Lab 2 Message Models and Working with XML Messages

2.1 Overview

In this lab, the IntroMessageFlow will be modified to identify the parser (XMLNSC) to be used to process the message.

The steps are very simple.

The properties of the Input node will be modified.

The Test Client will be used to run another test.

The trace file contents will be viewed to see the difference.

2.2 Using the XML Parser

- __1. Return to the IBM Integration Toolkit.
- ___2. Click on the IntroMessageFlow tab to bring the message flow into view.

🖽 IntroMessageFlow.msgflow 🛛 📄 IntroLab

___3. Click on the **XML_Input** node to bring its properties into view.



The message flow will be modified so that it uses the XMLNSC parser to process the input message.

- __4. On the **Properties** view at the bottom of the screen, click the **Input Message Parsing** tab. Since nothing was specified when the node was added, the Message domain (i.e. the parser) defaults to **BLOB** – which you saw in the trace.
- __5. Click the pull-down for the **Message domain**. The various parsers are listed along with a short description. Depending on the Message domain selection, the other fields may be enabled or disabled.
- ___6. Select the XMLNSC parser. The XMLNSC parser that supports Namespaces (the NS part) and builds a more efficient or compact tree (the C part). The compact tree uses less memory.

Properties 🛛 🔝 🛛	Problems 🔠 Deploy	yment Log	2 - 6
😰 MQ Input Node	Properties - XM	L_Input	
Basic			
Input Message Parsing	Message domain	XMLNSC : For XML messages (namespace aware, validation, low memory use)	-
Parser Options	Message model	DEDL : For binany or text messages with a Data Format Description Language schema model	
Advanced	messagemoder	XMLNSC : For XML messages (namespace aware, validation, low memory use)	
Validation	Message	JSON : For JavaScript Object Notation messages	
Security	Physical format	BLOB : For messages with an unspecified format MIME : For MIME wrapped data including multipart	
Instances		MRM : For binary or text messages that are modeled in a message set	
Monitoring		JMSMap : For JMS MapMessage messages (XML)	
		XMLNS : For XML messages (namespace aware)	

7.

Save the message flow (Ctrl+S).

Key Idea: Parsers and Message Domains

IBM Integration Bus supplies a range of parsers to parse and write messages in different formats.

A parser is called when the bit stream that represents an input message must be converted to the format that is used internally by the broker; this process is known as parsing. The input is a bit stream, and the output is a logical tree representation of the message.

A serializer is called when a logical tree structure must be converted into a bit stream (for example on an Output node). This process is known as serializing.

Each parser is suited to a particular class of messages, known as a message domain. The following list contains some examples of the message domains used in IBM Integration Bus:

- XMLNSC for XML documents
- DFDL for general text or binary data streams including industry standards
- JSON for JSON documents
- DataObject for data without a stream representation (used by adapters)

Now, let's re-run the Test Client.

- ___8. Switch back to the Test Client (IntroLab tab).
- ___9. Select one of the Invoke Message Flow items.
- ___10. Press the right mouse button.
- ___11. Select **Re-run** from the menu.

🖽 IntroMessageFlow.msgflow 🛛 📔 IntroLab 🛛		- 8
Events		
Message Flow Test Events	General Properties	
🕨 🗉 🏇 🚮 🔛 🖽 📮 🎢	 Detailed Properties 	
E Invoke Message Flow	Message flow: //IntroLab/IntroMessageFlow.msgflow	Ŧ
Re-run Invoke	node: XML_Input sage	Ŧ
Duplicate Rename	y: View as source	v
Remove	Show in hexadecimal viewer (Read Only)	<u> </u>
Remove All	<pre>?xml version="1.0" encoding="utf-8"?></pre>	-
Show flow view	rt Source	age

You again see the same output message from the Test Client.

1essage Flow Test Events	General Prop	erties		_
>	✓ Detailed Prop	perties		
Invoke Message Flow	Host:	localhost		
🗄 🔡 Invoke Message Flow	Port:	0		
🖂 🔡 Invoke Message Flow	Queue manager:	MBSQMGR		
Message flows deployment successfully completed	Oueue:	LAB.SEND.AS.	XML	
🖃 🐕 Starting	m Message	1		
Sending Message to MQ Queue "LAB.IN"	h Header			
MQ Queue Monitor "LAB.SEND.AS.XML"	r header			
En, scopped listening for response	Body: View as	XML structure		
Stopped	Name		Value	
	🖃 tns:JKE_I	n_Request		
	xmins	:xsi	http://www.w3.org/2001/XMLSchema-ins	
	xmins	:xsd	http://www.w3.org/2001/XMLSchema	
	xmins	tns:	http://www.ibm.lab.com	
	Action	nRequest	0	
	DateR	tequest	10/12/2005	
	custo	merNumber	1	
	custo	merName	ACME Hardware	
	E custo	merDetails		
	a	scomerAddress	1254 Main St	
	0	scomerAddress	Dute 12	
		istomerCity	TY	
		istomerCountry	1154	
		istomerPostalCr	76543	
	a	ustomerCreditLin	1200	
		ustomerCreditSc	123	
	🖃 conta	ctDetails		
	0	IntactFirstName	Freddy	
()				

This is only what the output message looks like *after* it arrives on the output queue. Let's see what the message looked like *while* it was passing through the message flow.

- ___12. Return to Windows Explorer.
- ___13. Navigate to the file at C:\XML_Input_Trace.txt.
- ___14. Double click on XML_Input_Trace.txt file.

rganize 👻 🥥 Open	 Print New folder 				• =	1
Favorites	Name	Date modified	Туре	Size		
🔜 Desktop	BGInfo	6/30/2010 12:23 PM	File folder			
Downloads	DB2	7/31/2012 1:06 PM	File folder			
🔚 Recent Places	SRDBM01	7/31/2012 3;08 PM	File folder			
	📕 Firefox	10/3/2012 6:06 PM	File folder			
Libraries	\mu IBM	7/8/2013 4:22 PM	File folder			
Documents	🍶 idsinstinfo	7/31/2012 3:01 PM	File folder			
J Music	📥 idsslapd-dsrdbm01	7/31/2012 3:07 PM	File folder			
E Pictures	퉬 PerfLogs	7/13/2009 11:20 PM	File folder			
🛃 Videos	🌽 Program Files	5/16/2013 12:59 PM	File folder			
	퉬 Program Files (x86)	7/18/2013 4:01 AM	File folder			
Computer	퉬 student	7/30/2013 6:37 PM	File folder			
🚨 Local Disk (C:)	퉬 temp	5/16/2013 1:03 PM	File folder			
🚽 Shared Folders (\\vn	🕕 Users	7/31/2012 12:57 PM	File folder			
	퉬 Windows	7/9/2013 1:07 PM	File folder			
Network	WMQv75-POT	4/28/2013 2:09 AM	File folder			
	VS_EXPBSLN_x64_enu	3/19/2010 7:55 PM	Cabinet File	2,026 KB		
	VS_EXPBSLN_x64_enu	3/19/2010 7:58 PM	Windows Installer	539 KB		
	XML_Input_Trace	7/31/2013 1:28 PM	Text Document	7 KB		

___15. Scroll to the end of the file (Ctrl + End).

Trace output is placed at the end of any existing content in a file so scroll down to the bottom of the file and view the results. Much more pleasing...here is a nicely formatted message tree that will allow you to conveniently access the fields in the XML message by name. Notice:

- The XMLNSC Domain name (which is what we set on the input node).
- All of the element types are String represented by the (CHARACTER)! Why is that? The answer requires understanding both Parsers and Message Models.



__16. Close the Notepad window.

___17. Minimize Windows Explorer.

2.3 Creating a Message Model from an XSD

In this portion of the lab, we will use a message model to parse the XML message.

Key Idea: Message Models

Much of the business world relies on the exchange of information between applications. This information is contained in messages that have a defined structure that is known and agreed by the sender and the receiver.

Applications typically use a combination of message formats, including those message formats that are defined by the following structures or standards:

- Comma Separated Values (CSV)
- COBOL, C, PL1, and other language data structures
- Industry standards such as SWIFT, X12 or HL7
- XML including SOAP

You can model a wide variety of message formats so that they can be understood by IBM Integration Bus message flows. When the message format is known, the broker can parse an incoming message bit stream and convert it into a logical message tree for manipulation by a message flow.

Some message formats are self-defining and can be parsed without reference to a model. However, most message formats are not self-defining, and a parser must have access to a predefined model that describes the message if it is to parse the message correctly.

An example of a self-defining message format is XML. In XML, the message itself contains metadata in addition to data values, and it is this metadata that enables an XML parser to understand an XML message even if no model is available. Another example of a self-defining format is JSON.

Examples of messages that do not have a self-defining message format are CSV text messages, binary messages that originate from a COBOL program, and SWIFT formatted text messages. None of these message formats contain sufficient information to enable a parser to fully understand the message. In these cases, a model is required to describe them.

Even if your messages are self-defining, and do not require modeling, message modeling has the following advantages:

- <u>Runtime validation of messages</u>. Without a message model, a parser cannot check whether input and output messages have the correct structure and data values.
- <u>Enhanced parsing of XML messages</u>. Although XML is self-defining, all data values are treated as strings if a message model is not used. If a message model is used, the parser is provided with the data type of data values, and can cast the data accordingly.
- <u>Code completion assistance when coding transformation</u>. When you are creating ESQL programs for your message flows, the ESQL editor can use message models to provide code completion assistance.
- <u>Graphical mapping</u>. Without message models, you cannot use the Message Mapping editor.
- <u>Reuse of message models</u>, in whole or in part, by creating additional messages that are based on existing messages.
- Generation of documentation.

• <u>Provision of version control and access control</u> for message models by storing them in a central repository.

Message models allow the full use of the facilities that are offered by IBM Integration Bus.

To speed up the creation of message models, importers are provided to read metadata such as C header files, COBOL copybooks, and EIS (Enterprise Information System, such as SAP®) metadata, and to create message models from that metadata. Additionally, predefined models are available for common industry standard message formats such as SWIFT, EDIFACT, X12, FIX, HL7, and TLOG.

The XML Parser was run in *programmatic* mode where it parsed the XML message, so it assumed everything was a string. By parsing with a model, we can get a message with typed elements and one that is subject to constraints (such as required fields, max field lengths, etc.). The toolkit provides wizards to import your existing models (such as WSDLs, XSDs, copybooks, etc.)

- _1. In the Application Development view (project navigator) on the left, right click the whitespace.
- __2. Select New→Message Model... from the menu.



- ___3. Select the **Other XML** radio button (under **XML**).
- ___4. Check out some of the other options for which there are import wizards.
- __5. Click Next.

<u>~ .</u>	
Create a new messag	ge model file
Select the message mode	type or format
XML	
SOAP XML	XML data for use in Web Services.
Other XML	All other XML data.
Text and binary	
CSV text	Comma Separated Values data, a delimited text format commonly used as an export format by spreadsheets and databases.
Record-oriented text	Text data formats where delimited fields are grouped into records.
COBOL	Data for COBOL programs
© C	Data for C programs
Other text or binary	All other text or binary data formats.
Enterprise Information Sy	ustems
C SAP	Data from SAP systems including IDoc and BAPI
Siebel	Data from Siebel systems
PeopleSoft	Data from PeopleSoft
JD Edwards	Data from JD Edwards systems
Other	
CORBA IDL	Data from CORBA
⑦ Database record	Records from relational databases
🖱 MIME	Data for extended email format
IBM supplied	Predefined data format
0	Nets Coul

___6. Select the I already have an XML schema for my data radio button.

___7. Click Next.

New Message Model					
Other XML					VCD
Choose how you would	like to create your X	ML message mo	del.		ASU A
WebSphere Message Bro model if you want to va broker applications by e	oker can parse and se idate the XML is con nabling ESQL conter	erialize your XML rect. A message r nt assist and grap	documents without nodel also speeds u hical maps.	a message mod p development o	el, but requires a f your message
Create an XML schen	na file using an XML	document as an	example		
Create an empty XM	schema file, I will m	nodel my data us	ing the XML schema	editor	
Create an XML schen	na by importing an X		-		
I already have an XM	schema file for my	data			
		_			
(?)	[< Back	Next >	Finish	Cancel
~	L.	×			

___8. Click the **New.** button next to the **Application or Library** field.

New Message Model	Contract Division Name of Contract On Street or other	
Create a new mess Identify the XML Scher	age model file from an XML schema file	
Application or Library:		▼ New
Folder :	<specifying a="" folder="" is="" optional=""></specifying>	Browse
XML Schema file :	<default below.="" file's="" imported="" is="" name.="" one="" select="" the=""></default>	
MQ_Servi	ce_Library ent Resources de workspace:	▼ Browse
?	Sack Next > Fi	nish Cancel

- ___9. In the popup dialog, select **Library**.
- ___10. Click OK.



- ___11. In the popup dialog, type IntroLabLib as the Library name.
- ___12. Click Finish.

New Library	-	the same time of the same	
Create a new library A library allows re-usable artifact	s to be grouped and manag	ged together. Enter a name	for the new library.
Library name IntroLabLib			
?	< <u>B</u> ack	<u>N</u> ext >	<u>Finish</u> Cancel

___13. Back in the Message Model wizard, check the radio button Select file from outside workspace.

___14. Click Browse...

Application or Library:	IntroLabLib	▼ New
-older :	<specifying a="" folder="" is="" optional=""></specifying>	Browse
(ML Schema file :	<default below.="" file's="" imported="" is="" name.="" one="" select="" the=""></default>	
 BARs Independ 	ent Resources	
Contraction of the second		

- __15. Navigate to C:\Student\Intro_XML_Message folder.
- __16. Select IN_Request.xsd.
- ___17. Click **Open**.



___18. Back in the Message Model Wizard, click Finish.

		·		
Application or Library:	IntroLabLib		•	New
older :	<specifying a="" folder="" is="" optio<="" th=""><td>nal></td><td></td><td>Browse</td></specifying>	nal>		Browse
(ML Schema file :	IN_Request.xsd		1	
BARs	ent Resources			
Select file from outsi	de workspace: nt\Intro_XML_Message\IN_Re	quest.xsd	•	Browse.
Location Callone				

You now have a Library project with the XML message model for the input message.



Key Idea: Library Projects

Applications and libraries are deployable containers of resources, such as message flows, message definitions (DFDL, XSD files), JAR files, XSL style sheets, and WebSphere Adapters files.

A library is a logical grouping of related code, data, or both. A library contains references to reusable resources, such as a message model or map. A library can refer to a resource that is contained in another library. Libraries are optional. They are typically used to reuse resources. Libraries can be either embedded in an application (private) or obtained by a message flow (that is not part of an application) dynamically at run time (execution group level). Use multiple libraries to group related resources (for example, by type or function).

Consider using libraries if you want to share routines and definitions across multiple teams, projects, or brokers. Libraries are also useful if you need to use different versions of a coherent set of routines and definitions.

Using a library is typically not necessary if you do not need to regularly reuse IBM Integration Bus routines or definitions.

Notice that the XSD is opened for you after import and is visible using the XML Schema Editor, an editor in the toolkit which shows you both a GUI representation of your XML schema as well as the source. **In_Request** is the only Global element. If you double click on it, you can drill down into its structure.

To:	19.	Double click on the In_	_Request element to view	the message elements.
-----	-----	-------------------------	--------------------------	-----------------------

IntroMessageFlow.msgflow	
Views	Adva
S Schema : http://www.ibm.lab.com	
Directives	
😢 Elements 📴 Types	
In Request	

The message model should now be visible.

e In_Request		😰 (In_RequestType)	
		ActionRequest	string
		DateRequest	string
		customerNumber	int
		customerName	string
	-6	customerDetails	(customerDetailsType)
		contactDetails	(contactDetailsType)
		requestDecision	string
		comments	string

_20. Select the **Detailed** view.



Some elements, such as customerNumber and customerCreditScore, are integers (ints) and not strings.



_21. Close the IN_Request.xsd tab.



Now, let's update the message flow to use the message model when parsing incoming messages.

- ___22. In the project view on the left, select the IntroLab application.
- ___23. Press the right mouse button.
- ___24. Select Manage Library references.



_25. Select the IntroLabLib check box.

_26. Click **OK**.



We need to tell the parser to run in *schema-driven* mode, rather than operate in *programmatic* mode.

_27. Click on the Intro_MessageFlow.msgflow tab to return to the message flow editor.



- ___28. Single click on the XML_Input node, in order to edit its properties.
- ____29. In the **Properties** view, click on the **Validation** tab.
- ___30. In the Validation dropdown, select Content.

×

___31. Select the Parser Options tab.

___32. Select the **Build tree using XML schema data types** check box.

Description Basic Parse timing On Demand Input Message Parsing Use MQRFH2C compact parser for MQRFH2 header Parser Options XMLNSC Parser Options Advanced Image: Security Validation Use XMLNSC compact parser for XMLNS domain Security Retain mixed content Instances Retain comments	•
Basic Parse timing On Demand Input Message Parsing Use MQRFH2C compact parser for MQRFH2 header Parser Options -XMLNSC Parser Options Advanced Image: Build tree using XML schema data types Validation Use XMLNSC compact parser for XMLNS domain Security Retain mixed content Instances Retain comments	•
Input Message Parsing Use MQRFH2C compact parser for MQRFH2 header Parser Options XMLNSC Parser Options Advanced Image: Security Validation Use XMLNSC compact parser for XMLNS domain Security Retain mixed content Instances Retain comments	
Parser Options XMLNSC Parser Options Advanced Image: Security Validation Use XMLNSC compact parser for XMLNS domain Security Retain mixed content Instances Retain comments	
Advanced Image: Build tree using XML schema data types Validation Use XMLNSC compact parser for XMLNS domain Security Retain mixed content Instances Retain comments	
Validation Use XMLNSC compact parser for XMLNS domain Security Retain mixed content Instances Retain comments	
Security Retain mixed content Instances Retain comments	
Instances Retain comments	
Monitoring Retain processing instructions	

2.4 Re-running the Test Client

The flow will now be run again. The trace output will then be examined.

___1. In the editor, select the IntroLab.mbtest tab (or re-open from the App in the navigator).

📧 IntroMessageFlow.msgflow 🛛 📔 *IntroLab

- ___2. Right click Invoke Message Flow.
- __3. Click Re-run.

The tooling will automatically re-build and redeploy the App with the dependent Library included.

⊢ G ▼ D Me: Inp
-

- __4. Return to the Windows Explorer window.
- _5. Double click the C:\XML_Input_Trace.txt file.

Organize 👻 🥘 Open	Print New folder				 ₩ • 🗖	1 (
🚖 Favorites	Name	Date modified	Туре	Size		
Desktop	3 BGInfo	6/30/2010 12:23 PM	File folder			
📕 Downloads	\rm DB2	7/31/2012 1:06 PM	File folder			
🔛 Recent Places	JSRDBM01	7/31/2012 3:08 PM	File folder			
	J Firefox	10/3/2012 6:06 PM	File folder			
🗃 Libraries	🔐 IBM	7/8/2013 4:22 PM	File folder			
Documents	🌡 idsinstinfo	7/31/2012 3:01 PM	File folder			
J Music	didsslapd-dsrdbm01	7/31/2012 3:07 PM	File folder			
Pictures	DerfLogs	7/13/2009 11:20 PM	File folder			
📑 Videos	Program Files	5/16/2013 12:59 PM	File folder			
	Program Files (x86)	7/18/2013 4:01 AM	File folder			
y Computer	퉬 student	7/30/2013 6:37 PM	File folder			
Local Disk (C:)	退 temp	5/16/2013 1:03 PM	File folder			
👷 Shared Folders (\\vn	🎒 Users	7/31/2012 12:57 PM	File folder			
	퉬 Windows	7/9/2013 1:07 PM	File folder			
🙀 Network	WMQv75-POT	4/28/2013 2:09 AM	File folder			
	VS_EXPBSLN_x64_enu	3/19/2010 7:55 PM	Cabinet File	2,026 KB		
	BVS_EXPBSLN_x64_enu	3/19/2010 7:58 PM	Windows Installer	539 KB		
		7/21/2012 1.22 8/4	Text Decument	12 VP		

___6. Scroll down to the end of the file (or use Ctl+End), and view the new trace output.



- ___7. Close the Notepad window.
- 8. Minimize Windows Explorer.

This is the end of lab 2.