** and **Rule Name** connected by a dot (.) as in:

`TestRule.MyRule`

This full name, rather than the **Rule Name**, is the correct value to use in the **BusinessRuleName** field when configuring an [HL7 routing process](#). The Business Process Wizard sets this up automatically.

- **Routing Engine Class** — The default **Router Class** from the wizard; do not change it:

```
EnsLib.HL7.MsgRouter.RoutingEngine
```

2. You may also enter additional fields for your own rule reporting purposes:

- **Report Group** — Value to be used to group rules for reporting
- **Report Name** — Display value for the rule report group
- **Short Description** — Optional short description of this Rule Definition.

3. See “[Using the Rule Constraint Editor](#)” in *Developing Business Rules* for details on entering the constraints of a routing rule.

4. Once you have created a new rule, you may add **Conditions** and **Actions** to it. For details, see chapter “[Creating and Editing Rule Sets](#)” in *Developing Business Rules*. As you add **Conditions** to each rule, remember these tips:

- **Conditions** evaluate AND operators first, then OR.
- **Conditions** can refer to properties of the HL7 message object. Within **Conditions**, the special variable `Document` represents the HL7 message object, as in the following examples. For HL7 batch documents, you can use the special variable `Document.Parent` to represent the parent message object.

```
Document.Name  
Document.Parent.DocType  
Document.{PIDgrp.PV1grp.PV1:18}  
Document.{PIDgrp.PID:PatientName.familylastname}  
Document.{ORCgrp(1).OBRuniongrp.OBRunion.OBR:4.3}
```

Note: For compatibility with past releases, the special variables `HL7` and `HL7.Parent` are supported as alternatives to `Document` and `Document.Parent`.

A dot separates the `Document` variable from the property name. This name may be:

- Any of the class properties: `DocType`, `TypeCategory`, `BuildMapStatus`, or `Name`.
- A virtual property, referenced using any of the following conventions:

- [Curly brackets](#)

```
{segmentPath:field}
```

- [Square brackets](#)

```
[segmentName:field]
```

- [Round brackets](#) or parentheses

```
(multi-valued-property-path)
```

- [Angle brackets](#)

```
<context|expression>
```

Details are available in the “[Syntax Guide](#)” section of *Ensemble Virtual Documents*. However, you do not need to enter properties by typing; you can browse for properties as described in “[Creating and Editing Rule Sets](#)” in *Developing Business Rules*.

5. Complete the instructions in the next several topics for creating any **Source**, **Target**, or **Transform** items that your routing rule set needs. The items might be:

- An [HL7 business service](#), to route messages to the rule
- A [DTL data transformation](#), to transform the message before sending it
- An [HL7 business operation](#), to route the message to an external application
- An [HL7 routing process](#) may be either a **Source** or a **Target**

As you create items, return to the Message Routing Rule Editor to add them to the appropriate fields of the rule definition.

- Return to the diagram on the Production Configuration page. Select the corresponding [HL7 routing process](#). In the **BusinessRuleName** field, enter the full name of the new routing rule set.

4.2 Defining DTL Data Transformations for HL7

Each interface may require some number of data transformations.

Important: Do not manually change HL7 escape sequences in the data transformation; Ensemble handles these automatically.

You use the Data Transformation Builder page to create data transformations (navigate to **Ensemble > Build > Data Transformations**). For general information on using this page, see [Developing DTL Transformations](#).

The following figure shows this page, displaying `MsgRouter.ADTLastNameTransform` from the `ENSDEMO` namespace:

The screenshot displays the Data Transformation Builder window. The main area shows a mapping between a source message structure and a target message structure. The source is `EnsLib.HL7.Message` (Schema: `ADT_A01`) and the target is `EnsLib.HL7.Message` (Schema: `2.3.1:ADT_A01`). The mapping shows fields like `MSH`, `EVN`, `PID`, `PD1`, `NK1()`, `PV1`, `PV2`, `DB1()`, `OBX()`, `AL1()`, `DG1()`, `DRG`, and `PR1grp()` being mapped from source to target.

Below the mapping, the **Actions** table is visible:

#	Action	Condition	Property	Value	Key / Transform
1	set		target.{MSH}	source.{MSH}	--
2	set		target.{MSH:9.1}	"ADT"	--
3	set		target.{MSH:9.2}	"A01"	--

On the right, the **Details for the overall data transformation** panel shows the following information:

- Name:** `Demo.HL7.MsgRouter.ADTLastNameTransform`
- Create:** new
- Source Class:** `EnsLib.HL7.Message`
- Source Doc Type:** `Demo.HL7.MsgRouter.Schema:ADT_A01`
- Target Class:** `EnsLib.HL7.Message`
- Target Doc Type:** `2.3.1:ADT_A01`
- Language:** objectscript
- Report Errors:** ☐
- Description:**

Note the following tips:

- Have [Developing DTL Transformations](#) handy. This book explains how to add each kind of DTL action.

- Be clear about the value you want for the **create** option of the data transformation class. **create** may have one of the following values:
 - **new** — Create a new object of the target type, before executing the elements within the data transformation. Any source segments that you do not explicitly assign to the target object are ignored. This is the default.
 - **copy** — Create a copy of the source object to use as the target object, before executing the elements within the transform.
 - **existing** — Use an existing object, provided by the caller of the data transformation, as the target object.

To create a target object that is an exact copy of the source, do not use an action like this:

```
<assign property='target' value='source' />
```

Instead use the `create='copy'` attribute in the data transformation class.

- Ensure that the data transformation class identifies the correct [schema category](#) for:
 - The *sourceDocType* attribute
 - The *targetDocType* attribute

The [schema category](#) may be the same or different for the source and target objects.

- Ensure that the **Transform** tab specifies the scripting *language* that you want to use for all the expressions and **code** actions within that data transformation. ObjectScript is the default language.
- Use **assign** actions to assign HL7 segments from the source message to the target message by drag and drop operations. Possibly, you will want to use a combination of techniques. You can first generate a line of code by dragging, then fine-tune the code by editing the text.
- The data transformation can refer to properties of the HL7 message object, including:
 - The class properties DocType, TypeCategory, BuildMapStatus, and Name.
 - Virtual document properties as described in “[Curly Bracket { } Syntax](#)” in the “[Syntax Guide](#)” section of *Ensemble Virtual Documents*.

Within the data transformation class code, the special variables `source` and `target` represent the respective HL7 message objects, as in the following examples:

```
source.Name  
target.DocType  
source.{PIDgrp.PV1grp.PV1:18}  
target.{PIDgrp.PID:PatientName.familylastname}  
source.{ORCgrp(1).OBRuniongrp.OBRunion.OBR:4.3}
```

- Assign any literal string values, or make conditional assignments. See the chapter “[Syntax Rules](#)” in *Developing DTL Transformations*.

Note: A string literal cannot include XML reserved characters. It also cannot include separator characters used by HL7.

Also, the HL7 null mapping code " " requires special handling. The following example tests for the null mapping code in the source message and replaces it with a blank string in the target message:

```
<if condition='source.{PV1:7().4}=" "'>
<true>
<assign property='target.{PV1:7().4}' value='' />
</true>
</if>
```

For details, see “[Null Mapping Codes](#)” in the chapter “Syntax for a Routing Production.”

- For simple calculations, the DTL data transformation can:
 - Invoke Ensemble utility functions
 - Use ObjectScript or Caché Basic expressions in the DTL [code](#) action
 - Invoke the DTL [sql](#) action

For more complex calculations, you can write your own class methods and invoke them from a [code](#) action, or from the value string in another DTL element.

- Include descriptions of the transformation and each action.
- Compiling the data transformation also saves it.
- To use the DTL data transformation in the production, simply enter its full package and class name in the **Transform** field of a [routing rule set](#).

4.2.1 Null Mapping Codes

There are HL7 applications that use a *null mapping* convention. According to this convention, the source application can send a field that consists of two consecutive double-quote characters (" ") to signify *if you have data in this field, delete it from your application*.

Many target applications do not expect these instructions and are not designed to respond to them. If this is the case, and the double quotes are saved in the target application as actual patient data, the application users see double-quote characters on their screens. This can be annoying and misleading.

When your source application uses a null mapping convention, your HL7 data transformation can check for null mapping entries in HL7 fields and replace them with empty strings, or otherwise fill in data for the benefit of the target application.

The following `<if>` statement represents the simplest case. It checks for a null mapping in the source and replaces it with an empty string in the target. The `<if>` *condition* tests for the null mapping code " " using a string of 2 quoted quotes. This results in a total of 6 double-quote characters, not including the single quotes that wrap the entire *condition* value. (Count carefully!)

```
<if condition='source.{PV1:7().4}=" "'>
<true>
<assign property='target.{PV1:7().4}' value='' />
</true>
</if>
```

In the above example, the `<assign>` *value* indicates a empty string using 2 consecutive double-quote characters, with single quotes wrapping the entire *value*.

The following syntax is equally valid:

```
<if condition='source.{PV1:7().4}="&quot;&quot;">
<true>
<assign property='target.{PV1:7().4}' value='&quot;&quot;' />
</true>
</if>
```

You can achieve more sophisticated goals for handling null mappings. The following example takes alternative actions based on whether there is actually a value in {PV1:3}. If the field contains a null mapping code the <true> element executes. Otherwise the <false> element executes.

XML

```
<if condition='source.{PV1:3}=""'>
<true>
<assign property='target.{PV1:3.1}' value='source.{PV1:PatientType}' />
<assign property='target.{PV1:3.2}' value='source.{PV1:PatientType}' />
</true>
<false>
<code><![CDATA[
// Dr Chart pulls subfields as follows:
// 1 location, 2 desc, 3 room, 4 bed, 5 wing, 6 floor
]]></code>
<assign property='target.{PV1:3.1}' value='source.{PV1:3.1}' />
<assign property='target.{PV1:3.2}' value='source.{PV1:3.1}' />
<assign property='target.{PV1:3.3}' value='source.{PV1:3.2}' />
<assign property='target.{PV1:3.4.1}' value='source.{PV1:3.3}' />
<assign property='target.{PV1:3.5}' value='source.{PV1:3.1}' />
</false>
</if>
```

4.2.2 Transforming Long Segment Fields

The ObjectScript method used by DTL transformations, GetValueAt, truncates HL7 segment fields at 32K. Therefore, when transforming a field that is longer than 32K, you cannot use the left-to-right drag action in the DTL Editor. For example, if an OBX:5 field exceeds 32K, you cannot use the DTL Editor to drag the source field to the target because it will be truncated. Similarly, custom code should not call GetValueAt if you are transforming fields longer than 32K.

To transform fields longer than 32K, you need to use a Code action to add custom code to the transformation. This custom code must include one of the following methods in the EnsLib.HL7.Segment class to read the value of the field from the source into a stream: GetFieldStreamRaw(), GetFieldStreamUnescaped(), or GetFieldStreamBase64().

These Get methods take 3 arguments: the stream output argument, the VDoc path of the field, and the pRemainder output argument. The pRemainder argument will be filled with all fields that come after the one being extracted. For example:

```
/// Segment: OBX|1|2|3|4|5|6|7
do GetFieldStreamRaw(.stream, "OBX:5", .rem)
/// rem contains: |6|7
```

Once the custom code has the field value in a stream, it must use one of the following methods of EnsLib.HL7.Segment to store the value in the target: StoreFieldStreamRaw(), StoreFieldStreamUnescaped(), or StoreFieldStreamBase64().

These Store methods accept three arguments: the stream to store in the field, the VDoc path of the field in which to store the stream, and a pRemainder argument. If the pRemainder argument is not specified, then all target fields after the field being stored are deleted. For example:

```
/// Before: OBX|1|2|3|4|5|6|7
do StoreFieldStreamRaw(stream, "OBX:5")
/// After: OBX|1|2|3|4|<stream>
/// |6|7 are gone because a remainder was not specified.
```

If pRemainder is specified, then all fields after the one being stored will be replaced with what is in pRemainder.

```
/// Before: OBX|1|2|3|4|5|6|7
do StoreFieldStreamRaw(stream, "OBX:5", "|six|seven")
/// After: OBX|1|2|3|4|<stream>|six|seven
```

Important: The target segment becomes immutable once a Store method is called, therefore it must be the last change made to the segment.

The following example shows how to extract the field from the source and store it in the target. It assumes that the custom code made edits to the target fields that come after the 32K+ field *before calling the Store method*; this approach is required because the segment becomes immutable after calling the Store method. The sample code does not take the remainder from the source and store that in the target because that would revert edits that were already made to fields that come after the target's long field. Therefore, you would take the stream field from the source, but the remainder from the target, and store both of those in the target:

```
/// previous code makes edits to fields that come after OBX:5 in the target
do source.GetFieldStreamRaw(.stream, "OBX:5")
do target.GetFieldStreamRaw(.dummy, "OBX:5", .rem)
do target.StoreFieldStreamRaw(stream, "OBX:5", rem)
///Segment is now immutable
```

Doing it this way prevents edits made to target fields that come after OBX:5 from being reverted or deleted.

4.3 Defining HL7 Search Tables

The HL7 search table class, `EnsLib.HL7.SearchTable`, automatically indexes popular HL7 properties; see the subsection “[Properties That Are Indexed by Default](#).”

If you need more items to search, you can create a subclass. The subclass inherits the `Identifier` property, plus the infrastructure that makes search tables work. For details, see “[Defining a Search Table Class](#)” in *Ensemble Virtual Documents*.

For HL7, Ensemble supports an additional value for `PropType`. You can use `DateTime:HL7` in addition to the types listed in *Ensemble Virtual Documents*.

4.3.1 Properties That Are Indexed by Default

When you choose `EnsLib.HL7.SearchTable` as your search table class, it enables Ensemble to search HL7 messages for the following virtual properties.

Choose...	To refer to this value...
MSHTypeName	<p>The message structure name. To create this string, Ensemble concatenates the following values from the HL7 message:</p> <ul style="list-style-type: none"> The MSH message header segment Field 9 (message type) Subfield 1 (message type: ADT, ORM, etc) The literal character _ The MSH message header segment Field 9 (message type) Subfield 2 (trigger event: A01, A12, O01_2, etc) <p>The result is a message structure name in the format ADT_A01, ADT_A12, ORM_O01_2, etc.</p>

Choose...	To refer to this value...
MSHControlID	<p>The unique identifying number for this message. Ensemble retrieves this value from:</p> <ul style="list-style-type: none"> The MSH message header segment Field 10 (message control ID) <p>Ensemble interprets this value as a case-sensitive string.</p>
PatientID	<p>The patient identifier for this message. This is a field whose location has shifted as the HL7 standard has evolved. For this reason, Ensemble looks for this value in <i>all</i> of the following locations. That way, the patient identifier can be found regardless of which HL7 schema category the message conforms to:</p> <ul style="list-style-type: none"> The PID patient identifier segment Field 2 (patient external identifier) Subfield 1 (patient identifier) The PID patient identifier segment Field 3 (patient identifier list), all entries in the list Subfield 1 (patient identifier) The PID patient identifier segment Field 4 (patient identifier list), all entries in the list Subfield 1 (patient identifier)
PatientName	<p>The PID patient identifier segment Field 5 (patient name)</p>
PatientAcct	<p>The PID patient identifier segment Field 18 (patient account number) Subfield 1 (ID)</p>

4.3.2 Examples

The following example consists of one virtual property path that uses {} syntax. This <Item> element refers to the value at Segment 1, Field 10 of the HL7 message:

XML

```
<Item DocType=" "
  PropName="MSHControlID"
  PropType="String:CaseSensitive"
  StoreNulls="true" >
  {1:10}
</Item>
```

The following more complex <Item> element uses the ObjectScript _ operator to concatenate three strings. From left to right, these are:

- The value within Segment 1, Field 4
- A literal – character

- The value within Segment 1, Field 3

XML

```
<Item DocType=" "
      PropName="SendingFacilApp" >
      {1:4}_"-_{1:3}
</Item>
```

The following <Item> example uses most of the possible syntax options: concatenation, virtual properties, a literal hyphen character (-), and the ObjectScript string function \$PIECE:

Class Member

```
XData SearchSpec [ XMLNamespace="http://www.intersystems.com/EnsSearchTable" ]
{
<Items>
  <Item DocType="Mater:ORM_001 "
        PropName="RelationKey" >
    $P(
      {ORCgrp(1).OBRuniongrp.OBRunion.OBR:UniversalServiceID.text}, "-", 1, 2
    )_"-_{MSH:12}
  </Item>
</Items>
```

The following sample search table class provides several examples of valid <Item> entries. This class inherits from EnsLib.HL7.SearchTable, as is required for HL7 search tables. Comments above each group of <Item> entries describe the purpose of that set of entries. For details about {} or [] syntax, see the “[Syntax Guide](#)” section of *Ensemble Virtual Documents*.

Class Definition

```
Class Demo.HL7.MsgRouter.SearchTable Extends EnsLib.HL7.SearchTable
{
XData SearchSpec [ XMLNamespace="http://www.intersystems.com/EnsSearchTable" ]
{
  <Items>
    <!-- Items that do not depend on DocType, indexing any HL7 message -->
    <Item DocType=" " PropName="SendingFacilApp" >{1:4}_"-_{1:3}</Item>
    <Item DocType=" " PropName="RecvingFacilApp" >{1:6}_"-_{1:5}</Item>
    <Item DocType=" " PropName="MSHDateTime" PropType="DateTime:HL7" >{1:7}</Item>

    <!-- Get fields from named segments found in any HL7 message -->
    <Item DocType=" " PropName="PatientName" >[PID:5]</Item>
    <Item DocType=" " PropName="InsuranceCo" >[IN1:4]</Item>

    <!-- Get patient name from any HL7 message declared type ADT_A05 -->
    <Item DocType=":ADT_A05" PropName="PatientName" >{3:5}</Item>

    <!-- Get specific field from specific segment when the -->
    <!-- HL7 message is assigned a specific DocType. Only in this -->
    <!-- case can you use names for segments, instead of numbers. -->
    <Item DocType="Demo.HL7.MsgRouter.Schema:ORM_001 " PropName="ServiceId" >
      {ORCgrp().OBRuniongrp.OBRunion.OBR:UniversalServiceID.text}
    </Item>
    <Item DocType="2.3.1:ORU_R01 " PropName="ServiceId" >
      {PIDgrpgrp().ORCgrp(1).OBR:UniversalServiceID.text}
    </Item>
  </Items>
}
}
```


5

Important HL7 Scenarios

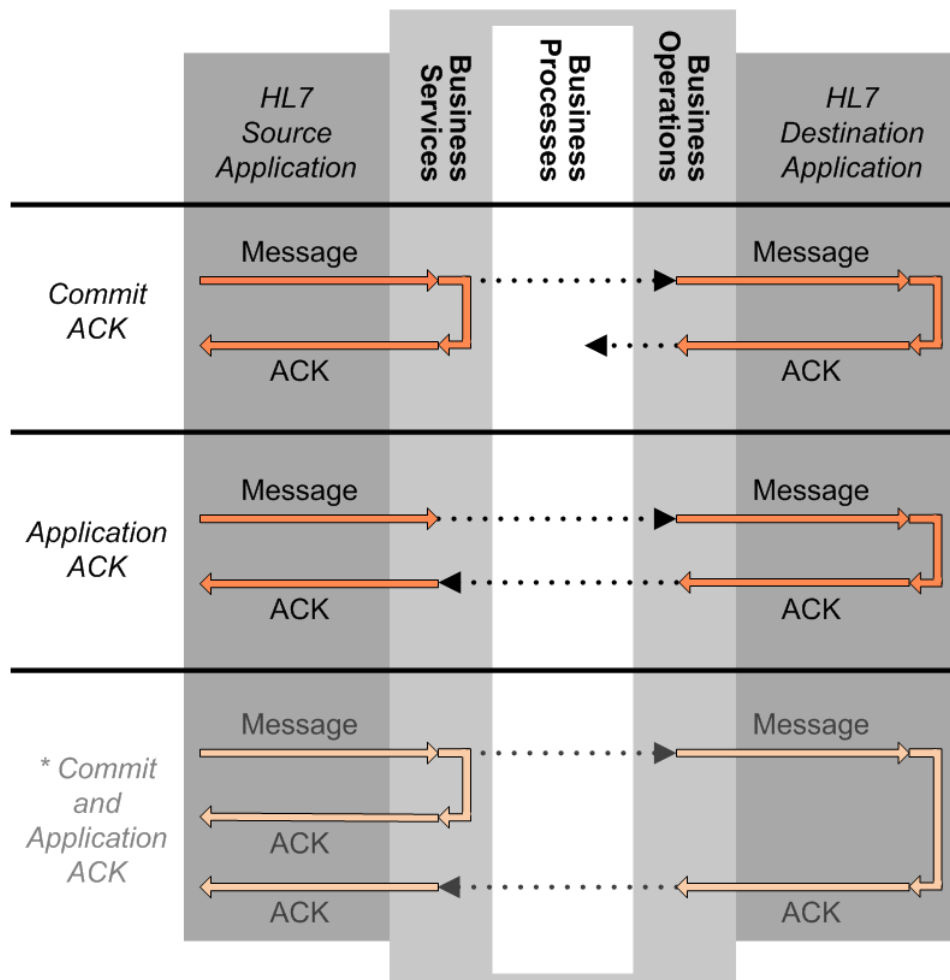
This chapter describes some key HL7 scenarios that affect your choice of configuration settings. Topics include:

- [HL7 Acknowledgment \(ACK\) Mode](#)
- [HL7 Dual Acknowledgment Sequences](#)
- [HL7 Batch Messages](#)

5.1 HL7 Acknowledgment (ACK) Mode

An HL7 acknowledgment (ACK) message acknowledges that a destination has received an HL7 message. A negative ACK (NACK) message acknowledges that the destination is aware of the transmission but did not capture the message.

The following figure illustrates how Ensemble upholds HL7 message conventions for sending ACK and NACK messages:



Commit ACK — The Ensemble business service returns an ACK to the source application as soon as it commits the transaction that saves the data received from that source. It sends a NACK if this proves impossible for some reason. The Ensemble business operation may be set up to interpret an ACK or NACK from the target application, but it does not return those messages to the source.

Application ACK — The Ensemble business service does not send an ACK or NACK to the source application until one returns from the target application by way of the Ensemble business operation. The business service returns the ACK or NACK that it receives from the business operation.

** Commit and Application ACK* — The third option only occurs in the rare condition where you set the **Ack Mode** to be MSH-determined and the MSH segment contains a value in both field 15 and 16 that requires an ACK. InterSystems recommends you avoid this condition.

You use these acknowledgment conventions to send one of the three main types of ACK message content:

- **Accept** — The message arrived, and was accepted.
- **Reject** — The message arrived, but has been rejected.
- **Error** — The message did not arrive successfully; try again.

Use the following configuration settings to control ACK processing in a business service:

- [Ack Target Config Names](#)
- [Ack Mode](#)
- [Use ACK Commit Codes](#)

- [Ignore Inbound ACK](#)
- [Add NACK ERR](#)
- [NACK Error Code](#)

Use the following configuration settings of a business operation to control ACK processing:

- [Reply Code Actions](#)
- [Get Reply](#)

5.2 HL7 Dual Acknowledgment Sequences

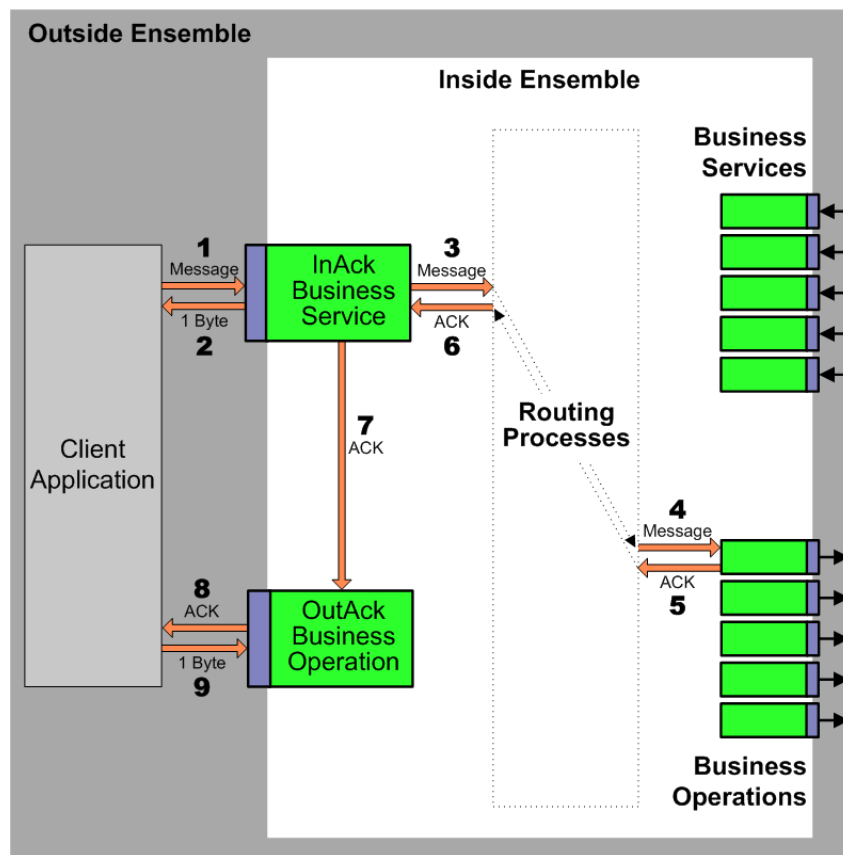
Some systems require a dual acknowledgment sequence from Ensemble: an immediate 1-byte ACK, followed later by the full ACK message. One such system is the dual-channel iSoft iCM application. If your configuration includes a client system such as iCM that requires a dual acknowledgment sequence, you must set up a paired business service and business operation to enable Ensemble to provide the expected ACKs.

Ensemble provides specialized business host classes that you can use to define a dual acknowledgment sequence over TCP and HTTP.

- `EnsLib.HL7.Service.TCPAckInService` is a specialized HL7 business service that receives ACKs on behalf of a paired HL7 TCP business operation. It also depends on this partner to send ACKs on its behalf.
- `EnsLib.HL7.Operation.TCPAckOutOperation` is a specialized HL7 TCP business operation that sends out ACKs on behalf of a paired HL7 TCP business service. It also depends on this partner to collect ACKs on its behalf. Each of these configuration items plays its usual role in addition to the work it does for its partner item.
- An `EnsLib.HL7.Service.HTTPAckInService` and `EnsLib.HL7.Operation.HTTPAckOutOperation` are also available.

5.2.1 Dual ACK Sequence for Incoming Messages

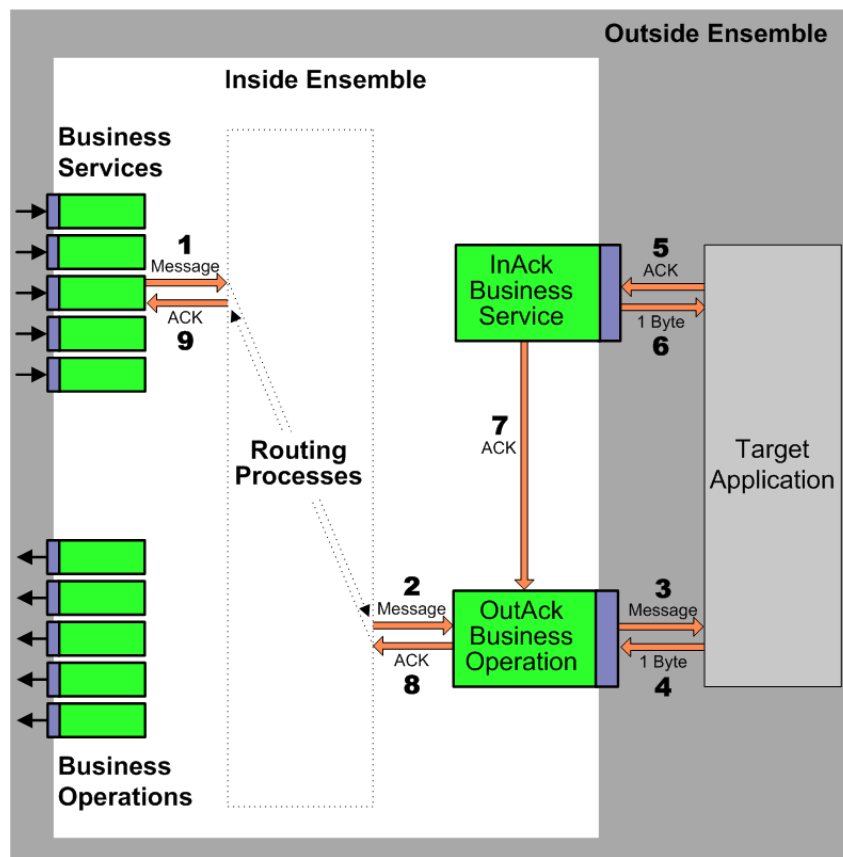
For messages entering Ensemble, the dual acknowledgment sequence works as shown in the following figure:



1. The client application sends a message into Ensemble.
2. The inbound ACK business service sends an immediate 1-byte ACK to the client application.
3. The inbound ACK business service sends the message to its routing process.
4. The routing process routes the message to its target via a business operation.
5. The target application returns an ACK message to the business operation.
6. Ensemble relays the ACK to the inbound ACK business service.
7. The business service relays the ACK to its paired business operation.
8. The business operation relays the ACK to the client application.
9. The client application acknowledges the ACK message by returning a 1-byte ACK.

5.2.2 Dual ACK Sequence for Outgoing Messages

For messages leaving Ensemble, the dual acknowledgment sequence works as shown in the following figure:



1. A business service sends a message to its routing process.
2. The routing process routes the message to the outbound ACK business operation.
3. The outbound ACK business operation relays the message to the target application.
4. The target application acknowledges the message by returning a 1-byte ACK.
5. The target application returns an ACK message to the inbound ACK business service.
6. The business service sends an immediate 1-byte ACK to the target application.
7. The business service relays the ACK to its paired business operation.
8. The business operation relays the ACK message back to the business service.
9. The business service receives the ACK to its original message.

5.2.3 Configuring a Dual ACK Sequence

To configure an Ensemble routing production to use the dual acknowledgment feature:

1. Add a business service to the production.

Choose `EnsLib.HL7.Service.TCPAckInService` or `EnsLib.HL7.Service.HTTPAckInService` as the business service class. It is not one of the standard **HL7 Input** options available from the Business Service Wizard, but you can choose it by selecting the **Other** option and identifying the class.

2. Add a business operation to the production.

Choose `EnsLib.HL7.Operation.TCPAckOutOperation` or `EnsLib.HL7.Service.HTTPAckOutOperation` as the business operation class. It is not one of the standard **HL7 Output** options available from the Business Operation Wizard, but you can choose it by selecting the **Other** option and identifying the class.

3. Configure the following settings for the business service:

- Set **Immediate Byte ACK** to True. Then, in addition to forwarding a full ACK message according to the **Ack Mode** setting, the business service also returns an immediate 1-byte ACK on its TCP or HTTP connection.
- For a **Partner Operation**, choose the business operation that you added in Step 2. Whenever you specify a **Partner Operation** value, the business service ignores any inbound ACK messages that it receives directly, to avoid creating an ACK feedback loop.

The business operation must exist and have the underlying class `EnsLib.HL7.Operation.TCPAckOutOperation` or `EnsLib.HL7.Operation.HTTPAckOutOperation`, respectively.

4. Configure the following setting for the business operation:

- Set the **Partner ACK Timeout** to the number of seconds for the business operation to wait for its partner business service to supply an ACK that corresponds to the normal outbound message that the business operation sent. The default is 600 seconds (10 minutes).

5.3 HL7 Batch Messages

Ensemble supports nested child documents (batch formats) in HL7. Each of the child documents is an Ensemble virtual document in its own right. This section discusses the details, which are controlled by two settings. It includes the following sections:

- [Supported Batch Formats](#)
- [Processing Incoming Batch Documents](#)
- [Sending Batch Messages](#)
- [Batch Modes](#)
- [Custom Outbound Batch Handling](#)

5.3.1 Supported Batch Formats

Ensemble supports the following HL7 batch formats:

- BHS MSH ... MSH ... BTS

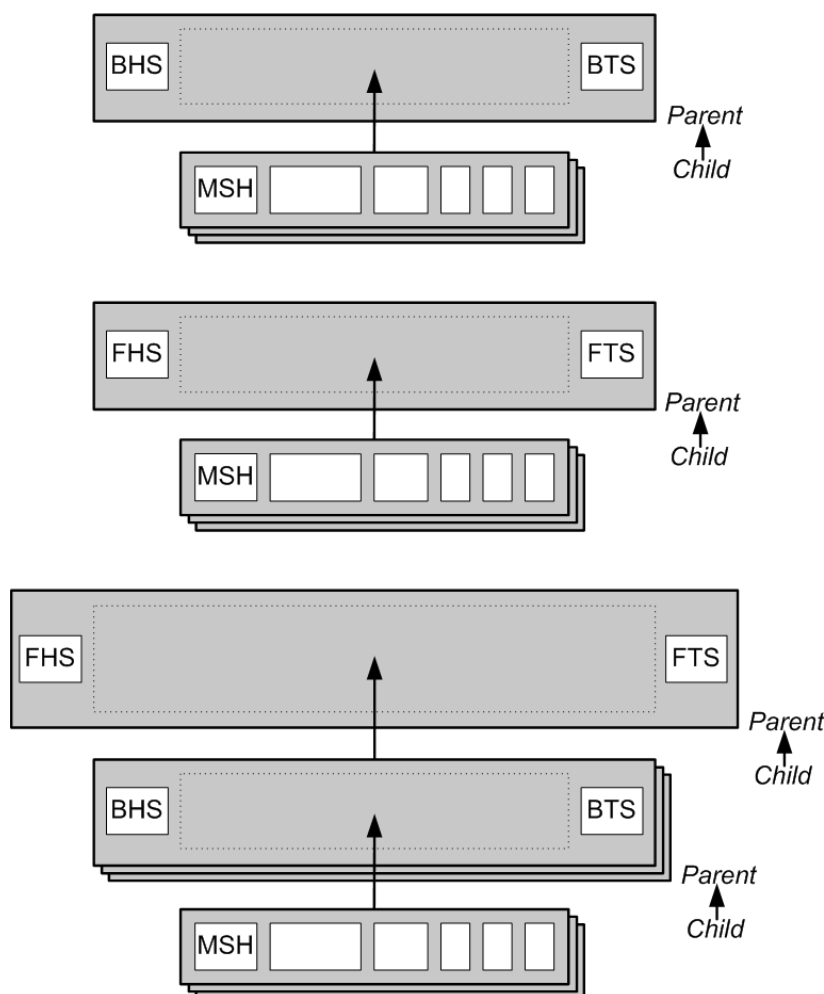
Ensemble recognizes BHS as a batch header segment, and BTS as a batch trailer segment. Within this container, Ensemble recognizes each MSH message header segment as the beginning of a child document.

- FHS MSH ... MSH ... FTS

Ensemble recognizes FHS as a batch header segment, and FTS as a batch trailer segment. Within this container, Ensemble recognizes each MSH message header segment as the beginning of a child document.

- FHS BHS MSH ... MSH ... BTS BHS MSH ... MSH ... BTS FTS

When FHS and BHS begin the message together, Ensemble recognizes FHS as the first-level parent document and each BHS as the beginning of a child document. BHS then becomes a second-level parent and the contents following each MSH segment become its child documents.



5.3.2 Processing Incoming Batch Documents

HL7 business services have the [Batch Handling](#) configuration setting, which determines how to process incoming batch documents. The options are:

- **Whole Batch** — Do not process child documents individually; accumulate and send the whole batch as one composite document.
- **Single-Session Batch** — Forward all child documents in the batch together in one session; the session includes objects representing the batch headers and trailers. **Single-Session Batch** is the default if no **Batch Handling** value is specified.
- **Multi-Session Batch** — Forward each child document in the batch in its own session, with a unique session ID.
- **Individual** — Forward each child document in the batch in its own session; do not forward objects representing the batch headers and trailers.

In responding to received messages, the default behavior is to send an acknowledgment to the sender as a batch document that contains an ACK message for each child document. This works for most situations. However, HL7 business services also have a property, not exposed as a configuration setting, called **NoBatchReply**. Its default value is 0 (false), which provides the default behavior. If you edit your business service's **OnInit()** method to include this statement:

```
Set ..NoBatchReply = 1
```

Then batch replies are inhibited; each individual message gets a separate unwrapped ACK. Alternatively, you could set the value by overriding the definition of the property in a subclass by including the following code:

```
Property NoBatchReply As %Boolean [ InitialExpression = 1 ];
```

5.3.3 Sending Batch Messages

On the outgoing side, HL7 **File** and **FTP** business operations have the **Auto Batch Parent Segs** configuration setting. When **Auto Batch Parent Segs** is **False** (the default) the business operation outputs child documents, but does not output the batch headers and trailers. When **Auto Batch Parent Segs** is **True**, while outputting a message that has a batch parent, the business operation outputs the batch headers first, then the child documents, then follows up with the batch trailers when triggered by the final batch header message or by a file name change.

5.3.4 Batch Modes

The combination of **Batch Handling** and **Auto Batch Parent Segs** enables the following modes of operation for HL7 batch documents:

Batch Handling	Auto Batch Parent Segs	Results
Whole Batch	(any)	Business service sends only the parent document; all child documents are referenced to it but not sent individually. Operation outputs entire batch at one time when it receives the parent document.
Single-Session or Multi-Session	True	Service sends each child document as it receives and parses it, followed by the parent document when all children have been sent. The business operation outputs parent headers when it receives the first child document, then finishes up with trailers when it receives the parent document object. Trailer segments automatically contain the correct child count values.
Single-Session or Multi-Session	False	This results in double output: the business operation sends out each child document individually, followed by the parent document containing each child document (again).
Individual	False	Business service forwards each child document in the batch in its own session and does not forward objects representing the batch headers and trailers. On the outgoing side, the business operation does the same.

5.3.5 Custom Outbound Batch Handling

If you wish to add custom code to your routing process to handle batch documents specially on the output side, you can do so. The following are two possibilities:

- Your routing process code creates new parent and child documents and links them, then sends each child to the business operation. The business operation must have **Auto Batch Parent Segs** set to **True**. The business operation outputs parent headers when it receives the first child document, then finishes up with trailers when it receives the parent document object. Trailer segments automatically contain the correct child count values.

- Your routing process code creates new parent and child documents and links them, but sends only the parent object via the business operation.

Reference

This section provides the following reference information:

- [Settings for HL7 Business Services](#)
- [Settings for HL7 Routing Processes](#)
- [Settings for HL7 Sequence Managers](#)
- [Settings for HL7 Business Operations](#)
- [HL7 Escape Sequences](#)

Settings for HL7 Business Services

Provides reference information for settings for HL7 business services.

Summary

HL7 business services have the following settings:

Group	Settings	See
Basic Settings	Target Config Names	“Settings for Business Services” in <i>Ensemble Virtual Documents</i>
	Ack Target Config Names , Message Schema Category	<i>sections in this topic</i>
Connection Settings	Framing	<i>section in this topic</i>
Additional Settings	Search Table Class	“Settings for Business Services” in <i>Ensemble Virtual Documents</i>
	Local Facility Application , ACK Mode , Use ACK Commit Codes , Ignore Inbound ACK , Add NACK ERR , NACK Error Code , Batch Handling , Default Char Encoding , DocTypeResolution , Save Replies	<i>sections in this topic</i>

The remaining settings are either common to all business services or are determined by the type of adapter. For information, see:

- [“Settings for All Business Services”](#) in *Configuring Ensemble Productions*
- [“Settings for the File Inbound Adapter”](#) in *Using File Adapters with Ensemble*
- [“Settings for the HTTP Inbound Adapter”](#) in *Using HTTP Adapters with Ensemble*
- [“Settings for the FTP Inbound Adapter”](#) in *Using FTP Adapters with Ensemble*

The difference for the special-purpose adapter called `EnsLib.HL7.Adapter.TCPInboundAdapter` is that it has **JobPerConnection** set to `False`, which is usually appropriate for HL7.

The most important settings for HL7 are as follows:

- **Pool Size** — The default value of 1 makes it possible to support FIFO (First In, First Out) processing. In many cases, multiple patient demographic updates must be received in order. For example, many applications require receipt of an ADT Registration message before they can process an Order message, an Order message must be received before a Result message, and so on.
- **Category** — This text label permits configuration items to be sorted in the configuration diagram.
- **Append Timestamp** — (**File** only) Appends a time stamp to filenames in the **Archive Path**.
- **Archive Path** — (**File** and **FTP** only) Specifies where to archive HL7 messages.
- **Call Interval** — The number of seconds to wait before looking for more input. The default is 5 seconds. The minimum is 0.1 seconds.

Ack Mode

Helps to establish the format and conventions for issuing HL7 acknowledgment messages in response to HL7 messages received. For business services, this setting can have one of the values shown in the following table.

Ack Mode	Meaning
Never	Do not send back any ACK.
Immediate	Return a Commit ACK reply message immediately upon receipt of the inbound message. This is the default if nothing is specified.
Application	<p>If message passes validation, wait for the ACK reply message from the target application and return this ACK when it arrives.</p> <p>In the situation where the caller is requesting a response and the Ensemble routing engine is not configured to forward back a response from any of its targets, Ensemble creates and returns ACK or NACK objects to return to the caller. If validation fails and the Ack Mode is <code>Application</code>, Ensemble does not contact the target application. Instead, it sends the caller an immediate <i>Validation NACK</i>.</p>
MSH-determined	<p>Return ACK reply messages as requested in the MSH header fields 15 and 16. Either field may contain one of the following four control codes:</p> <ul style="list-style-type: none"> AL — Always NE — Never ER — Error or reject conditions only SU — Successful completion only <p>MSH 15 (<code>AcceptAcknowledgmentType</code>) controls Commit ACK and MSH 16 (<code>ApplicationAcknowledgmentType</code>) controls Application ACK. Depending on how they are set in the incoming message MSH segment, one, both or neither may occur.</p>
Byte*	Send back a single ACK-code byte instead of an ACK message, immediately upon receipt of the inbound message. ASCII 6 means OK; ASCII 21 means Error. This option is not available for any of the built-in HL7 business services (TCP, File, HTTP, and so on) but it is available if you write a custom business service that subclasses <code>EnsLib.HL7.Service.Standard</code> without overriding the Ack Mode setting.

* Ensemble business operations automatically treat a single byte ASCII 6 as an HL7 ACK with an AA commit code and ASCII 21 as an HL7 ACK with an AE commit code.

For a general discussion of ACK handling, see “[HL7 Acknowledgment \(ACK\) Mode](#),” earlier in this book.

Ack Target Config Names

(**File** and **FTP** only) Unlike TCP business services, File and FTP business services have no persistent connection on which to send HL7 acknowledgment messages (ACK or NACK). For this reason, the default [Ack Mode](#) for File and FTP business services is `Never`, which is usually appropriate. However, when you do want to send ACKs from a File or FTP business service, use the **Ack Target Config Names** setting to identify an Ensemble routing process or business operation to receive the ACK messages.

See [Ack Mode](#) for more information. For a general discussion of ACK handling, see “[HL7 Acknowledgment \(ACK\) Mode](#),” earlier in this book.

Add NACK ERR

If True, when generating NACK messages, append an ERR segment that containing the Ensemble error codes and error text; otherwise do not embed internal error state information in NACK messages.

For a general discussion of ACK handling, see “[HL7 Acknowledgment \(ACK\) Mode](#),” earlier in this book.

Batch Handling

How to treat received message batches. The options are:

- `Whole Batch` — Do not process message documents individually; accumulate and send the whole batch as one composite document.
- `Single-Session Batch` — Forward all messages in the batch together in one session; the session includes objects representing the batch header and trailer segments. This is the default.
- `Multi-Session Batch` — Forward each message in the batch in its own session; each session includes objects representing the batch header and trailer segments.
- `Individual` — Forward each child message in the batch in its own session; do not forward objects representing the batch header and trailer segments.

For more about batch handling, see “[HL7 Batch Messages](#).”

Default Char Encoding

The character encoding of the inbound HL7 messages. Ensemble automatically translates the characters from this encoding.

Supported encoding values are UTF-8, Latin1, and any other NLS definitions installed on the Ensemble server. The value `Native` means to use the default encoding of the Ensemble server. You can also directly use an Ensemble translation table; to do so, use the value `@tablename`, where *tablename* is the name of the table.

By default, if an incoming HL7 message has a non-empty MSH:18 (Character Set) field, Ensemble uses that value rather than the setting. To force Ensemble to ignore MSH:18 and use this setting instead, place a ! (exclamation point) character at the beginning of the setting value. For example: `!UTF-8`

The default value depends on the adapter.

For background information on character translation in Caché, see “Localization Support” in the *Caché Programming Orientation Guide*.

DocTypeResolution

Specifies how to resolve a DocType based on the message type from MSH:9. Choose one of the following:

- `Standard` — Combine the effective [Message Schema Category](#) value with a message structure name looked up for the MSH:9 message type value in the corresponding schema category. This is the default.
- `Ignore 9.3` — Like 'Standard' except that if MSH:9 has three or more pieces, ignore the additional ones. The standard behavior is to use piece 3 as part of the type name if it has no sub-pieces because some schemas contain three-part type names.
- `Use 9.3` — Like 'Standard' except that if MSH:9 has three or more pieces, use the additional piece as the literal name of the document structure within the applicable schema category. Use with caution because messages may arrive with MSH:9.3 values for which no structure is present in the chosen schema category.
- `Literal` — Combine the effective [Message Schema Category](#) value with the literal MSH:9 message type value interpreted as the name of a message structure. Use only with custom schemas where every message type has a corresponding structure definition.

Framing

Controls how the HL7 business service interprets the incoming HL7 message packet. If you are unsure what value to use, accept the default of `Flexible` framing for HL7 business services.

The following table lists the valid values for this setting.

Framing Type	Inbound / Outbound	Meaning
Flexible	Inbound	Determine framing style from the content of received data.
None	Both	No framing; each line that begins with the string <code>MSH</code> is the start of a new message.
MLLP	Both	Minimal Lower Level Protocol — Frame each HL7 message with an ASCII 11 prefix and a suffix that consists of ASCII 28 followed by ASCII 13.
MLLP[nn]/[mm]	Both	MLLP using nonstandard ASCII values. Frame each HL7 message with a prefix that consists of the ASCII character value indicated by <i>nn</i> . Also provide a suffix that consists of the ASCII character value indicated by <i>mm</i> , followed by ASCII 13, the carriage return character.
AsciiLF	Both	Frame messages with ASCII 10, the line feed character, separating each message from the subsequent one.
AsciiCR	Both	Frame messages with an extra ASCII 13, the carriage return character, separating each message from the subsequent one.
Ascii[nn]	Both	Frame messages with a suffix separating each message from the subsequent one. This suffix consists of the ASCII character value indicated by <i>nn</i> .
Ascii[nn]/[mm]	Both	Frame messages with a prefix character before each message. This prefix consists of the ASCII character value indicated by <i>nn</i> . Also provide a suffix that consists of the ASCII character value indicated by <i>mm</i> , but with no trailing ASCII 13.
LLP	Both	(<i>Obsolete</i>) Lower Level Protocol — Frame each HL7 message in a redundant checksum block.
MsgEnvelope	Outbound	Use the message Envelope property verbatim if it is present. If the string <code><!--HL72MSG--></code> is present in the envelope, Ensemble replaces it with the Message content; otherwise the Message follows the envelope text.
MLLPMsgEnvelope	Outbound	Same as <code>MsgEnvelope</code> , but with the MLLP prefix and suffix also around the message inside the envelope.

When the framing type is `MLLP`, Ensemble automatically detects extra carriage returns (ASCII 13) that occur in the message before the close framing. This indicates to Ensemble that a blank line is *not* being used to separate messages, so it assumes that any blank line is part of the message content and can be safely ignored.

You can specify multiple characters. For example, if you require nonstandard framing such as `$Char(2)` for message start and `$Char(3,4)` for message end for your HL7 messages, you can use the `Ascii[nn]/[mm]` framing option as follows:

```
Ascii2/3,4
```

Note: ASCII values must be given as numeric values when you enter them in **Framing** fields. For example, enter lowercase `x` as `Ascii120`, not as `Ascii 'x'`.

Ignore Inbound ACK

If True, the business service ignores any inbound ACK messages to avoid creating an ACK feedback loop.

For a general discussion of ACK handling, see “[HL7 Acknowledgment \(ACK\) Mode](#),” earlier in this book.

Local Facility Application

Colon-separated *LocalFacility:LocalApplication* code that represents the facility and application that receive HL7 messages via this business service. If this business service creates its own ACK, **Local Facility Application** provides the *SendingFacility:SendingApplication* codes for the ACK message; otherwise, this setting is ignored.

Message Schema Category

Category to apply to incoming message types to produce a complete DocType specification. Combines with the document type *Name* (MSH:9) to produce a MessageType specification which Ensemble then uses to look up a MessageStructure / DocType in the MessageTypes section of the given HL7 schema category.

This setting may also contain multiple comma-separated type Names followed by = and then a DocTypeCategory or full DocType value to apply to HL7 messages containing that type Name. A trailing asterisk (*) at the end of a given partial type Name matches any types beginning with the entry.

DocType property of an HL7 message object

For example: MessageSchemaCategory='2.3.1, ADT_*=2.5, BAR_P10=2.4, ORM_O01_6=2.4:RDE_O01'

Note that a DocType assignment may be needed for Validation or [Search Table Class](#) indexing.

If you have not yet prepared a custom schema definition for the HL7 business service, you may leave this field blank for now. However, do not leave it blank permanently unless you also disable validation for the routing process, or validation errors will automatically occur. See “[Validation](#).”

NACK Error Code

Controls the error code in MSA:1 of the NACK message this service generates when there is an error processing the incoming message. The default is ContentE, which is to return code E for errors with the message content, and code R for system errors Ensemble encounters while attempting to process the message.

This distinction is important because Ensemble expects system errors to be solved so that if the remote client tries again at a later time they may not occur, while message content and validation errors are expected to require correction at the source and not to be worth retrying in the same form. Some client systems may expect or require a different error behavior; therefore, Ensemble provides three additional behaviors. The following table describes the four options.

Code	Meaning
ContentE	Use MSA error code E to report errors in message content and code R to reject due to (retryable) system errors
ContentR*	Return R for content errors, E for system errors
AllE	Return E for all content and system errors
AllR	Return R for all content and system errors

*Earlier versions of Ensemble used the ContentR behavior exclusively.

For a general discussion of ACK handling, see “[HL7 Acknowledgment \(ACK\) Mode](#),” earlier in this book.

Save Replies

Specifies whether to save a copy of reply messages sent back to the remote system. Also optionally specifies whether to index them using the configured search table class, if any. Choose one of the following:

- **None** — Do not save or index any reply messages.
- **NotOKs** — Save only the replies that are not a simple OK ACK message, for example, save error NACKS and query responses.
- **All** — Save a copy of all reply messages sent to the remote system.
- **IndexNotOKs** — Save replies that are not a simple OK ACK message *and* index them using the configured search table. This is the default behavior, unless you have overridden the *IndexReplies*, *SaveOKACKs*, or *IndexACKs* parameters in this class. Note that *IndexReplies*, *SaveOKACKs*, and *IndexACKs* are now deprecated.
- **IndexAll** — Save a copy of all reply messages *and* index them using the configured search table.

Use ACK Commit Codes

True or False. If True, when creating an ACK message for HL7 messages version 2.3 or higher, the business service places one of the *enhanced-mode* ACK commit codes in the MSA segment AcknowledgmentCode field.

HL7 business services have a **Use ACK Commit Codes** setting that applies if the **Message Schema Category** is 2.3 or higher. It can be True or False. If True, when creating an ACK message for HL7 messages version 2.3 or higher, the business service places one of the *enhanced-mode* ACK commit codes in the MSA segment AcknowledgmentCode field. This code may be one of the following two-letter sequences:

Code	Meaning in Original Mode	Meaning in Enhanced Mode
AA	Application Accept	Application acknowledgment: Accept
AE	Application Error	Application acknowledgment: Error
AR	Application Reject	Application acknowledgment: Reject
CA	—	Accept acknowledgment: Commit Accept
CE	—	Accept acknowledgment: Commit Error
CR	—	Accept acknowledgment: Commit Reject

For a general discussion of ACK handling, see “[HL7 Acknowledgment \(ACK\) Mode](#),” earlier in this book.

Settings for HL7 Routing Processes

Provides reference information for settings of an HL7 routing process.

Summary

HL7 routing processes have the following settings:

Group	Settings	See
Basic Settings	Business Rule Name	“Settings for Routing Processes” in <i>Ensemble Virtual Documents</i>
	Validation	<i>section in this topic</i>
Additional Settings	Act On Transform Errors , Act On Validation Errors , Alert On Bad Message , Bad Message Handler , Response From , Response Timeout , Force Sync Send	“Settings for Routing Processes” in <i>Ensemble Virtual Documents</i>
	Local Facility Application , Ack Type , NACK Code , Add NACK ERR , Response Target Config Names ,	<i>sections in this topic</i>
Development and Debugging	Rule Logging	“Settings for Routing Processes” in <i>Ensemble Virtual Documents</i>

The remaining settings are common to all business processes. See [“Settings for All Business Processes”](#) in *Configuring Ensemble Productions*.

Ack Type

Determines the ACK type (that is, AA or CA) if creating an ACK or NACK reply message locally. A is the default code. For further details, see [“HL7 Acknowledgment \(ACK\) Mode.”](#)

Add NACK ERR

If True, when generating NACK messages, append an ERR segment that containing the Ensemble error codes and error text; otherwise do not embed internal error state information in NACK messages.

Local Facility Application

Colon-separated *LocalFacility:LocalApplication* code that represents the facility and application that receive HL7 messages via this routing process. If this routing process creates its own ACK or NACK messages, **Local Facility Application** provides the *SendingFacility:SendingApplication* codes for the messages; otherwise, this setting is ignored.

NACK Code

Determines the NACK code type (that is, AE or AR) if creating a NACK reply message locally to report an error. E is the default code. For further details, see [“HL7 Acknowledgment \(ACK\) Mode.”](#)

Response Target Config Names

A comma-separated list of configured items within the production. If specified, this list identifies the destinations, in addition to the caller, to which responses will be forwarded. If empty, responses are only returned to the caller. This setting takes effect only if the **Response From** field has a value.

Validation

The **Validation** setting of an HL7 Message Router controls how the router validates the incoming message. If the incoming message fails the specified validation, Ensemble reports the failure in the event log, and the HL7 routing process passes the message to its bad message handler only; see the [Bad Message Handler](#) setting. If the message fails the specified validation but there is no bad message handler, an error is logged but the message is not sent to any target. If the message passes validation, the HL7 routing process sends the message to the targets specified by the routing rules.

Ideally, you can use routing rules and data transformations to ensure each message is acceptable to the target system, and can, consequently, avoid using validation. This ensures that all messages are processed by the appropriate target. If you enable validation, Ensemble applies the validation tests before the routing rules. Any message that fails validation will not be sent to a target based on the routing rules; it will be sent only to the Bad Message Handler. However, there are some environments where HL7 message validation is the preferred way to filter messages. For example, in the following situations, using HL7 validation is a good choice:

- You are developing or debugging an interface and want to determine the kind of message variants that your system needs to handle.
- The target application cannot handle messages that have variances from the specification, and the routing rules and transformations cannot resolve those variances.
- There is a regulatory or other business requirement that the messages conform to the specification.

HL7 validation does add overhead to the routing process. This overhead can be significant and can reduce the maximum load of messages that your production can handle.

The Validation property allows you to specify flags that control the following:

- Whether the message has a valid document type.
- Whether the message structure is validated.
- Whether trailing Z segments that are not specified in the schema are validated.
- What specific field validations should be applied.
- Whether the validation stops after reporting the first error or continues to validate the remaining segments and fields in the message.

If you specify an empty string as the Validation property value, the message router skips validation and routes all messages. When you create a new HL7 routing process in the Management Portal, the **Validation** setting is initialized to an empty string.

For compatibility with previous releases, setting **Validation** to 1 is treated the same as setting it to dm-z and setting **Validation** to 0 is treated the same as setting it to an empty string.

The message validation sequence is as follows:

1. Validate that the message object has a DocType property value.
2. Validate the segment order.
3. Validate the fields within segments.

If at any time during this sequence the message fails validation and the -x flag is not specified, the HL7 routing process does not validate the remaining segments of the message.

Note: A message can pass validation and not conform exactly to the schema definition depending on the **Validation** flags specified.

Validating the DocType Property

The business service sets the DocType property value based on the **Message Schema Category** defined for the service and the document name specified in the HL7 message MSH:9 field. If the Validation property value includes `d`, the message router tests if the DocType property has a value. If it does not have a value, the message is a bad message.

The business service does not assign a value to the DocType property under any of the following circumstances:

- The **Message Schema Category** is not specified for the business service.
- The document name in the HL7 message is not defined in the specified schema or, for custom schemas, in either the specified schema or the base schema.

Validating Segment Order

This section describes how routing engines validate segments.

If the Validation property has a value of `dm`, `dmz`, or `dm-z`, the routing process validation either validates the message or declares it bad, by the following tests:

1. If the segments in the message conforms to the message structure specified by the DocType value that the business service assigned to the message, the `m` validation is satisfied. For example, if the message has a DocType value of 2.3.1 and the MSH:9 value in the message is `ORU^R01` the message is verified against the `ORU_R01` message structure under the 2.3.1 category. If the message data is missing some required segments, or if it has extra undefined segments, as compared to the 2.3.1 `ORU_R01`, then it is a bad message. HL7 allows custom local variations to be added to standard messages as trailing Z segments. A trailing Z segment in a message is validated governed by the following rules:
 - a. If the schema specifies the trailing Z segment or specifies the “Zxx” wildcard segment, the trailing Z segment passes validation.
 - b. If the `-z` flag is specified, the trailing Z segment in the message passes validation.
 - c. If the `-z` flag is not specified and the trailing Z segment is not defined in the schema, the trailing Z segment fails validation.
 - d. If the schema requires a trailing Z segment and the message does not contain it, then the message fails validation.
2. If the DocType refers to a custom schema, the same rules apply as in test 1, except that the message is verified against the custom definition. Each custom schema has a base category defined (for example `base="2.3.1"`). If a message arrives that is not explicitly defined in the custom schema, the base category is used.

If MSH:9 field can contain a nonstandard value, you must define a custom schema to avoid having messages with this nonstandard value fail validation. For example, if MSH 9 is equal to `ORU^Z22` instead of `ORU^R01` a custom schema is required to specify that the message name `ORU^Z22` should be verified against the structure `ORU_R01`.

For information on creating custom schema categories, see “[Creating Custom Schema Categories](#)” in *Ensemble Virtual Documents*.

Validating Fields

The `f` flag enables all field validation. If you specify `f`, the message router performs the following field validation:

- Tests that field and component order in the message conform to the schema.
- Tests that required fields, components, and subcomponents are present.
- Test that field sizes in the message conform to the schema.
- Tests that field data values conform to the data type defined in the schema.
- Tests that field data values conforms to the code tables, but allows any value if the code table includes the `...` value or if the code table is empty.

Specifying Validation Flags

Each flag can be preceded with the `-` hyphen character to negate the meaning of the flag. For example, the `x` flag specifies that the routing process should stop validation after encountering the first error, and the `-x` flag specifies that the routing process should continue validation until the entire message is validated and all errors detected. Some validation flags are equivalent to specifying a series of flags. For example, the `s` flag is equivalent to specifying the `jpw` flags. You can remove one of the series by specifying a negating flag. For example `s-p` is equivalent to `jw`. To negate a flag, you must enter a hyphen immediately before the flag character. For example, if you want to negate both the `z` and `x` flags, you must specify `-z-x`.

The following table lists the HL7 validation flags and describes how the routing process validates the message when each is specified.

Flag	Routing Process
a	Enforces field array repetition limits.
b	Interprets <code>. . .</code> in a code table to allow any field value to pass validation.
c	Performs all component and subcomponent validation. Is equivalent to <code>gijopw</code> .
d	Requires that the DocType property be defined. Note that every other flag requires that DocType is defined in order to perform the specified validation.
e	Performs every validation. Is equivalent to <code>abdgi jlmnoprtuwy</code> .
f	Performs all field validation within segments. Is equivalent to <code>abgi jlnoprtuwy</code> .
g	Enforces data structures within segments.
i	Enforces component size restrictions.
j	Ensures that required subcomponents are present. These are governed by the optionality setting in the schema.
l	Enforces field size restrictions.
m	Ensures that all required segments in the schema definition are included in the message and that the message does not contain any misplaced segments that are not allowed by the schema.
n	Ensures that the segments contain the correct number of fields.
o	Ensures that required components are present. These are governed by the optionality setting in the schema.
p	Enforces component-level data structures.
r	Ensures that required fields are present.
s	Performs all subcomponent-level validation. Is equivalent to <code>jpw</code> .
t	Enforces code tables. This flag can be modified by the <code>b</code> and <code>u</code> flags.
u	Ignores code tables that do not list any permissible value. Permits any value for these empty code tables.
w	Enforces subcomponent size restrictions.
x	Stops validating after the first error is encountered. This is the default. <code>-x</code> specifies that the entire message is to be validated and all errors returned in the status.
y	Enforces field data types, such as numeric.

Flag	Routing Process
z	Requires that Z segments be explicitly specified in the schema or specified by the wildcard <code>Zxx</code> in the schema. <code>-z</code> allows trailing Z segments even if they are not specified in the schema.
(empty string)	Skips validation and routes all messages.

Settings for HL7 Sequence Managers

Provides reference information for settings of an HL7 sequence manager.

Summary

HL7 sequence managers have the following settings:

Group	Settings
Additional Settings	Enabled Duplicated Message Check, Perform Sequence Number Check On, Perform Output Transformation On, Output Sequence Number Index Field, Output Target Config Names, Message Wait Timeout, Passthrough Message Types, Out Of Sequence Message Notice Type, Out Of Sequence Message Target, Duplicated Message Notice Type, Duplicated Message Target, Output Facility Application, Message Resendable Time Window, Large Gap Size, Bypass Check On Internal Resent

The remaining settings are common to all business processes. See “[Settings for All Business Processes](#)” in *Configuring Ensemble Productions*.

Bypass Check On Internal Resent

If this setting is true, the sequence manager bypasses checks on internal resent messages.

Duplicated Message Notice Type

Specifies the message type for a duplicated message notice. Specify either `WorkflowRequest` or `OriginalMessage`

Duplicated Message Target

The configured **Name** of an item within the production, usually a business operation. If specified, this is where the sequence manager sends any duplicate messages that it receives.

Enable Duplicated Message Check

If True, the sequence manager checks to see if any incoming messages are duplicates of a previously received message. It does this by checking the following three fields in the MSH segment of each message:

- MSH:3 SendingApplication
- MSH:4 SendingFacility
- MSH:10 MessageControlId

A sequence manager can check for duplicate messages *and* messages out of sequence, duplicate messages *or* messages out of sequence, or neither. This makes **Enable Duplicated Message Check** entirely independent of **Perform Sequence Number Check On** in that either, one, or both of these settings may be True.

Large Gap Size

A number that indicates a significant gap in the sequence of messages; the default is 100.

A *late message* is a message that arrives with a sequence number larger than the previously received message in the sequence, but not exactly sequential. For example, when message 102 arrives after message 101, 102 is not late; it is the next message in sequence. However, when message 110 arrives after message 101, 110 is a late message.

When there is a late message, and the sequence numbering gap between the preceding and subsequent messages is less than the configured **Large Gap Size**, this is a *small gap*. A gap larger than this is a *large gap*.

- For any small gap, the sequence manager waits for **Message Wait Timeout** seconds to see if messages arrive to fill the gap. If the messages arrives within the waiting period, the sequence manager aligns them in their proper sequence. If they do not arrive and the **Message Wait Timeout** expires, the sequence manager stops waiting and sends whatever it has for messages in the current sequence.
- When there is a large gap, the sequence manager does not wait for any period of time. It immediately sends an alert, then sends whatever it has for messages in the current sequence.

Message Resendable Time Window

Time window during which the sequence manager considers a duplicate message to be a duplicate. The reason for the **Message Resendable Time Window** is that sometimes a duplicate message is not a mistake. Sometimes it is the result of deliberately resending a message sequence.

Suppose a message arrives and the sequence manager detects that it is a duplicate of a previously received message. In this case:

- If it has been more than **Message Resendable Time Window** seconds since the sequence manager first saw this message, it interprets the new arrival as a deliberately resent copy. The sequence manager begins tracking the messages in this resent sequence independently from its tracking of the main sequence of messages.

Suppose the sequence manager is processing a sequence that includes numbers such as 1001, 1002, 1003, and 1004, but at the same time it begins receiving messages with the sequence numbers 1, 2, and 3. If the older messages with smaller sequence numbers were first received more than **Message Resendable Time Window** seconds ago, the sequence manager interprets them as messages in a resent sequence, rather than as messages out of sequence. It begins tracking the main sequence and the resent sequence simultaneously. While doing so, it keeps the two sequences separate, appropriately ignoring gaps between unrelated sequence numbers such as 2 and 1001.

- If it has been less than **Message Resendable Time Window** seconds since the sequence manager first saw this message, the new arrival is understood to be a true duplicate, and the sequence manager handles it appropriately. See **Enable Duplicated Message Check**.

The default **Message Resendable Time Window** is 300 seconds (5 minutes).

Message Wait Timeout

For definitions of large gaps, small gaps, and late messages, see the **Large Gap Size** setting.

For any small gap, the sequence manager waits for **Message Wait Timeout** seconds to see if messages will arrive to fill the gap between the preceding message and the late message. If they do not arrive and the **Message Wait Timeout** expires, the sequence manager stops waiting and sends whatever it has for messages in the current sequence.

The default **Message Wait Timeout** is 60.

Out Of Sequence Message Notice Type

Specifies the message type for an out-of-sequence message notice. Specify either `WorkflowRequest` or `OriginalMessage`

Out Of Sequence Message Target

The configured **Name** of an item within the production, usually a business operation. If specified, this is where the sequence manager sends any out-of-sequence message notices.

Output Facility Application

Specifies the facility and application to be used in the message transformation when **Perform Output Transformation On** is set to `Sender` or `Receiver`.

The format for the **Output Facility Application** value is:

Facility:Application

Output Sequence Number Index Field

Controls the fields on which the output sequence number is incremented. Specify one of the following values:

- **Sender** — The index fields are MSH:3 SendingApplication and MSH:4 SendingFacility.
- **Receiver** — The index fields are MSH:5 ReceivingApplication and MSH:6 ReceivingFacility.
- **Auto** — The value of this setting is controlled by the **Perform Output Transformation On** setting. If **Perform Output Transformation On** is **Receiver**, then **Output Sequence Number Index Field** is **Receiver**. Otherwise, **Output Sequence Number Index Field** is **Sender**.

Output Target Config Names

A comma-separated list of configured items within the production. If specified, this list identifies the destinations to which the sequence manager will send any message that passes its tests.

Passthrough Message Types

A comma-separated list of message types that are exempt from duplicate checks, sequence checks, or output transformations. The sequence manager always passes these messages through unchanged. The default list is:

QBP_Q21,QBP_Q22,RSP_K21,RSP_K22,ACK

Perform Output Transformation On

Specifies fields to transform before sending out a message. The output transformation provides new facility, application, and sequence number fields as described in the following list, then sends the message to its configured targets. Possible values for **Perform Output Transformation On** are:

- **Sender** — This is the default. The sequence manager copies *Facility* and *Application* from the **Output Facility Application** setting and assigns a new sequence number. The result is that the following fields are changed in the outgoing message:
 - MSH:3 SendingApplication
 - MSH:4 SendingFacility
 - MSH:13 SequenceNumber
- **Receiver** — The sequence manager copies *Facility* and *Application* from the **Output Facility Application** setting and assigns a new sequence number. The result is that the following fields are changed in the outgoing message:
 - MSH:5 ReceivingApplication
 - MSH:6 ReceivingFacility
 - MSH:13 SequenceNumber
- **None** — The sequence manager does not perform any transformation before sending the message.

Perform Sequence Number Check On

Specifies the fields to check to see if any of the incoming messages are out of sequence. If so, the sequence manager resequences the messages and sends them to its configured targets.

Possible values are:

- **Sender** — This is the default. The sequence manager checks:
 - MSH:3 SendingApplication
 - MSH:4 SendingFacility
 - MSH:13 SequenceNumber
- **SendingFacility** — The sequence manager checks:
 - MSH:4 SendingFacility
 - MSH:13 SequenceNumber
- **SendingApplication** — The sequence manager checks:
 - MSH:3 SendingApplication
 - MSH:13 SequenceNumber
- **Receiver** — The sequence manager checks:
 - MSH:5 ReceivingApplication
 - MSH:6 ReceivingFacility
 - MSH:13 SequenceNumber
- **ReceivingFacility** — The sequence manager checks:
 - MSH:6 ReceivingFacility
 - MSH:13 SequenceNumber
- **ReceivingApplication** — The sequence manager checks:
 - MSH:5 ReceivingApplication
 - MSH:13 SequenceNumber
- **None** — The sequence manager does not check for messages out of sequence.

A sequence manager can check for duplicate messages *and* messages out of sequence, duplicate messages *or* messages out of sequence, or neither. This makes **Enable Duplicated Message Check** entirely independent of **Perform Sequence Number Check On** in that, either one, or both of these settings may be True.

Settings for HL7 Business Operations

Provides reference information for settings of an HL7 business operation.

Summary

HL7 business operations have the following settings:

Group	Settings	See
Basic Settings	File Name	<i>section in this topic</i>
Connection Settings	Get Reply , Framing	<i>sections in this topic</i>
Additional Settings	Search Table Class	<i>“Settings for Business Operations” in <i>Ensemble Virtual Documents</i></i>
	Auto Batch Parent Segs , Separators , Default Char Encoding , Reply Code Actions , No Fail While Disconnected , Retry Interval	<i>sections in this topic</i>

The remaining settings are either common to all business operations or are determined by the type of adapter. For information, see:

- “[Settings for All Business Operations](#)” in *Configuring Ensemble Productions*
- “[Settings for the File Outbound Adapter](#)” in *Using File Adapters with Ensemble*
- “[Settings for the FTP Outbound Adapter](#)” in *Using FTP Adapters with Ensemble*
- “[Settings for the HTTP Outbound Adapter](#)” in *Using HTTP Adapters with Ensemble*
- “[Settings for the TCP Outbound Adapter](#)” in *Using TCP Adapters with Ensemble*

EnLib.HL7.Adapter.TCPOutboundAdapter has the following settings configured appropriately for HL7:

- **Get Reply** is set to True. This means the adapter will wait to read a reply message from the socket before returning.
- **Connect Timeout** has its usual default of 5 seconds, but has a maximum limit of 30,000 seconds.
- **Response Timeout** has a default of 30 instead of its usual 15, and has a maximum limit of 30,000 seconds.

For an HL7 business operation, the most important settings are these:

- **Pool Size** — The default **Pool Size** value of 1 makes it possible to support FIFO (First In, First Out) processing. FIFO processing is important to ensure correct data in the receiving applications. Multiple patient demographic updates must be received in order; otherwise, obsolete, incorrect data may appear in the receiving application. Also, many applications require receipt of an ADT Registration message before they can process an Order message, an Order message must be received before a Result message, and so on.
- **Failure Timeout** — The number of seconds during which to continue retry attempts. HL7 business operations automatically set this value to -1 for “never time out” to ensure that no HL7 message is skipped.
- **Category** — This text label permits configuration items to be sorted in the configuration diagram.
- **Reconnect Retry** — (**TCP** only) Number of retries at which to drop the connection and try reconnecting again. A value of 0 (zero) means never disconnect. The default is 5.
- **Stay Connected** — The default of -1 means to stay permanently connected, even during idle times. Adapters are assumed idle at startup and therefore only auto-connect if they are configured with a StayConnected value of -1.

Auto Batch Parent Segs

(**File** and **FTP** only) If True, when writing a message that has a batch parent, output the batch headers first, then child documents, then follow up with the batch trailers when triggered by the final batch header message or by a file name change. If False, omit headers and trailers and output child documents only. The default is False.

For more about batch handling, see “[HL7 Batch Messages](#).”

Default Char Encoding

The desired character encoding of the outbound HL7 messages. Ensemble automatically translates the characters to this encoding. See “[Default Char Encoding](#)” for business services.

File Name

(**File** and **FTP** only) The target file name. The **File Path** adapter setting determines the path for this file; **File Name** determines the name. **File Name** can include Ensemble time stamp specifiers. If you leave **File Name** blank, the default uses the time stamp specifier %f_%Q where:

- %f is the name of the data source, in this case the input filename
- _ is the literal underscore character, which will appear in the output filename
- %Q indicates ODBC format date and time

For full details about time stamp conventions, including a variety of codes you can use instead of the default %f_%Q, see “[Time Stamp Specifications for Filenames](#)” in *Configuring Ensemble Productions*.

Framing

Controls how the HL7 business operation creates the outgoing HL7 message packet. For a list of framing options, see “[Framing](#)” in “[Settings of an HL7 Business Service](#).” If you are unsure what value to use, accept the default of MLLP framing for HL7 business operations.

Get Reply

(TCP business operations only) If True, the business operation waits to read an ACK or other reply message from the socket before returning. It also applies any defined [Reply Code Actions](#).

For a general discussion of ACK handling, see “[HL7 Acknowledgment \(ACK\) Mode](#),” earlier in this book.

No Fail While Disconnected

(TCP only) If True, suspend counting seconds toward the **Failure Timeout** while disconnected from the TCP server. This setting does not apply if **Failure Timeout** is -1 or if **Stay Connected** is 0.

Reply Code Actions

Allows you to supply a comma-separated list of code-action pairs, specifying which action the business operation takes on receipt of various types of ACK response messages. The format of the list is:

code=action , code=action , ... code=action

Where *code* (starting with a :) represents a literal value found in the MSA:1 (Acknowledgment Code) field of the response message, or one of the following special code values:

Code	Meaning
:?A	Matches AA or CA values (Accept)
:?E	Matches AE or CE values (Error)

Code	Meaning
:?R	Matches AR or CR values (Reject)
:_	Matches replies with an empty MSA:1 field. An empty or whitespace code value is the same as _ (underscore)
:*	Matches any MSA:1 value not matched otherwise (default=S)
:~	Matches replies that do not contain an MSA segment
:I?	Matches where the reply MSA:2 ControllId does not match the ControllId of the original message
:T?	Matches where the reply MSH:9 Type name does not match the schema's declared reply type for the original message

You can also use the standard codes described in “[Reply Code Actions](#)” in the reference section of *Configuring Ensemble Productions*.

The following values for *action* may be used alone or combined to form strings. F is the default *action* if no other is given, except for :?A whose default action is C:

Action	Meaning
C	Treat the message as Completed OK.
W	Log a warning but treat the message as Completed OK.
R	Retry the message according to the configured RetryInterval and FailureTimeout; finally Fail unless a different action is also specified
S	Suspend the message, log an error, and move on to try the next message
D	Disable the Operation, log an error and restore the outbound message to the front of the Operation's queue
F	Fail with an error and move on to try the next message

`code=action, code=action, ... code=action`

The default value for **Reply Code Actions** is:

`:?R=RF, :?E=S, :~=S, :?A=C, :*=S, :I?=W, :T?=C`

This default indicates that Ensemble retries messages with acknowledgment codes AR or CR; for those with codes AE or CE, it suspends the current message, logs an error, and moves on to the next message. The default also treats any message with codes AA or CA as *Completed OK* and suspends messages that have a value in field MSA:1 that is not matched by any other listed reply code.

Note: The default for retry is ?R=RF. In most cases, this fails immediately after an error, but if the error is "ERROR #5005: Cannot open file", the operation continues to retry until it reaches the FailureTimeout. In many cases this error is caused by a transient problem and retrying fixes the problem, but, in some cases, such as an incorrect directory, the operation fails repeatedly.

For batch ACK messages, the business operation determines the correct **Reply Code Actions** based on the first child ACK found within the batch.

For a general discussion of ACK handling, see “[HL7 Acknowledgment \(ACK\) Mode](#),” earlier in this book.

Retry Interval

Number of seconds to wait between attempts to connect with a destination outside Ensemble.

Separators

HL7 separator characters to use in the outgoing message. If you leave this field blank, the default is:

|^~\&

Basics

An HL7 message uses special characters to organize its raw contents. These characters may vary from one clinical application to another. For this reason, the HL7 standard requires that each HL7 message list the five specific characters that it is using as separators at the start of the MSH segment, in order from left to right:

1. Field separator (FS)
2. Component separator (CS)
3. Repetition separator (RS)
4. Escape character (ESC)
5. Subcomponent separator (SS)

A sixth character, the segment terminator character, is not specified in MSH and is generally assumed to be a carriage return (ASCII 13).

Details

For **Separators**, you must supply a string of characters which Ensemble assigns to HL7 separators in left to right order: FS, CS, RS, ESC, SS as described in the previous list.

Beyond positions 1 through 5 of the **Separators** string, you can supply additional characters to override the default segment terminator character, the carriage return (ASCII 13). After position 5, use `\r` for the carriage return (ASCII 13) and `\n` for the line feed (ASCII 10).

You can use `\x` in positions 1 through 5 if you need to specify segment terminators in positions 6 and higher but want your output messages to use fewer than 5 separators. Separators designated by `\x` in positions 1 through 5 are not used. The purpose of `\x` is simply to extend the length of the list of separators so that position 6 is interpreted correctly as the first segment terminator.

HL7 Escape Sequences

Provides reference information for HL7 escape sequences.

Details

When separator characters need to appear within the data contents of an HL7 message, HL7 provides escape sequences to replace the separator characters.

The character most likely to require this feature is the & (ampersand) character. It is often used as an HL7 separator. At the same time, & is likely to appear in HL7 document data, either within the legal names of employers (Parker & Sons) or as shorthand in treatment orders (XR CHEST PA&LAT).

Thus, within the HL7 data stream you often see & characters replaced by an escape sequence, such as the \T\ in the following samples:

```
Parker \T\ Sons
XR CHEST PA\T\LAT
```

The following table lists the standard HL7 escape sequences and their meanings. HL7 escape sequences begin and end with the escape character as defined in the MSH segment. HL7 escape sequences are case-sensitive, and they use the specific characters F R S T E as well as .br, X0A, and X0D as shown in the table. In the examples shown, the \ (backslash) is the escape character.

This character sequence...	If found within HL7 data, indicates...	For example...
\.br*	Carriage return	
\F\	Field separator character	
\R\	Repetition separator character	~
\S\	Component separator character	^
\T\	Subcomponent separator character	&
\E\	Escape character	\
\X0A\	Line feed	
\X0D\	Carriage return	

* Only used for unescaping.

If you <assign> a literal string to a target field using DTL, and that string needs to include HL7 separator characters, either *as separators* or *as literal characters*, some special handling is required. The procedures are as follows.

Suppose you have a source message in which ^, \, and & are separators as listed in the MSH segment:

```
MSH|^~\&
```

Now suppose that, for all ADT_A01 messages that arrive from a certain clinical source, your DTL data transformation needs to assign a literal value to the InsuranceCompanyName field. The correct value has been agreed upon by all concerned parties, including the insurance company, whose name is Pierre DuRhône & Cie. The value is to appear in the HL7 data stream as follows:

```
Pierre DuRho^ne & Cie
```

In this case, you need HL7 escape sequences to replace ^ and & within the data, as follows:

XML

```
<assign
  property='target.{PIDgrp.INlgrp(1).INl:InsuranceCompanyName.organizationname}'
  value='"Pierre DuRho\S\ne \T\ Cie"' />
```

The only other case in which you need to be concerned about separator characters is if you wish to migrate data into or out of an Ensemble HL7 virtual document and you are not using an HL7 to HL7 data transformation. For example, you need to explicitly deal with separator characters if you are transforming from XML to HL7 or if you are not using a transformation at all. The `EnsLib.HL7.Segment` class provides explicit **Escape()** and **Unescape()** methods that you can invoke to accomplish manually what a DTL data transformation does automatically to unescape and escape HL7 messages. A **replace()** method also exists.