IDoc Packaging and Mapping Techniques

Applies To:

SAP NetWeaver 04 – SAP NetWeaver Exchange Infrastructure 3.0 (SP14)

Summary

This paper is intended for all those who are familiar with the basics of SAP XI and want to learn to configure complex scenarios. In this BOK we have given details of how to configure a File to IDoc packaging scenario and show some important mapping techniques in SAP XI with some important transactions at the end.

Table of Contents

IDoc Packaging and Mapping Techniques

SAP DEVELOPER NETWORK

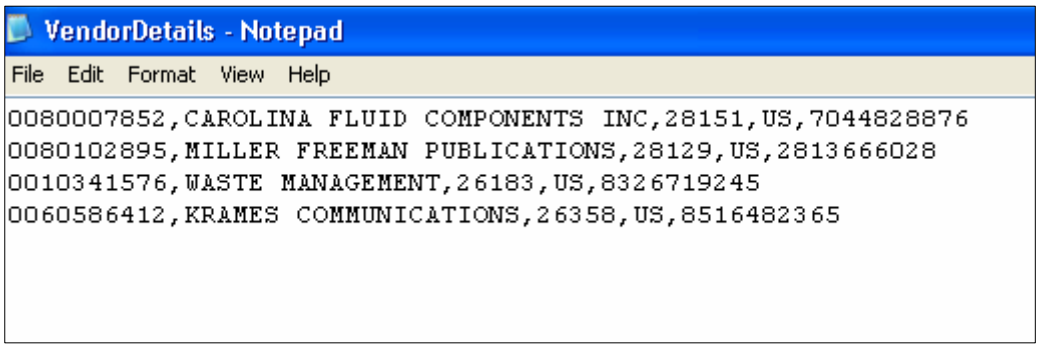
1. File to IDoc Packaging scenario

Posting single IDoc can be done by a simple Message Mapping between source message type and the imported IDoc structure in XI. But in case of multiple record-set from the source side it is not possible to create multiple IDocs instances in XI message with a the same imported IDoc.
For this purpose the IDoc structure needs to be changed to have occurrence 0:unbounded. This is done by exporting IDoc to XSD and adding maxOccurs expression. This XSD is reimported to XI as External Definition in Interface Objects and further used in the Message Mapping.

This interface scenario gives step by step approach for multiple IDocs instances in an XI message.

Follow the steps for File to IDoc packaging (Multiple IDoc instances in an XI message) scenario.

File: The source comma delimited file structure looks like this. This file is used to post data to XI.



1.1. SLD

1.3.1. Under Software Catalog create software component and Product.
1.3.2. Under Technical Landscape create technical systems.
1.3.3. Under Business Landscape create business systems, IDESDEV_600.
1.3.4. Import this software component in IR.

1.2. Integration Repository (IR)

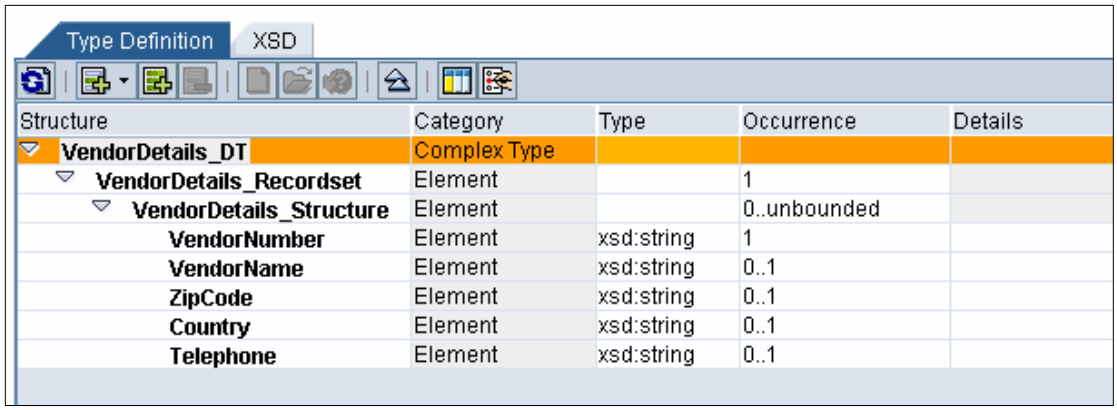
In the Integration Repository (Designer) create the following objects.

1.2.1. From the menu bar, Tools -> Transfer from SLD-> Import software component versions Select the software component 'BP_TRNG_SANDBOX_ONGL_XI' and import it.

1.2.2. After importing it select it from the navigation bar and right click on it to create a namespace http://infosys.com/xi/OneNGL/training/file/file-idoc.

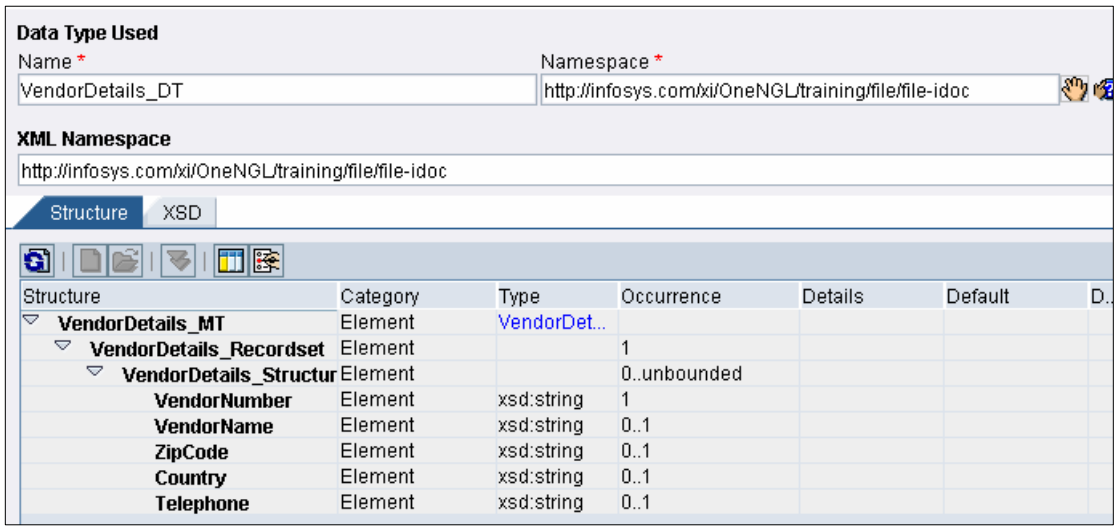
Interface Objects:

1.2.3. Create Data Type: Under the above namespace -> Interface Objects-> Data Types, right click and create '**VendorDetails_DT**' data type as below.



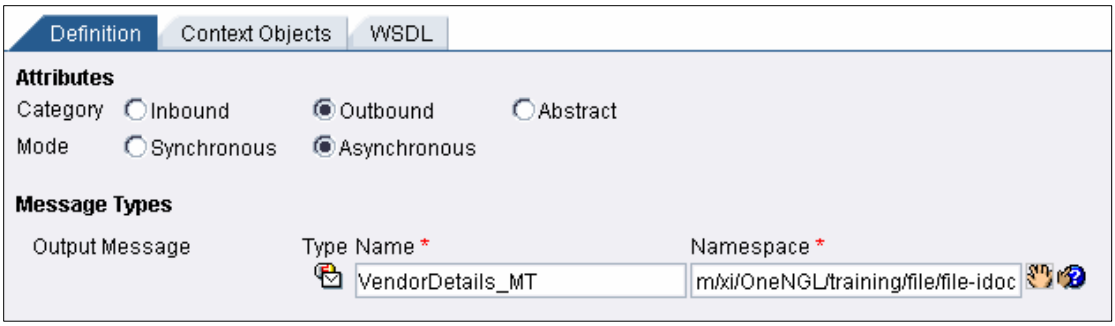
Structure	Category	Type	Occurrence	Details
VendorDetails_DT	Complex Type			
VendorDetails_Recordset	Element		1	
VendorDetails_Structure	Element		0..unbounded	
VendorNumber	Element	xsd:string	1	
VendorName	Element	xsd:string	0..1	
ZipCode	Element	xsd:string	0..1	
Country	Element	xsd:string	0..1	
Telephone	Element	xsd:string	0..1	

1.2.4. Create Message Type: Under the above namespace -> Interface Objects-> Message Types right click and create '**VendorDetails_MT**'. Select the data type created in previous step by F4 help.

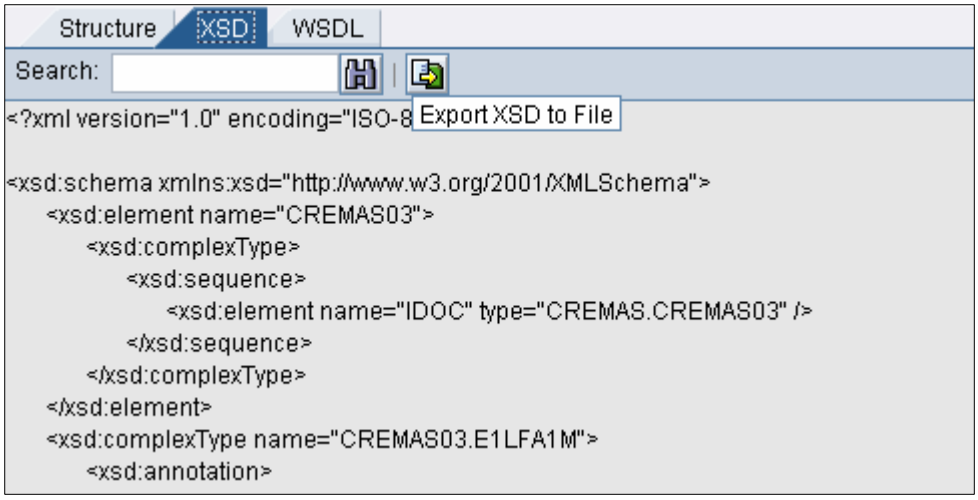


Structure	Category	Type	Occurrence	Details	Default	D..
VendorDetails_MT	Element	VendorDet...				
VendorDetails_Recordset	Element		1			
VendorDetails_Structur	Element		0..unbounded			
VendorNumber	Element	xsd:string	1			
VendorName	Element	xsd:string	0..1			
ZipCode	Element	xsd:string	0..1			
Country	Element	xsd:string	0..1			
Telephone	Element	xsd:string	0..1			

1.2.5. Create Message Interface: Under the above namespace -> Interface Objects-> Message Interfaces right click and create outbound asynchronous '**VendorDetails_MI**'. Select the message type created in previous step by F4 help. This interface is outbound because it will be used to send data from File system to XI. We are not receiving any response back so it is asynchronous interface.



1.2.6. Importing IDOC from SAPR/3 system: Import the CREMAS03 IDoc under Imported Objects -> IDocs. Open CREMAS.CREMAS03, go to XSD tab and export XSD structure to file by clicking the button as shown below.



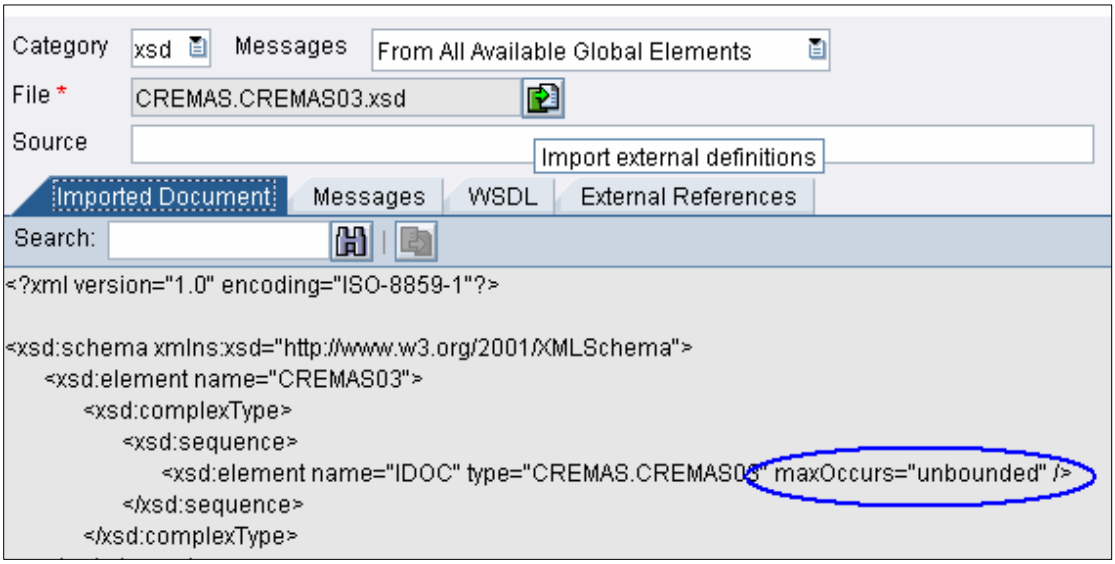
1.2.7. Changing the IDoc XSD: Open the exported XSD file in edit mode and change
<xsd:element name="IDOC" type="CREMAS.CREMAS03" />

to

<xsd:element name="IDOC" type="CREMAS.CREMAS03" maxOccurs="unbounded"/>

1.2.8. Creating External Definition for IDOC: In this step we will create a external definition for CREMAS03 IDoc. Go to Interface Objects -> External Definitions and create a new External Definition 'CREMAS03_ED'.

Select category as XSD and import the changed XSD file. The new structure will look like this.



Mapping Objects:

1.2.9. Create Message Mapping: In this step we will create Message Mapping '**VendorDetails_TO_CREMAS03_ED_MM**' between source message type '**VendorDetails_MT**' and External Definition '**CREMAS03_ED**' of IDoc.
Map node 'VendorDetails_Structure' to node 'IDOC' so that for every 'VendorDetails_Structure' occurrence it will create one 'IDOC'.
Disable EDI_DC40 segment.

Map following fields as shows

VendorNumber	-> LIFNR
VendorName	-> NAME1
ZipCode	-> PSTLZ
Country	-> LAND1
Telephone	-> TELF1
BEGIN	-> 1

1.2.10. Create Interface Mapping: In this step we will create Interface Mapping between '**VendorDetails_MI**' and **'CREMAS.CREMAS03'** IDoc.

The Message Mapping **'VendorDetails_TO_CREMAS03_ED_MM'** is between **VendorDetails_MT** and External definition **CREMAS03_ED,** so the mapping program will not come automatically on F4 help in Mapping Program, so you will need to drag and drop Message Mapping to Mapping Program.

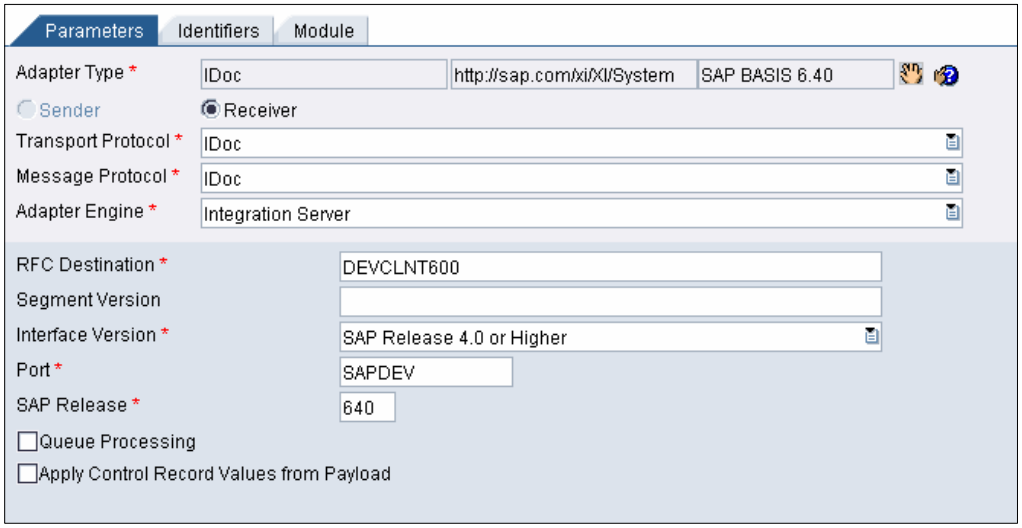
Save and activate all the changes.

1.3. Integration Directory (ID)

1.3.1. Create Configuration Scenario: In the Integration Directory from the menu bar select Object-> New and create a scenario called “INFY_FILE_TO_MULTIPLEIDOC” and save it.

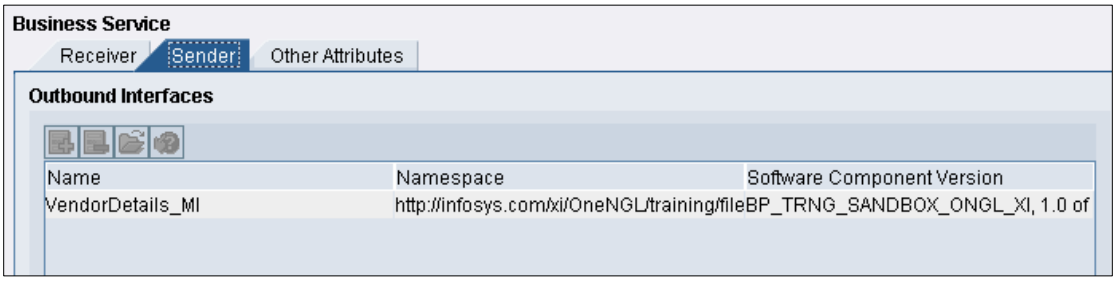
1.3.2. Assign Business System: Select Service Without Party-> Business System. Right click say Assign Business System and select business system IDESDEV_600. This system will act as a receiver of IDocs.

1.3.3. Create Communication Channel for receiver: We will create communication channel for this System. Right click on communication channel, click NEW. Give the name as IDOC_Receiver. Select the type as IDOC. Enter the parameters RFC Destination, Port and SAP Release as shown below.



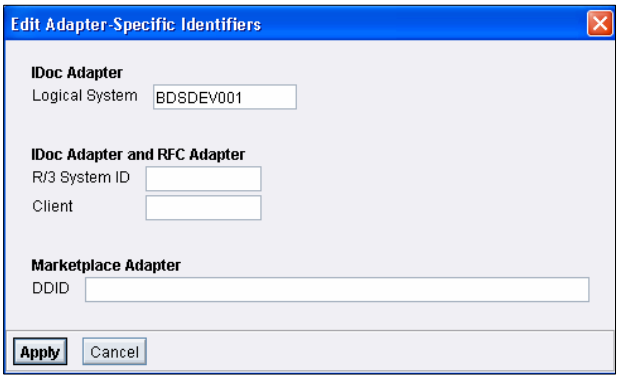
1.3.4. Create Business Service: We will create a Business Service for Source file system. Right click on Business Service click NEW, create BDSDEV001 service and save it. This service will be used to send the data from file to XI.

1.3.5. Assign OutBound Interfaces: Double click on the Business Service, go to Sender tab and add the outbound interface ‘**VendorDetails_MI**’ which was created in Repository. This will be used in ID configuration.



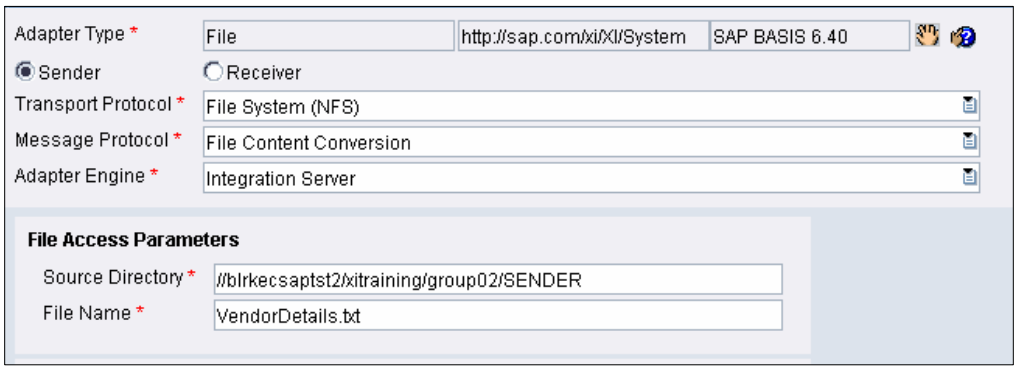
1.3.6. Create Adapter specific details: Go to Service -> Adapter specific identifiers, give any unique logical system name. (You can ignore other details). This is necessary for SAP R/3 system to identify the system from which data is coming.

If these details are not given XI will give following error.
ERROR: Unable to convert sender service BDSDEV001 to an ALE logical system

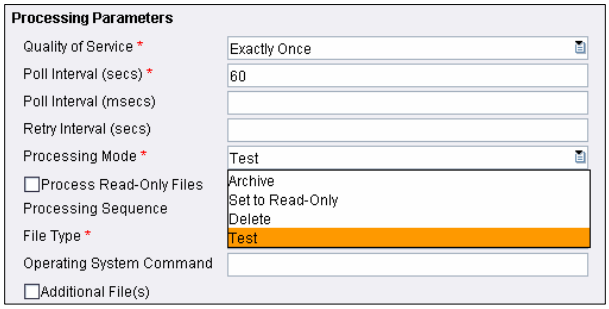


1.3.7. Create Communication Channel for sender: We will create communication channel for this System. Right click on communication channel, click NEW. Give the name as FILE_Sender. Select the type as FILE Sender. Select Message protocol as File Content Conversion. This is necessary because the file is coming in text format which needs to be converted to XML structure in by the adapter. Enter the parameters Target directory, File Name etc as shown below.

Note: Use "/" instead of windows "\" in source directory.



1.3.8. Give other details for FILE_Sender: Provide the details for sender file an adapter as show below.



Quality of Service Exactly once means sender does not wait for the response. The message is sent and processed exactly once. This is guaranteed delivery.

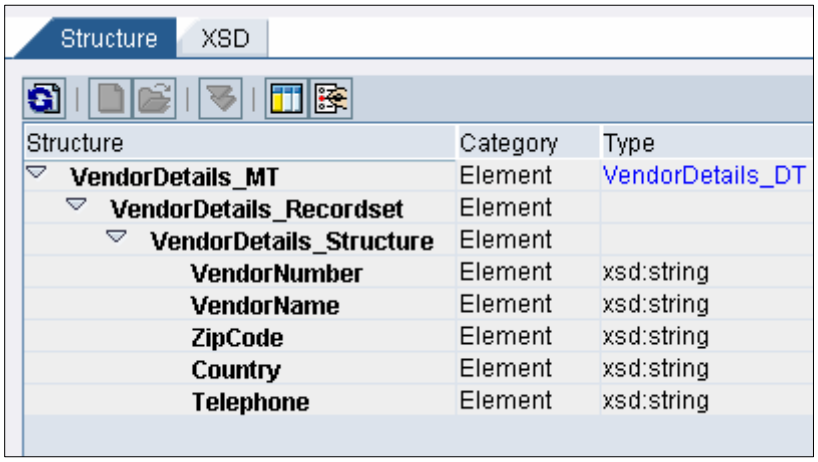
Poll Interval is number of seconds that the adapter must wait if no files are found for processing.

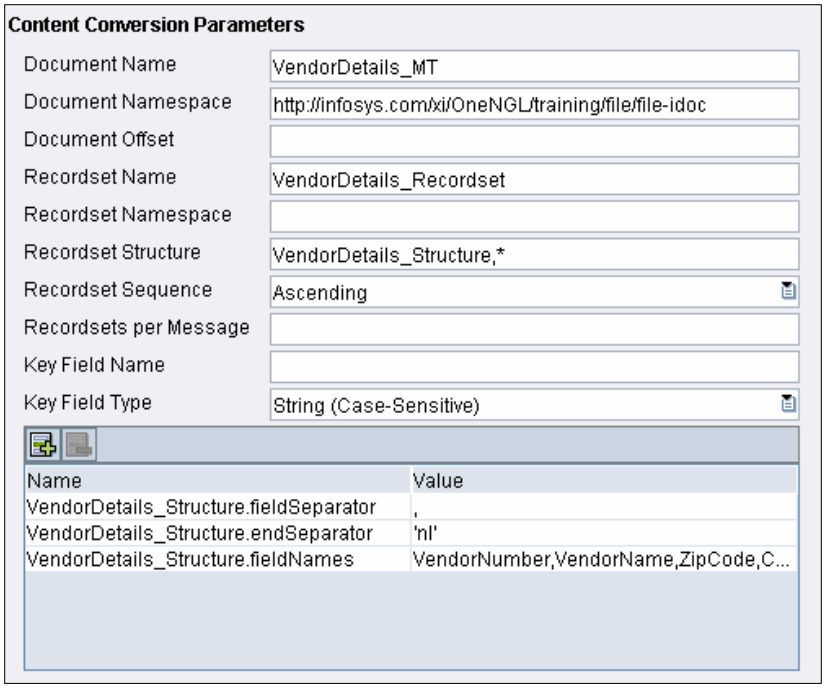
Processing Mode: This indicates the action on the file that needs to be done after polling.

Document name is Message type name.

RecordSet Name and RecordSet Structure are sub nodes of sender message type which we created in repository. Please refer to below screen shots. ,* indicates that there can be one or more structures.

fieldSeparator (,), endSeparator (“nl”-New Line) and fieldnames are given as shown.

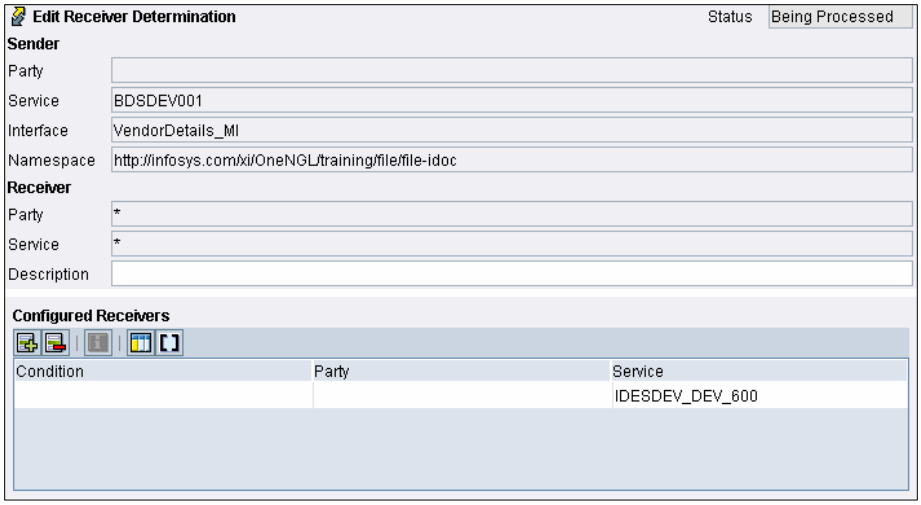




1.3.9. Create Sender Agreement: Next step is to create Sender agreement. Sender agreement defines binding between communication channel and outbound interface.

Right click on Sender Agreement -> New. Select the sender system BDSDEV001 and outbound interface VendorDetails_MI. In screen Edit Sender Agreement use F4 to select Sender communication channel.

1.3.10. Create Receiver Determination: In this step we will create Receiver Determination. Select sender system BDSDEV001 and message interface VendorDetails_MI. The receiver system is IDESDEV_DEV_600.



1.3.11. Create Interface Determination : Now that we have defined receiver for this message, we need to assign an inbound interface and interface mapping. In the area “Configuration Overview for Receiver Determination” at the bottom, hit refresh.

In the column “Receiver (Partner/Services) right click and select “New Specific”. You are now in to Edit Interface determination mode. Select inbound interface which will be ‘CREMAS.CREMAS03’ and interface mapping ‘VendorDetails_TO_CREMAS03_IM’. Save and close.

You can also create Interface determination by Right click -> New on Interface Determination.

1.3.12. Create Receiver Agreement: In this step we will create receiver agreement which will allow us to assign receiver communication channel to the receiver system and interface. Select sender service ‘BDSDEV001’, receiver service as ‘IDESDEV_DEV_600’, inbound interface as ‘CREMAS.CREMAS03’ and communication channel ‘IDOC_Receiver’.

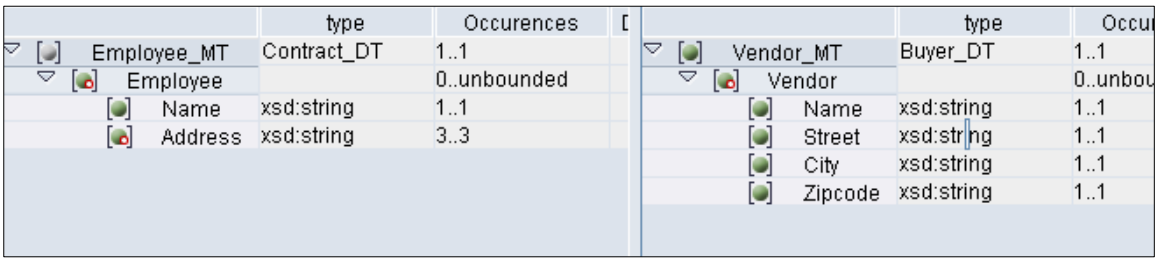
Save and activate all changes.

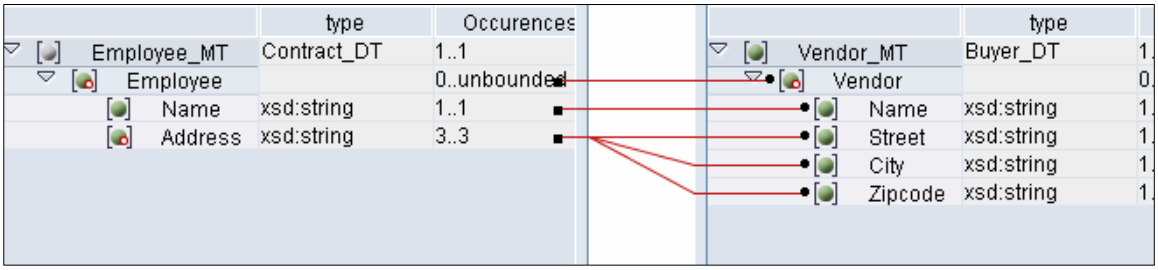
2. Mapping functions

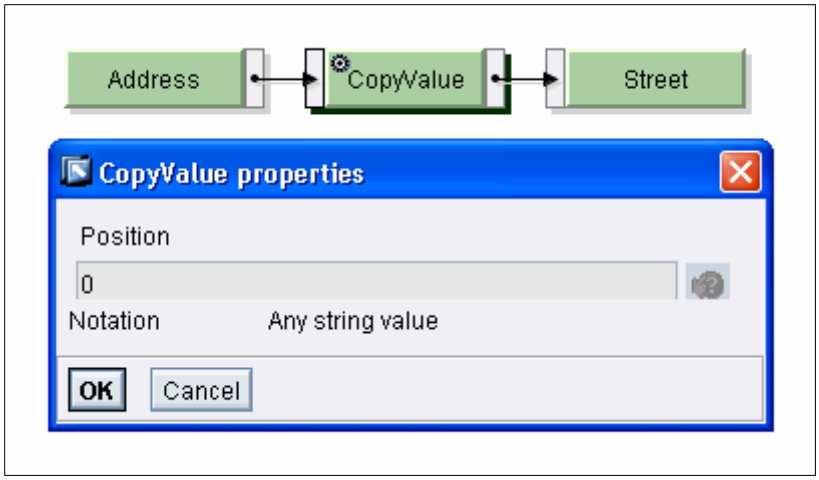
2.1. Copy Value

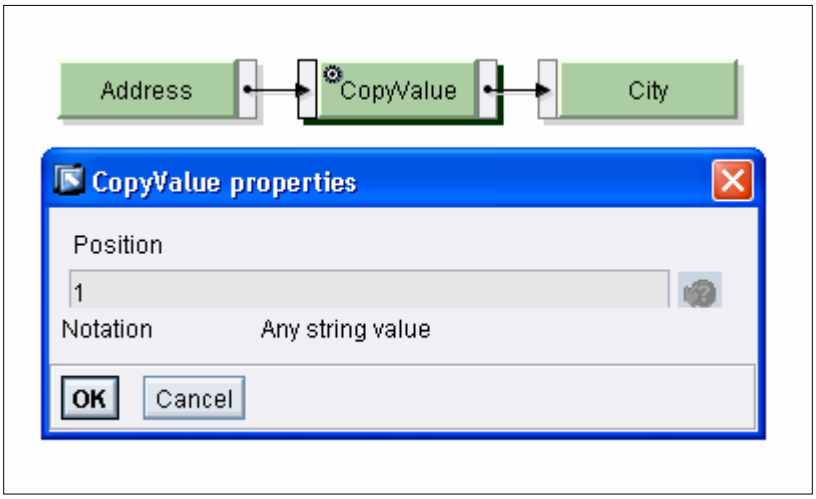
You can use **CopyValue** to copy the value of a position in the source structure and assign it to a target field for a frequently occurring source structure element. The value is copied each time the target field occurs in the target structure. However, it is copied a maximum of maxOccurs times

Mapping:

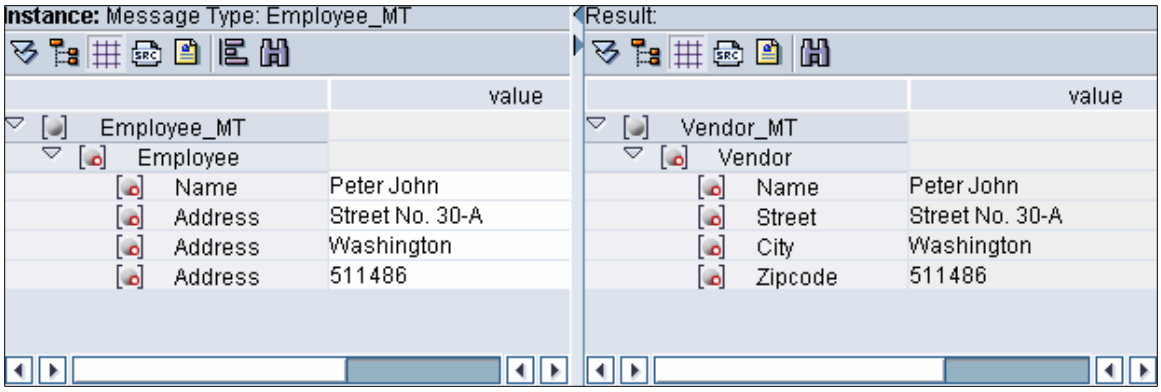








Testing:



Description:

In the example above, source structure represents Address, which occurs three times. However, in the target structure, this information is separated into three individual fields viz Street, City, Zipcode.

Use CopyValue function to assign values of Address field to three different target fields. In this case Address behaves like arrays.

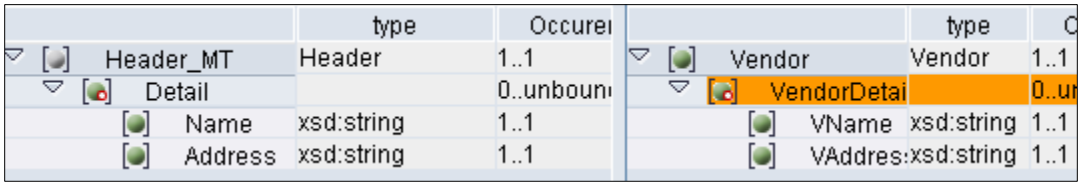
Double click on CopyValue to assign the position of source field to be assigned to a particular target field.

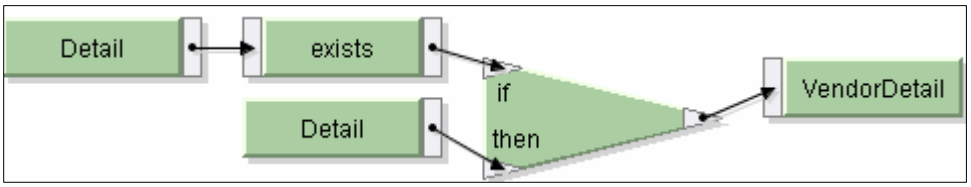
2.2. Exists

Use this function to determine whether a particular source field exists in the XML instance to be processed. If it does, exists() returns the value true, otherwise it returns the value false.

Example: Suppose VendorDetails node needs to be created only if Detail node exists, in this case you can use exists function. Here node VendorDetails has been mapped to node Detail with exists() function.

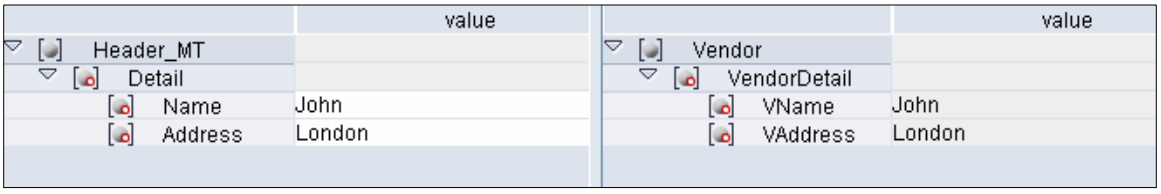
Mapping:

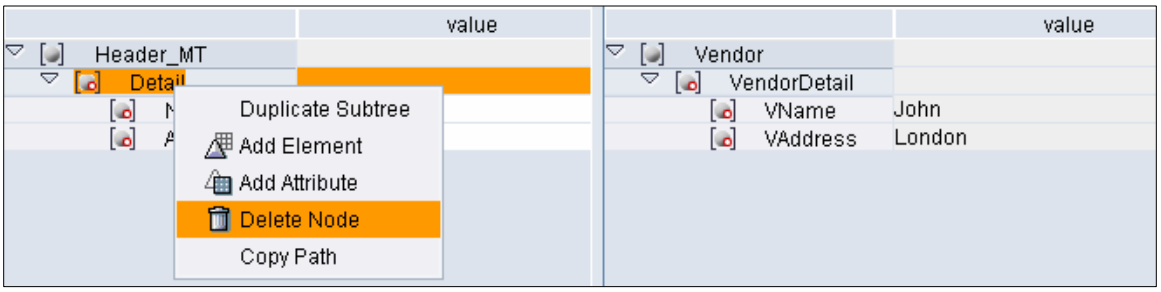


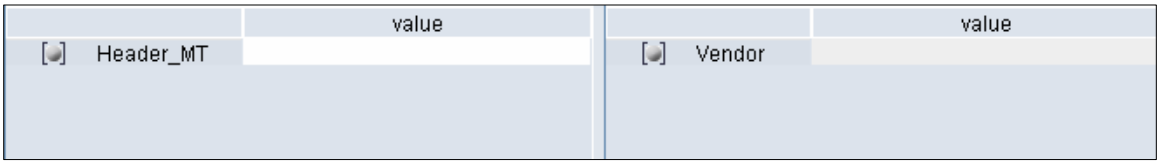


In the above mapping, exists will return TRUE if the Detail node exists in the source structure and the VendorDetail node at the target side will be created. This can be checked in below 2 test cases

Test:





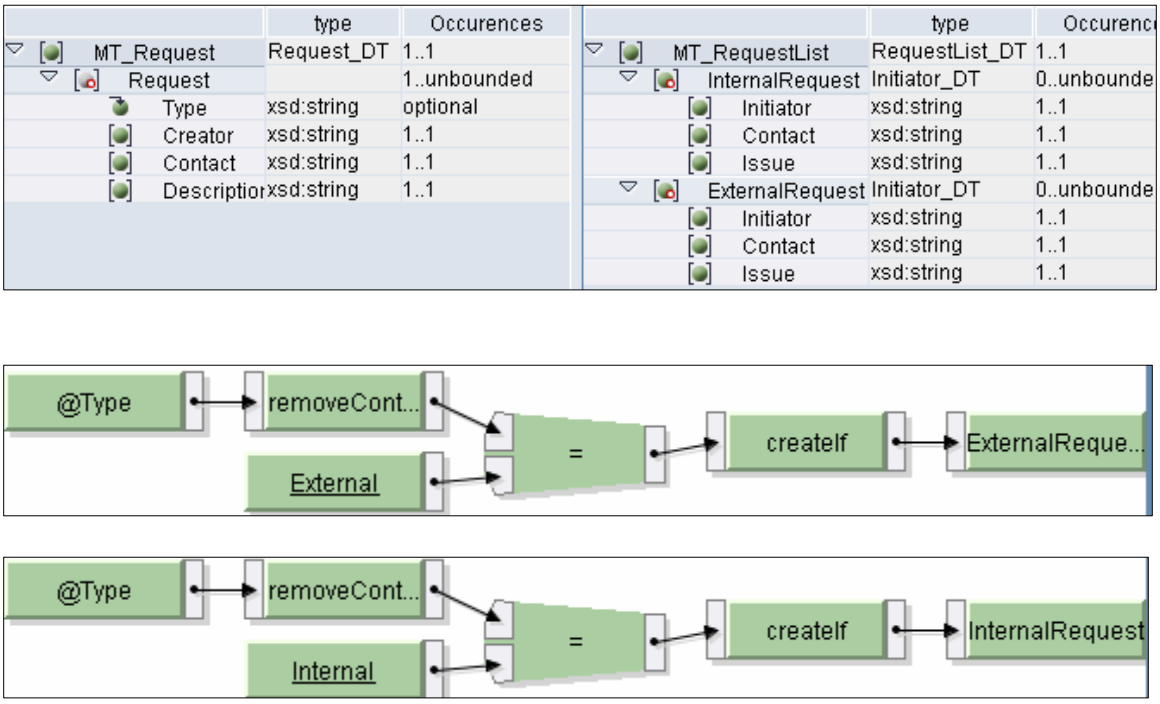


2.3. Create If

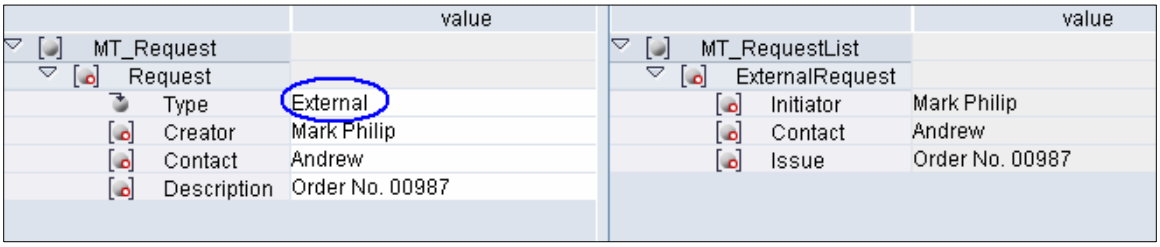
Use createIf() to create a tag in the target structure depending on the condition. Use createIf, if the nodes at the target side have to be created only after certain condition is true.

Here nodes InternalRequest and ExternalRequest are created depending on the value of Type attribute. InternalRequest node is created if value Type is 'Internal' and ExternalRequest node is created if value Type is 'External'.

Mapping:



Test:



	value
MT_Request	
Request	
Type	External
Creator	Mark Philip
Contact	Andrew
Description	Order No. 00987
Request	
Type	Internal
Creator	Michael
Contact	Daren
Description	Order No. 00854

	value
MT_RequestList	
InternalRequest	
Initiator	Michael
Contact	Daren
Issue	Order No. 00854
ExternalRequest	
Initiator	Mark Philip
Contact	Andrew
Issue	Order No. 00987

2.4. Remove Contexts

Use RemoveContexts to remove header contexts from source side.

If you do not want Header context to repeat in the target structure, use RemoveContexts. Here all the items under different headers are transformed to the separate target field item. RemoveContexts removes repeated header contexts and all the items come under same contexts.

Mapping:

	type	Occuren
Test_Out_Remove	Test_Out_Rem...	1..1
Header		0..unbound
Item	xsd:string	0..unbound

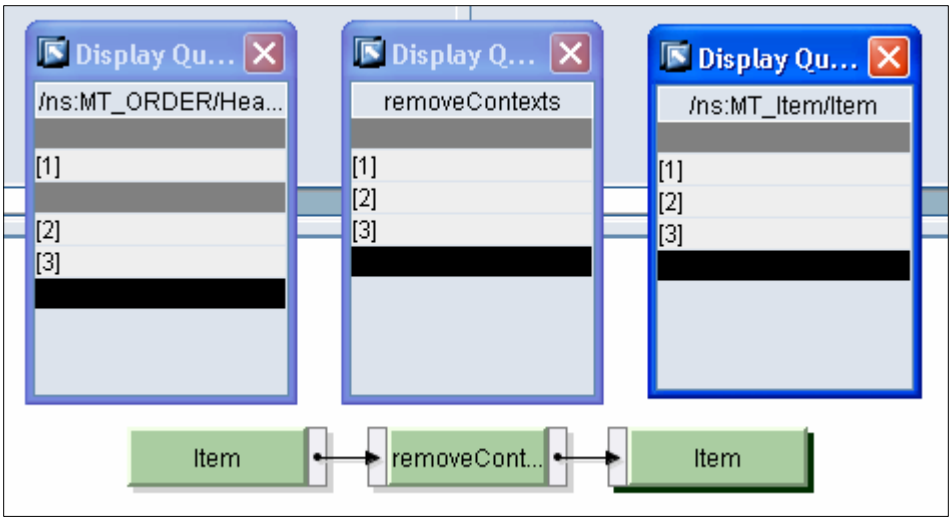
	type	Occurence
Test_In_Remove	Test_In_Remove	1..1
Item	xsd:string	0..unbounde

Test:

	value
Test_Out_Remove	
Header	
Item	1
Header	
Item	2
Item	3

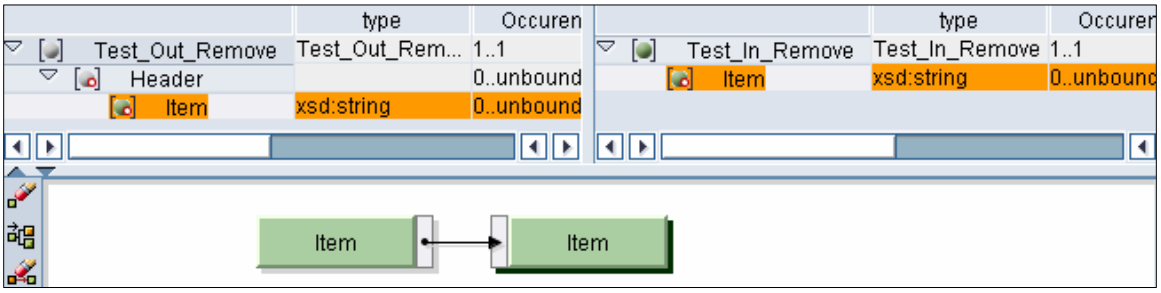
Test_In_Remove	
Item	1
Item	2
Item	3

In the figure below the first display queue displays Items in 2 different contexts. After execution of RemoveContexts function, the item values are displayed in the same contexts

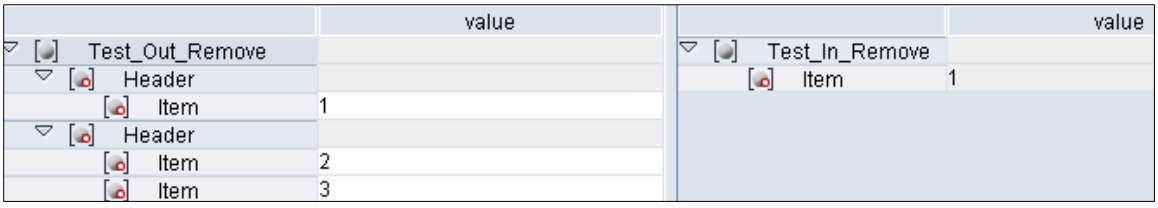


With Out RemoveContexts : If RemoveContexts is not used only Item field from the first header will appear in the target side because items are in different contexts.

Mapping:



Test:

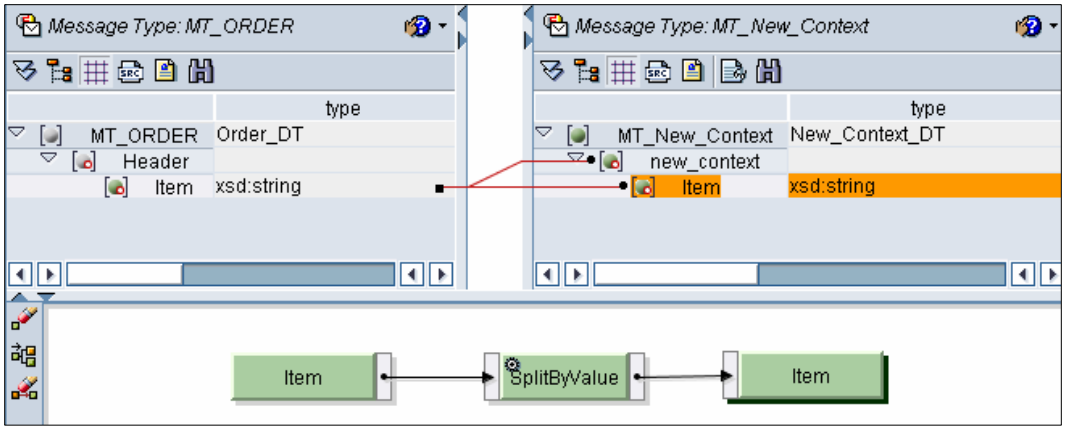


2.5. Split by value

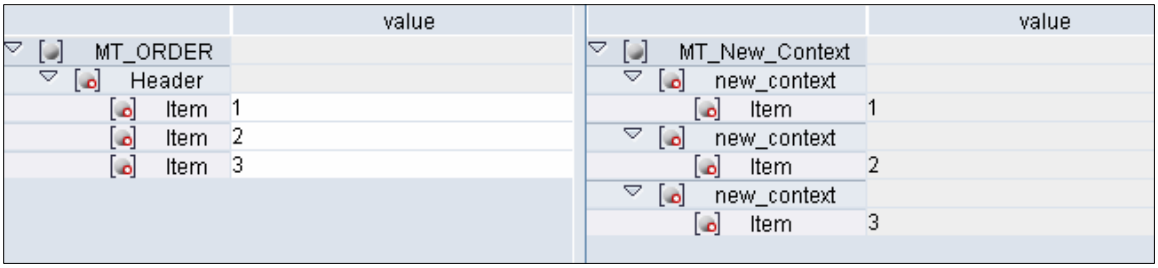
This is counter part of RemoveContexts. Use SplitByValue to add the header context at the target side. Suppose you want separate header contexts for every value of Item at source side use SplitByValue. Here all the values of Item from target side are transformed to the target side under separate new_context node.

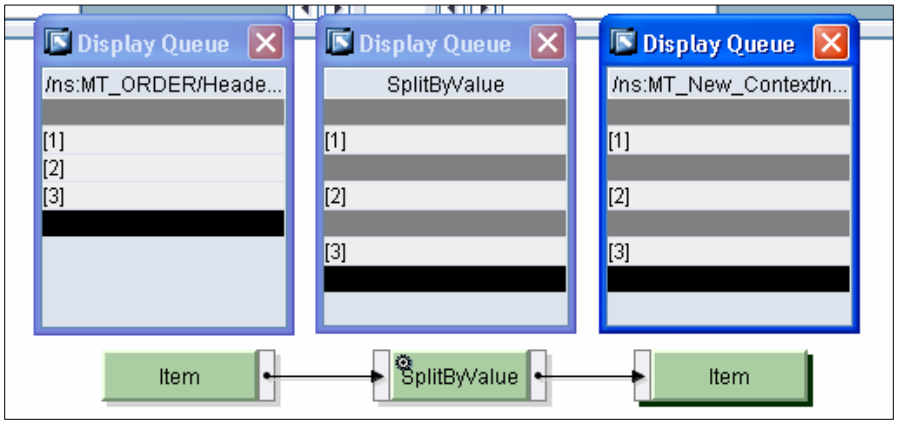
With SplitByValue:

Mapping:

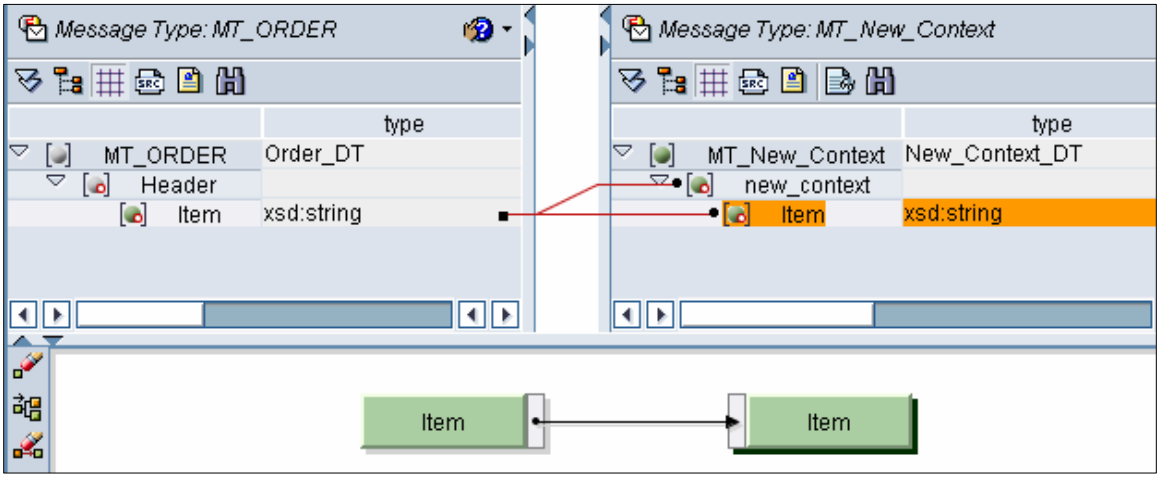


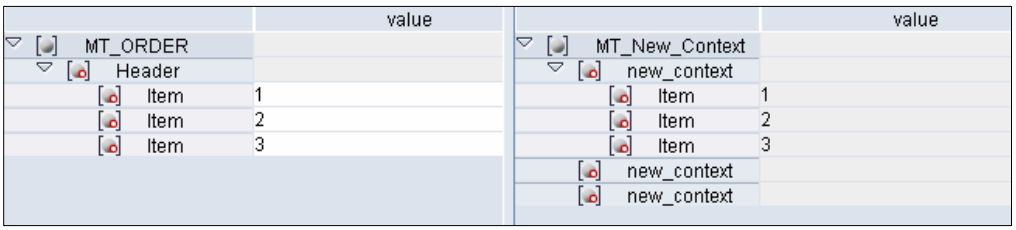
Testing:





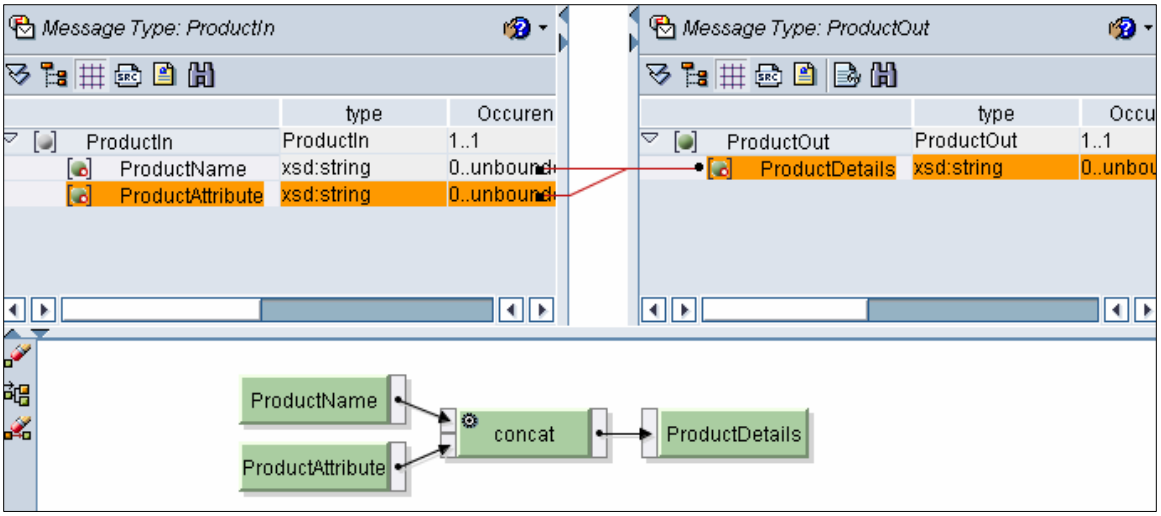
WithOut SplitByValue:

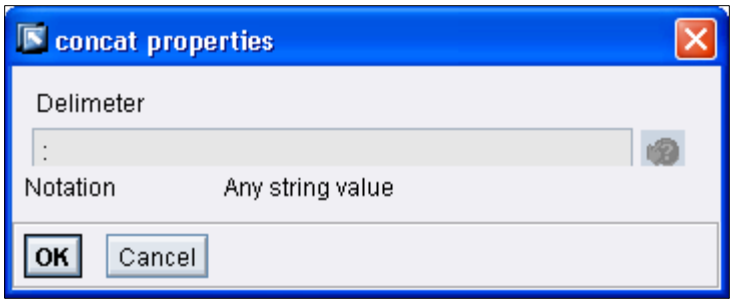




2.6. Concat

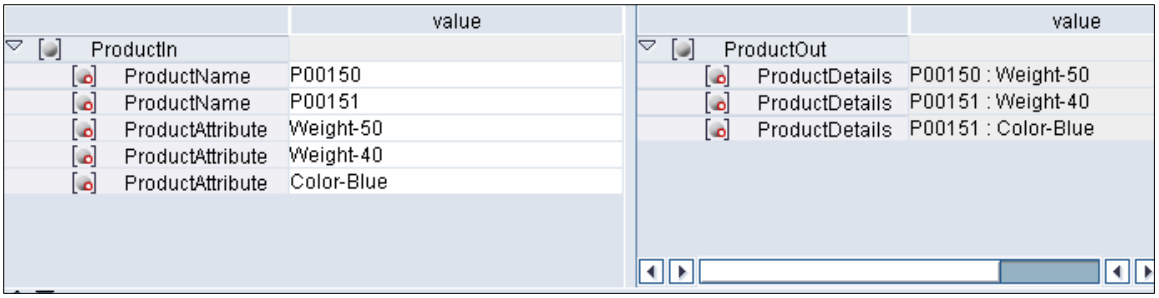
Use this function to concat 2 fields from input side.Here ProductName and ProductAttribute have been concatenated with the Delimeter ' : '





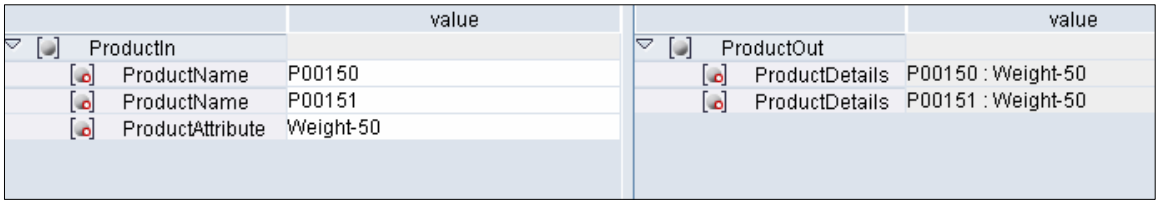
TEST:

Consider Multiple occurrence of ProductName (2) and ProductAttribute (3). If the occurrences are unequal, the last value of field having less number of occurrences is repeated.



	value
ProductIn	
ProductName	P00150
ProductName	P00151
ProductAttribute	Weight-50
ProductAttribute	Weight-40
ProductAttribute	Color-Blue

	value
ProductOut	
ProductDetails	P00150 : Weight-50
ProductDetails	P00151 : Weight-40
ProductDetails	P00151 : Color-Blue



	value
ProductIn	
ProductName	P00150
ProductName	P00151
ProductAttribute	Weight-50

	value
ProductOut	
ProductDetails	P00150 : Weight-50
ProductDetails	P00151 : Weight-50

3. SAP Transactions:

3.1. SAP R/3 Transactions:

WE31: Transaction for segment creation.

WE30: Transaction for IDoc creation.

WE20: Transaction for Partner Profile creation.

WE21: Transaction for Port creation.

SM59: Transaction for remote destination creation.

WE02: Transaction for IDoc list display.

WE05: Transaction for IDoc list display.

WE81: Transaction for message type creation

WE82: Transaction for Message Type and IDoc type linking.

3.2. SAP XI Transactions:

SXMB_MONI: Transaction for Integration Engine message monitoring

SXI_MONITOR: Transaction for Integration Engine message monitoring

IDX1: Transaction for Port maintenance in IDoc adapter.

IDX2: Transaction for MetaData overview for IDoc Aapter (Caching IDoc metadata)

IDX5: Transaction for XML message in adapter. (IDoc details in XI)

4. Notes

1. Once the IDoc is created in SAP R/3, it should be linked to message type by using transaction WE82. If this is not done the IDoc will not appear in the list while Imported Objects from SAP.
2. Whenever the IDoc scenario is executed, IDoc gets cached in XI system. So if the IDoc structure in SAP R/3 is change after this, the scenario may not work even after re-importing the IDoc as the old one is cached in XI. In this case you need to delete the cached IDoc structure using transaction IDX2.

Author Bio

Sagar and Chandra are SAP XI developers in Infosys technologies LTD, India. They have greatly contributed in design, development, testing phases of SAP XI projects in Infosys.