SAP NetWeaver
How-To Guide

How To... Perform XML Validations in SAP NetWeaver Process Integration 7.1

Applicable Releases:

SAP NetWeaver Process Integration 7.1 SP5

Topic Area:

SOA Middleware

Capability:

Service Bus

Version 1.0

June 2008

THE BEST-RUN BUSINESSES RUN SAP™

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

These materials are provided “as is” without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

SAP shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials.

SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.

SAP NetWeaver “How-to” Guides are intended to simplify the product implementation. While specific product features and procedures typically are explained in a practical business context, it is not implied that those features and procedures are the only approach in solving a specific business problem using SAP NetWeaver. Should you wish to receive additional information, clarification or support, please refer to SAP Consulting.

Any software coding and/or code lines / strings (“Code”) included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.

Disclaimer

Some components of this product are based on Java™. Any code change in these components may cause unpredictable and severe malfunctions and is therefore expressively prohibited, as is any decompilation of these components.

Any Java™ Source Code delivered with this product is only to be used by SAP’s Support Services and may not be modified or altered in any way.

THE BEST-RUN BUSINESSES RUN SAP™

Document History

Document Version	Description
1.00	First official release of this guide

Typographic Conventions

Type Style	Description
Example Text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation
Example text	Emphasized words or phrases in body text, graphic titles, and table titles
`Example text`	File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example text`	User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, `F2` or `ENTER`.

Icons

Icon	Description
⚠	Caution
💡	Note or Important
❖	Example
⬆	Recommendation or Tip

Table of Contents

1. Business Scenario

We will use SAP NetWeaver Process Integration 7.1 to validate XML documents using the Integration Server or the Adapter Engine.

Using the example below, we will send an XML document from file to file. A file will be picked up by the sender file communication channel from the OS directory and sent to the Integration Server. XML validation will be done in either the Adapter Engine or the Integration Server. After validation, if no error, the message will be sent to the file receiver communication channel to be created on another OS file directory. During this process, we will show validation of the sender's XML document at the Integration Server, then, at the Adapter Engine. The receiver XML validation, not shown in this guide, can be configured similarly.

XML validation can be performed using any adapter, in the same manner as demonstrated in this guide. File adapter is selected here to simplify the communication channel configuration and to focus on the XML validation features.

2. Introduction

Validating the XML documents is a new feature introduced in SAP NetWeaver Process Integration 7.1. The validation can be performed at two different locations - Integration Server or the Adapter Engine. Validations can be done in both synchronous and asynchronous operations.

In the synchronous scenario or using an adapter that can handle synchronous messages (e.g. HTTP and SOAP adapters), when a validation error occurs, the error message will be returned to the sender. In the case of asynchronous scenarios (e.g. file adapter), the error message will be logged to the SXI_MONITOR when the validation is done in the Integration Server or to the Runtime Workbench (RWB) when the validation is done in the Adapter Engine. In both cases, the messaging will be terminated with error.

Both Integration Server and Adapter Engine can be used for validating XML from the sender. However, only the Integration Server can be used to validate the XML when sending to the receiver. The XML validation configurations are done in either the Sender Agreement or the Receiver Agreement.

In the current release, as of PI 7.1, SPS5, the XML schema (or XSD) to be used for the XML validation has to reside in a file directory under the JEE. The XSD file has to be explicitly copied into a specific directory, depending on where the validation is to be performed.

⚠ CAUTION

Due to resource requirement of the validation process, it is recommended that XML validation is done when it is really required, especially when high volume is involved. Based on performance benchmarks, XML validation can take up as much as 50% of the mapping run time.

3. Step-by-Step Procedure

Note

XML Validation details can be referenced in the SAP Help Portal via the URL: http://help.sap.com/saphelp_nwpi71/helpdata/en/44/0bf1b3ec732d2fe10000000a11466f/frameset.htm

To validate an XML document, we must first have the XSD. The XSD can be constructed externally by using tools such as XMLSpy, or by exporting or copying the XSD of a Message Type in the Enterprise Service Repository (ESR).

Then, this XSD must be copied to the appropriate system directory depending on where the validation will be performed. The directory paths are as follow:

- Validation in the Integration Server:

  <sysdir>/xi/runtime_server**/validation/schema/<GUID of software component version to which Service Interface is assigned>/<Repository Namespace of Service Interface>**

  Example

  /usr/sap/<***sid***>/SYS/global/xi/runtime_server/validation/schema/<***guid***>/<***namespace1***>/<***service_interface_name***>/<***namespace2***>

- Validation in the Adapter Engine (for a non clustered installation of AS Java):

  <SAP installation directory>/<system id>/<instance number>/j2ee/cluster/server0**/validation/schema/<GUID of software component version to which Service Interface is assigned>/<Repository Namespace of Service Interface>**

  Example

  /usr/sap/<***sid***>/DVEBMGS00/j2ee/cluster/server0/validation/schema/<***guid***>/<***namespace1***>/<***service_interface_name***>/<***namespace2***>

  Important

  If multiple server nodes have been installed, the XSD file will have to be copied to each one of the server directory, e.g. under server1, server2, etc.

  Note

  The file system schema setup can be found in the SAP Help Portal via the URL: http://help.sap.com/saphelp_nwpi71/helpdata/en/44/23641dc2b36491e10000000a1553f7/frameset.htm

Where:

<*sid*>: System ID of the server instance.

<*guid*>: The GUID of the SWCV.

<*namespace1*>: The namespace within which the service interface is defined.

<*service_interface_name*>: The name of the service interface.

<*namespace2*>: The namespace of the Message Type used by the service interface. In many instances, namespace1 and namespace2 are the same.

CAUTION

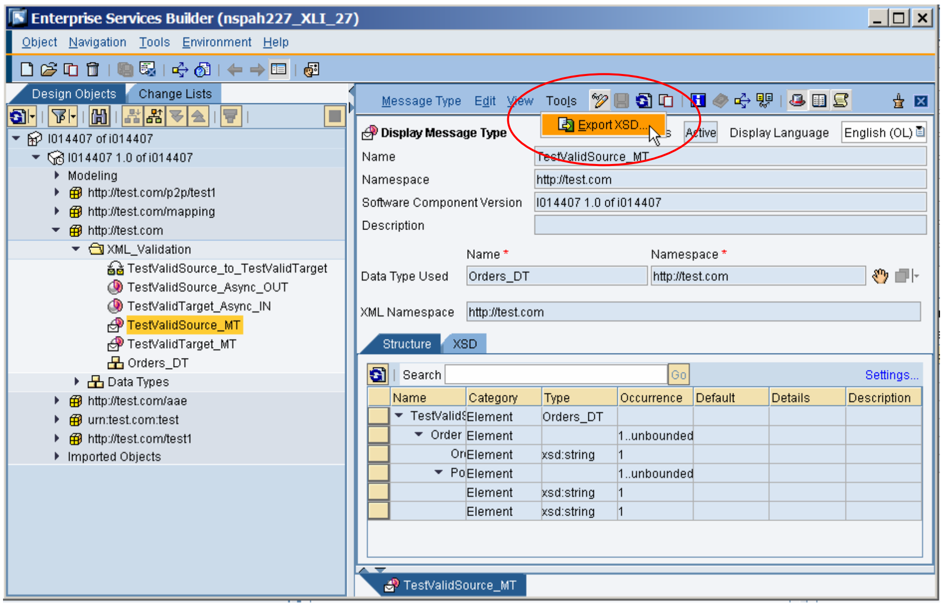
In most situations, the PI developer will not be allowed access to the OS directory in order to copy an XSD. If this is the case, the server system administrator can create a share for a Windows system, or an FTP user to allow the PI developer to administer the XSD files.

3.1 Validation Using the Integration Server

3.1.1 Obtain the XSD File

The XSD can be created using an external tool, e.g. XMLSpy, or in our case, we will export it from the ESR's Message Type.

In the ESR, open the Message Type which needs to be validated, from menu: Tools ➪ Export XSD



Display Message Type

Display Language: English (OL)

Name: TestValidSource_MT

Namespace: http://test.com

Software Component Version: I014407 1.0 of i014407

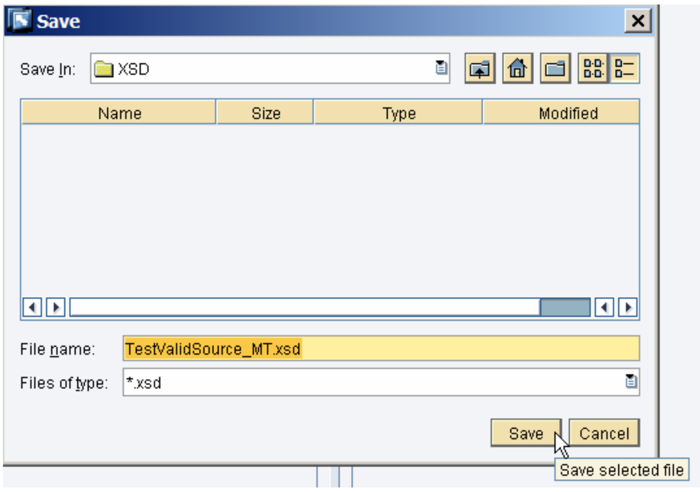
Description:

Data Type Used: Name * Orders_DT; Namespace * http://test.com

XML Namespace: http://test.com

Name	Category	Type	Occurrence	Default	Details	Description
TestValid$	Element	Orders_DT				
Order	Element		1..unbounded			
Or	Element	xsd:string	1			
Po	Element		1..unbounded			
	Element	xsd:string	1			
	Element	xsd:string	1			

In this case, I am exporting it to the /tmp/XSD directory, using the file name: TestValidSource_MT.xsd



Save In: XSD

File name: TestValidSource_MT.xsd

Files of type: *.xsd

Note

The filename must contain the Message Type name, and ".xsd" as the extension.

In some situations, a WSDL file may have been given to us. The XSD is contained in the WSDL. Please reference Appendix C for extraction of the XSD from WSDL.

Note

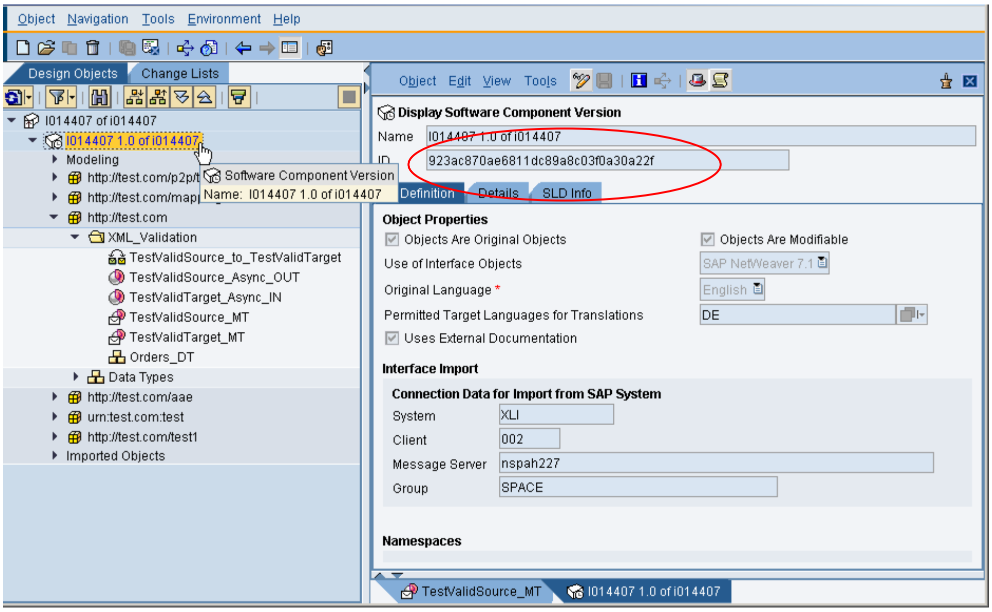
The export of WSDL from the Enterprise Service Repository can be referenced in the SAP Help Portal via the URL: http://help.sap.com/saphelp_nwpi71/helpdata/en/44/23641dc2b36491e10000000a1553f7/frameset.htm

3.1.2 Copy the XSD File to the Integration Server System

In this example, PI 7.1 is installed on a Windows system. The server system administrator has created a share, runtime_server_validation, using the path: /usr/sap/<*sid*>/SYS/global/xi/runtime_server/validation

But, before we can copy the XSD, we have to get all the directory names to be used under the share:

<*guid*>: 923ac870ae6811dc89a8c03f0a30a22f, the GUID of the SWCV, which can be obtained by opening SWCV and get the ID field value.



Display Software Component Version

Name: I014407 1.0 of i014407

ID: 923ac870ae6811dc89a8c03f0a30a22f

Definition | Details | SLD Info

Object Properties

☑ Objects Are Original Objects ☑ Objects Are Modifiable

Use of Interface Objects: SAP NetWeaver 7.1

Original Language *: English

Permitted Target Languages for Translations: DE

☑ Uses External Documentation

Interface Import

Connection Data for Import from SAP System

System: XLI

Client: 002

Message Server: nspah227

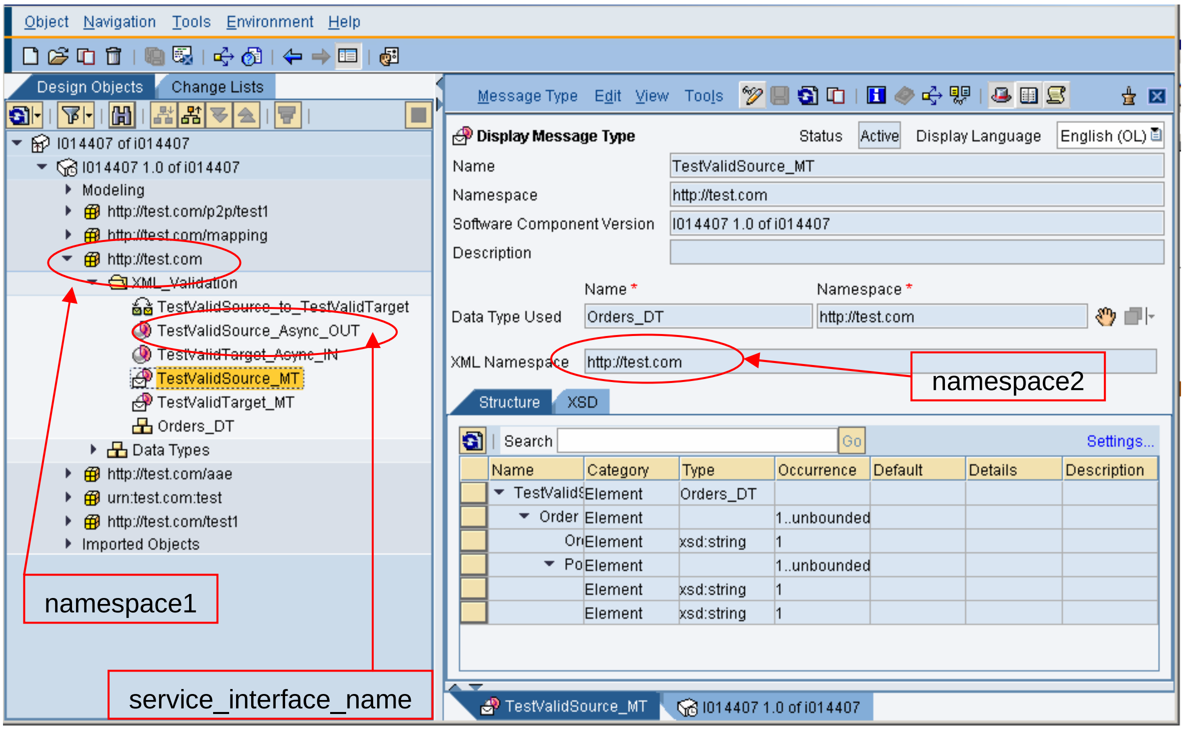
Group: SPACE

Namespaces

<*namespace1*>: http://test.com, the namespace within which the service interface is located.

<*service_interface_name*>: TestValidSource_Async_OUT, the name of the service interface.

<*namespace2*>: http://test.com, the target namespace of the Message Type.



Display Message Type Status: Active Display Language: English (OL)

Name: TestValidSource_MT

Namespace: http://test.com

Software Component Version: I014407 1.0 of i014407

Description:

Data Type Used: Name *: Orders_DT Namespace *: http://test.com

XML Namespace: http://test.com

Structure | XSD

Name	Category	Type	Occurrence	Default	Details	Description
TestValidS	Element	Orders_DT				
Order	Element		1..unbounded			
Or	Element	xsd:string	1			
Po	Element		1..unbounded			
	Element	xsd:string	1			
	Element	xsd:string	1			

Also, to avoid invalid characters in the name of the namespace, we have to replace the invalid characters by "~". Therefore, http://test.com will be replaced by http~test.com.

Note

For a list of all the characters NOT allowed in a file path or folder name, please reference the SAP Help Portal via the URL: http://help.sap.com/saphelp_nwpi71/helpdata/en/44/23641dc2b36491e10000000a1553f7/frameset.htm

As a result, the XSD file, TestValidSource_MT.xsd, will have to be copied to:

\\<server>\runtime_server_validation\schema\923ac870ae6811dc89a8c03f0a30a22f\http~test.com\TestValidSource_Async_OUT\http~test.com\TestValidSource_MT.xsd

-or-

\usr\sap\<*sid*>\SYS\global\xi\runtime_server\validation\schema\923ac870ae6811dc89a8c03f0a30a22f\http~test.com\TestValidSource_Async_OUT\http~test.com\TestValidSource_MT.xsd

Important

When the Message Type namespace is different from the Data Type namespace, the Message Type's XSD will contain an import statement. The handling of imports in the XSD must be done correctly. Please reference Appendix B for more information.

3.1.3 Configure the Sender Agreement in the Integration Directory

Note

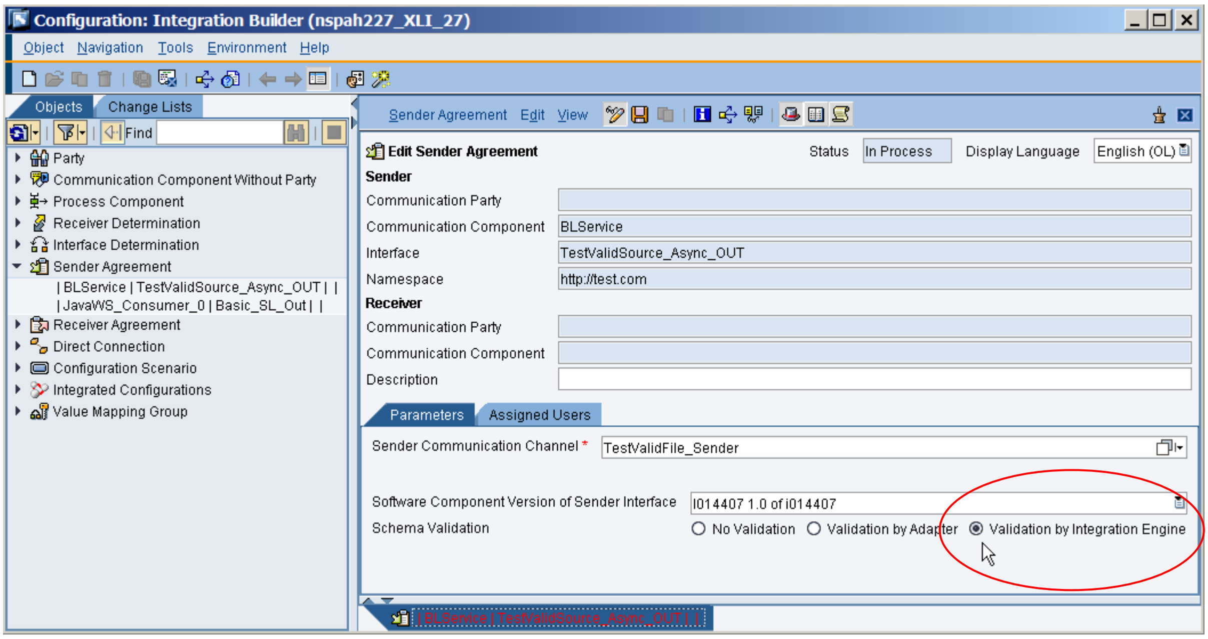
More detailed information of configuring the Sender and Receiver Agreements can be found on SAP Help Portal via the URLs:

http://help.sap.com/saphelp_nwpi71/helpdata/en/5d/112d20f6ce6c46ba66afb98d278fbd/frameset.htm

and

http://help.sap.com/saphelp_nwpi71/helpdata/en/b1/f29e7a56e18a439984a3c6630951d2/frameset.htm

We need to go through all the required configurations in the Integration Directory; the only additional configuration required is to check the XML validation in the Sender Agreement. In this case, we need to check the option "Validation by Integration Engine".


Configuration: Integration Builder (nspah227_XI_27)
Object Navigation Tools Environment Help
Objects Change Lists
Find
Party
Communication Component Without Party
Process Component
Receiver Determination
Interface Determination
Sender Agreement
| BLService | TestValidSource_Async_OUT | |
| JavaWS_Consumer_0 | Basic_SL_Out | |
Receiver Agreement
Direct Connection
Configuration Scenario
Integrated Configurations
Value Mapping Group
Sender Agreement Edit View
Parameters Assigned Users
| BLService | TestValidSource_Async_OUT | |

Edit Sender Agreement

Status: In Process

Display Language: English (OL)

Sender

Communication Party:

Communication Component: BLService

Interface: TestValidSource_Async_OUT

Namespace: http://test.com

Receiver

Communication Party:

Communication Component:

Description:

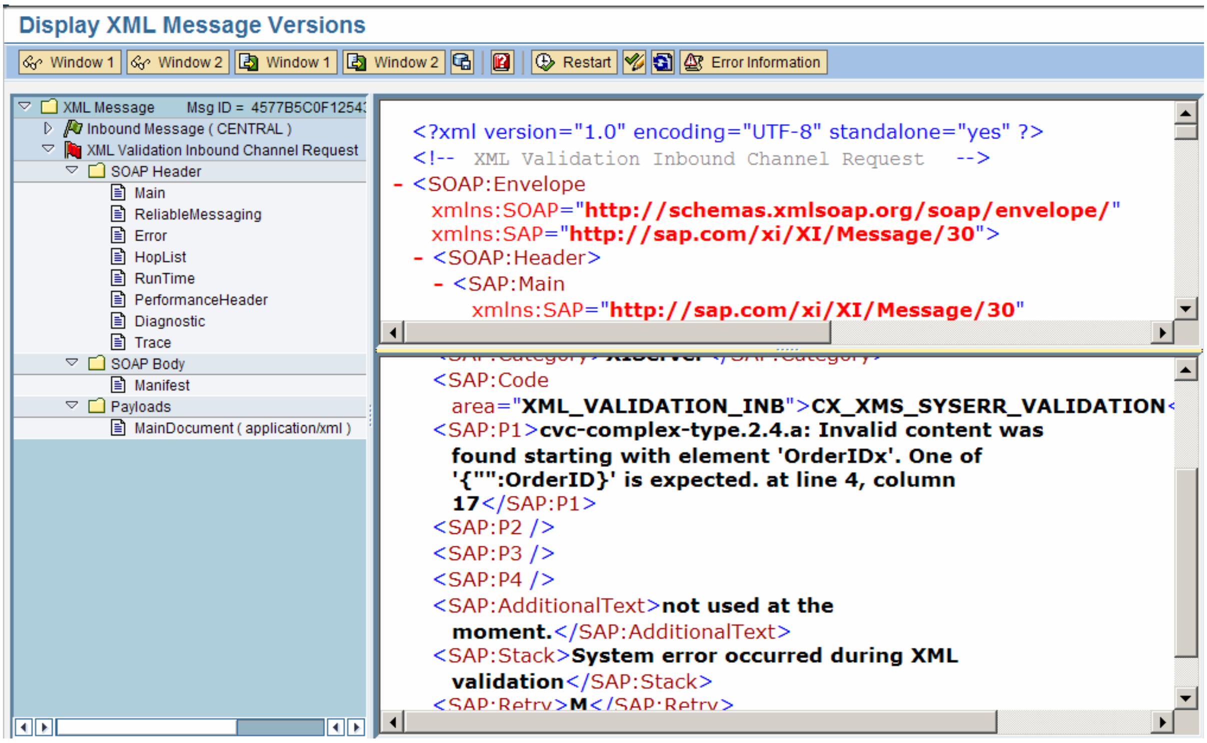
Sender Communication Channel *: TestValidFile_Sender

Software Component Version of Sender Interface: I014407 1.0 of i014407

Schema Validation: ○ No Validation ○ Validation by Adapter ● Validation by Integration Engine

3.1.4 Error Monitoring in the Integration Server

Once the message is executed, we can view the error in SXI_MONITOR.



```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!-- XML Validation Inbound Channel Request -->
- <SOAP:Envelope
   xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:SAP="http://sap.com/xi/XI/Message/30">
 - <SOAP:Header>
  - <SAP:Main
     xmlns:SAP="http://sap.com/xi/XI/Message/30"
```

```
  <SAP:Code
    area="XML_VALIDATION_INB">CX_XMS_SYSERR_VALIDATION<
  <SAP:P1>cvc-complex-type.2.4.a: Invalid content was
    found starting with element 'OrderIDx'. One of
    '{"":OrderID}' is expected. at line 4, column
    17</SAP:P1>
  <SAP:P2 />
  <SAP:P3 />
  <SAP:P4 />
  <SAP:AdditionalText>not used at the
    moment.</SAP:AdditionalText>
  <SAP:Stack>System error occurred during XML
    validation</SAP:Stack>
  <SAP:Retry>M</SAP:Retry>
```

It displays the XML element with error and the line and column position of the element.

Tip

After a validation error occurred and we wish to ignore it and continue the message processing, we can just click the “Restart” button.

From the RWB, you can skip validation.

In the RWB message editor, edit the message and restart, and Check the XSD.

Tip

Instead of following the Section 3.1.2 to determine the directory location to copy the XSD file, take a look at Appendix A on how we can let the Integration Server runtime tell us the complete path of the OS system directory.

3.2 Validation Using the Adapter Engine

3.2.1 Obtain the XSD File

Follow the same instructions in Section 3.1.1.

3.2.2 Copy the XSD File to the Adapter Engine

The basic directions are the same as in Section 3.1.2. The exception is that the directory path is different.

For the adapter engine, a share name, server0_validation, is created with the path:
\usr\sap\<sid>\DVEBMGS00\j2ee\cluster\server0\validation\schema

The XSD file, TestValidSource_MT.xsd, is copied to:

\\<server>\server0_validation\schema\923ac870ae6811dc89a8c03f0a30a22f\http~test.com\TestValidSource_Async_OUT\http~test.com\TestValidSource_MT.xsd

-or-

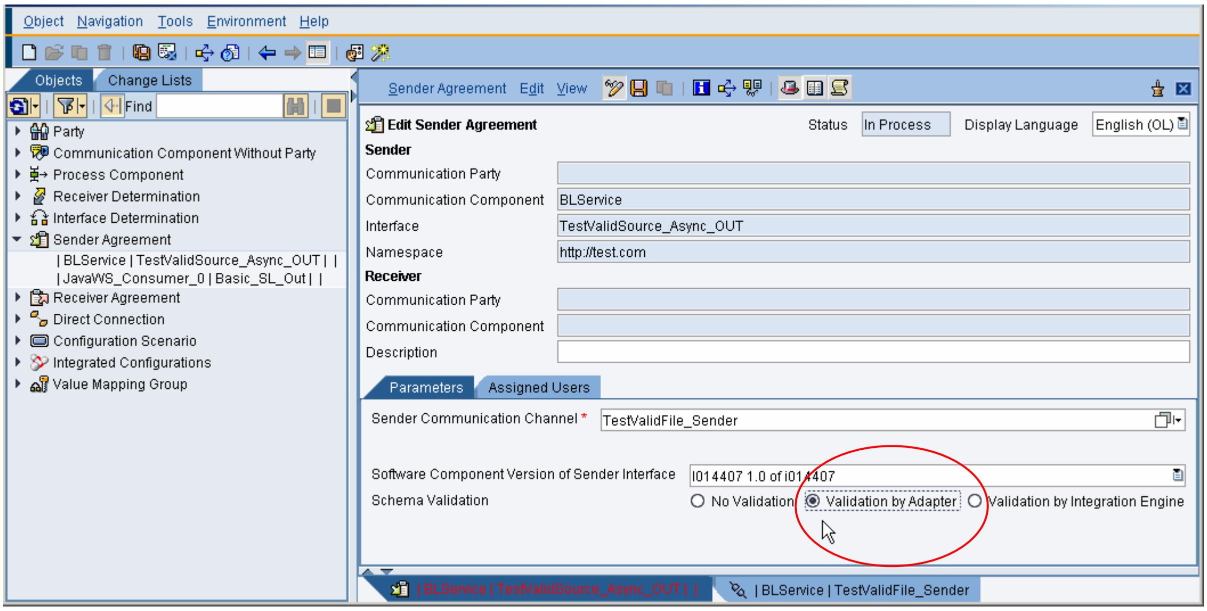
\usr/sap\<*sid*>\DVEBMGS00\j2ee\cluster\server0\validation\schema\923ac870ae6811dc89a8c03f0a30a22f\http~test.com\TestValidSource_Async_OUT\http~test.com\TestValidSource_MT.xsd

Important

If there are additional server nodes, the same copying process will have to be repeated for each one of those nodes.

3.2.3 Configure the Sender Agreement in the Integration Directory

As with the validation in the Integration Engine, the Sender Agreement will have to be configured to use the Adapter Engine for validation. The option “Validation by Adapter” needs to be selected.


Object Navigation Tools Environment Help
Objects
Change Lists
Find
Party
Communication Component Without Party
Process Component
Receiver Determination
Interface Determination
Sender Agreement
| BLService | TestValidSource_Async_OUT | |
| JavaWS_Consumer_0 | Basic_SL_Out | |
Receiver Agreement
Direct Connection
Configuration Scenario
Integrated Configurations
Value Mapping Group
Sender Agreement Edit View
Parameters
Assigned Users
| BLService | TestValidSource_Async_OUT | |
| BLService | TestValidFile_Sender

Edit Sender Agreement

Status: In Process

Display Language: English (OL)

Sender

Communication Party	
Communication Component	BLService
Interface	TestValidSource_Async_OUT
Namespace	http://test.com

Receiver

Communication Party	
Communication Component	
Description	

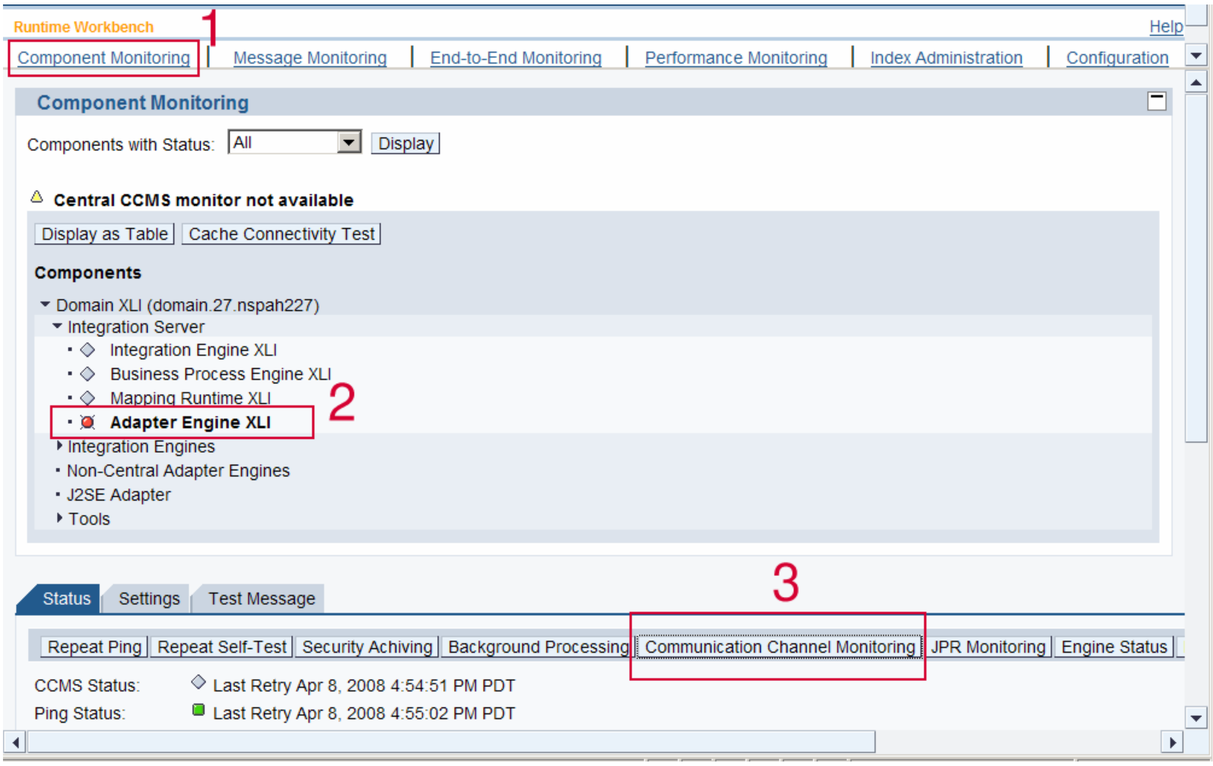
Sender Communication Channel *: TestValidFile_Sender

Software Component Version of Sender Interface: I014407 1.0 of i014407

Schema Validation: ○ No Validation ● Validation by Adapter ○ Validation by Integration Engine

3.2.4 Error Monitoring in the Runtime Workbench (RWB)

When the message is executed, we can find the error in the RWB.



Component Monitoring

Components with Status: All

Central CCMS monitor not available

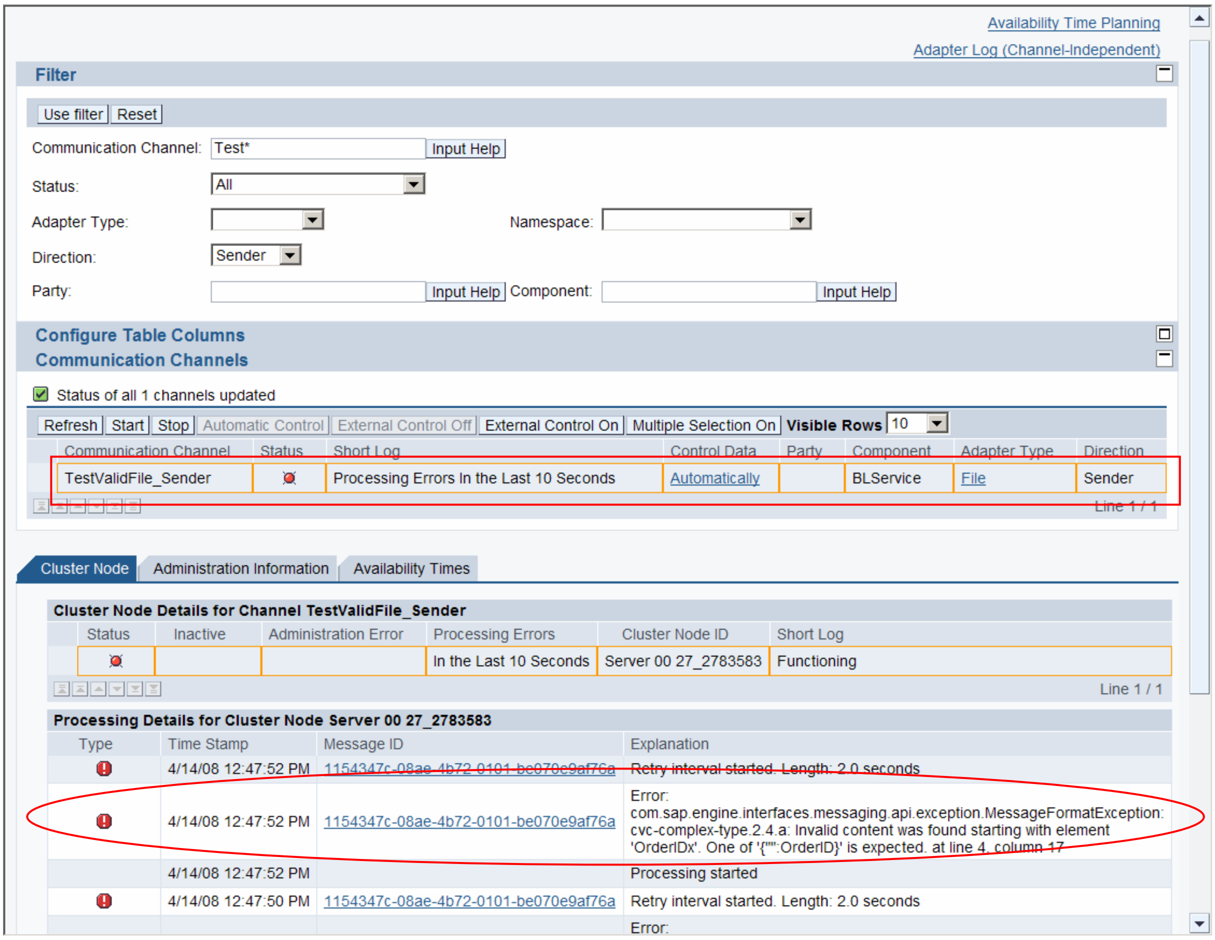
Components

- Domain XLI (domain.27.nspah227)
  - Integration Server
    - Integration Engine XLI
    - Business Process Engine XLI
    - Mapping Runtime XLI
    - **Adapter Engine XLI**
  - Integration Engines
  - Non-Central Adapter Engines
  - J2SE Adapter
  - Tools

CCMS Status: Last Retry Apr 8, 2008 4:54:51 PM PDT

Ping Status: Last Retry Apr 8, 2008 4:55:02 PM PDT

In the Communication Channel Monitoring, select the sender communication channel and display the log:



Communication Channel: Test*

Status: All

Adapter Type:

Namespace:

Direction: Sender

Party:

Component:

☑ Status of all 1 channels updated

Visible Rows 10

Communication Channel	Status	Short Log	Control Data	Party	Component	Adapter Type	Direction
TestValidFile_Sender	●	Processing Errors In the Last 10 Seconds	Automatically		BLService	File	Sender

Line 1 / 1

Cluster Node Details for Channel TestValidFile_Sender

Status	Inactive	Administration Error	Processing Errors	Cluster Node ID	Short Log
●			In the Last 10 Seconds	Server 00 27_2783583	Functioning

Line 1 / 1

Processing Details for Cluster Node Server 00 27_2783583

Type	Time Stamp	Message ID	Explanation
!	4/14/08 12:47:52 PM	1154347c-08ae-4b72-0101-be070e9af76a	Retry interval started. Length: 2.0 seconds
!	4/14/08 12:47:52 PM	1154347c-08ae-4b72-0101-be070e9af76a	Error: com.sap.engine.interfaces.messaging.api.exception.MessageFormatException: cvc-complex-type.2.4.a: Invalid content was found starting with element 'OrderIDx'. One of '{"":OrderID}' is expected. at line 4, column 17
	4/14/08 12:47:52 PM		Processing started
!	4/14/08 12:47:50 PM	1154347c-08ae-4b72-0101-be070e9af76a	Retry interval started. Length: 2.0 seconds
			Error:

It displays the XML element with error and the line and column position of the element. The communication channel error can be alerted via PI’s Alert Framework.

By using the “Status” filter of “Channels with Errors”, we can get a list of all communication channels with errors, including those with XML validation errors. There is currently no way to get a list of communication channels with a specific error, such as XML validation error.

4. Implementation Considerations

XML validation is a much-requested feature enhancement for PI 7.1. But, due to its current design we must take its implementation and management into consideration.

Considerations:

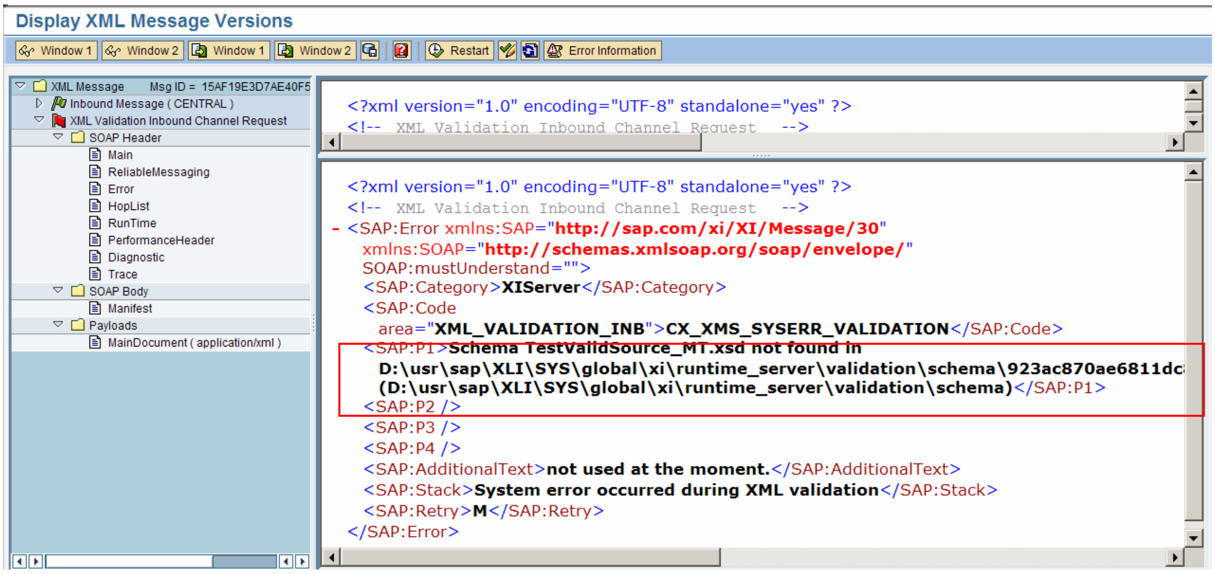
1. The XSD cannot be managed via PI tools. It must be copied to the OS file system, with directory paths hard to decipher.
2. Because of the use of the OS file system, lifecycle management becomes an issue. There are no transport mechanisms to support it.
3. The Message Type and the XSD to validate it are disjointed. This means that changes to the Message Type will not automatically update the XSD; these 2 objects are not synchronized. Manual update process must be performed.
4. Access to the OS file system by the developers can be problematic, especially from the security perspective.
5. The maintenance of the XSDs becomes increasingly complex as the XSDs become more complex, especially when imports are used in the XSDs with different namespaces.
6. Due to resource requirement during the validation process, hardware sizing must be considered during the planning phases.
7. Be extremely careful when using asynchronous/synchronous bridge in the sender JMS adapter. Since the service interface is configured as synchronous, the validation error will be returned to the sender. However, JMS is an asynchronous adapter, therefore, will not be able to receive the response containing the error. As a result, the error cannot be seen any place. At the same time, the message will not go thru and there is no monitoring to indicate so in RWB or the Integration Server.
8. If there are multiple errors in the same XML document, then the RWB only displays the first error.
9. Since messages are not persisted after XML validation error occurs in the Adapter Engine the alerts and navigation using message IDs will not work.

5. Appendix

Appendix A – Use the Integration Server Runtime to Determine the Directory Path for the XSD File

The location of the OS directory path to copy the XSD file is quite long and easy for us to make mistakes. What we can do is to use the runtime engine to tell us the path.

Follow all the steps in Section 3, except skip Section 3.1.2. By doing so, the XML validation by the Integration Engine will still take place, except there is no XSD to use for the validation. As a result, an error will be issue due to missing XSD file. In the error message of SXI_Monitor, the complete directory path will be displayed.



```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!--  XML Validation Inbound Channel Request   -->
```

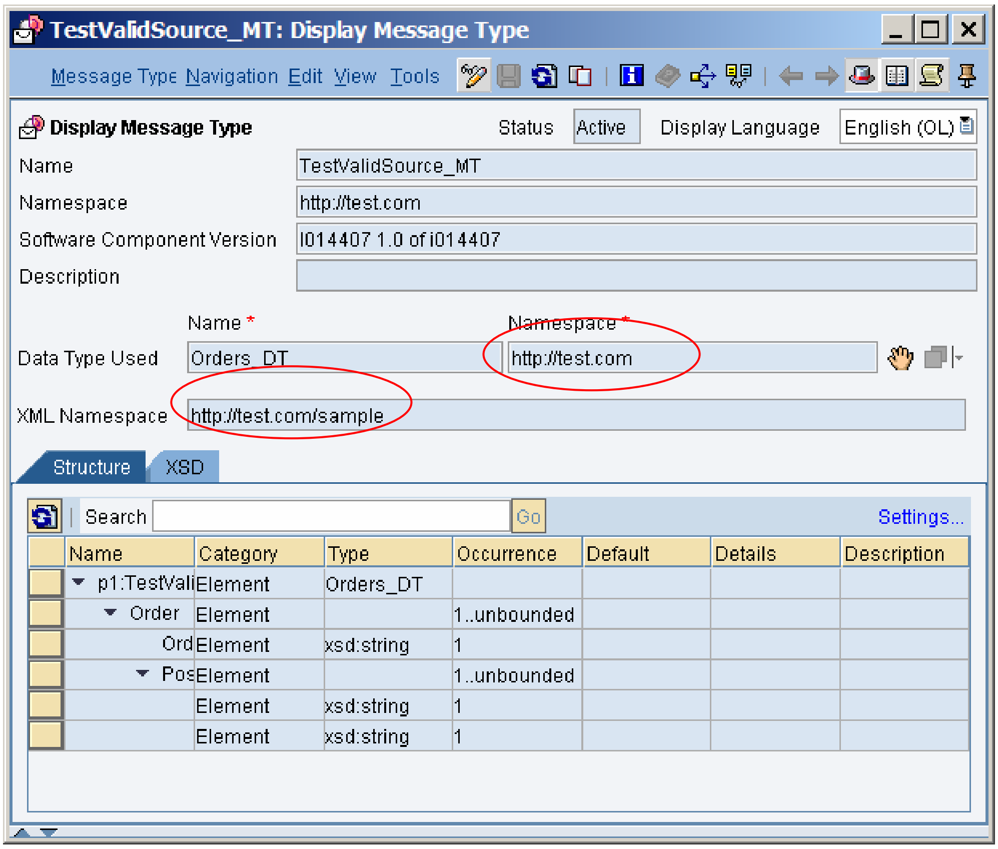
```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!--  XML Validation Inbound Channel Request   -->
- <SAP:Error xmlns:SAP="http://sap.com/xi/XI/Message/30"
    xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
    SOAP:mustUnderstand="">
    <SAP:Category>XIServer</SAP:Category>
    <SAP:Code
      area="XML_VALIDATION_INB">CX_XMS_SYSERR_VALIDATION</SAP:Code>
    <SAP:P1>Schema TestValidSource_MT.xsd not found in
      D:\usr\sap\XLI\SYS\global\xi\runtime_server\validation\schema\923ac870ae6811dc
      (D:\usr\sap\XLI\SYS\global\xi\runtime_server\validation\schema)</SAP:P1>
    <SAP:P2 />
    <SAP:P3 />
    <SAP:P4 />
    <SAP:AdditionalText>not used at the moment.</SAP:AdditionalText>
    <SAP:Stack>System error occurred during XML validation</SAP:Stack>
    <SAP:Retry>M</SAP:Retry>
  </SAP:Error>
```

The above error gave the complete path as:

D:\usr\sap\XLI\SYS\global\xi\runtime_server\validation\schema\923ac870ae6811dc89a8c03f0a30a22f\http~test.com\TestValidSource_Async_OUT\http~test.com\TestValidSource_MT.xsd

Appendix B – Handling XSD Imports in XSD

If the Message Type's namespace is different than the Data Type's namespace, as shown below, the XSD export will create a ZIP file. The content of the ZIP file must be handled properly.



Display Message Type — Status: Active — Display Language: English (OL)

Name: TestValidSource_MT

Namespace: http://test.com

Software Component Version: I014407 1.0 of i014407

Description:

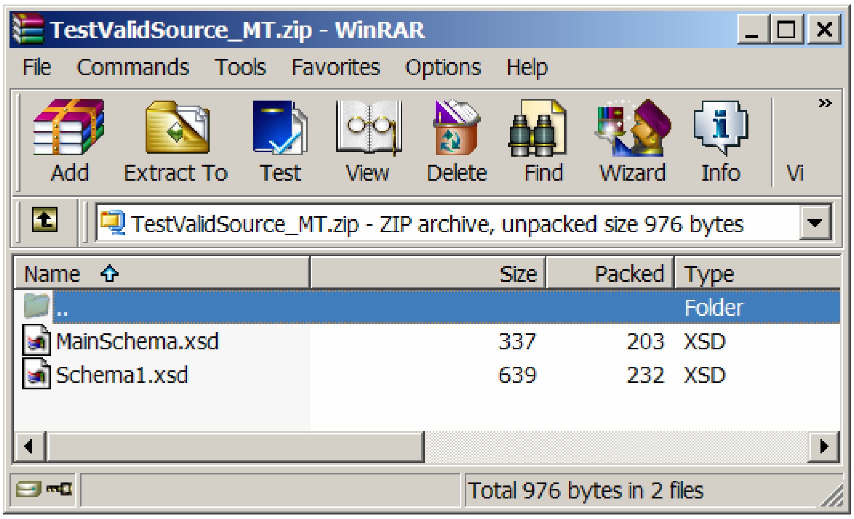
Data Type Used — Name *: Orders_DT — Namespace *: http://test.com

XML Namespace: http://test.com/sample

Search

	Name	Category	Type	Occurrence	Default	Details	Description
	▾ p1:TestVali	Element	Orders_DT				
	▾ Order	Element		1..unbounded			
	Ord	Element	xsd:string	1			
	▾ Pos	Element		1..unbounded			
		Element	xsd:string	1			
		Element	xsd:string	1			

Figure: Message Type namespace is different from Data Type's namespace.



TestValidSource_MT.zip - ZIP archive, unpacked size 976 bytes

Name	Size	Packed	Type
..			Folder
MainSchema.xsd	337	203	XSD
Schema1.xsd	639	232	XSD

Total 976 bytes in 2 files

Figure: ZIP file created when exporting the XSD of the Message Type.

The content of MainSchema.xsd contains an import statement for the Data Type:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://test.com/sample" xmlns:p7="http://test.com"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://test.com/sample">
    <xsd:import namespace="http://test.com" schemaLocation="Schema1.xsd" />
    <xsd:element name="TestValidSource_MT" type="p7:Orders_DT" />
</xsd:schema>
```

After extraction of the files, the following steps are required: (using the example we are using in this guide)

1. Rename the file MainSchema.xsd to TestValidSource_MT.xsd.
2. Copy the file, TestValidSource_MT.xsd, to the OS directory:

   …\TestValidSource_Async_OUT**http~test.com~sample**

   Note

   The namespace used is the namespace of the Message Type.

3. Copy the file, Schema1.xsd, to the OS directory:

   …\TestValidSource_Async_OUT**http~test.com**

   Note

   The namespace used is the namespace of the Data Type.

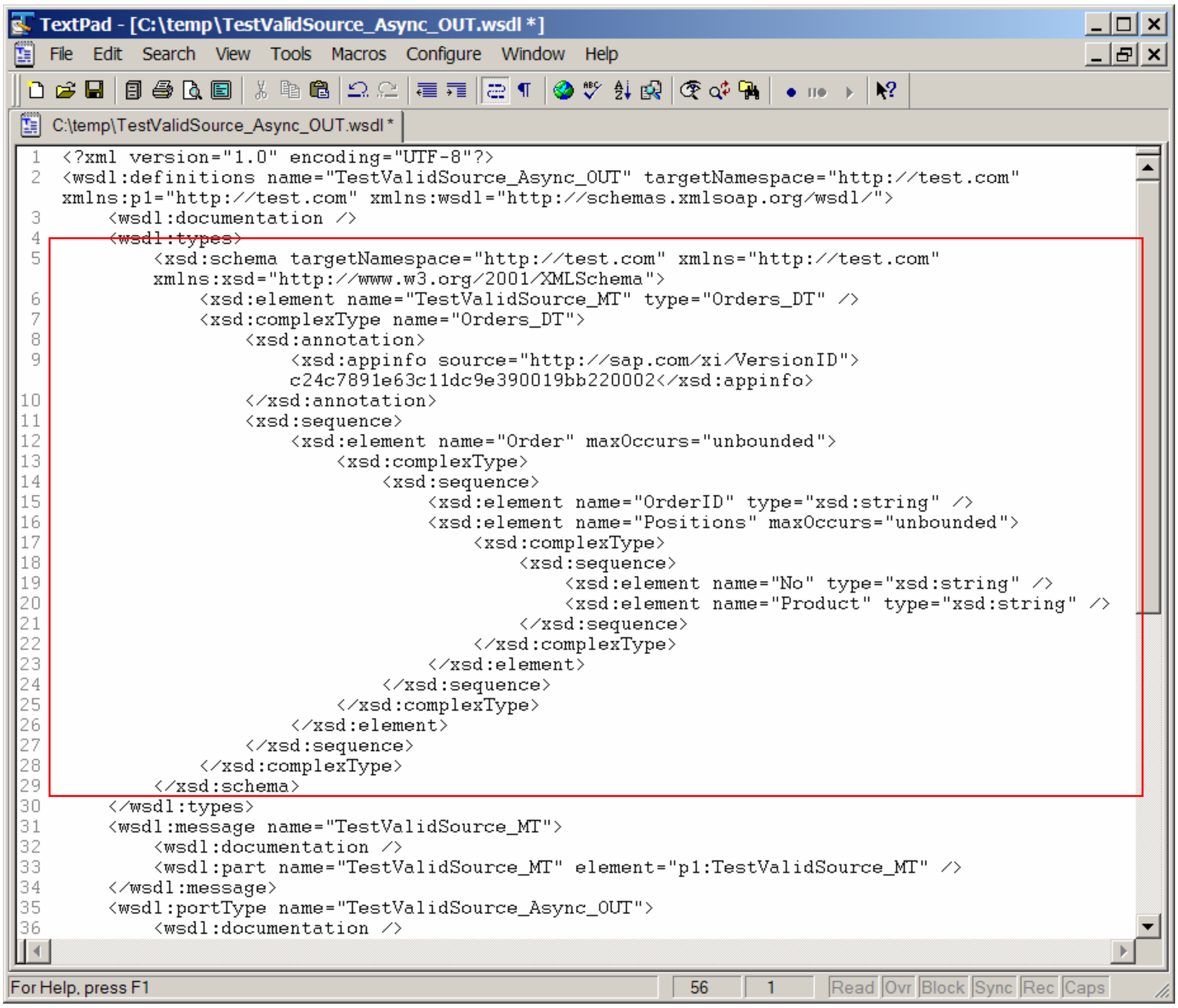
Important

If the SWCV and the namespace within the SWCV are also different between the Message Type and Data Type, the directory locations must also be adjusted to use the appropriate SWCV GUID and namespace name.

Appendix C – Obtaining the XSD from an WSDL

When a WSDL is given to us, the XSD is always included in the WSDL document. We will need to extract the XSD from the WSDL. Since WSDL is a flexible language, the XSD can be included in many variations. Below is a typical WSDL containing an XSD, and the extraction steps.

Only a portion of the complete WSDL will be displayed.


TextPad - [C:\temp\TestValidSource_Async_OUT.wsdl *]
File Edit Search View Tools Macros Configure Window Help
C:\temp\TestValidSource_Async_OUT.wsdl *
For Help, press F1
56
1
Read Ovr Block Sync Rec Caps

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="TestValidSource_Async_OUT" targetNamespace="http://test.com"
xmlns:p1="http://test.com" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
    <wsdl:documentation />
    <wsdl:types>
        <xsd:schema targetNamespace="http://test.com" xmlns="http://test.com"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <xsd:element name="TestValidSource_MT" type="Orders_DT" />
            <xsd:complexType name="Orders_DT">
                <xsd:annotation>
                    <xsd:appinfo source="http://sap.com/xi/VersionID">
                    c24c7891e63c11dc9e390019bb220002</xsd:appinfo>
                </xsd:annotation>
                <xsd:sequence>
                    <xsd:element name="Order" maxOccurs="unbounded">
                        <xsd:complexType>
                            <xsd:sequence>
                                <xsd:element name="OrderID" type="xsd:string" />
                                <xsd:element name="Positions" maxOccurs="unbounded">
                                    <xsd:complexType>
                                        <xsd:sequence>
                                            <xsd:element name="No" type="xsd:string" />
                                            <xsd:element name="Product" type="xsd:string" />
                                        </xsd:sequence>
                                    </xsd:complexType>
                                </xsd:element>
                            </xsd:sequence>
                        </xsd:complexType>
                    </xsd:element>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:schema>
    </wsdl:types>
    <wsdl:message name="TestValidSource_MT">
        <wsdl:documentation />
        <wsdl:part name="TestValidSource_MT" element="p1:TestValidSource_MT" />
    </wsdl:message>
    <wsdl:portType name="TestValidSource_Async_OUT">
        <wsdl:documentation />
```

The XSD is encapsulated between the XML tags <wsdl:types> and </wsdl:types>, or between <types> and </types>.

Copy the content between those tags to a new file and add, as the first line, the line:

<?xml version="1.0" encoding="UTF-8"?>

The resulting file will look as follow, which can be copied to the OS directory:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://test.com" xmlns="http://test.com"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="TestValidSource_MT" type="Orders_DT" />
    <xsd:complexType name="Orders_DT">
        <xsd:annotation>
            <xsd:appinfo
source="http://sap.com/xi/VersionID">c24c7891e63c11dc9e390019bb220002</xsd:appinfo>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="Order" maxOccurs="unbounded">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="OrderID" type="xsd:string" />
                        <xsd:element name="Positions" maxOccurs="unbounded">
                            <xsd:complexType>
                                <xsd:sequence>
                                    <xsd:element name="No" type="xsd:string" />
                                    <xsd:element name="Product" type="xsd:string"
/>
                                </xsd:sequence>
                            </xsd:complexType>
                        </xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

Since no two WSDL structures may be alike, it is possible some manipulations of the XSD may be necessary.

www.sdn.sap.com/irj/sdn/howtoguides