

SDN Community Contribution

(This is not an official SAP document.)

Disclaimer & Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.

Table of Contents

Table of Contents	2
Applies To:	3
Summary	3
Introduction	4
The Scenario.....	4
Process Flow: Overview.....	5
Prerequisites	5
Contivo.....	5
XI.....	5
SAP	5
Interface Objects.....	6
Create an External Definition.....	6
The Message Interface	12
Mapping Objects.....	16
Create an Imported Archive.....	16
Interface Mapping.....	20
Testing the Interface Mapping.....	28
Author Bio	34

Applies To:

SAP Exchange Infrastructure v 3.0, SP12, Integration Repository.

Summary

This document details step by step procedures for designing a standalone inbound EDI X12 820 Payment Advice in the XI Integration Repository using an imported XSD Schema and Java Mapping Program. The Mapping Program used in this example is generated in Contivo Analyst, a powerful and easy to use mapping tool that builds java classes specifically formatted for import into XI.

The XSD schema used as the source interface is imported from Contivo, which pulls the standard from EDIFICS SpecBuilder, an electronic EDI standards library that also provides an impressive environment for generating classic EDI specifications.

This nicely underlines the fact that development of a best of breed EDI solution involves the use of a number of complementary technologies each of which contributes its strengths to the overall effort.

These procedures can be adapted to any standalone inbound EDI transaction that does not require custom processing best handled by an XI Business Process Model.

By: Emmanuel Hadzipetros

Company: NBC Universal

Date: August 28, 2005

Introduction

These procedures describe the steps involved in designing a standalone inbound EDI transaction in SAP XI using an imported source interface in XSD format and a Java mapping program pulled and archived from Contivo Analyst. Inbound in this context means into SAP.

The purpose of this document is to provide clear procedures for a consistent and repeatable process for building standalone XI interfaces that do not require custom processes.

XI development object naming conventions are implied in the descriptions presented in this document but these should not be taken as recommendations for an approach to naming development objects in XI. Nor should the description of a standalone EDI transaction design, and the tools used to support it, be taken as recommending a particular EDI architecture, design or set of implementation tools.

These kinds of issues need to be determined on a case-by-case basis.

For a structured approach to creating an Integration Directory Scenario with Party to bring the 820 into SAP, please see my companion document, *How to Set up an XI Integration Directory Scenario with Party*, at:

<https://www.sdn.sap.com/irj/servlet/prt/portal/prtroot/com.sap.km.cm.docs/library/xi/xi-how-to-guides/How%20to%20Set%20up%20an%20XI%20Integration%20Directory%20Scenario%20with%20Party%20to%20Support%20EDI%20Partner%20Processing.pdf>.

The Scenario

Your employer has found a lucrative sideline in developing and distributing computer games that are sold through big box retailers such as Wal-Mart. Like most Wal-Mart suppliers, this business relationship is conducted entirely through EDI. Your backend business system is SAP.

It's no secret that Wal-Mart is one of the most important EDI players in the world and any company that's serious about doing business with them, follows their lead when it comes to EDI.

But as Wal-Mart EDI relationships go, your computer game business is a fairly simple one. Wal-Mart periodically sends you a Purchase Order in an X12 850 transaction which posts to SAP as a Sales Order. You respond with an Acknowledgement from the Order in an outbound 855.

When you're ready to ship, you send Wal-Mart an Advanced Ship Notification in an outbound 856 from the SAP Delivery Document. When Wal-Mart receives the shipment they send back a Shipping Confirmation in an inbound 856 that updates your Delivery Document and posts a Goods Issue to update inventory and accounting. You then send back your favorite document: an Invoice in an outbound 810.

Wal-Mart then deposits a payment on the Invoice to your bank in a timely manner and sends you an 820 Payment Advice, containing detailed information about the payment, including all deductions with reason codes. This posts a Payment Advice document in SAP which is then cleared by a standard program that updates all the appropriate accounts.

This is the beautifully balanced business eco-system that pays your bills and returns value to your shareholders, the context of the EDI interfaces we configure in SAP and design in the XI Integration Repository. It's important to understand this context: our inbound 820 is the end point of a process that began when our customer decided to buy something from us.

This illustrates an important point about EDI: while the technical component is necessary to facilitate it, EDI is all about communications and trade ... buying and selling ... maintaining a business relationship.

Process Flow: Overview

The XI EDI development is set up in namespace urn:X12_820_to_pexr2002:us:in.

A Source Interface is created in XI for the X12 820 Payment Advice from an XSD schema imported from Contivo, which was brought into Contivo from the EDIFECs SpecBuilder Standards Library in gXML format, stripped of all qualifiers. gXML is EDIFECs proprietary EDI Standards XML format.

This is critical: if qualifiers are included, the XSD is over 70 MB and XI chokes. Without qualifiers, the XSD is only 223 KB.

The XSD is brought into XI as an External Definition and associated with a Message Interface.

The target interface is IDoc REMADV.PEXR2002, imported from SAP at the Software Component level and available to all namespaces within that component. The IDoc is already recognized as an interface in XI so we do not create a Message Interface for the IDoc.

The Contivo map is brought into XI as an Imported Archive. One map is imported for each Trading Partner-EDI Transaction. The Contivo mapping program is then used in a Message Mapping to link Source and Target Interfaces to the Contivo map.

For purposes of clarity, all XI screens will be undocked in these procedures.

Prerequisites

Contivo

The X12 820 Source Interface, stripped of qualifiers, is exported in XSD format to the local desktop.

An XI mapping program is generated in Contivo Analyst in two java files, one with a .java and the other a .class extension. These must be archived in either a zip or a java jar file. The following archives also need to be present in XI at every Namespace level:

contivo_class_library.jar
contivo_library.jar
contivologging.zip

These contain supporting classes and files that must be present in XI at the Namespace level: they provide the Java Runtime Environment for execution of the Contivo mapping programs.

XI

IDoc REMADV.PEXR2002 is imported at the Software Component Version level, a valid Namespace is present and an 820 X12 XML input file for testing the Interface Mapping is saved on the local desktop.

SAP

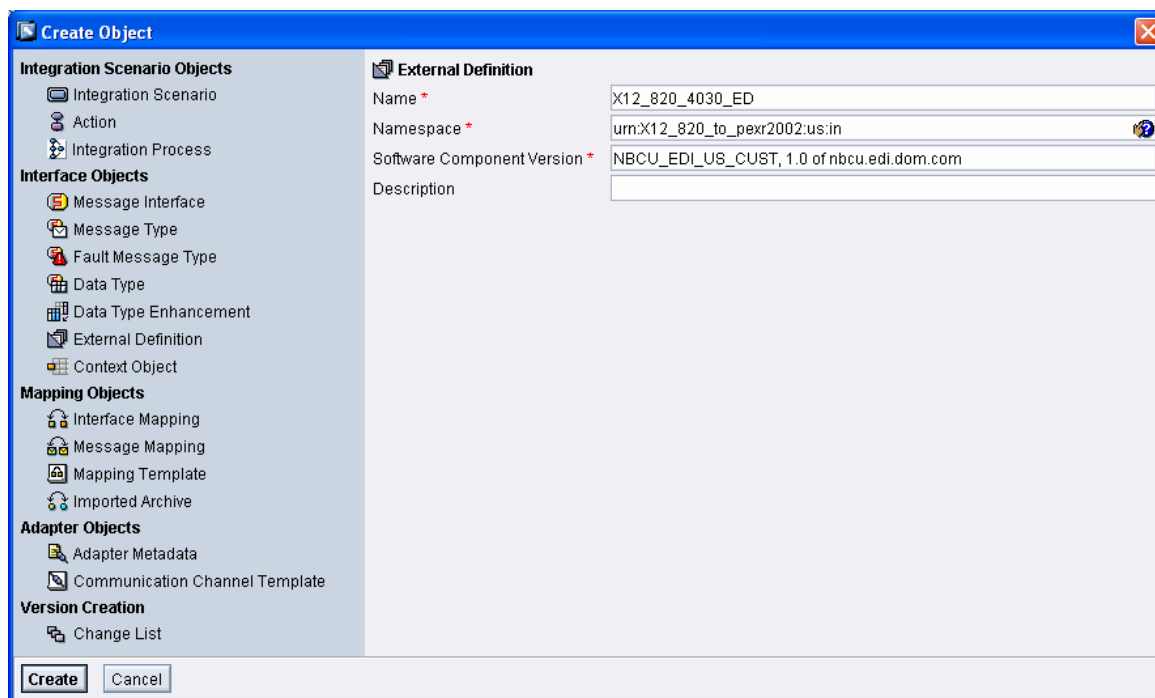
Inbound Partner Profile for Partner Type KU (Customer) for the Walmart Sold-to partner for Message Type REMADV with process code REMC, as well as supporting master and transactional data and configuration.

Interface Objects

Create an External Definition

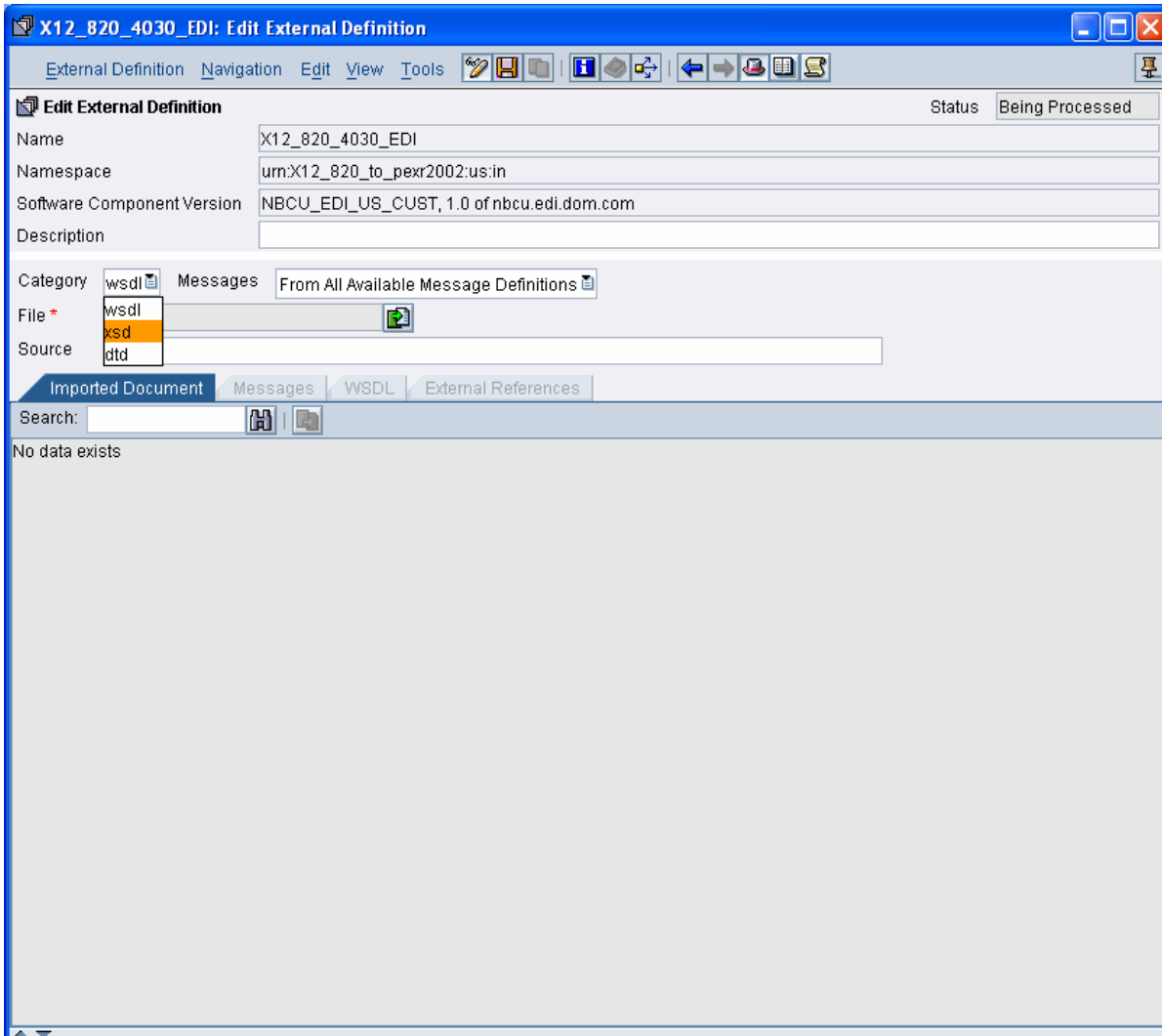
The External Definition stores the metadata for the Source interface in XSD format, in this case the X12 820 version 4030, providing the structure for the X12 820 Message Interface used in the XI transformation. The structure of the 820 must correspond exactly to the structure of the 820 used as the source interface in the Contivo map.

1. Right-click on External Definitions in the Navigation Pane and left-click New.
2. The Create External Definition dialog opens.
3. Enter X12_820_4030_ED in the Name field. This describes an External Definition for an X12 820 Payment Advice version 4030.
4. The External Definition can be used for any other Trading Partner that uses version 4030 of the 820.
5. The Namespace defaults to urn:X12_820_to_pexr2002:us:in.
6. Click Create.

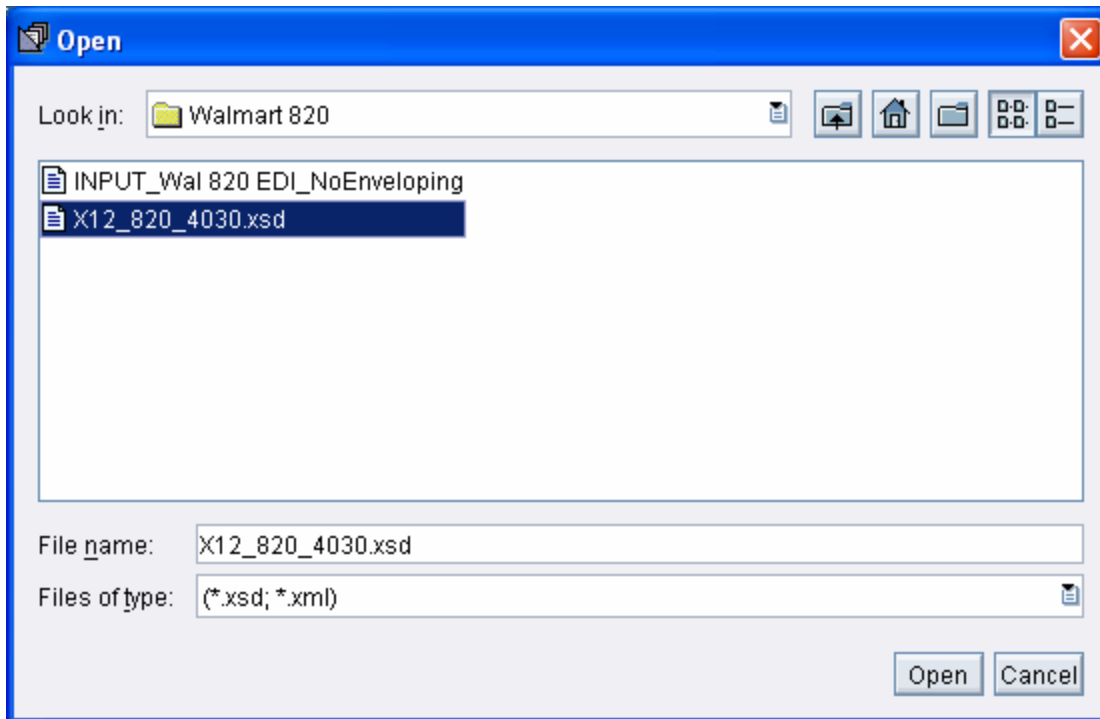


How to Build a Basic EDI Interface Using an Imported Schema and Map

7. The Edit External Definition screen opens.
8. From the Category drop down list, select xsd. The message name will be selected from the internal message definitions within the schema file.
9. Click the right-pointing green arrow next to the File field.

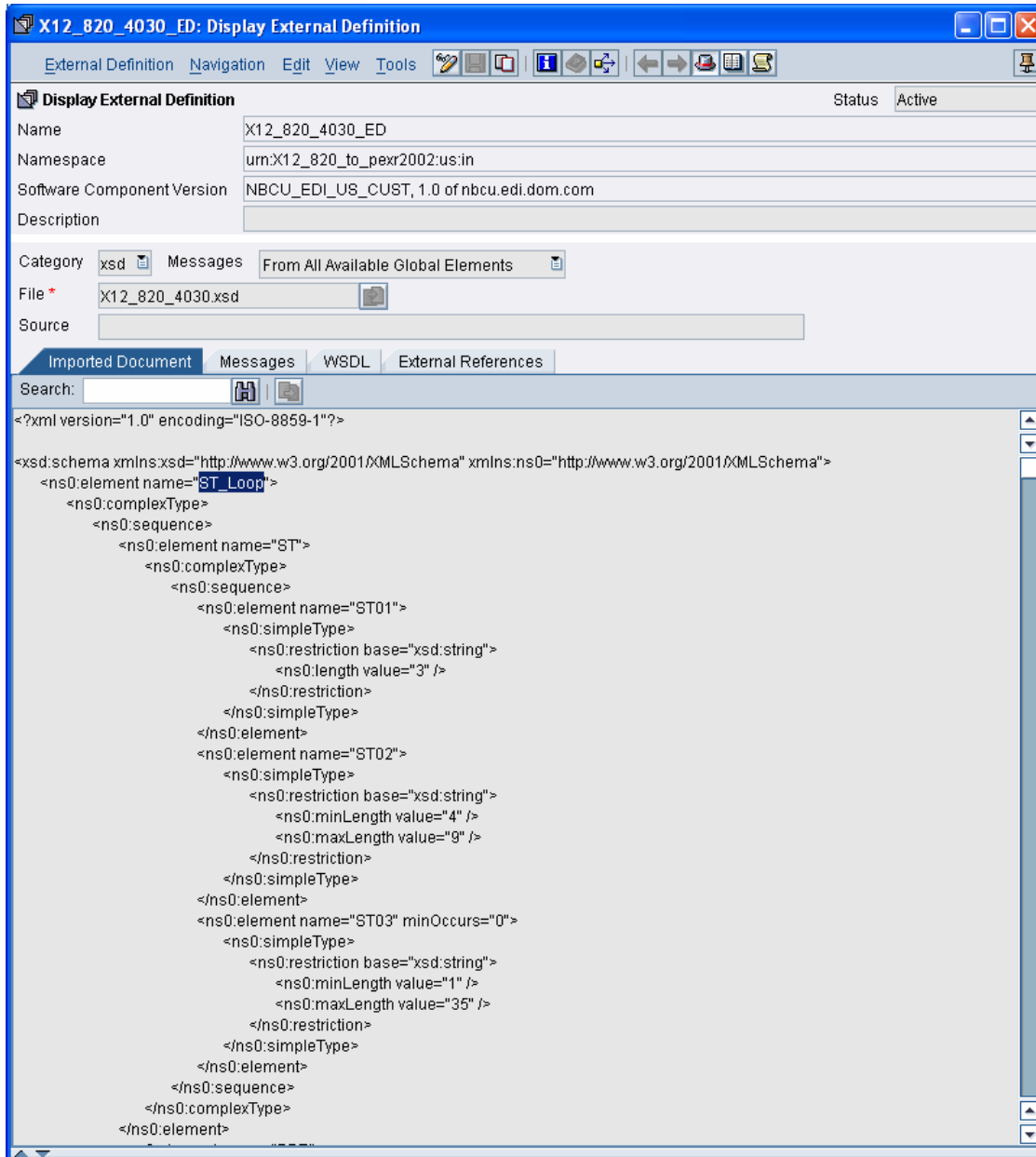


10. A file navigation window opens. Navigate to your desktop directory where the XSD file for the X12 820 is stored.
11. Select the file name and click Open.

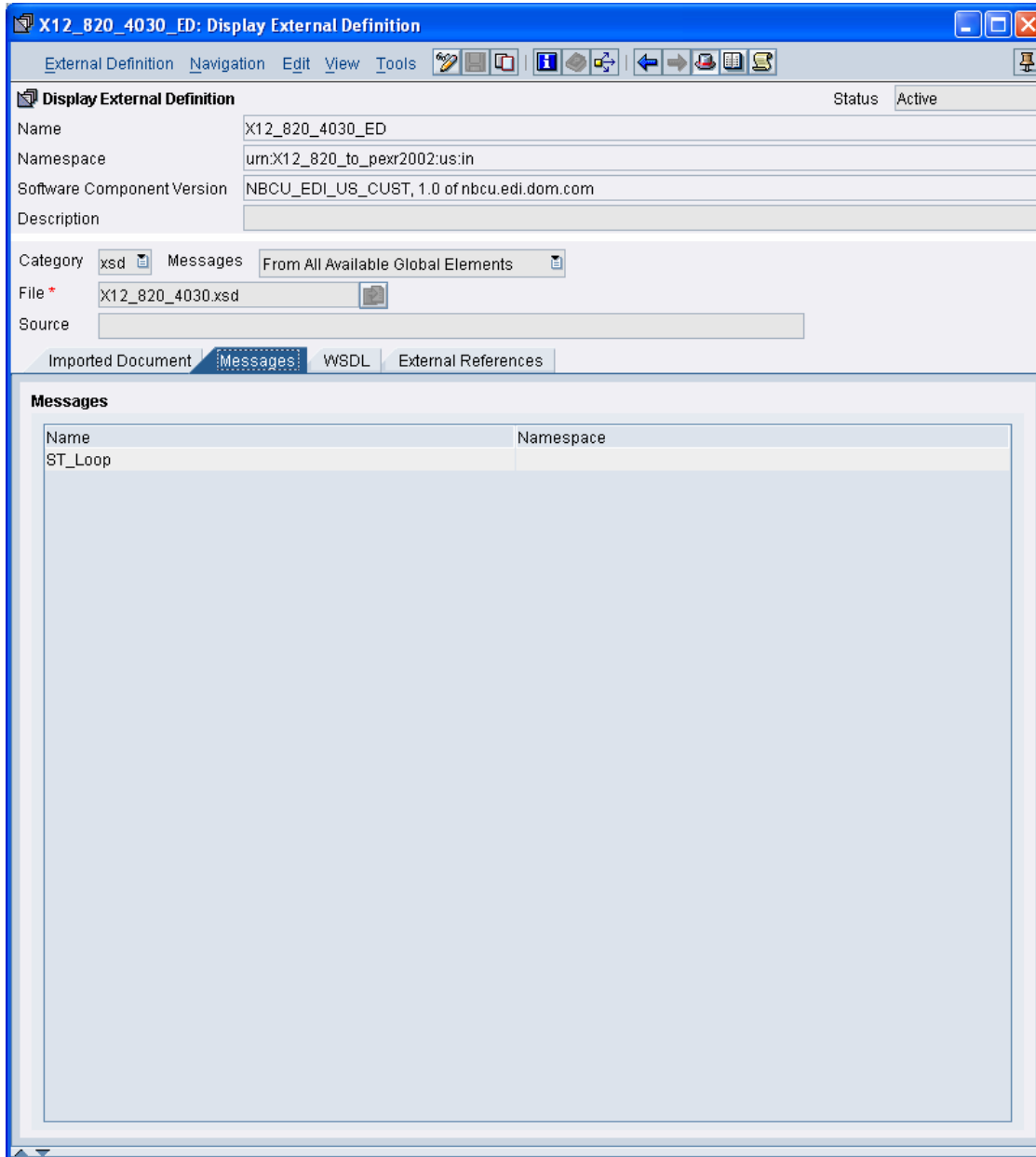


How to Build a Basic EDI Interface Using an Imported Schema and Map

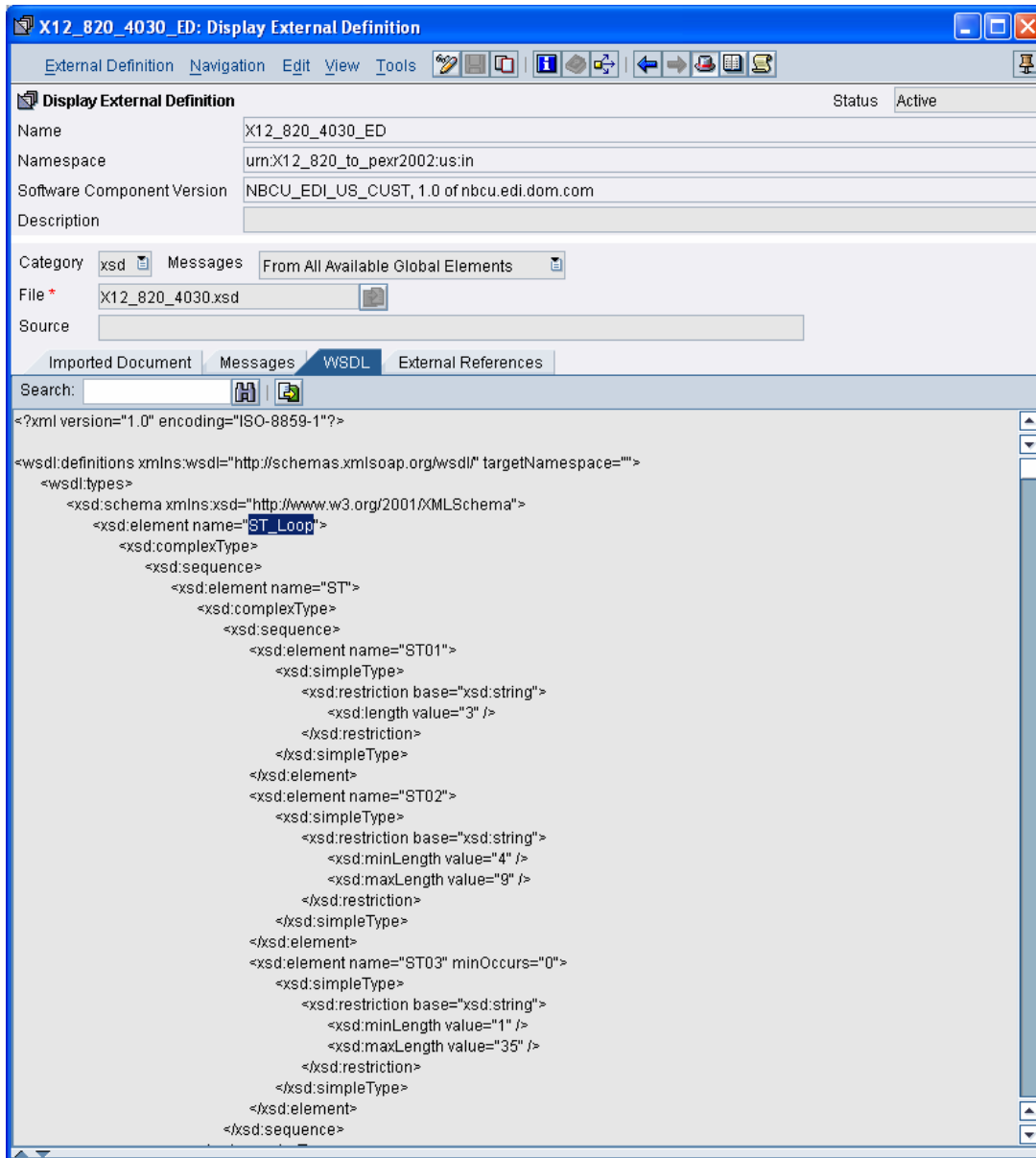
12. The XSD is loaded into the Imported Document tab of the Edit External Definitions screen. The element name – ST_LOOP – is just below the XSD schema xmlns reference at the top of the file.
13. Click Save to create the Object. Activate the External Definition.



- Click on the Messages tab. Note the Name of the listed messages. ST_LOOP will be used to identify the X12 820 in the Message Interface and Message Mapping. It is defined from the Global Element at the top of the XSD Schema file.



15. Click on the WSDL tab. XI has put a WSDL wrapper around the XSD schema under the hood.
16. This WSDL message is used in XI's own internal processing and message tracking and is also available for use in SOAP exchanges with SOAP aware systems.

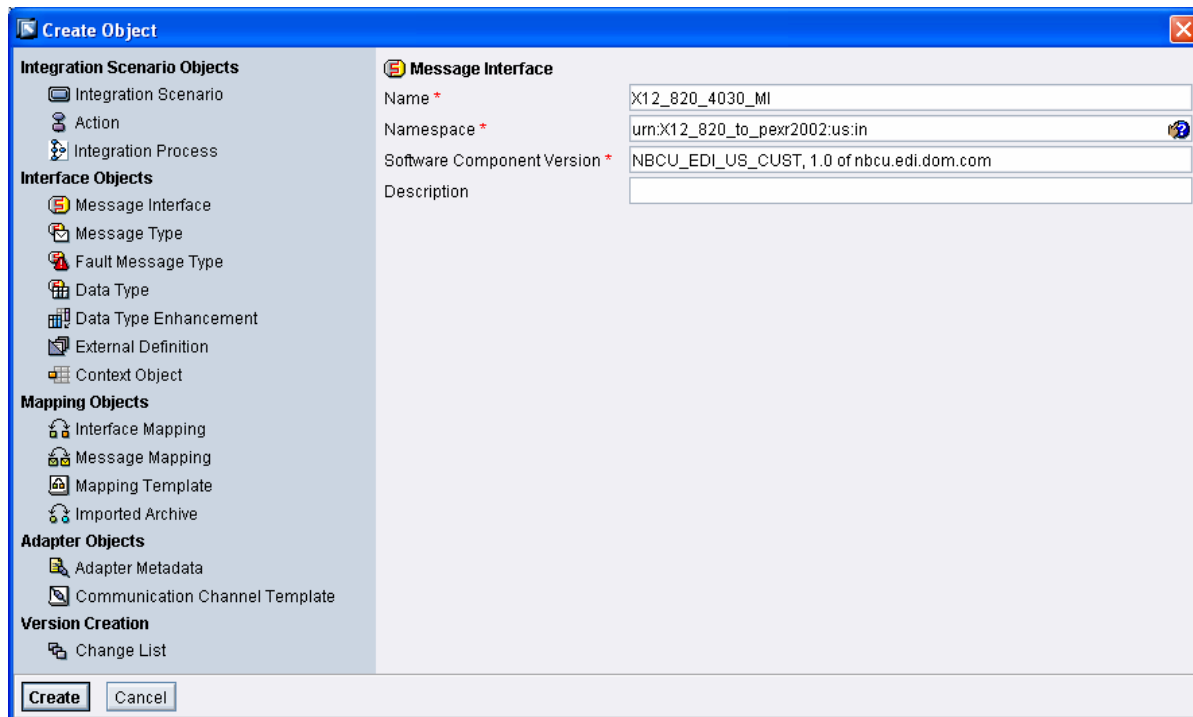


The Message Interface

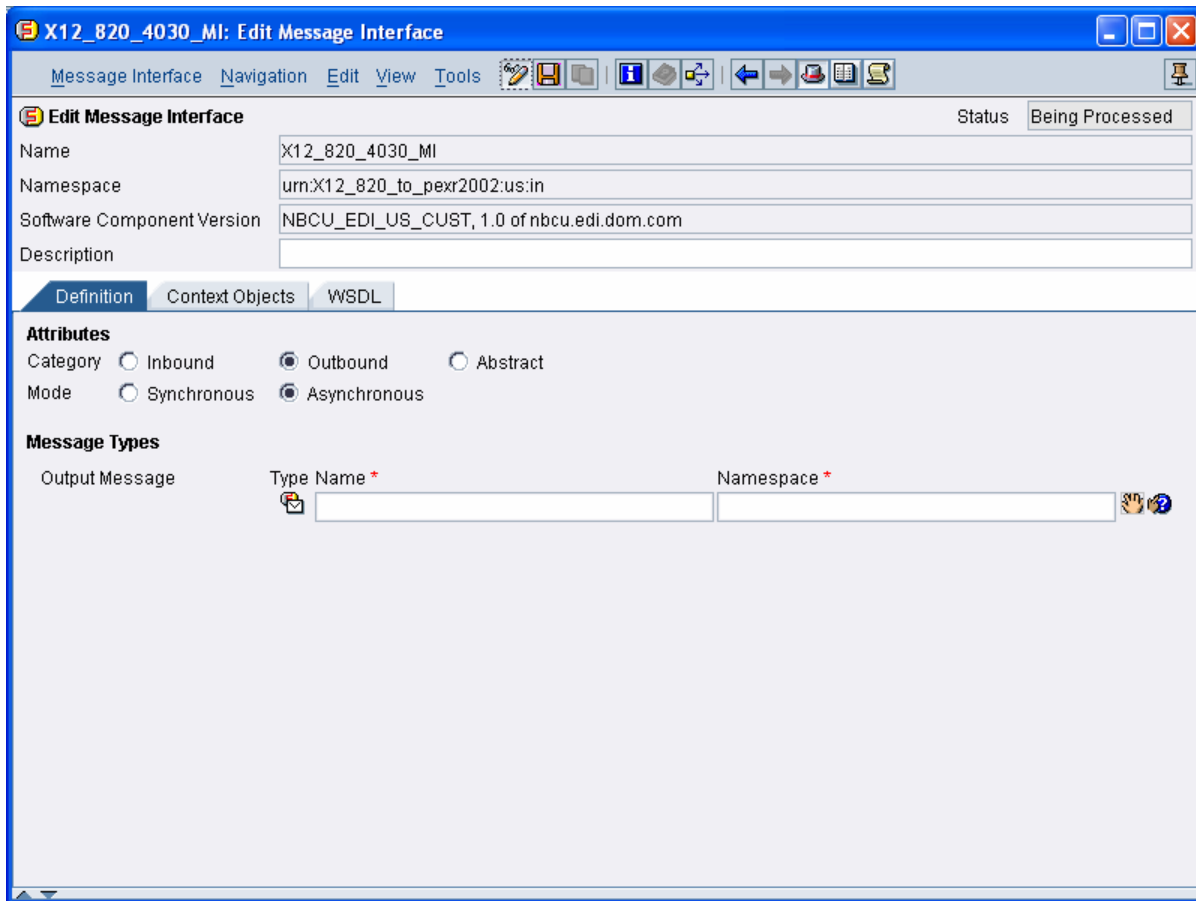
The Message Interface transforms the metadata stored in the External Definition into an input (Message Interface) that participates in an XI transformation through the Contivo mapping program.

The structure of the External Definition referenced by the Message Interface must correspond exactly to the structure of the source X12 820 interface used in the Contivo mapping program.

1. Right-click on Message Interface in the Navigation Pane and left-click New.
2. The Create Object dialog opens.
3. Enter the name X12_820_4030_MI for the name of the Message Interface.
4. The Message Interface just created is available to mapping programs for all Trading Partners that use version 4030 of the X12 820 message.
5. The namespace urn:X12_820_to_pexr2002:us:in is defaulted.
6. Click Create.

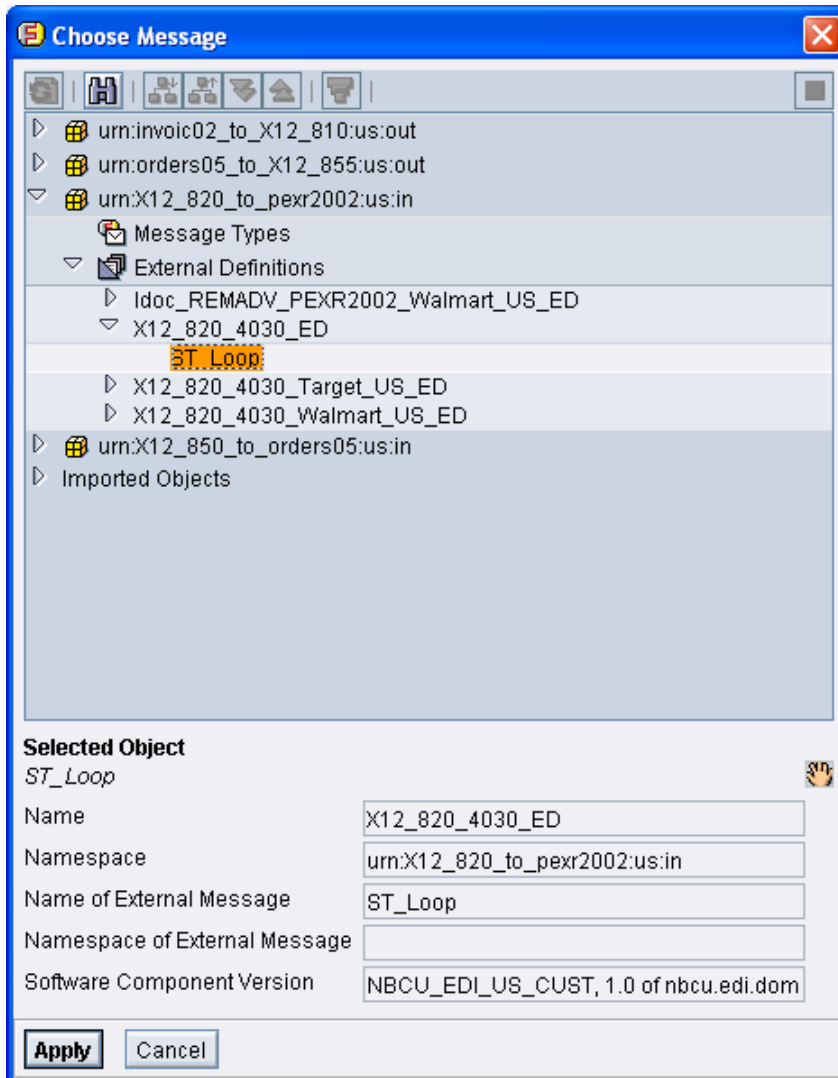


7. Under Attributes, check Outbound and Asynchronous.
8. Outbound is a little counter-intuitive but in this context it is defined in relation to the external system: the 820 is outbound from the Trading Partner's EDI system.



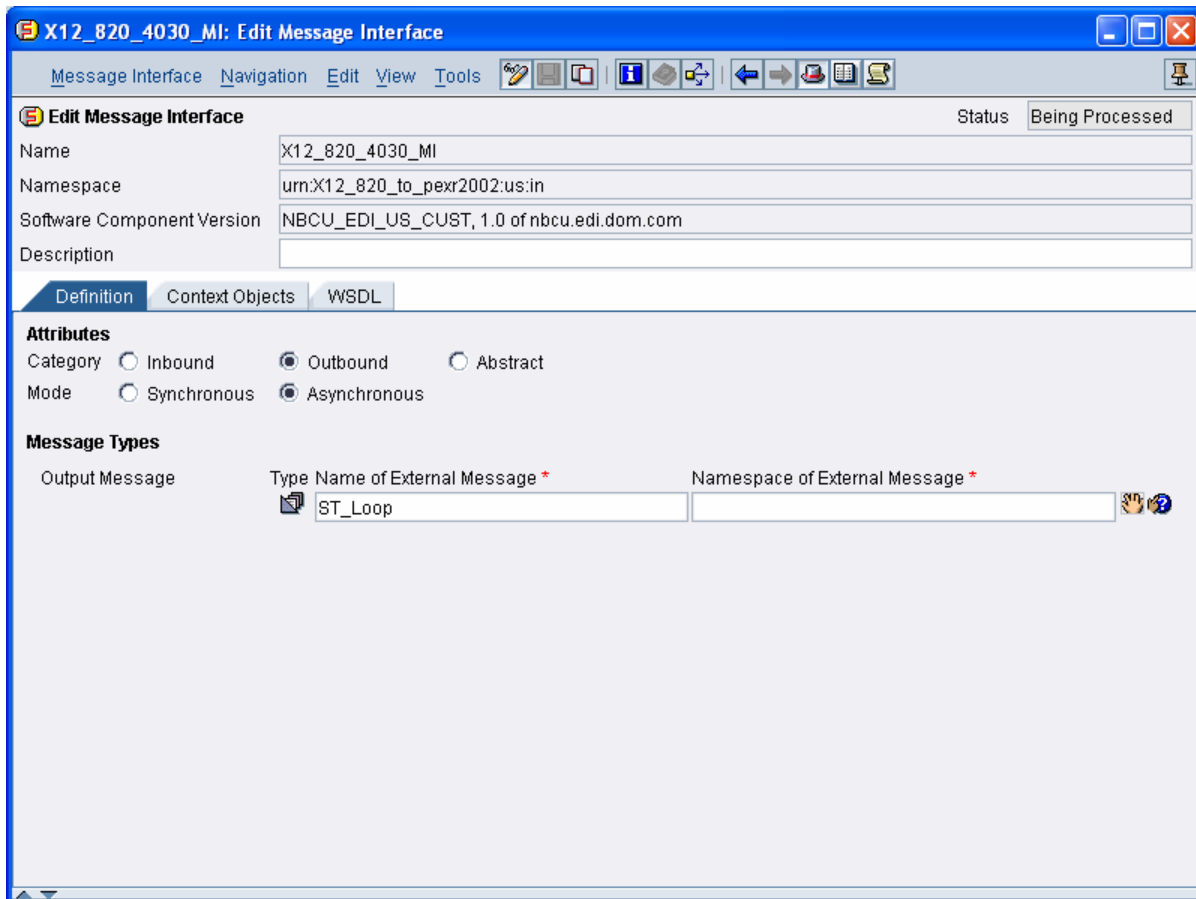
How to Build a Basic EDI Interface Using an Imported Schema and Map

9. To get the Message Type name, select the Question Mark icon next to the Hand beside the Namespace field.
10. The Choose Message dialog opens. Navigate to the 820 namespace and expand External Definition X12_820_4030_ED. Select Message Type ST_Loop.



How to Build a Basic EDI Interface Using an Imported Schema and Map

11. Click Apply to return to the Edit Message Interface screen.
12. Save and Activate the Message Interface.

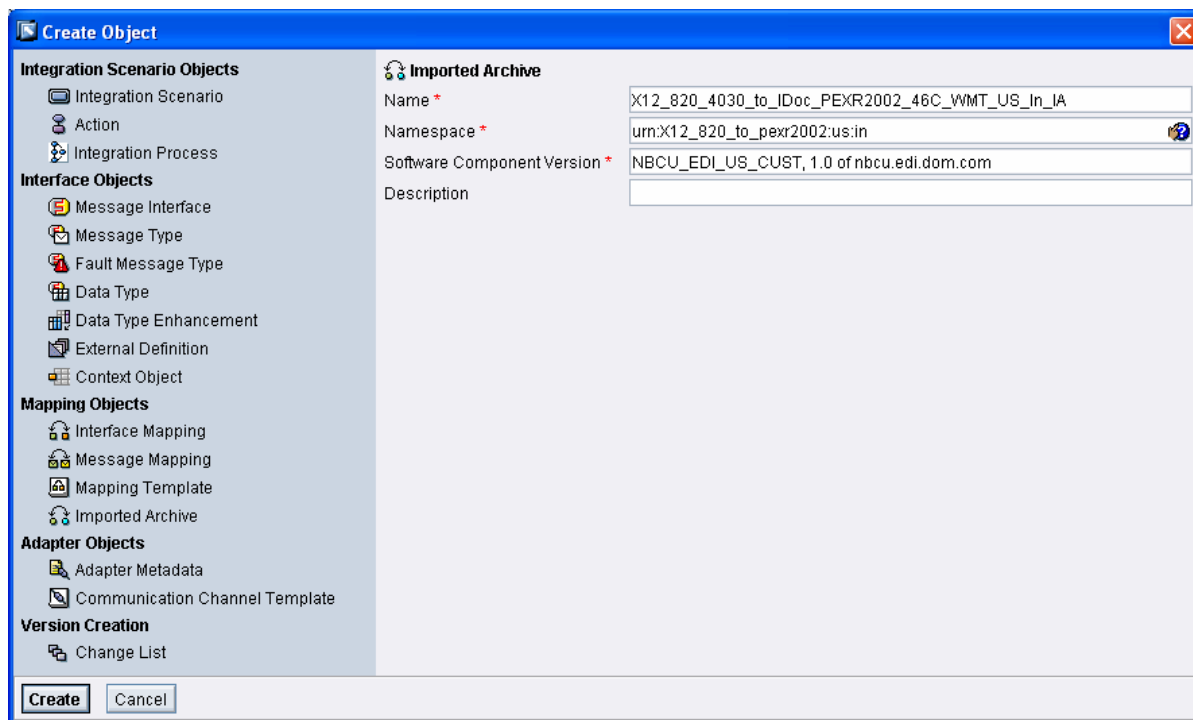


Mapping Objects

Create an Imported Archive

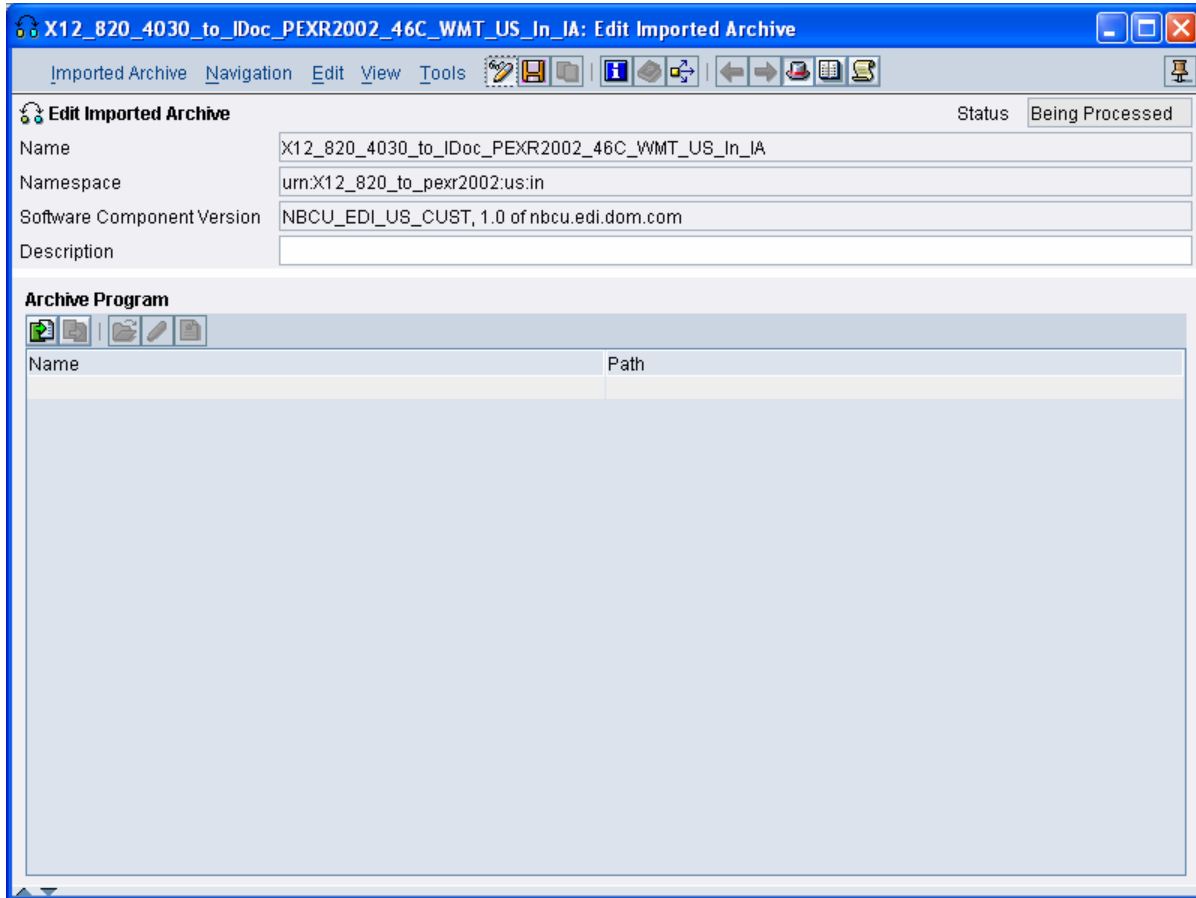
The Imported Archive stores the Java classes generated by Contivo for XI from the Contivo mapping program. These classes must be stored in zip or java jar files.

1. Right-click Imported Archives in the Navigation Pane and left-click New.
2. The Create Imported Archive Dialog opens.
3. Enter X12_820_4030_to_IDoc_PEXR2002_46C_WMT_US_In_IM in the Name field.
4. This describes the transformation in the Contivo mapping program: an X12 820 version 4030 is transformed into an IDoc REMADV.PEXR2002 for Trading Partner Walmart US.
5. This map will be used exclusively by Walmart US.
6. Click Create.

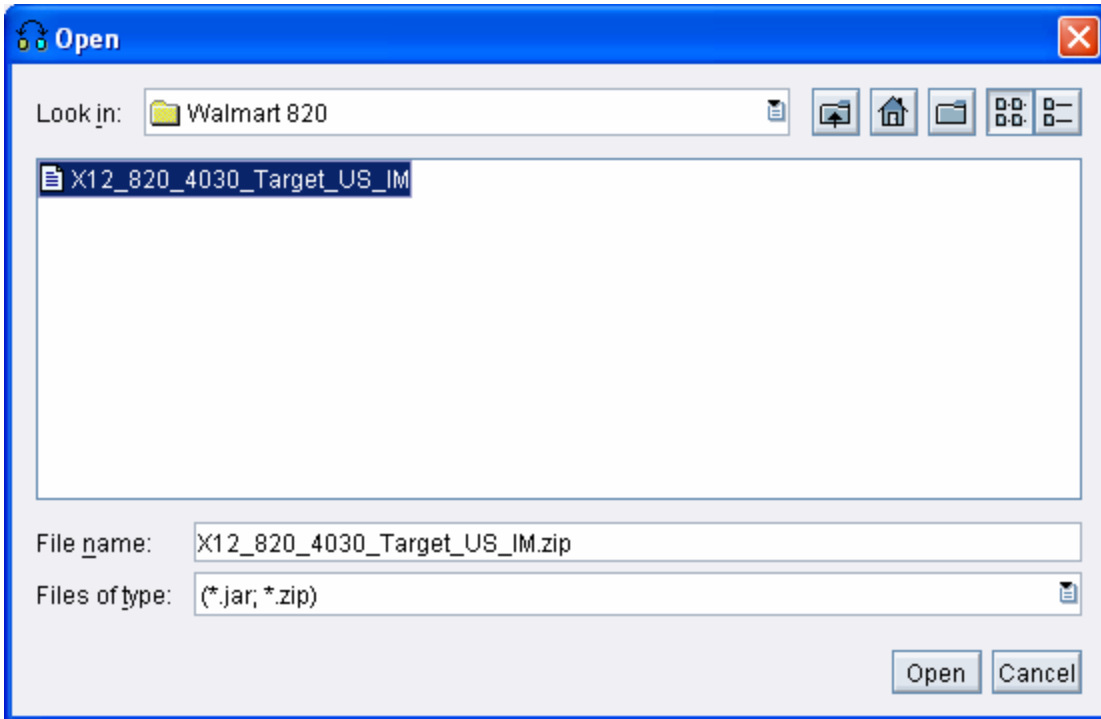


How to Build a Basic EDI Interface Using an Imported Schema and Map

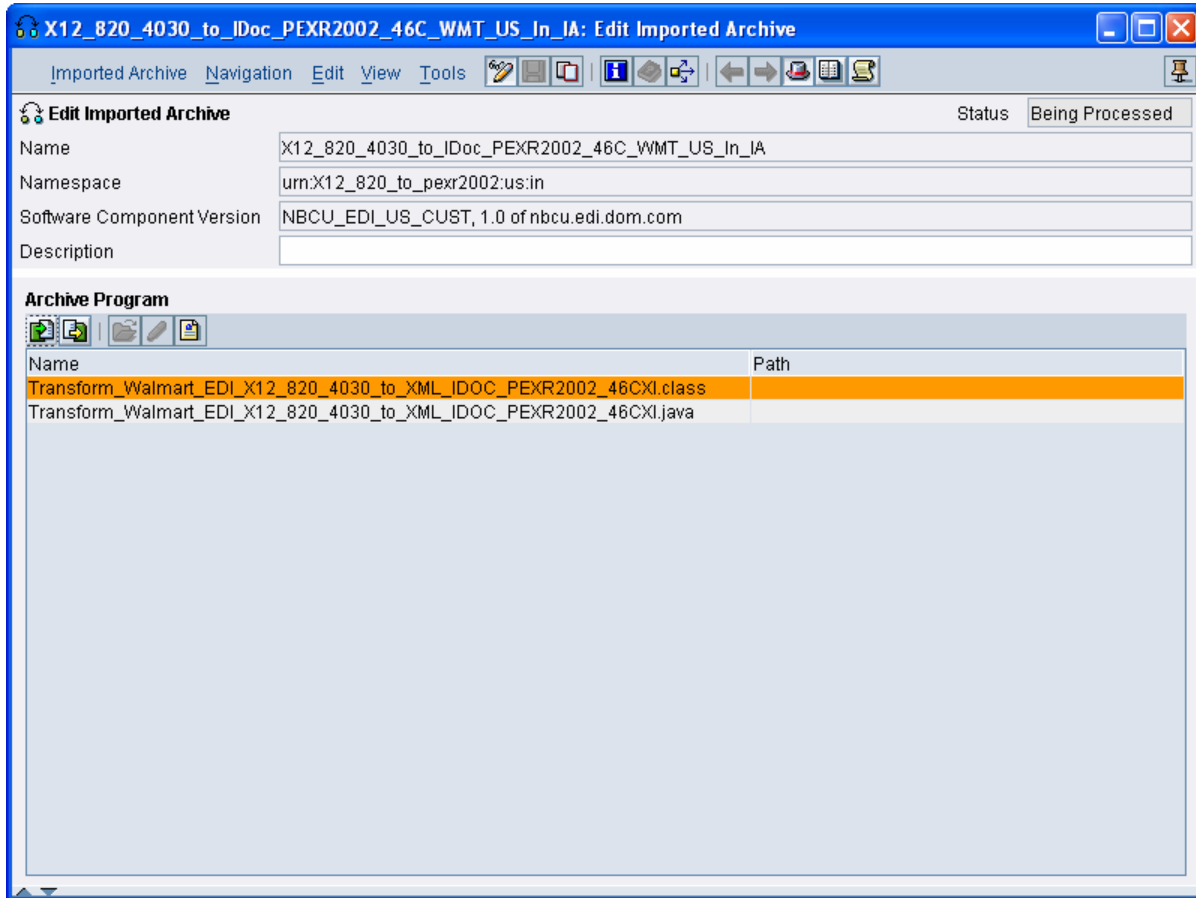
7. The Edit Imported Archive window opens.
8. Next step is to import the mapping program archive.
9. Click on the green arrow icon under Archive Program.



10. A file navigation window opens. Navigate to the directory where the mapping archive is stored, in either zip or jar format.
11. Select the archive and click Open.



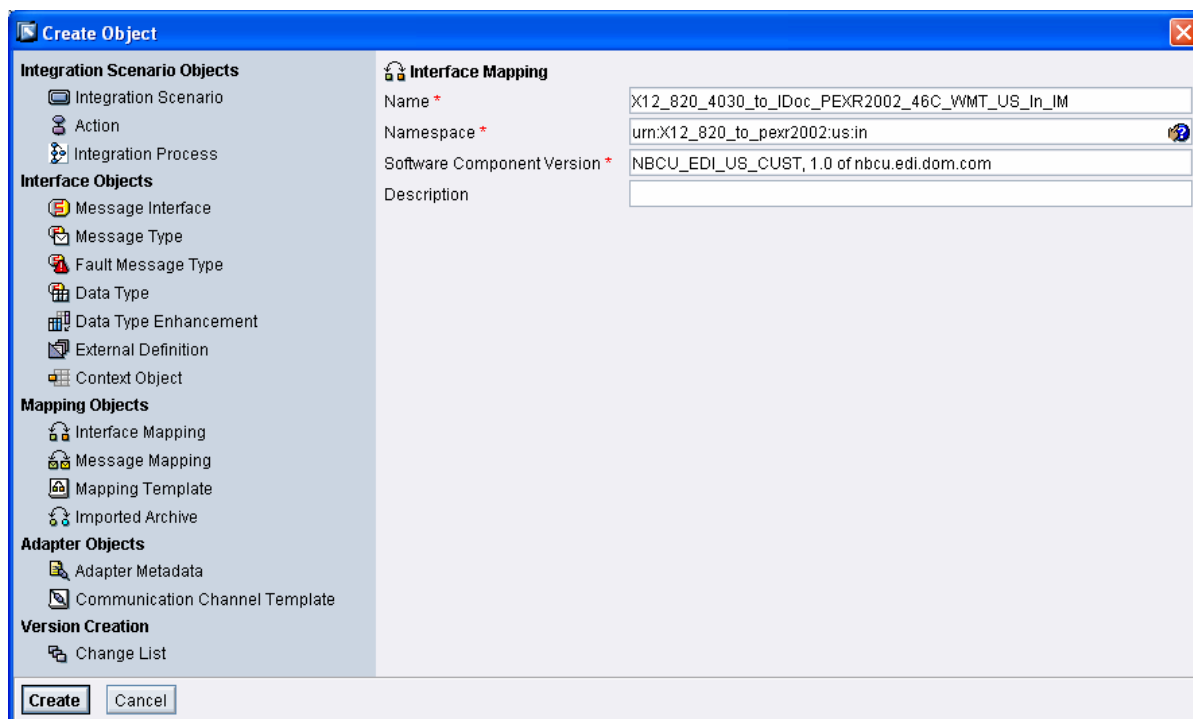
12. The archive is loaded.
13. Save and activate the Imported Archive.
14. Note the names of the Transform files in the Imported Archive.



Interface Mapping

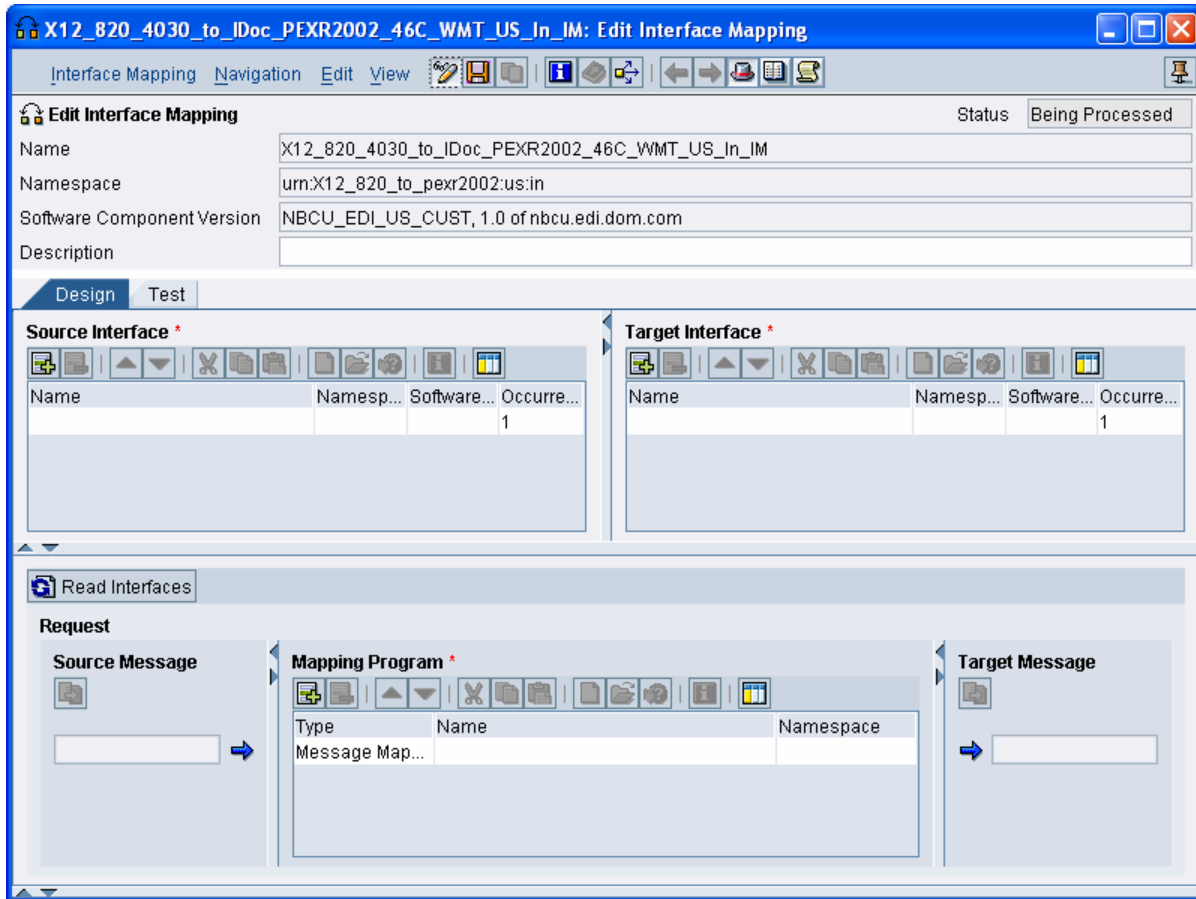
The final step is to set up an Interface Mapping. This maps the source to the target interfaces and links both with the mapping program generated in Contivo and stored in XI as an Imported Archive.

1. Right-click Interface Mappings in the Navigation Pane and left-click New.
2. The Create Interface Mapping dialog opens.
3. Enter X12_820_4030_to_IDoc_PEXR2002_46C_WMT_US_In_IM in the Name field.
4. Click Create.



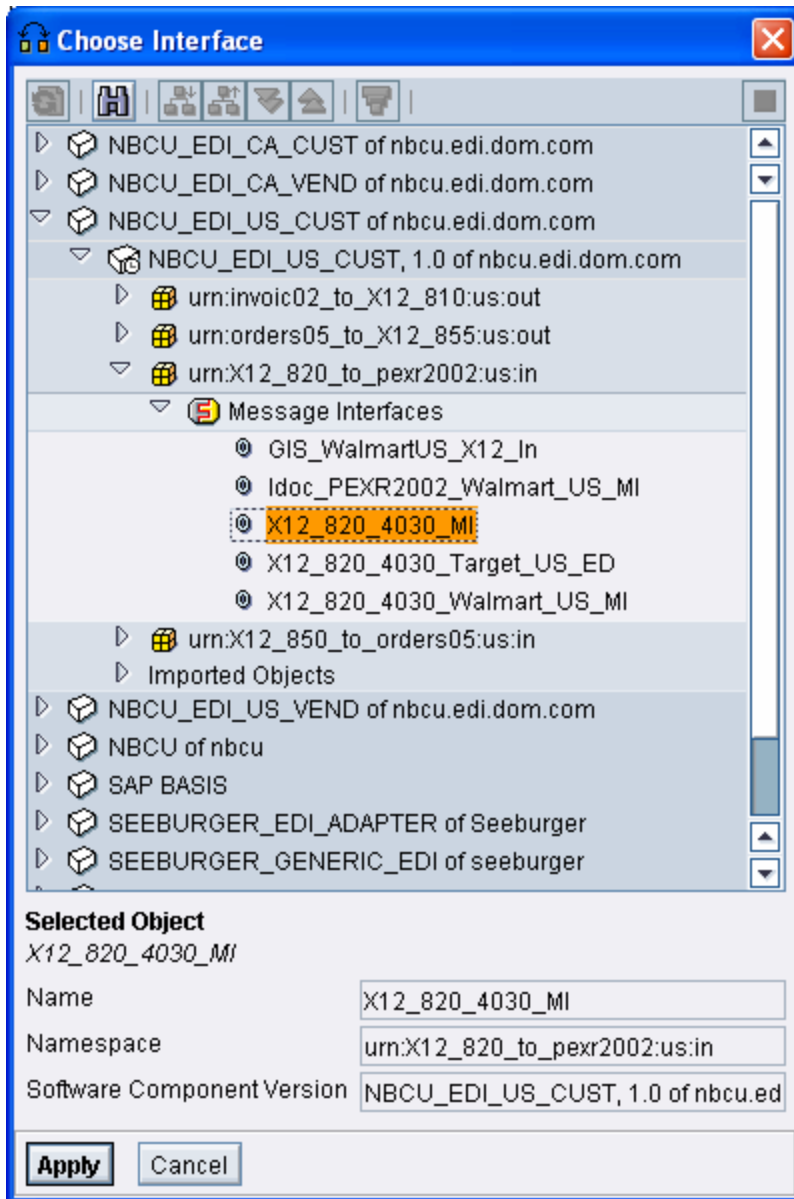
How to Build a Basic EDI Interface Using an Imported Schema and Map

5. The Edit Interface Mapping screen opens.
6. In the Source Interface panel of the Design subscreen, select the Software component, Namespace and Source Interface Name by clicking the Question Mark icon to the right of the Name field.



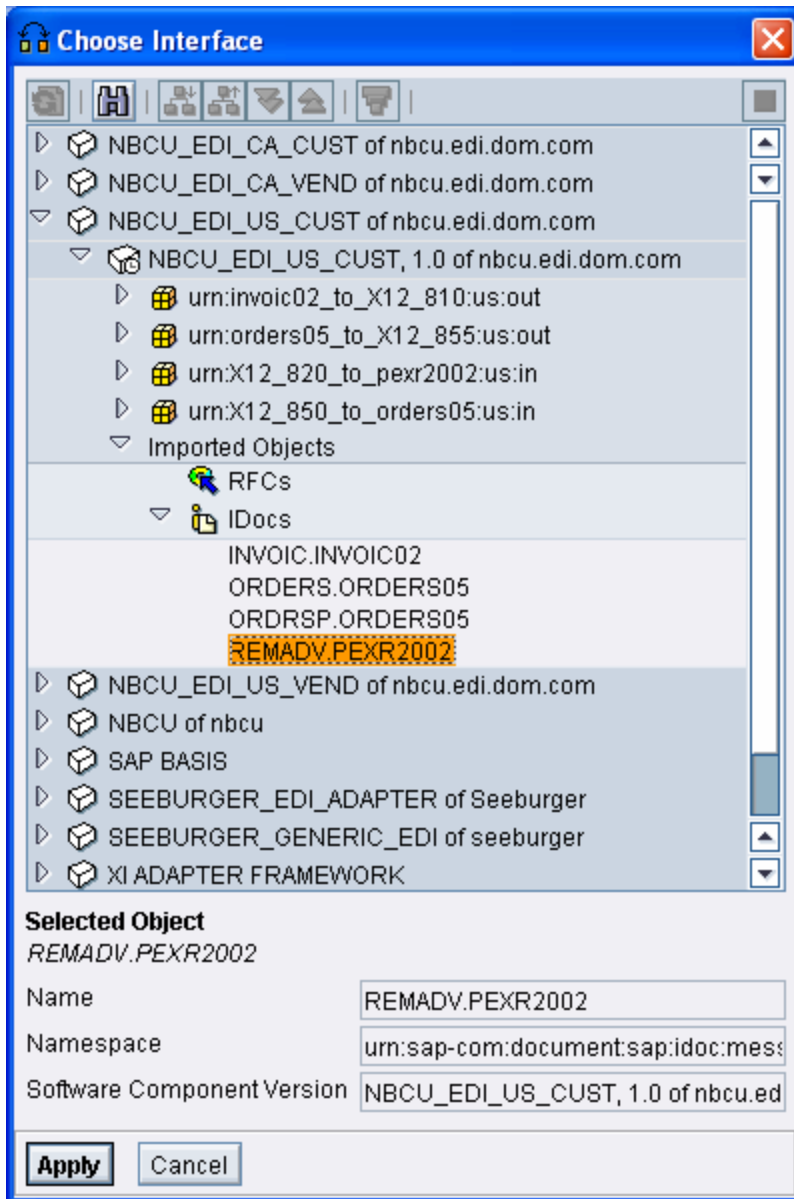
How to Build a Basic EDI Interface Using an Imported Schema and Map

7. The Choose Interface dialog opens.
8. Navigate through the Software Component and Namespace to the Message Interface X12_820_4030_MI.
9. Click Apply.



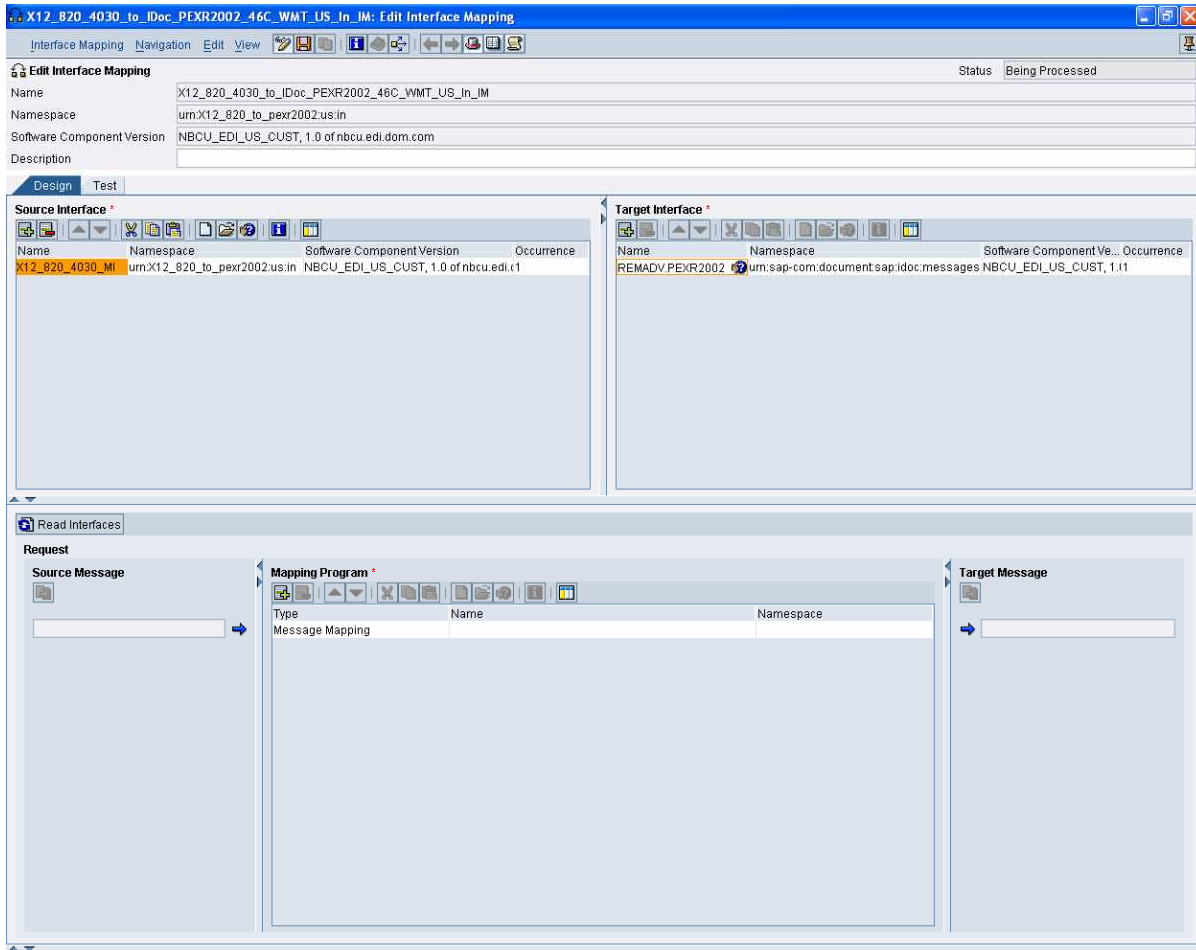
How to Build a Basic EDI Interface Using an Imported Schema and Map

10. Select the Target Interface in the same way: IDoc REMADV.PEXR2002 under Imported Objects at the Software Component Level.
11. Click Apply.



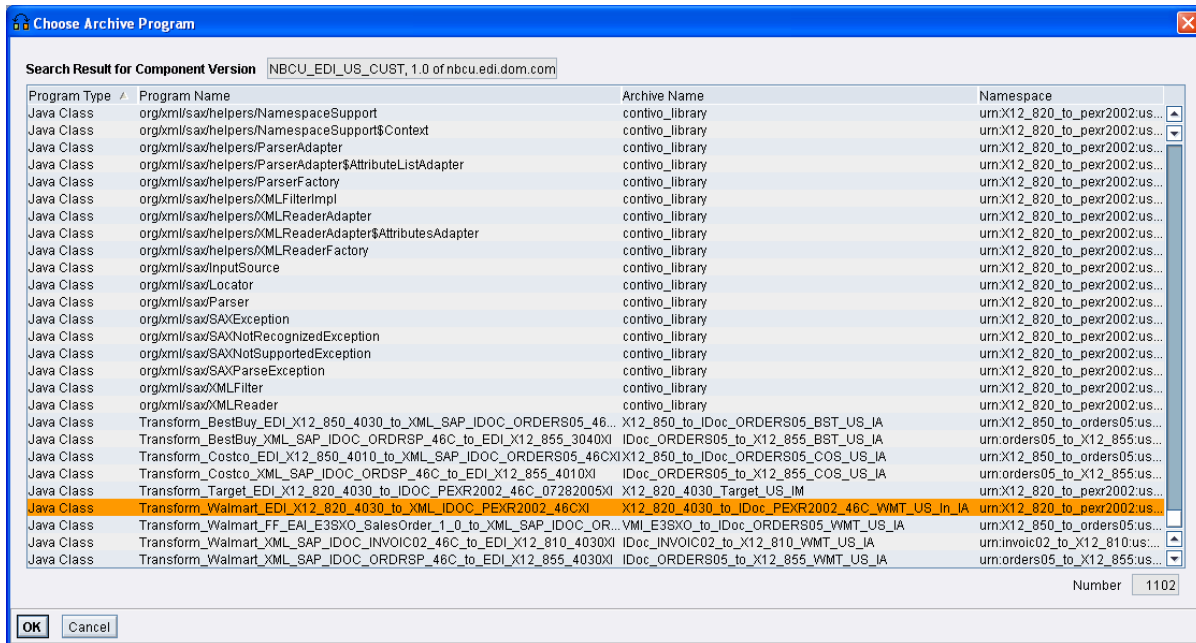
How to Build a Basic EDI Interface Using an Imported Schema and Map

12. The Source and Target Interfaces have been selected.
13. Next step is to identify the Mapping Program. In the Mapping Program subscreen, select the program type from the drop-down list under Type.
14. Program type is Java Class.
15. Click the Question Mark icon to the right of the Program Name field.



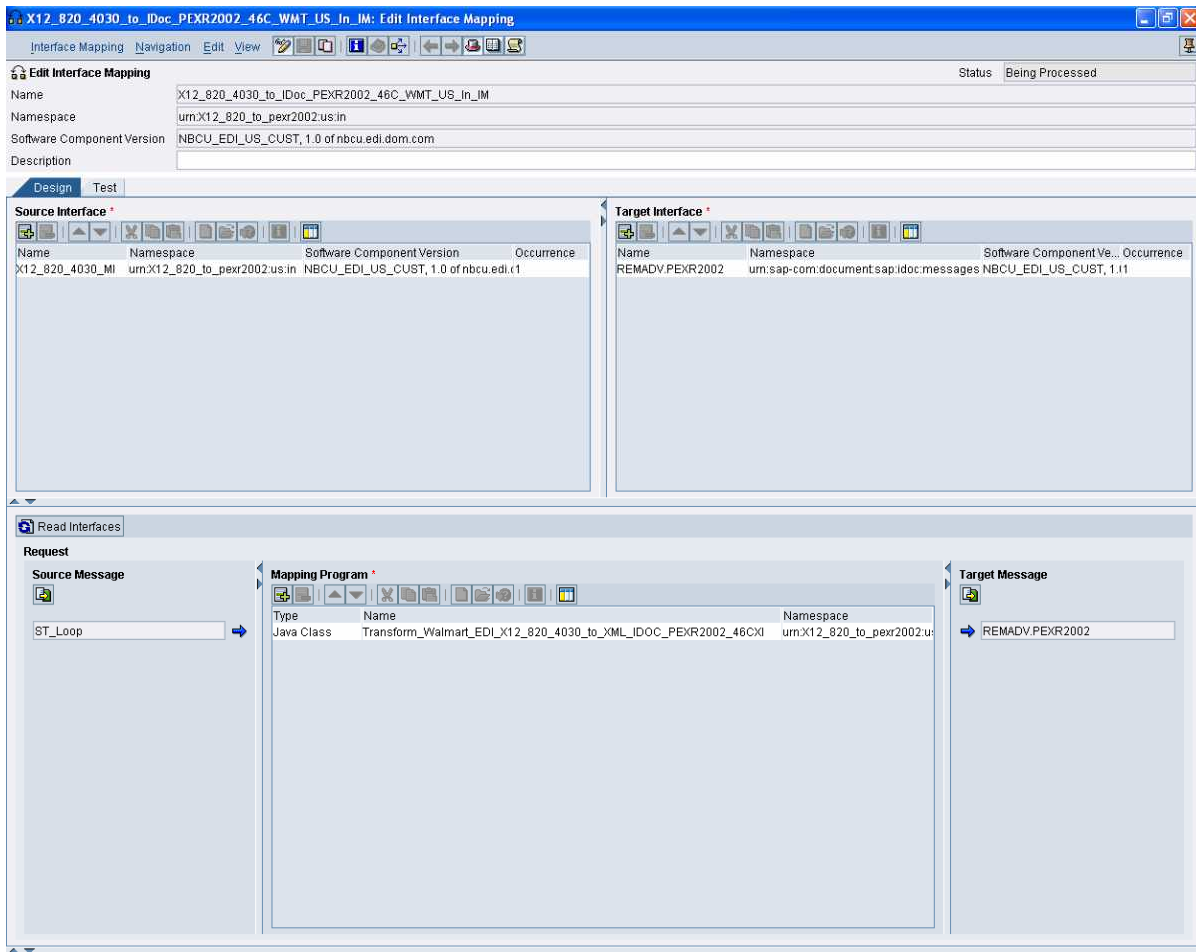
How to Build a Basic EDI Interface Using an Imported Schema and Map

16. The Choose Archive Program dialog opens.
17. The Transform Classes are generally at the bottom of the list of classes.
18. Select the Transform class for the Walmart US 820 inbound and click OK.



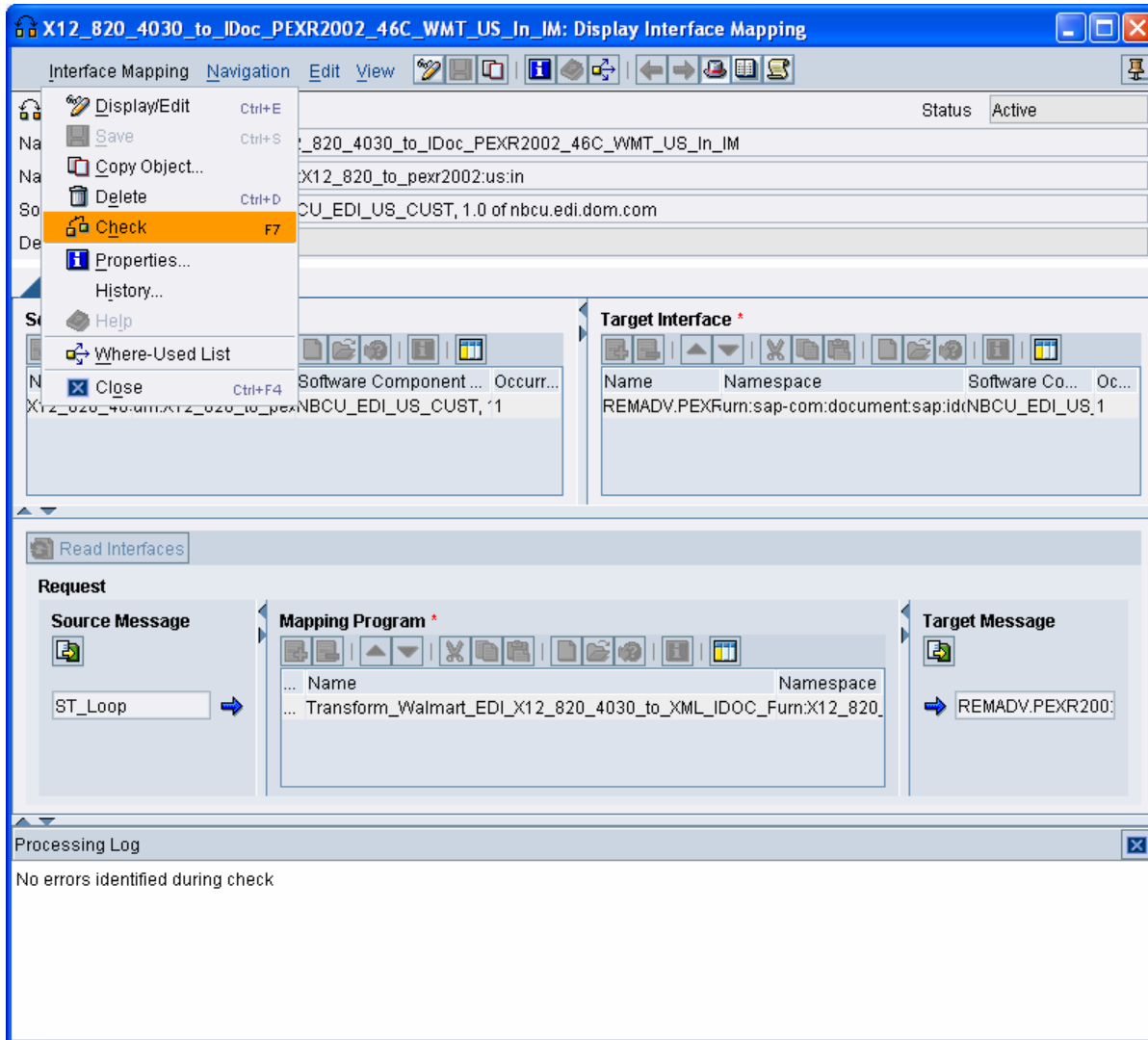
How to Build a Basic EDI Interface Using an Imported Schema and Map

19. The Edit Interface Mapping screen returns.
20. Click the Read Interfaces button to retrieve the names of the Source and Target interfaces used in the Mapping Program.
21. Save and activate the Interface Mapping.



How to Build a Basic EDI Interface Using an Imported Schema and Map

22. Select Check from the Interface Mapping menu to run a syntax check on the map.
23. If no errors are identified, move on to testing execution of the map against X12 820 data in XML format.

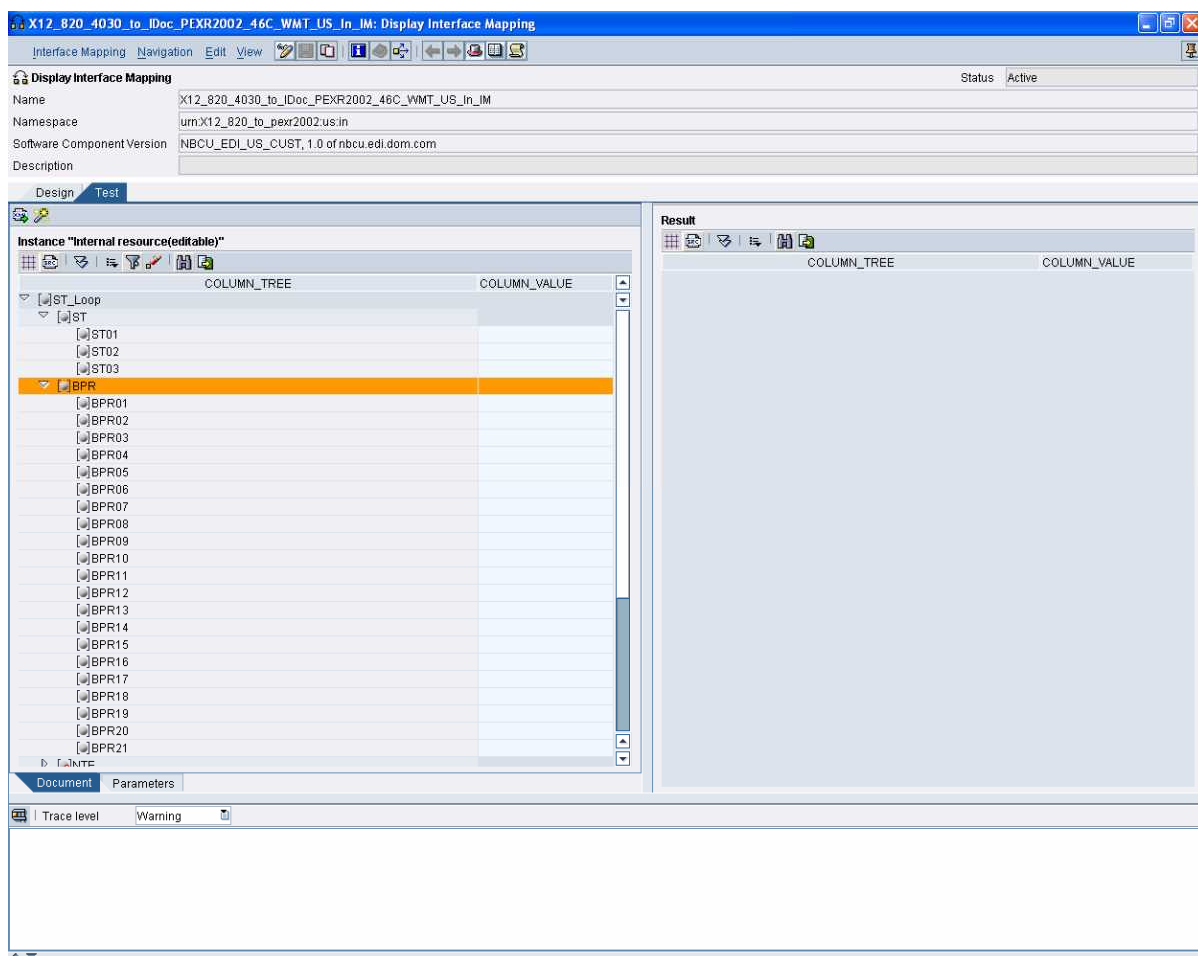


Testing the Interface Mapping

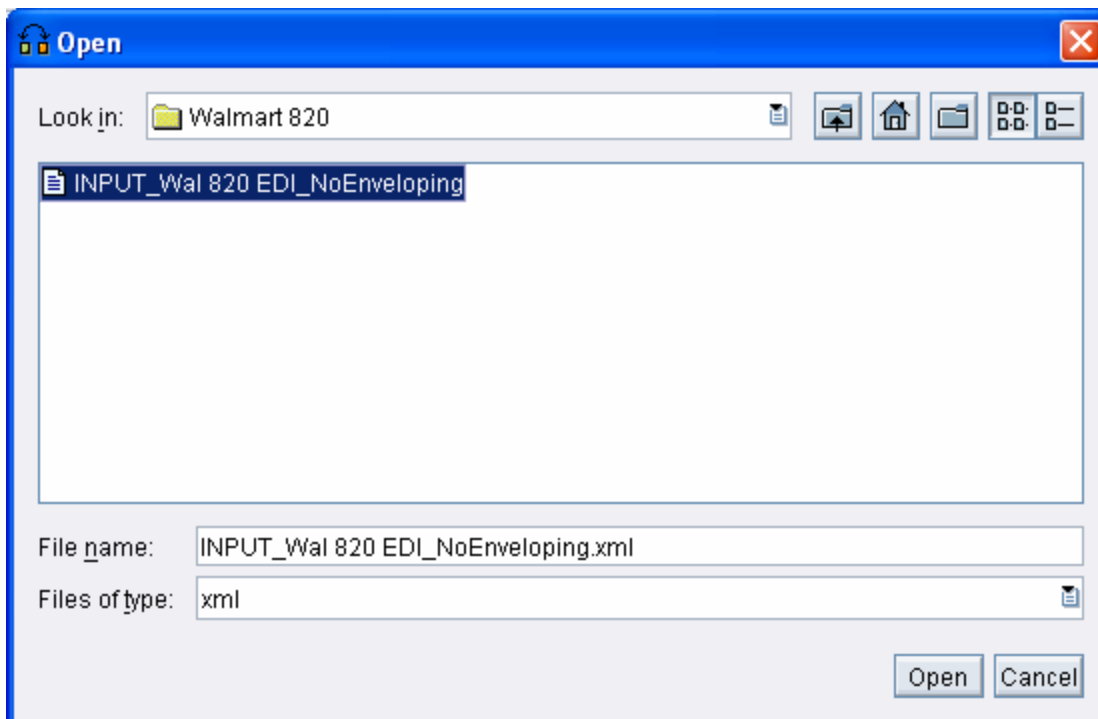
We can now test the map against an X12 data file in XML format to ensure that it will cleanly map to an IDoc.

If the test passes, we can move on to configuring the interface in the XI Integration Directory. If not, XI provides extensive Trace functionality to help us identify where the error fell.

1. Click on the Test tab.
2. Note the Source interface on the left: the X12 820 transaction as defined in our Contivo mapping program and Message Interface.
3. To load the test file, click on the XML icon to the far left of the bar just beneath the Design and Test tabs.

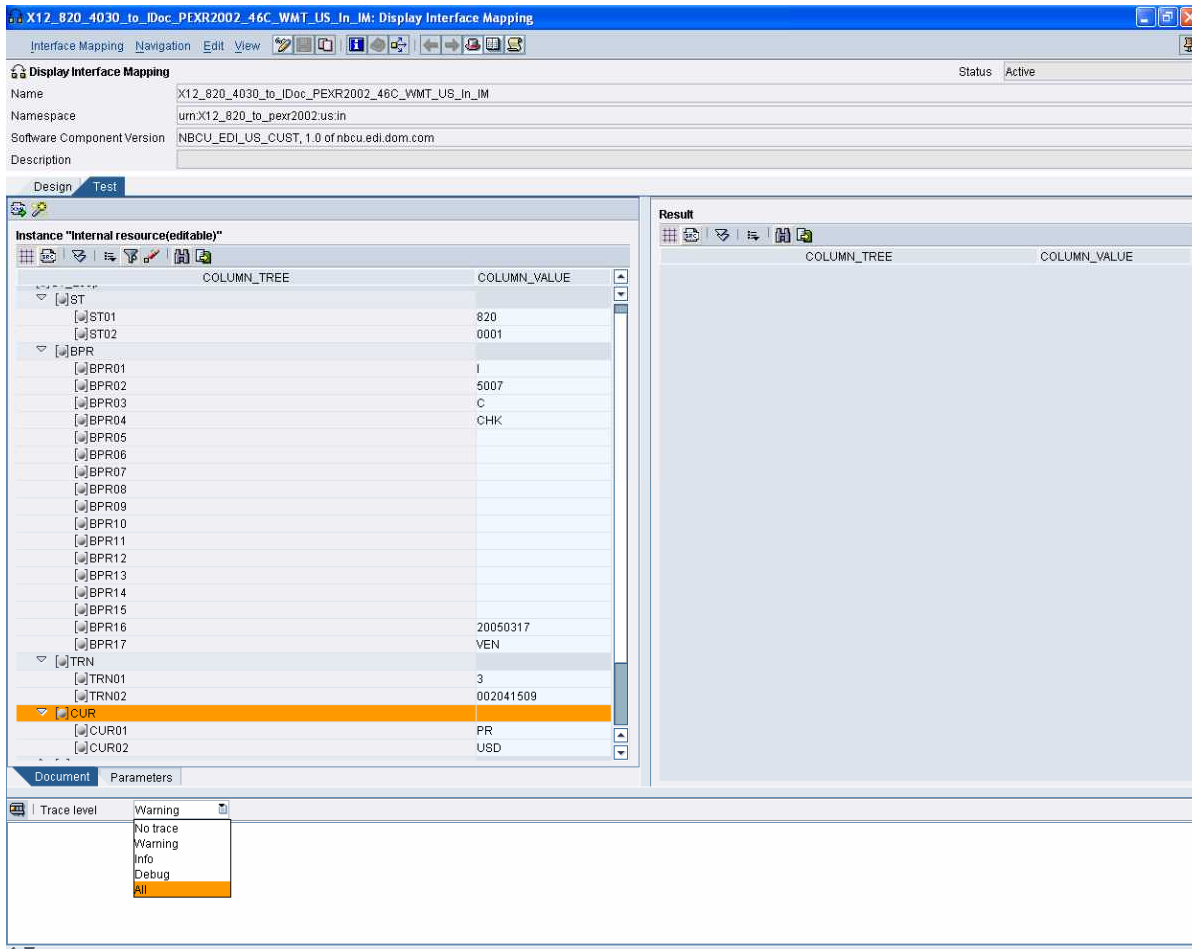


4. A file navigation window will open.
5. Navigate to where the XML test input is stored and click Open.

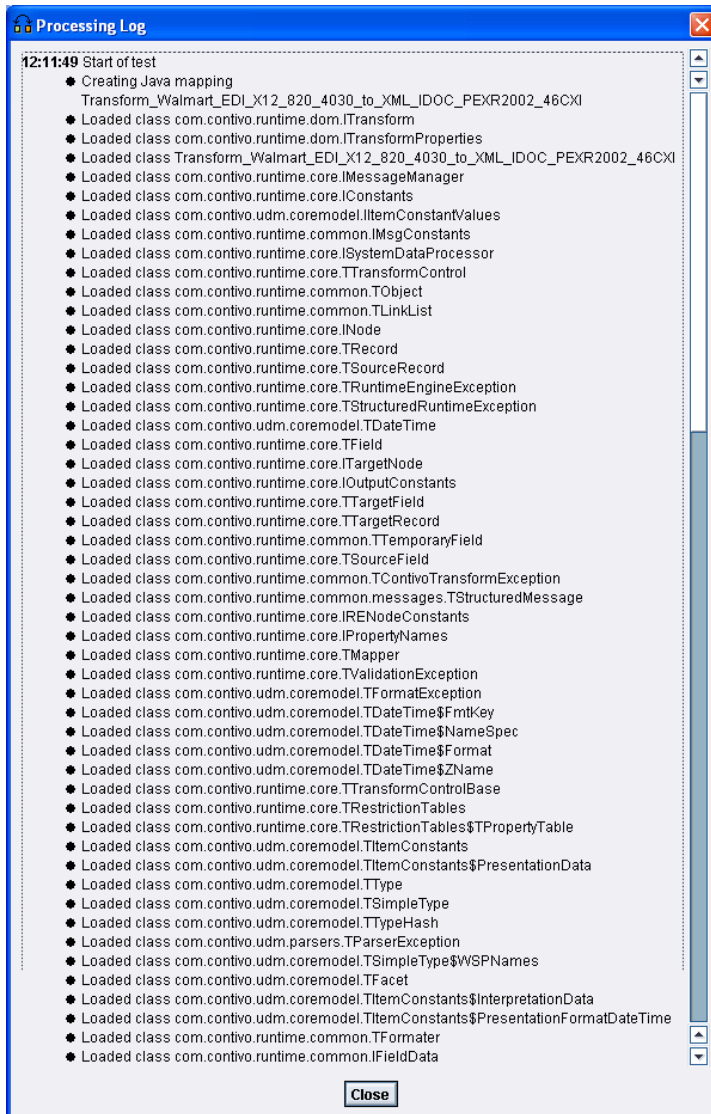


How to Build a Basic EDI Interface Using an Imported Schema and Map

6. Note that the X12 820 is now populated with data from the XML input file.
7. Select the Trace Level from the Trace drop-down list in the Test subscreen.
8. Click the Execute icon next to the Trace level drop-down list to run the test.



9. If Trace Level All is selected, a Processing Log dialog opens listing every operation executed in running the test. Success or failure will be reported at the bottom at the processing stack. An error will end processing at the point of failure.



How to Build a Basic EDI Interface Using an Imported Schema and Map

10. Trace data is also written to the Trace Window at the bottom of the screen.
11. A successful run will return a successful transformation in the right pane: an instance of IDoc REMADV.PEXR2002 correctly populated with data from the 820 XML input file.
12. Navigate through the IDoc and check as many segments as you need to confirm that the X12 820 data has been correctly mapped to the corresponding IDoc segment and field.

The screenshot displays the SAP Display Interface Mapping tool. The main window is titled "X12_820_4030_to_IDoc_PEXR2002_46C_WMT_US_In_IM: Display Interface Mapping". It shows the mapping configuration for the interface, including the name, namespace, and software component version.

The interface is divided into two main panes: "Instance 'Internal resource(editable)'" and "Result".

Instance 'Internal resource(editable)' shows a tree structure of columns and their values:

Column Tree	Column Value
[ST] Loop	
[ST]	
[ST01]	820
[ST02]	0001
[BPR]	
[BPR01]	1
[BPR02]	5007
[BPR03]	C
[BPR04]	CHK
[BPR05]	
[BPR06]	
[BPR07]	
[BPR08]	
[BPR09]	
[BPR10]	
[BPR11]	
[BPR12]	
[BPR13]	
[BPR14]	
[BPR15]	
[BPR16]	20050317
[BPR17]	VEN
[TRN]	
[TRN01]	3
[TRN02]	002041509
[CUR]	
[CUR01]	PR
[CUR02]	USD
[N1] Loop	
[N1]	
[N101]	PR

Result pane shows the resulting IDoc structure:

Column Tree	Column Value
[PEXR2002]	
[IDOC]	
[BEGIN]	1
[EDL_DC40]	
[E1DKU1]	
[SEGMENT]	1
[BGM TYP]	REM
[BGM NAME]	06
[BGM LEV]	ORG
[BGM ACC]	001
[E1EDK03]	
[SEGMENT]	1
[ID DAT]	017
[DATUM]	20050317
[E1EDK02]	
[SEGMENT]	1
[QUALF]	026
[BELNR]	002041509
[E1DKU3]	
[SEGMENT]	1
[PAIMED]	CHK
[E1DKU5]	
[SEGMENT]	1
[MOAQUAL]	001
[MOABETR]	5007
[CUXWAERZ]	USD
[E1EDKA1]	
[SEGMENT]	1
[PARVW]	AG
[PARTN]	V130
[LIFNR]	0003000001
[NAME1]	WAL-MART STORES, INC.

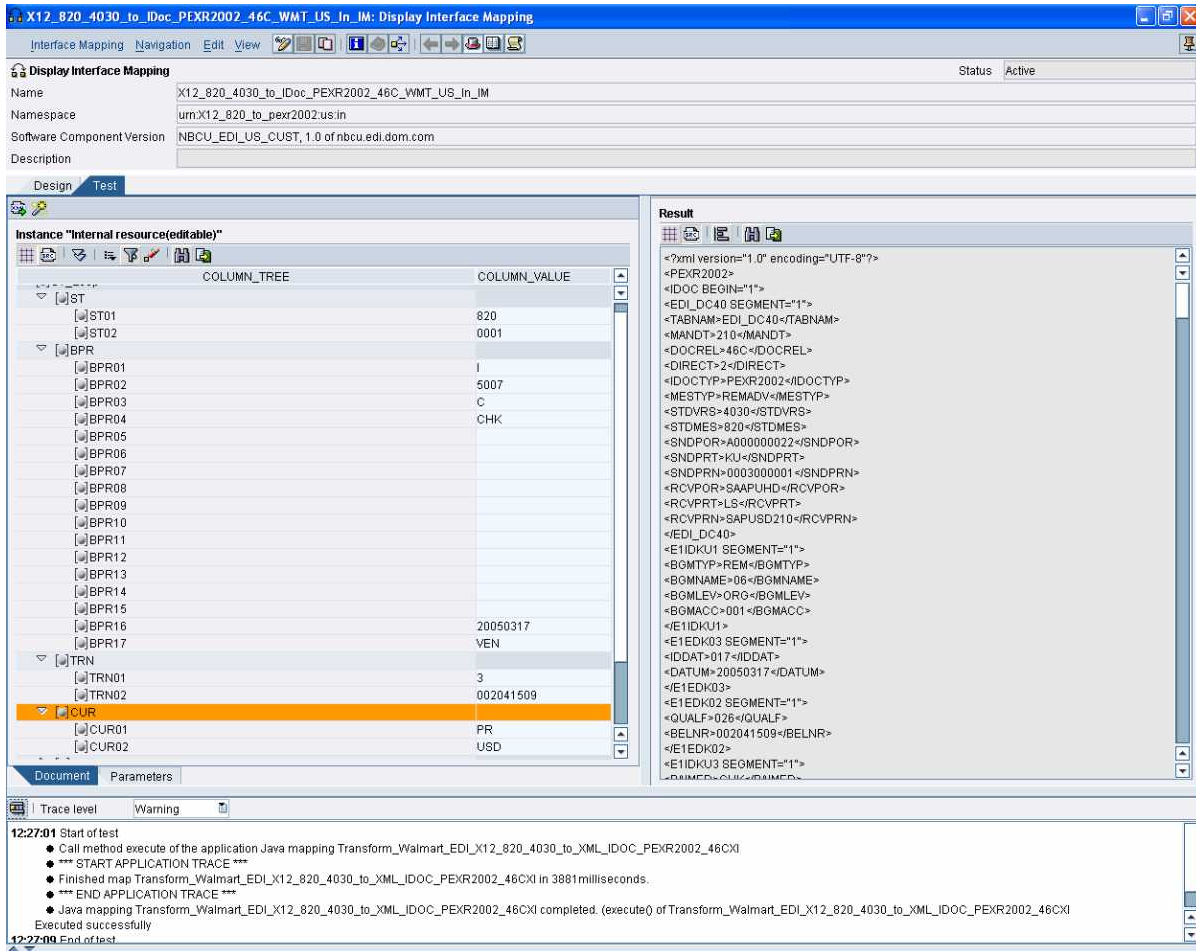
The bottom pane shows the trace level set to "All" and a log of test execution steps:

```

12:11:49 Start of test
  • Creating Java mapping Transform_Walmart_EDI_X12_820_4030_to_XML_IDOC_PEXR2002_46CXI
  • Loaded class com.contivo.runtime.dom.ITransform
  • Loaded class com.contivo.runtime.dom.ITransformProperties
    
```

How to Build a Basic EDI Interface Using an Imported Schema and Map

13. To view the XML source in the mapped IDoc, click the SRC icon above the IDoc display.
14. To export the IDoc source -- data and XML tags -- click the Export Source icon to the right of the tool bar above the IDoc.



The screenshot shows the SAP IDoc mapping tool interface. The main window is titled "X12_820_4030_to_IDoc_PEXR2002_46C_WMT_US_In_Im: Display Interface Mapping". It has a menu bar with "Interface Mapping", "Navigation", "Edit", and "View". Below the menu bar, there are tabs for "Design" and "Test".

The "Design" tab is active, showing an "Instance 'Internal resource(editable)'" with a table of columns and values:

COLUMN_TREE	COLUMN_VALUE
[+] ST	
[+] ST01	820
[+] ST02	0001
[+] BPR	
[+] BPR01	J
[+] BPR02	5007
[+] BPR03	C
[+] BPR04	CHK
[+] BPR05	
[+] BPR06	
[+] BPR07	
[+] BPR08	
[+] BPR09	
[+] BPR10	
[+] BPR11	
[+] BPR12	
[+] BPR13	
[+] BPR14	
[+] BPR15	
[+] BPR16	20050317
[+] BPR17	VEN
[+] TRN	
[+] TRN01	3
[+] TRN02	002041509
[+] CUR	
[+] CUR01	PR
[+] CUR02	USD

The "Test" tab is also visible, showing the resulting XML source code:

```
<?xml version="1.0" encoding="UTF-8"?>
<PEXR2002>
<IDOC BEGIN="1">
<EDI_DC40 SEGMENT="1">
<TABNAM>EDI_DC40</TABNAM>
<MANDT>210</MANDT>
<DOCREL>46C</DOCREL>
<DIRECT>2</DIRECT>
<IDOCITY>PEXR2002</IDOCITY>
<MESTYP>REMADV</MESTYP>
<STDVRS>4030</STDVRS>
<STDMES>820</STDMES>
<SNDPOR>A00000022</SNDPOR>
<SNDPRT>KUJ</SNDPRT>
<SNDRPN>0003000001</SNDRPN>
<RCVPOR>SAAPUHD</RCVPOR>
<RCVPRV>LS</RCVPRV>
<RCVPRN>SAPUSD210</RCVPRN>
<EDI_DC40>
<E1IDKU1 SEGMENT="1">
<BGMTYP>REM</BGMTYP>
<BGNAME>06</BGNAME>
<BGMLEV>ORG</BGMLEV>
<BGMACC>001</BGMACC>
<E1IDKU1>
<E1EDK03 SEGMENT="1">
<IDDAT>017</IDDAT>
<DATUM>20050317</DATUM>
<E1EDK03>
<E1EDK02 SEGMENT="1">
<QUALF>026</QUALF>
<BELNR>002041509</BELNR>
<E1EDK02>
<E1IDKU3 SEGMENT="1">
<AMFED>GUK</AMFED>
```

The "Trace level" section at the bottom shows the following log entries:

```
12:27:01 Start of test
● Call method execute of the application Java mapping Transform_WalmartEDI_X12_820_4030_to_XML_IDOC_PEXR2002_46CXI
● *** START APPLICATION TRACE ***
● Finished map Transform_WalmartEDI_X12_820_4030_to_XML_IDOC_PEXR2002_46CXI in 3881 milliseconds.
● *** END APPLICATION TRACE ***
● Java mapping Transform_WalmartEDI_X12_820_4030_to_XML_IDOC_PEXR2002_46CXI completed. (execute) of Transform_WalmartEDI_X12_820_4030_to_XML_IDOC_PEXR2002_46CXI
Executed successfully
12:27:00 End of test
```

Author Bio



Author Name: Emmanuel Hadzipetros

Company: NBC Universal, Universal City, CA

Emmanuel Hadzipetros has been an SAP technical contractor and consultant for more than 11 years, since R3 version 2.2. A certified ABAP consultant, he's long specialized in integrating SAP with external Legacy, Custom, Packaged and Business Partner systems on large and complex projects.

He has intimate and detailed knowledge of the IDoc interface and other standard SAP and external integration tools and technologies and has used this knowledge in more than 10 SAP implementations representing a wide variety of industries in four countries and three continents.

For the last four years Emmanuel has gone Hollywood as a key fixture on two major Studio implementations where he helped lead the SAP development effort in building complex VMI-based EDI transactional processing systems supporting billions of dollars a year in Video and DVD sales.

Emmanuel currently lives with his son Johnny in Westlake Village, California. He is SAP EDI Development Lead at NBC Universal Home Entertainment, where he is actively involved in all phases of designing and implementing a dynamic EDI architecture that includes Gentran Integration Server for EDI services, Contivo Analyst for mapping, SAP XI for integration logic and SAP R3 as the enterprise system of record.