Ensemble HL7 Version 3 Development Guide

Version 2018.1
2024-05-02

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

Ensemble HL7 Version 3 Development Guide
Ensemble Version 2018.1 2024-05-02

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

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 describes how to add HL7 Version 3 interfaces to Ensemble routing productions. It contains the following sections:

- Ensemble Support for HL7 Version 3
- Elements of a Routing Production
- Sample Routing Production

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* explains how the concept of virtual documents allows Ensemble to provide efficient support for EDI document exchange.

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

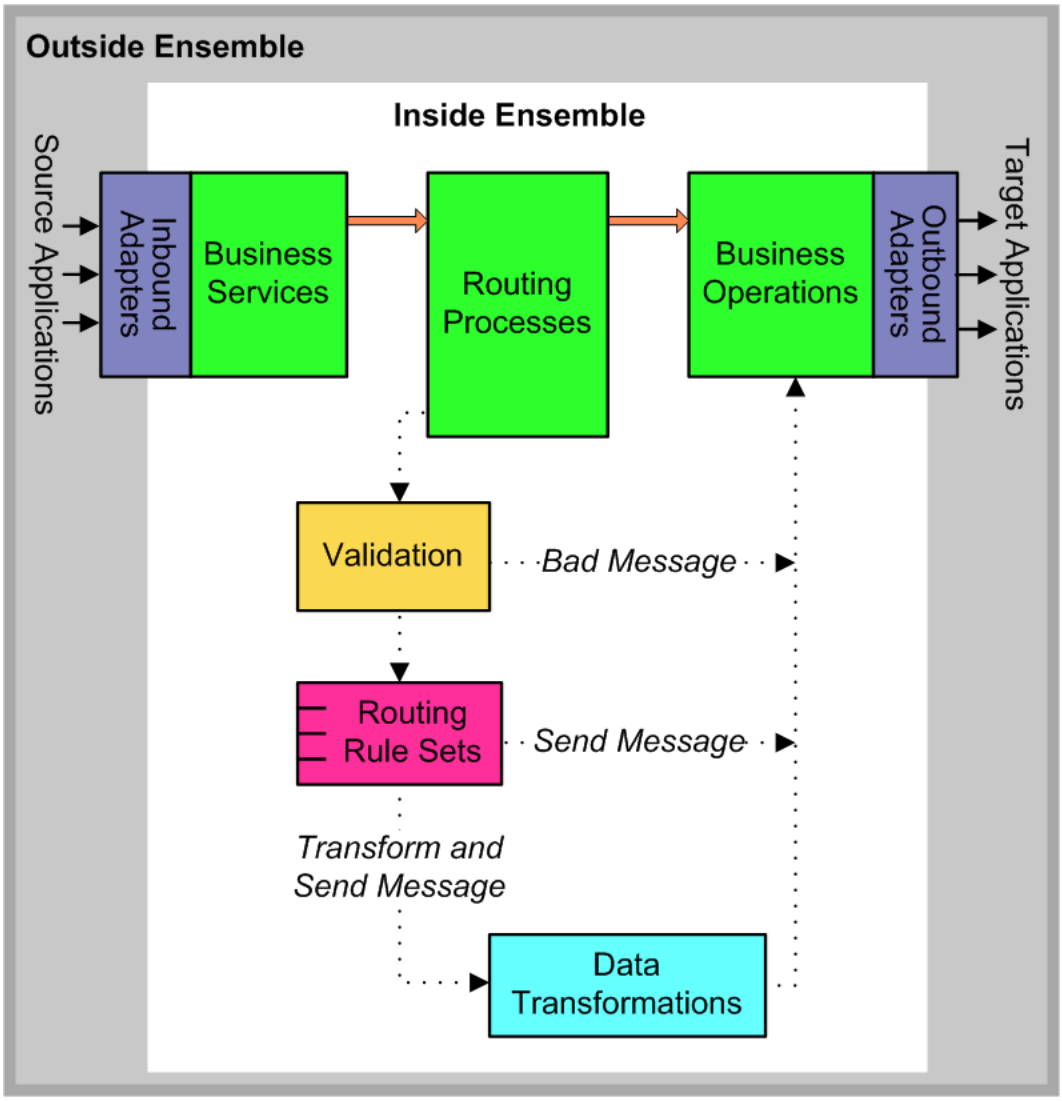
1

Ensemble Support for HL7 Version 3

Unlike previous versions of Health Level Seven, HL7 Version 3.0 uses only XML encoding for messages. This is a significant departure from HL7 Version 2.x, with several unique challenges.

This book explains how to route HL7 Version 3.0 messages from one application to another using Ensemble as the routing engine. The book explains how to transform the XML contents of these messages using a combination of XPath and XSLT expressions in Ensemble data transformations.

The following figure illustrates the flow of HL7 Version 3.0 messages through an Ensemble production that works as an HL7 interface routing engine.



An HL7 Version 3.0 message flows through the elements of an Ensemble HL7 routing production in the following order:

1. An *HL7 business service* receives an incoming data stream from the specific source application whose HL7 messages it is configured to accept. The HL7 business service generates an XML document from the incoming data stream and extracts from it the information needed to route and transform the message.
2. The business service packages the relevant data into an Ensemble message and sends it to a specific *HL7 routing process*. This is a special-purpose Ensemble business process that prepares HL7 Version 3.0 messages for delivery outside Ensemble.
3. The routing process validates the XML document structure. If validation fails, the routing process passes the message to the *bad message handler*. This is an HL7 business operation that disposes of any incoming HL7 messages that an interface was unable to parse, usually by saving the messages to a file so that they can be analyzed.
4. If validation succeeds, the HL7 routing process applies a *routing rule set* to the Ensemble message. The routing rule set provides the logic that determines:
   - A destination for the message
   - Whether or not the message data requires transformation before it is sent.
5. Each *data transformation* uses XPath to find the appropriate fields in the data stream, then uses XSLT to transform the fields as needed and write the result to a new XML document.

6. The routing process passes an Ensemble message containing the new XML document to the appropriate *HL7 business operation*. The business operation uses the routing properties of the Ensemble message to direct the new XML document to the target application as an outgoing data stream.

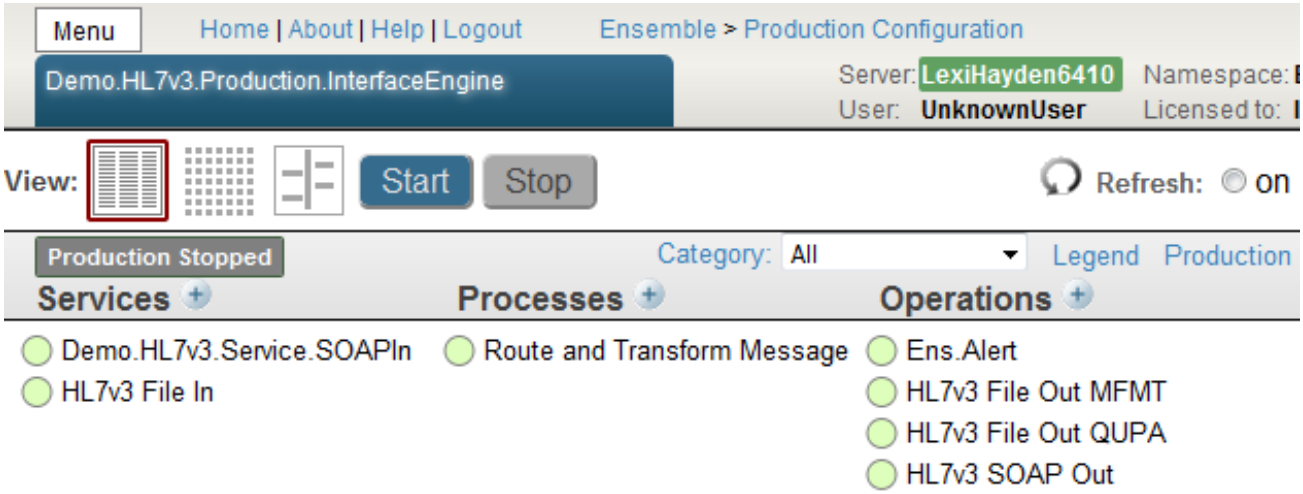
As further background information, *Ensemble Best Practices* describes best practices for organizing and developing Ensemble productions. It provides information that applies to all productions, including those that route HL7 Version 3.0 messages and other EDI format documents.

A useful sequel to *Ensemble Best Practices*, the book *Developing Ensemble Productions* describes specific Ensemble development practices in detail.

2

Elements of a Routing Production

In the ENSDEMO namespace, Ensemble provides a simple HL7 Version 3.0 message routing production in the package Demo.HL7v3. The production diagram for this example is shown below.



2.1 Business Services

Each business service in an HL7 Version 3.0 message routing production receives an incoming data stream from a specific source application. The business service generates an XML document from the incoming data stream. It extracts from this document the information it needs to route and transform the message.

The sample classes Demo.HL7v3.Service.FileIn and Demo.HL7v3.Service.SOAPIn are business services. You can find these classes in the ENSDEMO namespace. Demo.HL7v3.Service.FileIn is shown below.

Class Definition

```
Class Demo.HL7v3.Service.FileIn Extends Ens.BusinessService
{

Parameter ADAPTER = "EnsLib.File.InboundAdapter";
```

```
Method OnProcessInput(pInput As %FileCharacterStream,
                      pOutput As Ens.Response) As %Status
{
  Set $ZTrap = "OnProcessInputET"

  Set tStatus =
  ##class(%XML.XPATH.Document).CreateFromStream(pInput, .tDocument)
  Set tStatus = tDocument.EvaluateExpression("/*", "name()", .tResults)
  Set tStatus = pInput.Rewind()

  If (tResults.Count() > 0) Set tRoot = tResults.GetAt(1).Value
  Else  Set tRoot = "<errorNoRootElement>"

  Set tRequest = ##class(Demo.HL7v3.Message).%New()
  Set tRequest.Name    = tRoot
  Set tRequest.DocType  = ""
  Set tRequest.Source     = pInput.Attributes("Filename")
  Do tRequest.Content.CopyFrom(pInput)

  Set tStatus =
  ..SendRequestSync("Route and Transform Message", tRequest, .tResponse)

  Quit $$$OK

OnProcessInputET
  Set $ZTrap = ""

  Quit $$$ERROR($$$GeneralError,"Error in OnProcessInput():  "_$ZError)
}

}
```

Upon encountering an input file, Demo.HL7v3.Service.FileIn calls the **CreateFromStream()** method to create an XML document from the incoming character stream.

This document is an object of type %XML.XPATH.Document. As such, it has a method called **EvaluateExpression()**. The business service calls this method to evaluate the XPath context (`"/*"`) and expression (`"name()"`). The context selects the initial nodeset from the document and the expression further filters the node set. If **EvaluateExpression()** succeeds, it returns a list of results (`.tResult`) which can be queried for their types and values. The business service does this, accepting the name of the first XML element in the input stream as the message Name.

Tip: For full details of XPath syntax and usage, see the web site `https://www.w3.org/TR/xpath`

Demo.HL7v3.Service.FileIn completes creation of an Ensemble message by recording the name of the input file and copying the message data into an appropriate stream object.

The business service sends the message to its designated routing process, using the configured name of the routing process in a call to **SendRequestSync()**.

You can create a business service host class for an HL7 Version 3.0 message routing production by copying and customizing the sample classes in the Demo.HL7v3 package, or by creating your own subclass of Ens.BusinessService. After you create the class, add the business service to the production.

The Management Portal provides a New Business Service wizard with an **HL7** option. This option creates an HL7 Version 2.x business service that is *not* suitable for use with HL7 Version 3.0.

2.2 Messages

In designing an HL7 Version 3.0 message routing production, it can be convenient to define a message type that is suitable for both requests and responses. This is the case in the sample production in the Demo.HL7v3 package in the ENSDEMO namespace.

The sample message class Demo.HL7v3.Message contains the minimum properties required to achieve the goal of routing, transforming, and time-stamping an HL7 Version 3.0 message that originated in a external file. Demo.HL7v3.Message is shown below.

Class Definition

```
Class Demo.HL7v3.Message Extends (%Persistent, %XML.Adaptor)
{
Property Content As %GlobalCharacterStream;

Property Name As %String;

Property DocType As %String;

Property Source As %String;

}
```

Content is the data stream.

Name and DocType are properties expected by the HL7 routing process; both must be present in the message class. In the sample production, Name is the property used to route the message.

Source is used in the business operation to provide the timestamp. It varies depending on which business service originated the message data. It identifies either the input file, or the name of the web service that relayed the data into the SOAP business service.

You can create a message class for an HL7 Version 3.0 message routing production by copying and customizing the sample class Demo.HL7v3.Message, or by creating your own class.

2.3 Routing Processes

The routing process for an HL7 Version 3.0 message routing production validates the XML document that it receives from its associated business service. If validation fails, the routing process sends the XML document to its bad message handler. If validation succeeds, the routing process invokes its routing rule set to determine what to do with the XML document.

Upon finding a matching rule, the routing process carries out the instructions for that rule. It may invoke a data transformation class **Transform()** method or send an Ensemble message to a business operation (or both, in sequence).

The sample class Demo.HL7v3.MsgRouter.RoutingEngine has an empty **OnValidate()** method. You can find Demo.HL7v3.MsgRouter.RoutingEngine in the ENSDEMO namespace. If you copy this class to create a routing process class for an HL7 Version 3.0 message routing production, you should put meaningful code into the **OnValidate()** method.

Demo.HL7v3.MsgRouter.RoutingEngine performs a simple, implicit validation test by calling **CreateFromStream()** to convert the stream Contents of the message into an XML document. If this call fails, the routing process invokes its bad message handler. Otherwise, it invokes its routing rule set.

You can create a routing process class for an HL7 Version 3.0 message routing production by copying and customizing the sample class Demo.HL7v3.MsgRouter.RoutingEngine, or by creating your own subclass of EnsLib.MsgRouter.RoutingEngine. After you create the class, add the routing process to the production.

The Management Portal provides a New Business Process wizard with an **HL7 Message Router** option. This option creates an HL7 Version 2.x routing process that is *not* suitable for use with HL7 Version 3.0.

2.4 Routing Rule Sets

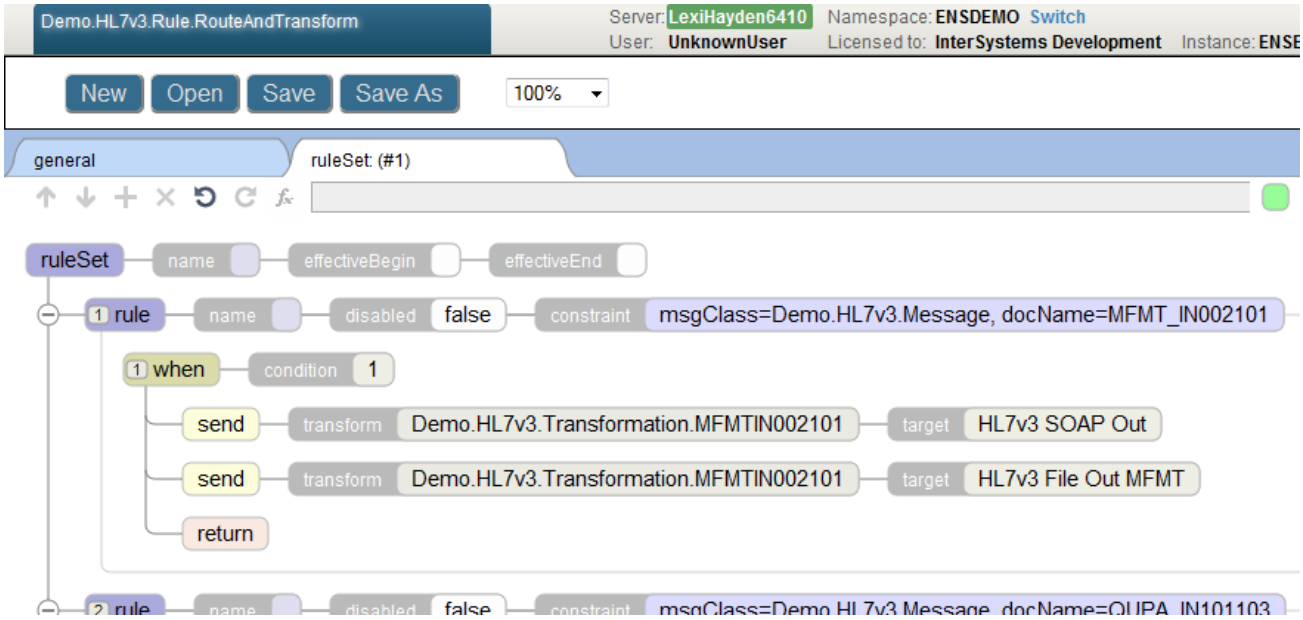
A routing rule set provides the logic that determines:

- Where to send the message
- Whether to transform, and if so, how to transform the message before sending it

For details, see "Defining Routing Rule Sets for HL7" in the *Ensemble HL7v2 Development Guide*.

Everything about routing rule sets for HL7 Version 3.0 message routing productions is the same as for HL7 Version 2.x, except that for Version 3.0 the **Message Class** is whatever class the business service sends to the routing process. In the sample code, this class is Demo.HL7v3.Message. You can find it in the ENSDEMO namespace.

The full name of the sample routing rule set in the Demo.HL7v3 package is Demo.HL7v3.Rule.RouteAndTransform. The following figure shows this rule definition as seen in the **Ensemble Rule Editor** page. (To access this page in the Management Portal, click **Ensemble** > **Build** > **Business Rules**.)



This rule definition takes different actions depending on the type of message. The Demo.HL7v3 sample expects each message to be one of two types, MFMT_IN002101 or QUPA_IN101103. Both of the sample business services contain code that determines the type of the incoming message based on the structure of the data stream. The business services then place the appropriate string in the Name property of the Demo.HL7v3.Message that they send to the routing process. This is how the message type, MFMT_IN002101 or QUPA_IN101103, becomes available to the routing rule set.

The sample data that you use to run the sample production must actually contain HL7 Version 3.0 messages of type MFMT_IN002101 or QUPA_IN101103, or no messages will be routed. A sample of each type of message is provided as an appendix to this book. If you are viewing the book online, you can copy the text of each appendix into a file and use that as your sample data to run the Demo.HL7v3.Production.InterfaceEngine production.

2.5 Data Transformations

Data transformations for HL7 Version 3.0 message routing productions do not use the Ensemble Data Transformation Language (DTL). Data transformations for HL7 Version 3.0 messages use XPath and XSLT to create a valid XML document.

The sample data transformation classes Demo.HL7v3.Transformation.MFMTIN002101 and Demo.HL7v3.Transformation.QUPAIN101103 provide detailed code and comments. You can find these classes in the ENSDEMO namespace. An overview of the **Transform()** method follows:

1. Get the message structure from an XData block contained in the class.
2. Get a stream of XSL from an XData block contained in the class.
3. Create an XML document from the input stream.

4. Set up the XSL transformation. For each operation in the transformation:
   - Use an XPath expression to obtain data to be transformed.
   - Transform the data, for example using ObjectScript string functions.
   - Bind this transformation to an XSL parameter.
5. Write the transformed data stream into the target message.

You can create a data transformation for an HL7 Version 3.0 message routing production by copying and customizing the sample classes Demo.HL7v3.Transformation.MFMTIN00210 or Demo.HL7v3.Transformation.QUPAIN101103, or by creating your own subclass of Ens.DataTransform, providing appropriate XData sections, and calling appropriate methods from classes in the %XML package.

To deploy a data transformation within your production, name it as the **Transform** action for a rule in a routing rule set.

2.5.1 Transforming an Attribute

The previously mentioned examples show how to transform elements in an HL7 V 3.0 message. To modify the MFMTIN00210 example so that it also transforms an attribute, add the following to the **Transform()** method:

```
// Get creation date/time, change to the current date, and bind to XSL parameter
Set tStatus=tDocument.EvaluateExpression("//MFMT_IN002101/creationTime", "@value",.tResults)
Set tParams("creationTime") = "'" _ $ZD($H,3) _ " " _ $P(tResults.GetAt(1).Value," ",2) _"'"
```

And add the following to the XData that defines the XSL transform:

```
<xsl:param name="creationTime"/>
<xsl:template match="//MFMT_IN002101/creationTime/@value">
 <xsl:attribute name="value">
  <xsl:value-of select="$creationTime"/>
 </xsl:attribute>
</xsl:template>
```

2.6 Business Operations

Each business operation in an HL7 Version 3.0 message routing production uses the routing properties of the Ensemble message to direct the new XML document to the target application as an outgoing data stream.

The sample classes Demo.HL7v3.Service.FileOut and Demo.HL7v3.Service.SOAPOut are business operations. You can find these classes in the ENSDEMO namespace. Demo.HL7v3.Service.FileOut is shown below.

Class Definition

```
Class Demo.HL7v3.Operation.FileOut Extends Ens.BusinessOperation
{

Parameter ADAPTER = "EnsLib.File.OutboundAdapter";

Parameter INVOCATION = "Queue";

Parameter SETTINGS = "Filename";

Property Filename As %String;

Method WriteToFile(pRequest As Demo.HL7v3.Message,
                   Output pResponse As Ens.Response) As %Status
{
  Do pRequest.Content.Rewind()
  Set tFilename=
  ..Adapter.CreateTimestamp(##class(%File).GetFilename(
     $Piece(pRequest.Source, $Char(13))),
     ..Filename)
```

```
   Set $ZTrap = "WriteToFileET"
  Set tStatus = ..Adapter.PutStream(tFilename, pRequest.Content)

  Quit $$$OK

WriteToFileET
  Set $ZTrap = ""

  Quit $$$ERROR($$$GeneralError,"WriteToFile("_tFilename_"):  "_$ZError)
}

XData MessageMap
{
<MapItems>
  <MapItem MessageType="Demo.HL7v3.Message">
    <Method>WriteToFile</Method>
  </MapItem>
</MapItems>
}

}
```

Demo.HL7v3.Service.FileOut uses the Source property from the Demo.HL7v3.Message to create a time stamp for the outgoing file.

You can create a business operation host class for an HL7 Version 3.0 message routing production by copying and customizing the sample classes from the Demo.HL7v3 package in the ENSDEMO namespace, or by creating your own subclass of Ens.BusinessOperation. After you create the class, add the business operation to the production.

The Management Portal provides a New Business Operation wizard with an **HL7** option. This option creates an HL7 Version 2.x business operation that is *not* suitable for use with HL7 Version 3.0.

2.7 Bad Message Handlers

A bad message handler is simply a business operation that does something useful with any HL7 Version 3.0 message that fails validation by the routing process. Often, a bad message handler saves the bad message to a file, using a timestamp and a file naming convention to make the source of the bad message easy to identify. The sample class Demo.HL7v3.MsgRouter.RoutingEngine (in the ENSDEMO namespace) has a **BadMessageHandler** property that appears in the configuration display on the **Production Configuration** page so that the user can specify the name of a configured business operation. The sample production leaves this field blank, so bad messages are simply ignored.

2.8 Alert Handlers

Alert handling for HL7 Version 3.0 message routing productions is the same as for HL7 Version 2.x. The following references provide detailed information:

- "Pager and Email Alerts" in the *Ensemble HL7 Version 2 Development Guide*
- "Configuring Alerts" in *Configuring Ensemble Productions*

3

Sample Routing Production

The sample data that you use to run the sample production Demo.HL7v3.Production.InterfaceEngine must actually contain HL7 Version 3.0 messages of type MFMT_IN002101 or QUPA_IN101103, or no messages will be routed. This appendix provides a sample of each type of message.

If you are viewing the book online, you can copy the text of each message into a file with a `*.xml` extension, and use that as your sample data to run the Demo.HL7v3.Production.InterfaceEngine production. You can find the sample production in the ENSDEMO namespace.

To use the sample production with these files:

1. Create three directories under the directory into which you installed Ensemble:
   - <*ensemblesys*>`\Dev\HL7v3\Source`
   - <*ensemblesys*>`\Dev\HL7v3\In`
   - <*ensemblesys*>`\Dev\HL7v3\Out`
2. Copy your `*.xml` files to the directory <*ensemblesys*>`\Dev\HL7v3\Source`.
3. Start the Demo.HL7v3.Production.InterfaceEngine production in the ENSDEMO namespace.
4. Copy your `*.xml` files to the directory <*ensemblesys*>`\Dev\HL7v3\In`. They should disappear when consumed by the production
5. Look for your transformed `*.xml` files in the directory <*ensemblesys*>`\Dev\HL7v3\Out`.

3.1 Sample Message MFMT_IN002101

XML

```
<MFMT_IN002101>
  <id extension="9223372036854775800" root="2.16.528.1.1007.3.2.700222.1" />
  <creationTime value="2006-01-01 12:00:00PM" />
  <versionCode code="NATCHEZEd2005-Okt" />
  <interactionId extension="MFMT_IN002101" root="2.16.840.1.113883.1.6" />
  <processingCode code="ER" />
  <processingModeCode code="T" />
  <acceptAckCode code="ER" />
<receiver>
<device>
  <id extension="000700856" root="2.16.528.1.1007.3.2" />
<name use="L">
  <given>ZIM Applicatie Regio Utrecht</given>
  </name>
<agencyFor classCode="AGNT">
```

```
<representedOrganization classCode="ORG" determinerCode="INSTANCE">
  <id extension="00100100" root="2.16.528.1.1007.3.3" />
<name use="L">
  <given>ZIM Beheersorganisatie Utrecht</given>
  </name>
  </representedOrganization>
  </agencyFor>
  </device>
  </receiver>
<sender>
  <telecom use="WP" value="tel:+31307236354" />
<device>
  <id extension="000700222" root="2.16.528.1.1007.3.2" />
<name use="L">
  <given>ABC-HIS Goodhope Hospital</given>
  </name>
<agencyFor classCode="AGNT">
<representedOrganization classCode="ORG" determinerCode="INSTANCE">
  <id extension="00600862" root="2.16.528.1.1007.3.3" />
<name use="L">
  <given>Goodhope Hospital</given>
  </name>
  </representedOrganization>
  </agencyFor>
  </device>
  </sender>
<ControlActProcess moodCode="EVN">
  <effectiveTime value="20040417" />
<authorOrPerformer typeCode="AUT">
<participant>
<AssignedPerson>
  <id extension="000120450" root="2.16.528.1.1007.3.1" />
<Organization>
  <id extension="00988137" root="2.16.528.1.1007.3.3" />
  </Organization>
  </AssignedPerson>
  </participant>
  </authorOrPerformer>
<overseer typeCode="RESP">
<AssignedPerson>
  <id extension="000120450" root="2.16.528.1.1007.3.1" />
<Organization>
  <id extension="00988137" root="2.16.528.1.1007.3.3" />
  </Organization>
  </AssignedPerson>
  </overseer>
<subject>
<registrationProcess classCode="REG" moodCode="RQO">
  <code code="722933" codeSystem="2.16.840.1.113883.2.4.15.4"
        codeSystemName="ActRegistryCodeNL" displayName="Voorschrift" />
  <statusCode code="active" codeSystem="2.16.840.1.113883.5.14" />
<effectiveTime>
  <low value="20040417" />
  </effectiveTime>
<subject2 typeCode="SUBJ">
<ActReference classCode="SBADM" moodCode="RQO">
  <id extension="9223372036854775800" root="2.16.528.1.1007.3.2.400416.16" />
  <statusCode code="active" />
<recordTarget>
<patient>
  <id extension="000197245" root="2.16.840.1.113883.2.4.6.3" />
  <statusCode code="" />
<Person>
<name use="L">
  <given>Tom</given>
  <prefix qualifier="VV">de</prefix>
  <family>Jong</family>
  </name>
  </Person>
<Organization>
  <id extension="00123456" root="2.16.528.1.1007.3.3" />
  </Organization>
  </patient>
  </recordTarget>
<authorOrPerformer typeCode="AUT">
  <time value="20040417151000" />
<assignedEntity>
  <id extension="0000120450" root="2.16.528.1.1007.3.1" />
<assignedPerson>
  <name />
<LocatedEntity>
  <Place />
  </LocatedEntity>
  </assignedPerson>
<Organization>
```

```
  <id extension="00988137" root="2.16.528.1.1007.3.3" />
  </Organization>
  </assignedEntity>
  </authorOrPerformer>
<overseer typeCode="RESP">
<assignedEntity>
  <id extension="000120450" root="2.16.528.1.1007.3.1" />
  <code code="01.015" codeSystem="2.16.840.1.113883.2.4.15.111"
        codeSystemName="RoleCode" displayName="GP" />
<assignedPerson>
  <name />
<LocatedEntity>
  <Place />
  </LocatedEntity>
  </assignedPerson>
<Organization>
  <id extension="00988137" root="2.16.528.1.1007.3.3" />
  </Organization>
  </assignedEntity>
  </overseer>
  </ActReference>
  </subject2>
  </registrationProcess>
  </subject>
  </ControlActProcess>
  </MFMT_IN002101>
```

3.2 Sample Message QUPA_IN101103

XML

```
<QUPA_IN101103>
 <id extension="1001" root="2.16.528.1.1007.3.2.2.233"/>
 <creationTime value="20040719140000"/>
 <versionCode code="NATCHEZEd2005-Okt"/>
 <interactionId extension="QUPA_IN101103" root="2.16.840.1.113883"/>
 <processingCode code="P"/>
 <processingModeCode code="T"/>
 <acceptAckCode code="ER"/>
 <receiver>
  <telecom use="WP" value="tel:+31299324874"/>
  <device>
   <id extension="000900478" root="2.16.528.1.1007.3.2"/>
   <name use="L">
    <given>ZIM Systeeem regio Utrecht</given>
   </name>
  </device>
 </receiver>
 <sender>
  <telecom use="WP" value="tel:+31307236354"/>
  <device>
   <id extension="1" root="2.16.528.1.1007.3.2"/>
   <name use="L">
    <given>ABC-HIS Goodhope Hospital</given>
   </name>
  </device>
 </sender>
 <ControlActProcess moodCode="EVN">
  <effectiveTime value="20040719135956"/>
  <authorOrPerformer typeCode="AUT">
   <participant>
    <AssignedPerson>
     <id extension="000120450" root="2.16.528.1.1007.3.1"/>
     <Organization>
      <id extension="00304845" root="2.16.528.1.1007.3.2"/>
     </Organization>
    </AssignedPerson>
   </participant>
  </authorOrPerformer>
  <reasonOf>
   <justifiedDetectedIssue>
    <code code="QNAT" codeSystem="2.16.840.1.113883.2.4.5.4"
          codeSystemName="ActCodeNL"/>
    <targetOf>
     <source moodCode="DEF">
      <code code="EMAUTH" codeSystem="2.16.840.1.113883.2.4.5.4"
            codeSystemName="ActCodeNL"/>
     </source>
```

```
    </targetOf>
   </justifiedDetectedIssue>
  </reasonOf>
  <queryByParameter>
   <queryId extension="5523264" root="2.16.528.1.1007.3.2.400893.15"/>
   <statusCode code="new"/>
   <person.addr>
    <value>
     <postalCode>1200 BR</postalCode>
    </value>
    <semanticsText>Person.addr</semanticsText>
   </person.addr>
   <person.birthTime>
    <value>
     <center value="19750103"/>
    </value>
    <semanticsText>Person.birthTime</semanticsText>
   </person.birthTime>
  </queryByParameter>
 </ControlActProcess>
</QUPA_IN101103>
```