

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

Applies To:	1
Summary	1
Table of Contents	1
1. File to IDoc Packaging scenario	2
1.1. SLD	2
1.2. Integration Repository (IR)	2
1.3. Integration Directory (ID)	6
2. Mapping functions	11
2.1. Copy Value	11
2.2. Exists	12
2.3. Create If	14
2.4. Remove Contexts	15
2.5. Split by value	17
2.6. Concat	18
3. SAP Transactions:	20
3.1. SAP R/3 Transactions:	20
3.2. SAP XI Transactions:	20
4. Notes	20
Author Bio	20

1. File to IDoc Packaging scenario

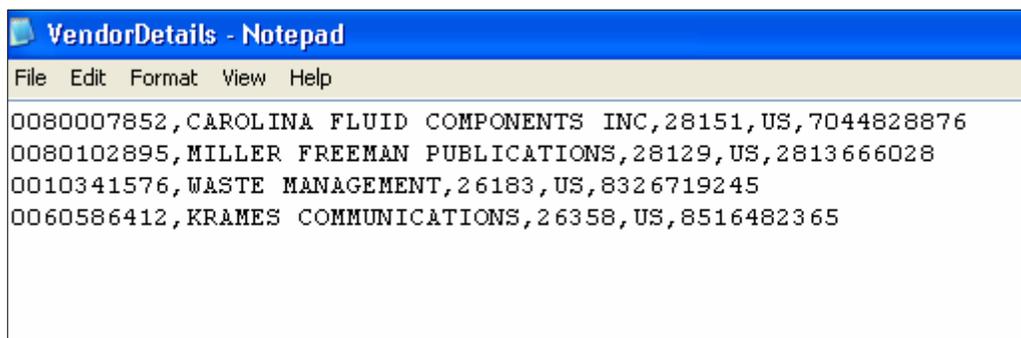
Posting single IDoc can be done by a simple Message Mapping between source message type and the imported IDoc structure in XI. But in case of multiple record-set from the source side it is not possible to create multiple IDocs instances in XI message with a the same imported IDoc.

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.



```
VendorDetails - Notepad
File Edit Format View Help
0080007852,CAROLINA FLUID COMPONENTS INC,28151,US,7044828876
0080102895,MILLER FREEMAN PUBLICATIONS,28129,US,2813666028
0010341576,WASTE MANAGEMENT,26183,US,8326719245
0060586412,KRAMES COMMUNICATIONS,26358,US,8516482365
```

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.

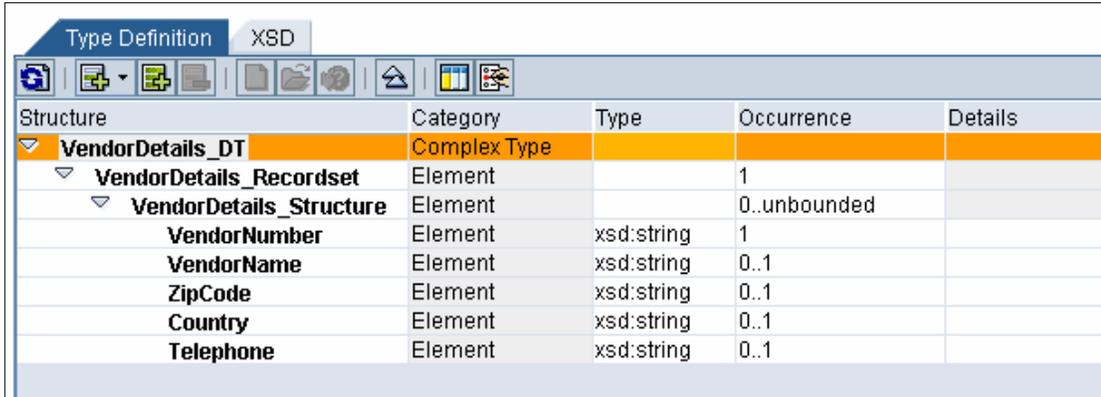
1.2. Integration Repository (IR)

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>.

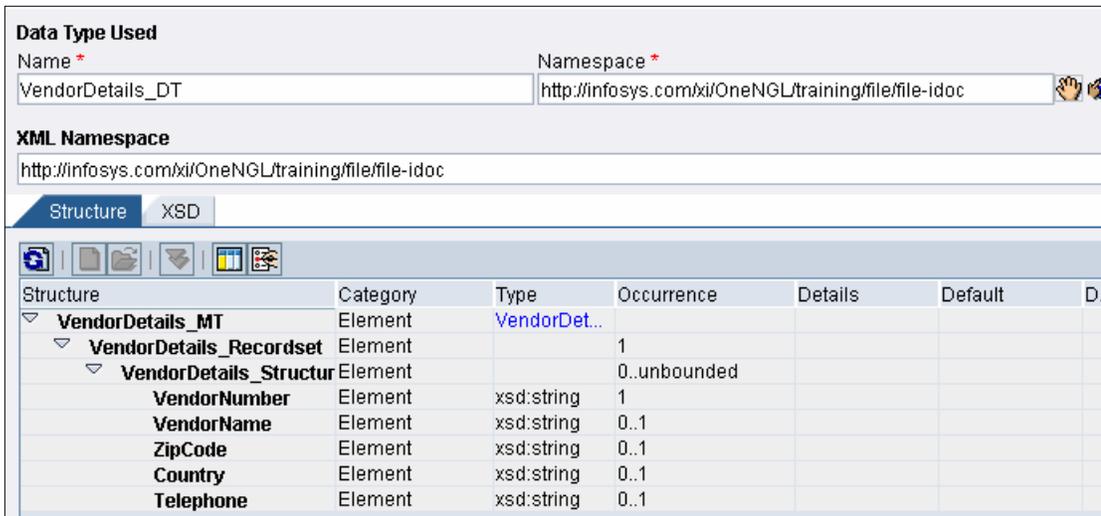
Interface Objects:

- 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.



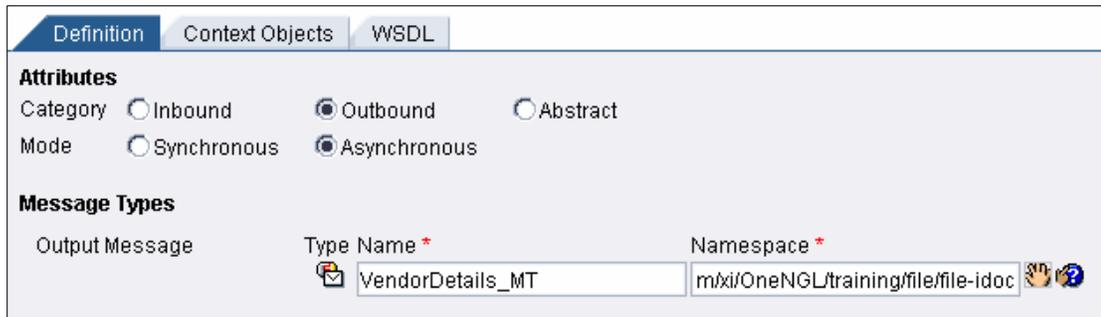
Data Type Used

Name * VendorDetails_DT Namespace * http://infosys.com/xi/OneNGL/training/file/file-idoc

XML Namespace
http://infosys.com/xi/OneNGL/training/file/file-idoc

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.



Definition Context Objects WSDL

Attributes

Category Inbound Outbound Abstract

Mode Synchronous Asynchronous

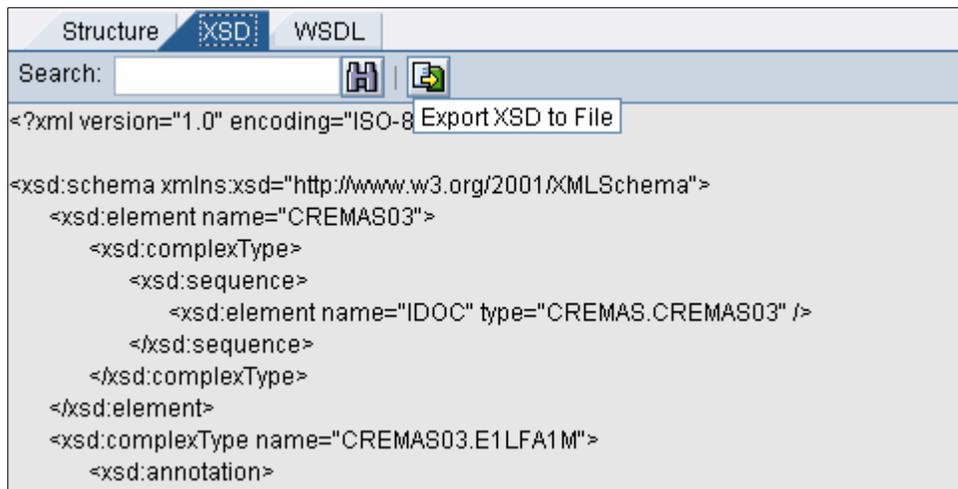
Message Types

Output Message

Type Name * Namespace *

 VendorDetails_MT m/xi/OneNGL/training/file/file-idoc  

- 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.



```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
  <xsd:element name="CREMAS03">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="IDOC" type="CREMAS.CREMAS03" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="CREMAS03.E1LFA1M">
    <xsd:annotation>
```

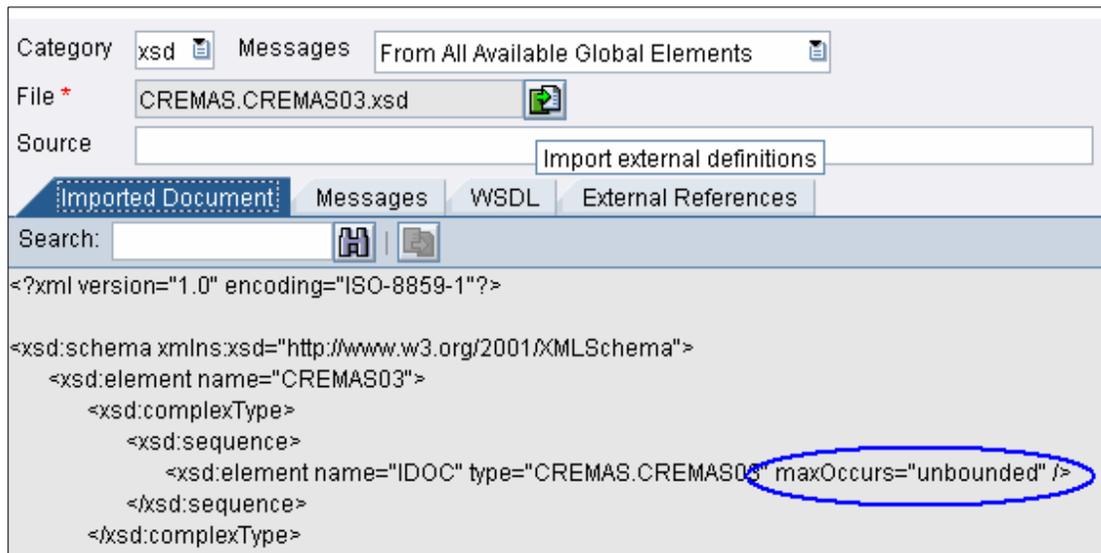
- 1.2.7. Changing the IDoc XSD:** Open the exported XSD file in edit mode and change
<xsd:element name="IDOC" type="CREMAS.CREMAS03" />

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

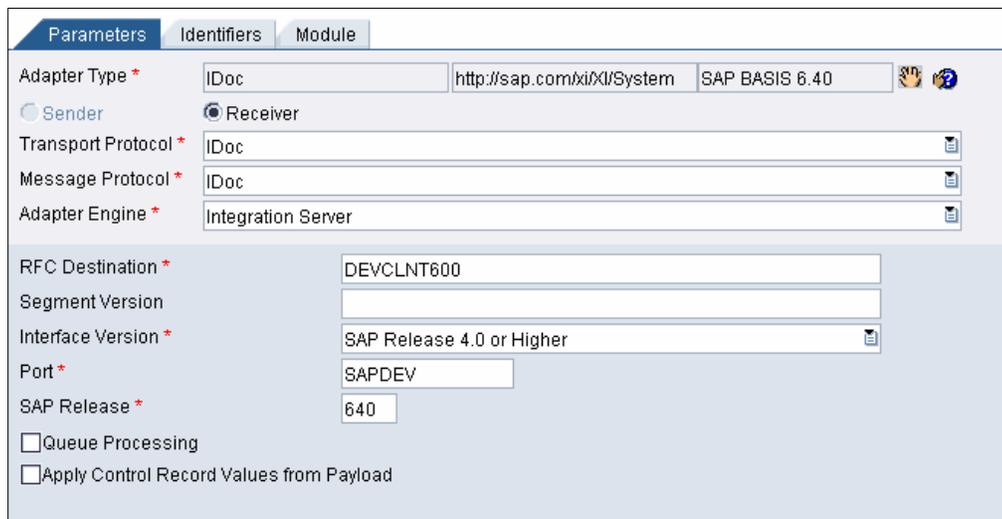
- 1.2.10. Create Interface Mapping:** In this step we will create Interface Mapping between 'VendorDetails_MI' and 'CREMAS.CREMAS03' IDoc.

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.



The screenshot shows the 'Parameters' tab of a configuration dialog. The 'Adapter Type' is set to 'IDoc' with a URL 'http://sap.com/xi/XI/System' and 'SAP BASIS 6.40'. The 'Receiver' radio button is selected. 'Transport Protocol' and 'Message Protocol' are both set to 'IDoc'. 'Adapter Engine' is 'Integration Server'. 'RFC Destination' is 'DEVCLNT600'. 'Interface Version' is 'SAP Release 4.0 or Higher'. 'Port' is 'SAPDEV' and 'SAP Release' is '640'. There are checkboxes for 'Queue Processing' and 'Apply Control Record Values from Payload', both of which are unchecked.

- 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.



The screenshot shows the 'Business Service' configuration dialog with the 'Sender' tab selected. Under 'Outbound Interfaces', there is a table with the following data:

Name	Namespace	Software Component Version
VendorDetails_MI	http://infosys.com/xi/OneNGL/training/fileBP_TRNG_SANDBOX_ONGL_XI	1.0 of

- 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



The screenshot shows a dialog box titled "Edit Adapter-Specific Identifiers". It contains three sections: "IDoc Adapter" with a "Logical System" field containing "BDSDEV001"; "IDoc Adapter and RFC Adapter" with "R/3 System ID" and "Client" fields; and "Marketplace Adapter" with a "DDID" field. At the bottom are "Apply" and "Cancel" buttons.

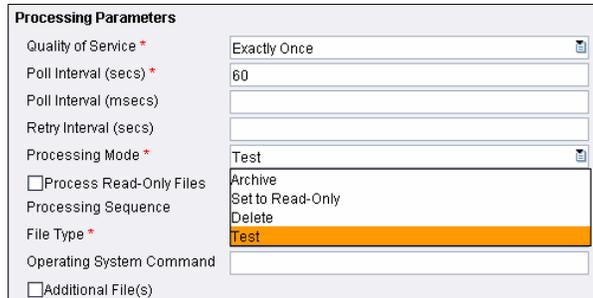
- 1.3.7. Create Communication Channel for sender:** We will create communication channel for this System. Right click on communication channel, click NEW. Give the name as FILE_Sender. Select the type as FILE Sender. Select Message protocol as File Content Conversion. This is necessary because the file 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.



The screenshot shows a configuration dialog for a communication channel. It includes fields for "Adapter Type" (File), "Transport Protocol" (File System (NFS)), "Message Protocol" (File Content Conversion), and "Adapter Engine" (Integration Server). The "File Access Parameters" section contains "Source Directory" (//blrkecsaptst2/xitraining/group02/SENDER) and "File Name" (VendorDetails.txt).

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



Processing Parameters	
Quality of Service *	Exactly Once
Poll Interval (secs) *	60
Poll Interval (msecs)	
Retry Interval (secs)	
Processing Mode *	Test
<input type="checkbox"/> Process Read-Only Files	Archive
Processing Sequence	Set to Read-Only
File Type *	Delete
Operating System Command	Test
<input type="checkbox"/> Additional File(s)	

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.

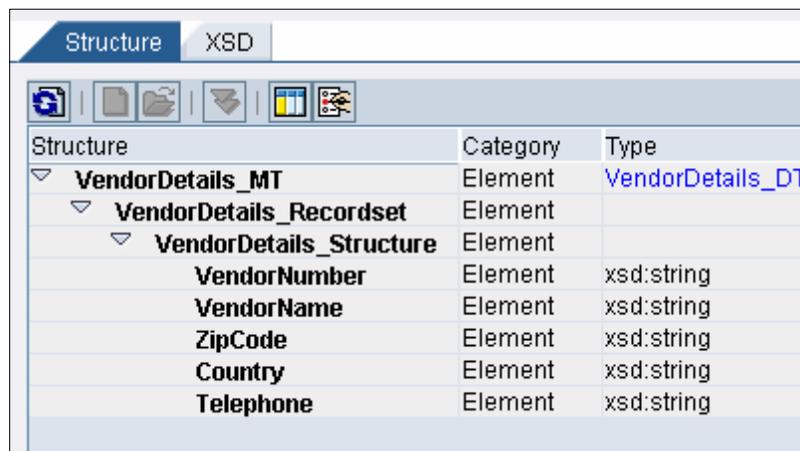
Poll Interval is number of seconds that the adapter must wait if no files are found for processing.

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** (“\n”-New Line) and **fieldnames** are given as shown.



Structure	Category	Type
VendorDetails_MT	Element	VendorDetails_DT
VendorDetails_Recordset	Element	
VendorDetails_Structure	Element	
VendorNumber	Element	xsd:string
VendorName	Element	xsd:string
ZipCode	Element	xsd:string
Country	Element	xsd:string
Telephone	Element	xsd:string

Content Conversion Parameters

Document Name: VendorDetails_MT
 Document Namespace: http://infosys.com/xi/OneNGL/training/file/file-idoc
 Document Offset:
 Recordset Name: VendorDetails_Recordset
 Recordset Namespace:
 Recordset Structure: VendorDetails_Structure,*
 Recordset Sequence: Ascending
 Recordsets per Message:
 Key Field Name:
 Key Field Type: String (Case-Sensitive)

Name	Value
VendorDetails_Structure.fieldSeparator	,
VendorDetails_Structure.endSeparator	'\n'
VendorDetails_Structure.fieldNames	VendorNumber,VendorName,ZipCode,C...

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.

Edit Receiver Determination Status: Being Processed

Sender

Party:
 Service: BDSDEV001
 Interface: VendorDetails_MI
 Namespace: http://infosys.com/xi/OneNGL/training/file/file-idoc

Receiver

Party: *
 Service: *
 Description:

Configured Receivers

Condition	Party	Service
		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

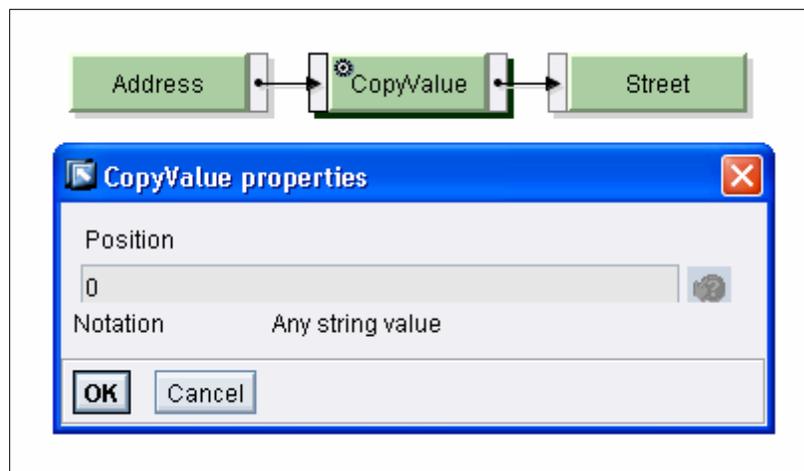
You can use **CopyValue** to copy the value of a position in the source structure and assign it to a target field for a frequently occurring source structure element. The value is copied each time the target field occurs in the target structure. However, it is copied a maximum of maxOccurs times

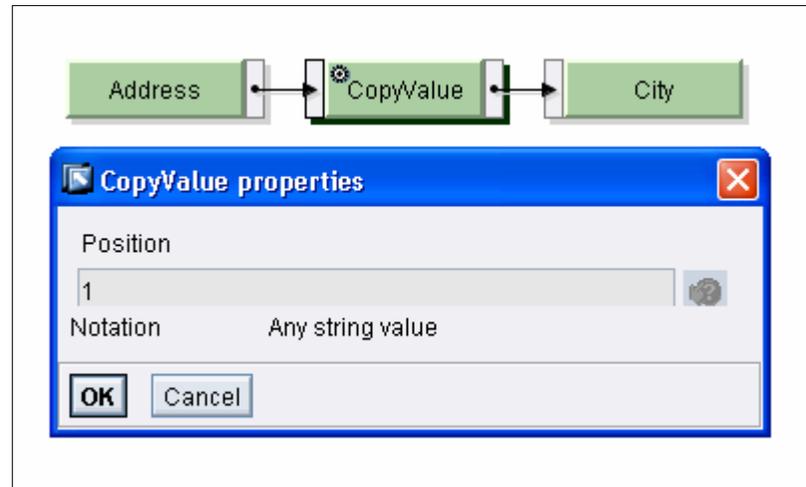
Mapping:

	type	Occurrences		type	Occurrences	
Employee_MT	Contract_DT	1..1		Vendor_MT	Buyer_DT	1..1
Employee		0..unbounded		Vendor		0..unbounded
Name	xsd:string	1..1		Name	xsd:string	1..1
Address	xsd:string	3..3		Street	xsd:string	1..1
				City	xsd:string	1..1
				Zipcode	xsd:string	1..1

	type	Occurrences		type	Occurrences	
Employee_MT	Contract_DT	1..1		Vendor_MT	Buyer_DT	1..1
Employee		0..unbounded		Vendor		0..unbounded
Name	xsd:string	1..1		Name	xsd:string	1..1
Address	xsd:string	3..3		Street	xsd:string	1..1
				City	xsd:string	1..1
				Zipcode	xsd:string	1..1

Red arrows indicate mapping from the source 'Address' field to the target 'Street', 'City', and 'Zipcode' fields.





Testing:

instance: Message Type: Employee_MT		Result:	
	value		value
Employee_MT		Vendor_MT	
Employee		Vendor	
Name	Peter John	Name	Peter John
Address	Street No. 30-A	Street	Street No. 30-A
Address	Washington	City	Washington
Address	511486	Zipcode	511486

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.

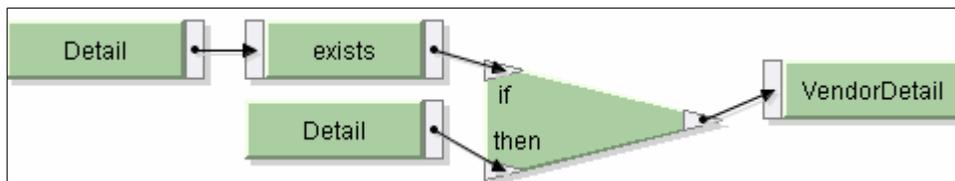
2.2. Exists

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:

	type	Occure		type	C
Header_MT	Header	1..1	Vendor	Vendor	1..1
Detail		0..unbound	VendorDetail		0..unbound
Name	xsd:string	1..1	VName	xsd:string	1..1
Address	xsd:string	1..1	VAddress	xsd:string	1..1



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:

	value		value
Header_MT		Vendor	
Detail		VendorDetail	
Name	John	VName	John
Address	London	VAddress	London

	value		value
Header_MT		Vendor	
Detail		VendorDetail	
Name		VName	John
Address		VAddress	London

A context menu is overlaid on the 'Detail' node in the source structure, with the 'Delete Node' option highlighted.

	value		value
Header_MT		Vendor	

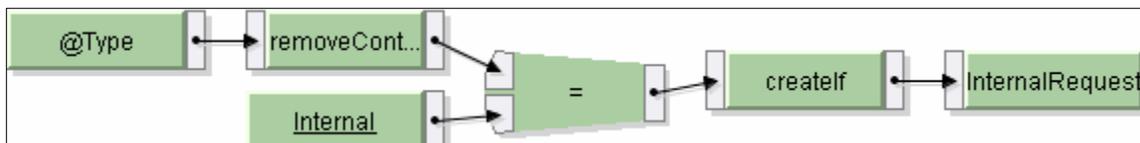
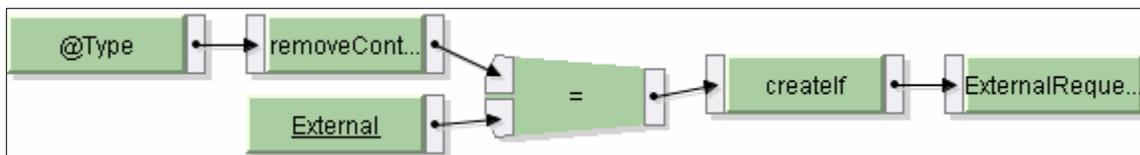
2.3. Create If

Use `createlf()` to create a tag in the target structure depending on the condition. Use `createlf`, 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:

	type	Occurrences		type	Occurrence
MT_Request	Request_DT	1..1		MT_RequestList	RequestList_DT 1..1
Request		1..unbounded		InternalRequest	Initiator_DT 0..unbounded
Type	xsd:string	optional		Initiator	xsd:string 1..1
Creator	xsd:string	1..1		Contact	xsd:string 1..1
Contact	xsd:string	1..1		Issue	xsd:string 1..1
Description	xsd:string	1..1		ExternalRequest	Initiator_DT 0..unbounded
				Initiator	xsd:string 1..1
				Contact	xsd:string 1..1
				Issue	xsd:string 1..1



Test:

	value		value
MT_Request		MT_RequestList	
Request		ExternalRequest	
Type	External	Initiator	Mark Philip
Creator	Mark Philip	Contact	Andrew
Contact	Andrew	Issue	Order No. 00987
Description	Order No. 00987		

		value			value
MT_Request	Request	Type: External	MT_RequestList	InternalRequest	Initiator: Michael
		Creator: Mark Philip			Contact: Daren
		Contact: Andrew			Issue: Order No. 00854
		Description: Order No. 00987		ExternalRequest	Initiator: Mark Philip
MT_Request	Request	Type: Internal			Contact: Andrew
		Creator: Michael			Issue: Order No. 00987
		Contact: Daren			
		Description: Order No. 00854			

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:

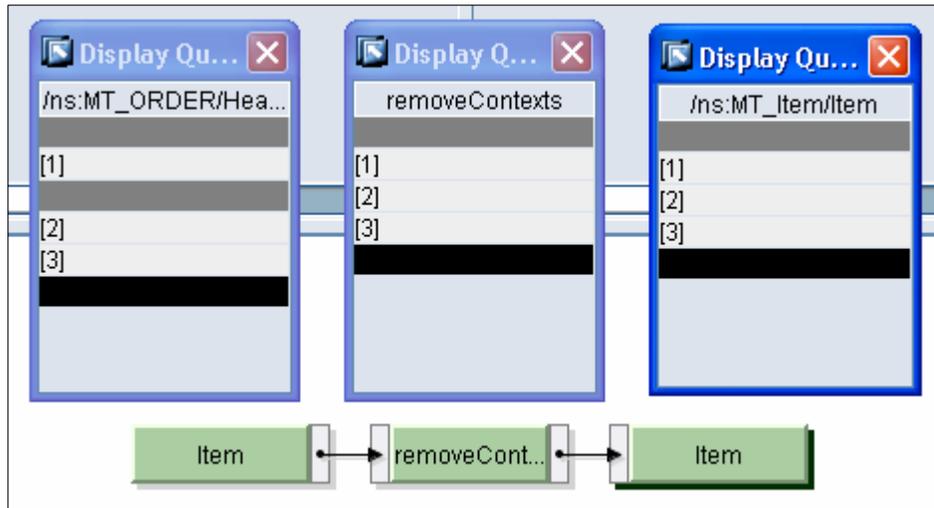
	type	Occuren		type	Occurrence
Test_Out_Remove	Test_Out_Rem...	1..1	Test_In_Remove	Test_In_Remove	1..1
Header		0..unbound	Item	xsd:string	0..unbound
Item	xsd:string	0..unbound			



Test:

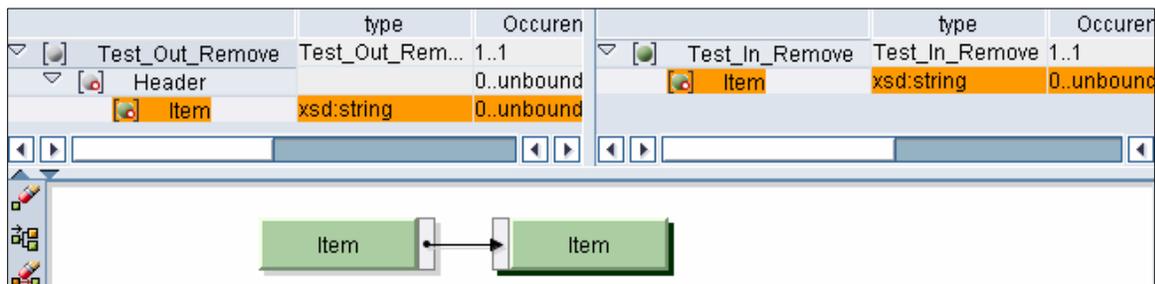
		value			value
Test_Out_Remove	Header		Test_In_Remove	Item	1
	Item	1		Item	2
Test_Out_Remove	Header			Item	3
	Item	2			
	Item	3			

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:

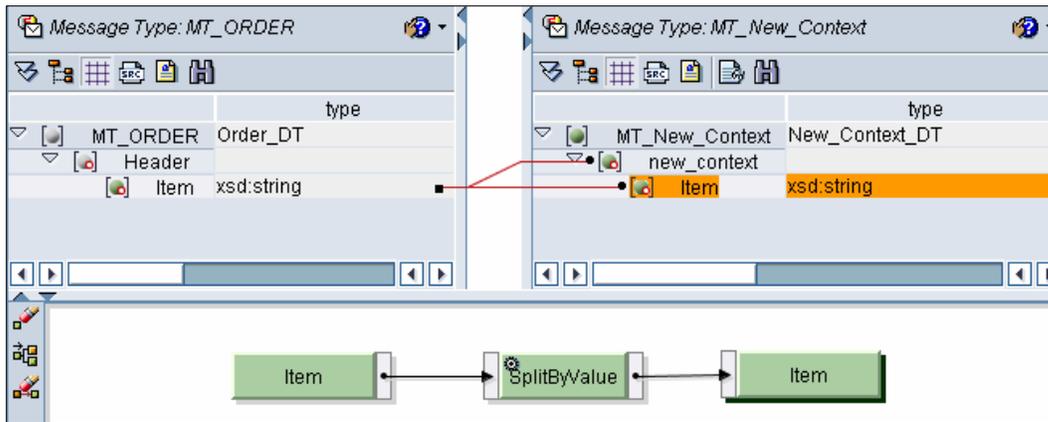
	value		value
Test_Out_Remove		Test_In_Remove	
Header		Item	1
Item	1		
Header			
Item	2		
Item	3		

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.

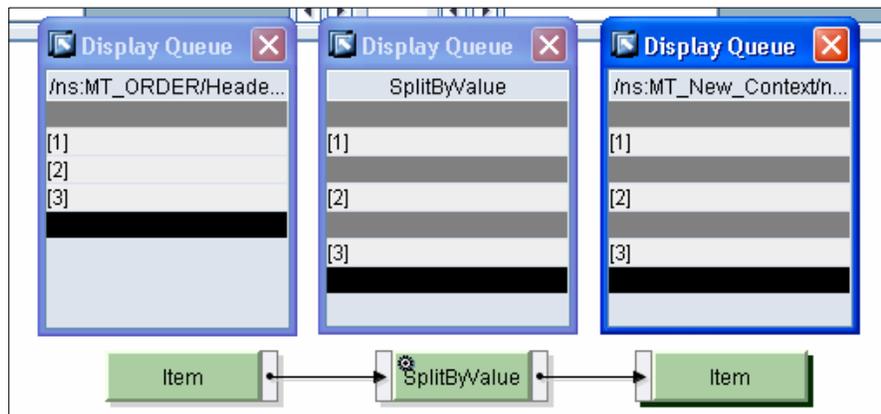
With SplitByValue:

Mapping:

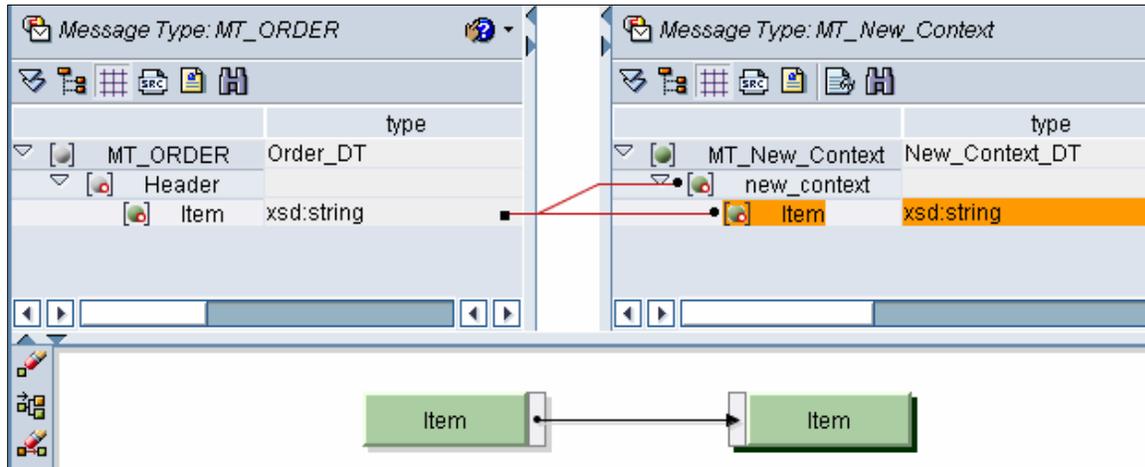


Testing:

value		value	
MT_ORDER		MT_New_Context	
Header		new_context	
Item	1	Item	1
Item	2	new_context	2
Item	3	Item	2
		new_context	3
		Item	3



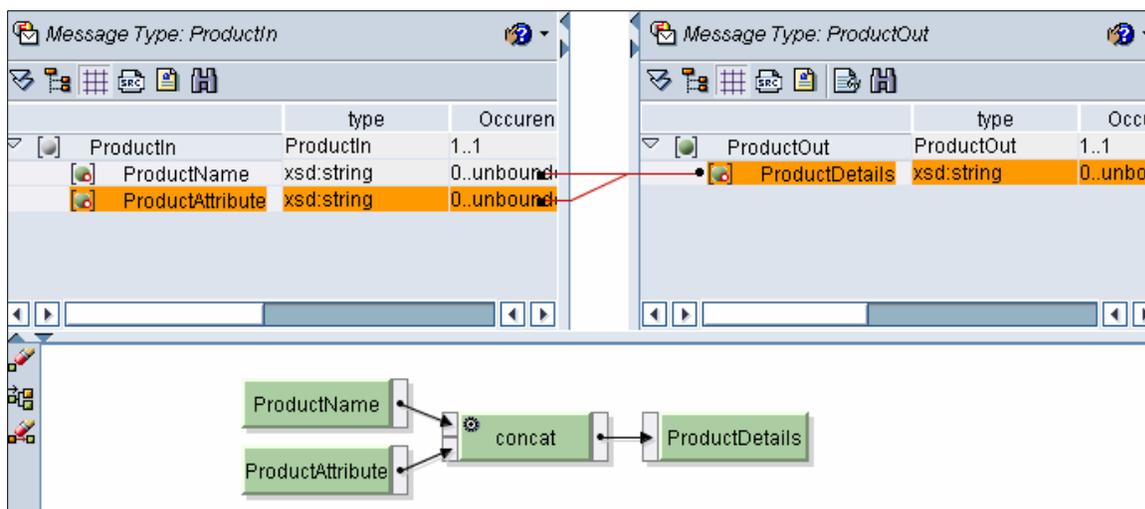
Without SplitByValue:



	value		value
MT_ORDER		MT_New_Context	
Header		new_context	
Item	1	Item	1
Item	2	Item	2
Item	3	Item	3
		new_context	
		new_context	

2.6. Concat

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





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

	value		value
ProductIn		ProductOut	
ProductName	P00150	ProductDetails	P00150 : Weight-50
ProductName	P00151	ProductDetails	P00151 : Weight-50
ProductAttribute	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 Adapter (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.