# Oracle SOA Suite 11g R1 PS5

# **SOA Suite for healthcare integration Series**

# **Exception Handling - Processing Endpoint Errors**

michael@czapski.id.au January 2013

# **Table of Contents**

ntroduction	.1
Solution Overview	.2
Modify Inbound Endpoint to use Functional ACKs and Validate	.3
Exception Handler Application	.5
Start the "Receiving" "External Systems"	15
Send valid ADT messages	16
Send invalid ADT messages	17
Summary	20

# Introduction

In this article we will add exception handling to the routing and transformation solution developed in the article entitled "SOA Suite for healthcare integration Series – Domain Value Map (DVM) – On-the-fly Code Mapping", to explore how message processing exceptions can be redirected to a logic component for handling. The exception handling components will handle exceptions form all SOA Suite for healthcare integration applications.

This article assumes that the reader has the SOA Suite for healthcare integration environment with all necessary components installed and ready to use. The Bill of Materials for such an environment and a discussion on where the components can be obtained is provided in the earlier article, "SOA Suite for healthcare integration Series - Overview of the Development Environment", to be found at

http://blogs.czapski.id.au/2012/08/soa-suite-for-healthcare-integration-series-overviewof-the-development-environment.

This article assumes that the reader completed the solution discussed in the earlier article, "SOA Suite for healthcare integration Series – Domain Value Map (DVM) – On-the-fly Code Mapping", to be found at

http://blogs.czapski.id.au/2013/01/soa-suite-for-healthcare-integration-series-domain-value-map-dvm-on-the-fly-code-mapping.

# **Solution Overview**

In healthcare messaging environments it is expected that exceptions in message processing will be handled in a reasonable way, not by simply dropping messages without some form of notification being provided to the sender or to the operations staff.

The SOA Suite for healthcare integration infrastructure will queue invalid incoming messages, or outgoing messages which cannot be delivered, to the JMS B2B\_IN\_QUEUE, if JMS is configured as the default transport (which it is for this article series). Messages in error are queued with a JMS User Defined Property "MSG\_TYPE" set to the value of "3". An inbound message can cause an exception in a number of circumstances, some of which are:

- message cannot be decoded (for example not a HL7 v2 delimited)
- message structure for the message cannot be identified (for example ORU coming in when no ORU document is defined)
- Endpoint cannot be identified
- ...

An outbound message can cause an exception in a number of circumstances, some of which are:

- Endpoint cannot be identified so the message cannot be sent
- External target systems is down and no retry is configured
- ...

The solution components are depicted in Figure 1 Solution Components.



#### Figure 1 Solution Components

The diagram uses the convention which clearly separates the external systems, the SOA Suite for healthcare integration-specific components and generic SOA Suite components using the "swim-line" analogy.

The solid lines from the "ADT Receiver", "ADT A08 Sender" and "ADT Broadcast Sender" to the "B2B\_IN\_QUEUE" JMS Queue depict implicit integration between the endpoints and the JMS infrastructure used to both pass messages between the endpoints and the SOA Composites and to queue exception messages to be processed as might be desired. This latter functionality is what this article will explore.

Exception handling functionality revolves around the SOA Suite for healthcare integration infrastructure passing to a SOA Composite a message and a property whose value allows the SOA Composite to de-queue the message from the queue and process it as might be required.

This topic is discussed in some detail in the Oracle "Technical Note #007, Exception Handling", available at

<u>http://www.oracle.com/technetwork/testcontent/b2b-tn-007-exception-handling-133087</u> <u>.pdf</u>. While it applies to the Oracle B2B infrastructure it is equally applicable to the "SOA Suite for healthcare integration". Importantly, the tech note provides the definition of the XML structure which is used to represent exception messages in the SOA Suite for healthcare integration.

To trigger exception processing we will submit an invalid HL7 v2 message to the existing solution, will receive a HL7 v2 message, validate it, acknowledge it with functional ACK and pass it onto the SOA Composite if it is valid. If the message is not valid an exception message will be delivered to the exception handler, which we will develop in this article, and will be subsequently written to a file in the file system. Clearly, a more imaginative / useful / elaborate handling of exception messages is possible. Developing such a sophisticated solution is a SOA Suite project and has nothing to do with HL7 processing as such.

We need to create a SOA Composite which will use JMS selectors to read messages from the pre-configured JMS Queue and will write them to a file in the file system.

### Modify Inbound Endpoint to use Functional ACKs and Validate

As configured, the HosA\_SystemA\_In endpoint, which receives ADT messages, will send a canned ACK as soon as it receives a message. It is also configured to not validate messages it receives so the endpoint will not trigger exception processing if the message is invalid. Even with this there will be circumstances when exceptions will occur and trigger exception handling. Some of these will be inability to deliver messages to the external systems by the outbound endpoint, for example. To make it easy for us to induce exception handling we will re-configure the inbound endpoint, HosA\_SystemA\_In, to use Functional Acknowledgements and to Validate messages. In this section we will go through the steps necessary to accomplish this.

- □ Start the "Healthcare Integration Console", <u>http://localhost:7001/healthcare</u>, and log in with your credentials (perhaps weblogic/welcome1)
- □ Navigate through "Designer"  $\rightarrow$  "Configuration"  $\rightarrow$  "Endpoint" and double-click the "HosA\_SystemA\_In" endpoint name to open its configuration properties

ORACLE: SOA Suite for healthcare integration				
Designer Dashboards Reports				
Configuration Administration	HosA_SystemA_In			
View ▼	Endpoint: HosA_SystemA_In (Enabled)			
Calout Mapset	Endpoint Properties         ✓ Hide Endpoint Configuration Parameters         Status       Enabled         Transport Protocol       MLLP 10       Transport Details			

□ In the "Document to Receive" check the "Functional ACK" and "Validate" checkboxes and choose "YES" from the "FA handled automatically" drop down

Document	Functional ACK	Validation	Translation	FA handled automatically	Internal Channel	
HL7-2.3.1-ADT-CMM_v1.0		<b>v</b>		YES		ŀ

□ Click the "Transport Details" button

Endpoint Properties ∀ Hide Endpoint Config	uration Para	ameters	Connection Made	Conver
Transport Protocol	MILD10	Transport Details	Host Name	Server
Transport Callout		Hanspore Details	Port	10Calh0st 22222
Transport Calibut				

□ Click the "Advanced" Tab, change "Immediate ACK" to "none" and click "OK"

ransport Protocol Parameters	
Connection Block Characters Advanced	
Immediate ACK None	
Custom Immediate ACK File Browse	
Map ACK Control ID	
Map Trigger Event	
Discard HL7 ACK None	
Sequencing Mode OneToOneMapping	
Polling Interval 10	
Timeout 300	
Sequencing	

□ Click "Apply" to complete the procedure

The endpoint is re-configured as required and changes are applied to the runtime. Now we will develop the composite application which will pick up exception messages form the JMS Queue and write them to a file in the file system.

# **Exception Handler Application**

In this section we develop and deploy the exception handler application, "ExceptionHandlerApp", to receive selected JMS messages and write them to files in the file system.

- □ Start JDeveloper Studio, if it is not already running
- □ Pull down the "Application" menu and choose "New..."
- □ Name the new application "ExceptionHandlerApp", choose "SOA Applicaiotn" and click "Next>"

lightary SOA Application	- Step 1 of 3
Name your applicatio	
Application Name	Application Name: ExceptionHandlerApp
	Directory:
<ul> <li>Project SOA Settings</li> </ul>	C:\JDeveloper\mywork\ExceptionHandlerApp Browse
	Application Package Prefix:
	pkgWSCvtDate
	Application Template:
	Davabeans technologies.
	Java Desktop Application (ADF) Creates a databound rich client application. The application consists of one project for the client (ADF Swing), and another project for the ADF Model (ADF Business Components).
	Java EE Web Application     Creates a databound web application. The application consists of one project for     the view and controller components (J3F), and another project for the data model     (E3B session beams and JPA entities).
	<b>50A Application</b> Creates a SOA (service-oriented architecture) application. The application consists of one SOA project for the SOA corposite, components, and adapters.
	< Back Next > Einish Cancel

 $\hfill\square$  Name the new project "ExceptionHandlerProj" and click "Finish"

SCREATE SOA Application	- Step 2 of 3	×
Name your project	01	25
Application Name	Project Name: ExceptionHandlerProj	rowse
Reference Project Name		_
Project SOA Settings	Project Technologies Generated Components Associated Libraries	
	Available: Selected:	
	ADF Business Components 🛛 🗠 SOA	
	ADF Desktop Integration ADF Faces ADF Library Web Application Support ADF Mobile Browser ADF Page Flow ADF Swing Ant BPM Database (Offline) EJB	
	Technology Description:	
	SOA is the Service Oriented Architecture to build composite applications.	
Help	< Back Next > Einish	Cancel

As mentioned in the introduction, the Oracle "Technical Note #007, Exception Handling", available at

<u>http://www.oracle.com/technetwork/testcontent/b2b-tn-007-exception-handling-133087</u> .pdf, provides the definition of the XML structure which is used to represent exception messages in the SOA Suite for healthcare integration.

We need to create an XML Schema Definition (XSD) document in our project to use as the definition of the exception message.

□ Right-click the "ExceptionHandlerProj" project name and choose "New..."



□ Click the "All Technologies" Tab, expand the "General" Tab, click "XML" to show the list of XML-related technologies, choose "XML Schema" and click "OK"



□ Name the new XML Schema "ipException.xsd" then click "OK"

Create XML Schema
Enter the details of your new file.
File Name:
ipException.xsd
Directory:
C:\JDeveloper\mywork\ExceptionHandlerApp\ExceptionHandlerProj Browse
Target Namespace:
http://www.example.org
Prefix:
Help OK Cancel

□ Click the "Source" Tab to switch to the source code mode



 Open a web browser, navigate to the Technical Note #007, <u>http://www.oracle.com/technetwork/testcontent/b2b-tn-007-exception-handling-133</u> 087.pdf, scroll down to the bottom of the document and copy the XML Schema definition – reproduced below for convenience

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"</pre>
xmlns="http://integration.oracle.com/B2B/Exception"
targetNamespace="http://integration.oracle.com/B2B/Exception">
  <xs:element name="Exception">
    <!--xs:complexType name="Exception"-->
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="correlationId"/>
        <xs:element ref="b2bMessageId"/>
        <xs:element ref="errorCode"/>
        <xs:element ref="errorText"/>
        <xs:element ref="errorDescription"/>
        <xs:element ref="errorSeverity"/>
        <xs:element ref="errorDetails" minOccurs="0" />
     </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="correlationId" type="xs:string"/>
  <xs:element name="b2bMessageId" type="xs:string"/>
  <xs:element name="errorCode" type="xs:string"/>
 <xs:element name="errorText" type="xs:string"/>
 <xs:element name="errorDescription" type="xs:string"/>
 <xs:element name="errorSeverity" type="xs:string"/>
 <xs:element name="errorDetails">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="parameter" maxOccurs="unbounded"/>
      </xs:sequence>
```

```
</rs:complexType>
</xs:element>
<xs:element name="parameter">
<xs:complexType>
<xs:attribute name="name" type="xs:string" use="required" />
<xs:attribute name="value" type="xs:string" use="required" />
</xs:complexType>
</xs:complexType>
</xs:element>
</xs:schema>
```

Paste the XSD text into the new XSD definition in JDeveloper, replacing the default content

Exc	ept:	ionHandlerApp Overview 🗴 📲 composite.xml 🗴 🛃 ipException.xsd 🗴
( 💏 -	Find	
1		<pre><?xml version="1.0" encoding="HTF-8" 2></pre>
2		<pre>(xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://integration.oracle.com/B2B</pre>
3		
4		<pre>xs:element name="Exception"&gt;</pre>
5		xs:complexType name="Exception"
6		<xs:complextype></xs:complextype>
7	Ξ	<xs:sequence></xs:sequence>
8		<xs:element ref="correlationId"></xs:element>
9		<xs:element ref="b2bMessageId"></xs:element>
10		<pre><xs:element ref="errorCode"></xs:element></pre>
11		<pre><xs:element ref="errorText"></xs:element></pre>
12		<xs:element ref="errorDescription"></xs:element>
13		<pre><xs:element ref="errorSeverity"></xs:element></pre>
14		<pre><xs:element minoccurs="0" ref="errorDetails"></xs:element></pre>
15		
16		
17		
18		<xs:element name="correlationId" type="xs:string"></xs:element>
19		<xs:element name="b2bMessageId" type="xs:string"></xs:element>
20		<pre><xs:element name="errorCode" type="xs:string"></xs:element></pre>
21		<pre><xs:element name="errorText" type="xs:string"></xs:element></pre>
22		<pre><xs:element name="errorDescription" type="xs:string"></xs:element></pre>
23		<pre><xs:element name="errorSeverity" type="xs:string"></xs:element></pre>
24		<xs:element name="errorDetails"></xs:element>
25		<xs:complextype></xs:complextype>
26		<xs:sequence></xs:sequence>
27		<pre><xs:element maxuccurs="unbounded" ref="parameter"></xs:element></pre>
20		
20		
31		<pre>//x.clement vere="neveneter"\ /veilgment vere="neveneter"\</pre>
32		<pre></pre>
33		<pre></pre> (se:attribute name="name" type="yes:string" use="required" />
34		<pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre>
35		
36		
37		
Design	S	ource History

□ In the project node tree right-click the "ipException.xsd" node and choose "Validate XML" to make sure that the schema XML is valid. Correct any errors you may find



- □ Click the "Save All" button to save the changes to the project so far and close the XSD editor to return to the composite editor
- □ From the "Component Palette" → "SOA" → "Service Adapters" drag the "JMS Adapter" to the "Exposed Services" swim line to start the JMS configurator wizard



□ Name the service "JMSException\_In" and click "Next>"

SIMS Adapter Configuration Wizard - Step 2 of 4	×
Service Name	*
Enter a Service Name.	
Service Type: JMS Adapter	
Service Name: JMSException_In	

□ Choose "Oracle Weblogic JMS" as the OEMS and click "Next>"



□ Choose your "AppServer Connection" and click "Next>"

SIMS Adapter Configuration	Wizard - Step 4 o	f 6			×
Service Connection					*
An Application Server Connection is required to configure this adapter. Select an application server connection already defined in your project or create a new connection.					
AppServer Connection:	con_single_serve	r_domain		- + /	X
	con_single_serve	r_domain	N		
User Name: weblogic	IntegratedWebLo	gicServer	13		
Host Name: localhost					

□ Accept default for "Adapter Interface" and click "Next>"

MS Adapter Configuration Wizard - Step 5 of 6					
Adapter Interface					
The adapter interface is defined by a wsdl that is generated using the operation name and schema(s) specified later in this wizard. Optionally, the adapter interface may be defined by importing an existing WSDL.					
Import an existing WSDL					
WSDL URL:	10 B				
Port Type:	•				

□ Choose "Consume Message" operaton and click "Next>"



□ Click the "Browse..." button alongside the "Destination Name"

े JMS Adapter Configu	ration Wizard - Step 7 o	f9	×
Consume Operatio	n Parameters	0101010101010101010000	*
Enter the parameters for t	he Consume Message opera	ation.	
Destination Name:			Browse
JMS <u>C</u> onnection Factory:	Not Specified		μÇ
Message Body <u>T</u> ype:	TextMessage		•]
Message Selector:			]
	example 1: "country in ('US'	', 'UK')", example 2: "origin = 'FR'"	

□ Scroll down and choose "B2B\_IN\_QUEUE" and click "OK"

Select Destination		×		
Select a destination an button.	d click OK. You may select different search criteria and click the Search			
Search				
Des <u>t</u> ination Type:	All Types 👻			
Destination Name:				
	Show <u>All</u>			
Destinations				
Measurer	mentTopic (topic)			
PeopleQu	ueryTopic (topic)			
🖻 🧁 SOAJMSModu				
TestFwk	Queue (queue)			
	onsenderQueue (queue)			
	tQueue (queue)			
B2BBroad	dcastTopic (topic)			

□ Change the "Message Body Type" to "BytesMessage", enter "MSG\_TYPE='3" into the Message Selector box and click "Next>"

🥧 JMS Adapter Configuration Wizard - Step 7 of 9			×		
Consume Operation	Parameters	01	*		
Enter the parameters for the	Enter the parameters for the Consume Message operation.				
Destination Name (Queue):	jms/b2b/B2B_IN_QUEUE		Browse		
JMS <u>C</u> onnection Factory:	weblogic.jms.ConnectionFactory				
Message Body <u>T</u> ype:	BytesMessage	-			
Message Selector:	MSG_TYPE='3'				
example 1: "country in ('US', 'UK')", example 2: "origin = 'FR'"					
Specify the JNDI name for the JMS Connection. The deployment descriptor for the deployed instance of the JMS Adapter must associate this JNDI name with a set of configuration properties needed by the JMS Adapter to access the JMS destination at runtime.					
the JMS destination at runtin	ne,	5			

□ Click the "Browse for schema file" button

r Message Schema	<b>b</b>
Native format translation is not required (Schema is Opaque)	– Se
URL 🗛 🔅	÷
Schema Element	ile [
	10 

 $\hfill\square$  Expand the node tree, choose "Exception" and click "OK"

	<u>×</u>
	칾 🙋
🔍 Type Explorer	
🖻 🗁 Project Schema Files	
🖮 🤮 ipException.xsd	
Exception	
<b>b2bMessageId</b> : string	
correlationId : string	
errorCode : string	
errorDescription : string	
www.errorDetails	
errorSeverity : string	
errorText : string	
🚯 parameter	
Project WSDL Files	

- □ Click "Next" and "Finish" to complete JMS Adapter confguratoin
- □ From the "Component Palette" → "SOA" → "Service Adapters" drag the "File Adapter" to the "Exposed Services" swim line to start the JMS configurator wizard
- □ Enter "FileException\_Out" as the service name and click "Next>"

FILE Adapter Configuration Wizard - Step 2 of 4	×
Service Name	*
Enter a Service Name.	
Service Type: File Adapter	
Service Name: FileException_Out	

□ Accept the default for the "Adapter Interface" and click "Next>"

📥 FILE Adapter Configuration Wizard - Step 3 of	4 X			
Adapter Interface	0101010101010101010104040000			
The adapter interface is defined by a wsdl that is generated using the operation name and schema(s) specified later in this wizard. Optionally, the adapter interface may be defined by importing an existing WSDL.				
Interface:      Define from operation and schema (specified later)				
◯ Import an existing WSDL				
WSDL URL:	1			
Port Type:	▼			
Operation:	▼			

 $\hfill\square$  Choose the "Write File" operation and click "Next>"

FILE Adapter Configuration Wizard - Step 4 of 7		
Operation	010101010101010101010100100	
The File Adapter supports four operations. There is a system, a Write File operation that creates outgoing fil contents of a file, and a List Files operation that lists fil Operation Name. Only one operation per Adapter Ser	Read File operation that polls for incoming files in your local file es, a Synchronous Read File operation that reads the current le names in specified locations. Specify the Operation type and vice may be defined using this wizard.	
Operation Type: <u>R</u> ead File <u>Write File</u> <u>Synchronous Read File</u> List Files		
Operation Name: Write		
Add Output Header		

□ Enter the file system path to a directory where the exception files will be written, enter the file name, "ipException\_\$SQ%.xml" and click "Next>" – files will have a name followed by an automatically-generated sequence number and the extension of ".xml"

💩 FILE Adapter Configuration \	Wizard - Step 5 of	7			×
File Configuration					*
Specify the parameters for the Wri	te File operation.				
Directory specified as Directory for Outgoing Files (physic c:\hl7\received	ysical Path O Log al path):	ical Name			Browse
Eile Naming Convention (po_%5EQ	%.txt): [ipExce	otion_%SEQ%.>			
Write to output file when any of t	hese conditions are	met			
✓ Number of Messages Equals:	1		₽		
Elapsed Time Exceeds:	1		minutes	-	•]
File Size Exceeds:	1000		kilobytes	+	

□ Click the "Browse for schema file" button to locate the xml schema

FILE Adapter Configuration Wizard - Step 6 or	f7 <u>×</u>		
Messages	01010101010101048496565		<b>—</b>
Define the message for the Write File operation	-ify the Scheme File Location and select the Scheme Element that		E C
defines the messages in the outgoing files. Use the Br	owse button to find an existing schema definition. If you check	erProj	SOA
'Schema is Opaque', then you do not need to specify a	a Schema.		<b>60</b>
-Message Schema-		P	- Ser
			🐣 А
Native format translation is not required (Schema	is Opaque)		<u>اللہ</u> A
	🔍 🌼 📗		🔯 В
Schema Element	V		👘 В
	browse fo	r schema f	ile D
			070

□ Expand the "ipException.xsd", choose the "Exception" element and click "OK"



- □ Click "OK" then "Next>" and "Finish" to complete the wizard
- □ From the "Service Components" list drag the "Mediator" component to the "Components" swim line



 $\hfill\square$  Name the new component "PassThroughMediator" and click "OK"

🚖 Create Mediator	×
Mediator Component Create a mediator component to perform routing, filtering, and transformations.	4
Name: PassThroughMediator <u>I</u> emplate: Define Interface Later	• @

#### Connect adapters to the component

ExceptionHandlerApp Overview	× Gacomposite.xml ×		V
🖌 🤌 🦛 🖶 🗶 🕕	🗟 諮 🗟 📽 🤣	Co	mposite: ExceptionHandlerProj
Exposed Services	P	Components	External References

□ Click the "Save All" button, right-click on the name of the project and deploy the project

Note that if you had exceptions in any of the work done so far, all accumulated exception messages will be written to files in the file system. Inspect some of them to see what they are pertaining to then delete them before proceeding.

# Start the "Receiving" "External Systems"

We will use the CMDHL7Listener command line client to receive HL7 ADT messages look at the output in the output directory specified on the listener's command line – for me c:\hl7\received. The CMDHL7Listener will display trace of message exchange in the console

window. The SOA Suite for healthcare integration will record message tracking information which we will look at a later stage.

Please note that in this solution the CMDHL7Listener returns an ACK as soon as it gets the message.

- $\hfill\square$  Check that your configured output directory is empty
- □ To start the listener for the "BcastB\_SystemA\_Out" sender endpoint, in a command / terminal window execute the following command

java -jar c:\tools\CMDHL7\CMDHL7Listener\_v0.7.jar -c ID\_ -p 22200 -s c:\hl7\received

□ Inspect the CMDHL7Listener console output making sure the listener started and is listening on the appropriate port

```
08/12/2012 10:45:36 AM au.id.czapski.hl7.CMDHL7Listener main
INFO: Port: 22200
08/12/2012 10:45:36 AM au.id.czapski.hl7.CMDHL7Listener main
INFO: Store in: c:\hl7\received
08/12/2012 10:45:36 AM ca.uhn.log.HapiLogImpl info
INFO: au.id.czapski.hl7.SimpleACKApplication registered to handle *^*
messages
08/12/2012 10:45:36 AM ca.uhn.log.HapiLogImpl info
INFO: SimpleServer running on port 22200
```

□ To start the listener for the "HA08\_SystemA\_Out" sender endpoint, in a command / terminal window execute the following command

```
java -jar c:\tools\CMDHL7\CMDHL7Listener_v0.7.jar -c ID_ -p 22100 -s
c:\hl7\received
```

### Send valid ADT messages

As before, we will use the CMDHL7Sender command line client to read files containing a single HL7 ADT message and submit them to the ADT Receiver endpoint. We will then look at the console output produced by the CMDHL7Listener which we started earlier, then look at the output in our configured output directory – for me c:\hl7\received, and finally review message tracking information in the Healthcare Integration Console.

Please note that in this solution the receiver endpoint returns immediate ACK as soon as it gets the message. There may be a delay, most noticeable the first time one executes the processing flow after application server restart, between the receipt of the ACK and the time the message is written to a file in the file system.

- □ Check that your configured output directory is empty
- □ Locate the input file containing a single HL7 message for me this will be C:\hl7\adt\sources\ADT\_A01\_output\_1.hl7

The content of my file, where each segment starting with the 3 character segment ID in bold text is a single line up to the next 3 character segment ID, looks like this:

□ In a command / terminal window execute the following command

java -jar c:\tools\CMDHL7\CMDHL7Sender\_v0.7.jar -a SystemA -b HosA -c ID\_ -n 1 -d \r\r\n -p 22222 -h localhost -t 30000 -f c:\hl7\adt\sources\ADT A01 output 1.hl7

□ Locate the output files in the received directory and inspect them to confirm that a) A01 or A03 and A08 have been written and b) that A01/A03 has the same content as the input file and the A08 has the same content as the input file except for the tree changes we made in the "TransformerMediator" component. Take particular note of the Gender value – which will be "Male"

The content of the A08 output file looks like this:

The content of the file named

"38313332333630313831353734383532\_ADT\_A08\_1356512791863.hl7" is the same as the message which was sent except for the fields and components which we explicitly modified.

□ Submit the ADT A03 file, ADT\_A03\_output\_1.hl7, and inspect the output.

Our solution works to the extent of receiving HL7 v2.3.1 messages, and acknowledging them, transforming ADT messages to A08 for one stream of processing and writing them to files in the file system.

### Send invalid ADT messages

As before, we will use the CMDHL7Sender command line client to read a file containing a single HL7 ADT message, which is invalid, and submit it to the ADT Receiver endpoint. We will look at the exception file in our configured output directory – for me c:\hl7\received, and then review message tracking information in the Healthcare Integration Console.

- □ Copy the ADT\_A01\_output-1.hl7 file and give a copy a new name ADT\_A01\_broken.hl7
- □ Using a text editor which \_does not\_ change line endings, for example "Notepad++", edit the message and change the name of the EVN segment to read EVX instead. An example is shown below.

In a command / terminal window execute the following command

```
java -jar c:\tools\CMDHL7\CMDHL7Sender_v0.7.jar -a SystemA -b HosA -c ID_
-n 1 -d \r\r\n -p 22222 -h localhost -t 30000 -f
c:\hl7\adt\sources\ADT A01 broken 1.hl7
```

□ Inspect the Negative ACK which the command line tool displays in the console window

```
MSH|^~\&|PI|MDM|SystemA|HosA|20130103||ACK^A01|1001|P|2.3.1
MSA|AE|ID_0000000
ERR|^^^100&0x810070:Rejecting:Unrecognized data was found in the data
file. The last~^^100&0x810070:Rejecting:Unrecognized data was found in
the data file. The last~EVN^3^100&0x81004d:Rejecting:Segment EVN (EVN
- event type segment) at guidelin~PID^3^100&0x81004d:Rejecting:Segment
PID (PID - patient identification segment)
```

□ Inspect the content of the exception file, ipException\_*nn*.xml – for me it looks like the text below

```
<?xml version="1.0" encoding="UTF-8" ?><Exception
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://integration.oracle.com/B2B/Exception">
  <correlationId>null</correlationId>
  <b2bMessageId>COA8E9E913BFE096F4E000001BB82D63-1</b2bMessageId>
  <errorCode>B2B-51507</errorCode>
  <errorText>
  Validator error - Extra data was encountered.
Validator error - Extra data was encountered.
A data segment with 'Mandatory' status is missing.
A data segment with 'Mandatory' status is missing.
  </errorText>
  <errorDescription>
 Machine Info: (R1PS5HCI)
Unrecognized data was found in the data file. The last known Segment was MSH at
guideline position 001. {br} {br} This error was detected at: {br} {tab} Segment
Count: 2{br}{tab}Characters: 80 through 125
Unrecognized data was found in the data file. The last known Segment was MSH at
guideline position 001. {br} {br} This error was detected at: {br} {tab} Segment
Count: 3{br}{tab}Characters: 126 through 275
Segment EVN (EVN - event type segment) at guideline position 002 is missing. This
Segment's standard option is 'Required'. {br}{br}This Segment was expected
after:{br}{tab}Segment Count: 3{br}{tab}Character: 275
Segment PID (PID - patient identification segment) at guideline position 003 is
missing. This Segment's standard option is 'Required'. {br}{br}This Segment was
expected after:{br}{tab}Segment Count: 3{br}{tab}Character: 275
  </errorDescription>
  <errorSeverity>2</errorSeverity>
  <errorDetails>
          <parameter name="hc.messageId"</pre>
value="C0A8E9E913BFE096F4E000001BB82D63-1"/>
          <parameter name="hc.documentTypeName" value="ADT"/>
          <parameter name="hc.documentProtocolVersion" value="2.3.1"/>
          <parameter name="hc.documentDefinitionName" value="CMM v1.0"/>
          <parameter name="hc.documentProtocolName" value="HL7"/>
```

□ In the "Healthcare Integration Console" switch to "Reports" Tab and select the entry with the state of "MSG\_ERROR"



□ Click at the "Error" "button" between the "Wite Message" and the "Business Message" and inspect the exception



Our exercise if concluded. The solution will now process valid messages and will dump exceptions for invalid messages, or any other endpoint and HL7 processing in the "SOA Suite for healthcare integration" exceptions to files in the file systems. We could have gone to the trouble of parsing these messages, constructing custom notification messages, sending them via a Short Message Service or email, and so on. These are left as an exercise

and are examples of the kinds of things one can do in the Oracle SOA Suite, whether using the healthcare integration components or not.

#### **Summary**

In this article we added exception handling to the routing and transformation solution developed in the article entitled "SOA Suite for healthcare integration Series – Domain Value Map (DVM) – On-the-fly Code Mapping", to explore how message processing exceptions can be redirected to a logic component for handling. The exception handling components will handle exceptions form all SOA Suite for healthcare integration applications.