Tutorial #004: Building Metadata for Oracle B2B 11g using Self-Service XML

Contents

Introduction

Self-Service XML is a simplified metadata of Oracle B2B 11g metadata. This helps user to create bulk amount of document protocol, trading partner, and agreements in a single command.

Purpose
The main purpose of this tech-note is to show Self-Service usage through creating XML based on the Selfservice.xsd, and running the tool using command-line script.

After reading this tech-note, users should be able to independently model Self-Service XML based on the use cases.

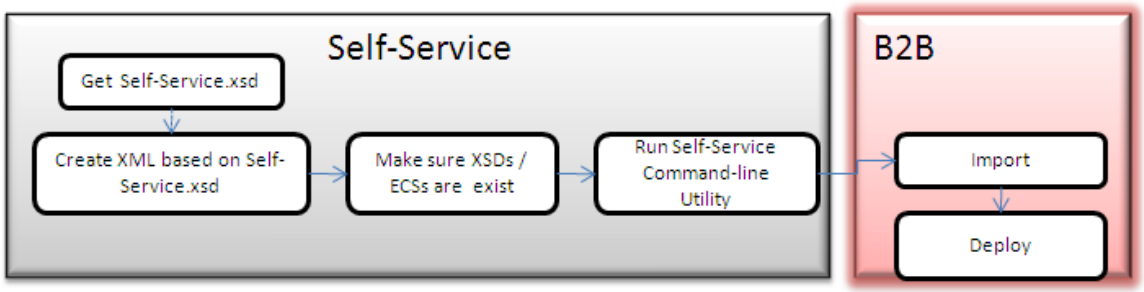
Audience
B2B users who want to create/upload bulk configuration using Oracle B2B as an Integration Gateway product and would like to understand various use cases and implementation.

Prerequisite
Basic knowledge of XML and Oracle B2B 11g metadata is must.

Assumptions
This document is based on Oracle Fusion Middleware B2B 11g R1 PS2.

Creating basic Self-Service XML

An XSD also shipped to explain the structure of the Self-Service XML. Appendix in Oracle B2B 11g documentation has the details of the Self-Service supported transport, document protocols, exchange, and its protocols details. Once the XML get created manually or modified from existing sample, this XML has to be given to the Self-Service Utility to convert to the Oracle B2B 11g metadata format. After successful conversion, converted Oracle B2B 11g metadata has to import using B2B UI Import or command-line import. Rest of steps of bringing the agreement to Runtime remains same.



Deploying agreements and modifying the existing configuration are similar for all the document protocols, which will not be explained in this document.

Get Self-Service XSD

XSD for Self-Service is shipped as part of Oracle SOA-B2B WLS installation. In SOA_HOME location, all the command-line utilities for Oracle B2B are bundled in single ant script. Using this ANT script's target, Self-Service.xsd will be created.

Prerequisite: Set environment variables

Variable Name	Value
ORACLE_HOME	/scratch/<env.USER>/fmwhome/AS11gR1SOA
ANT_HOME	$ORACLE_HOME/../modules/org.apache.ant_1.7.0
JAVA_HOME	$ORACLE_HOME/../jdk160_11

→ Run ant command to export Self-Service.xsd

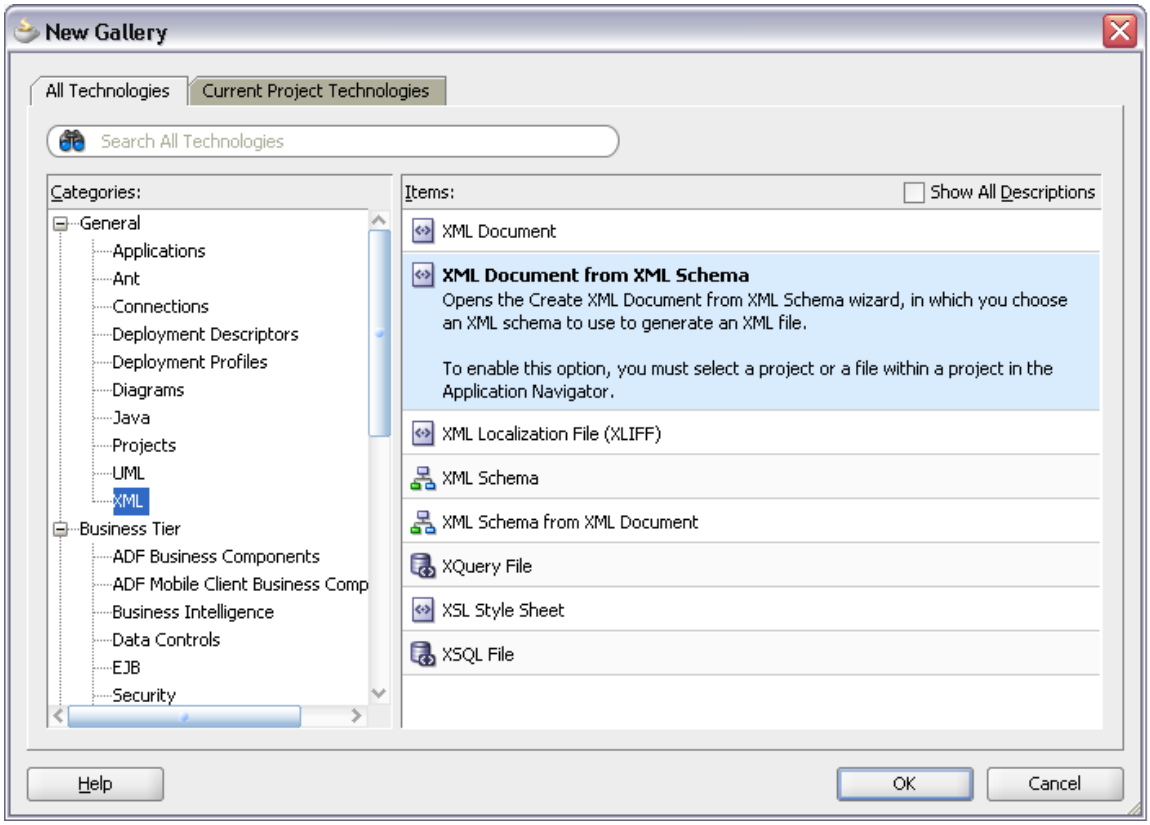
```
ant -f ant-b2b-util.xml b2bselfservicexsd
```

Now, a file with name selfservice.xsd will get created in the ORACLE_HOME/bin location.

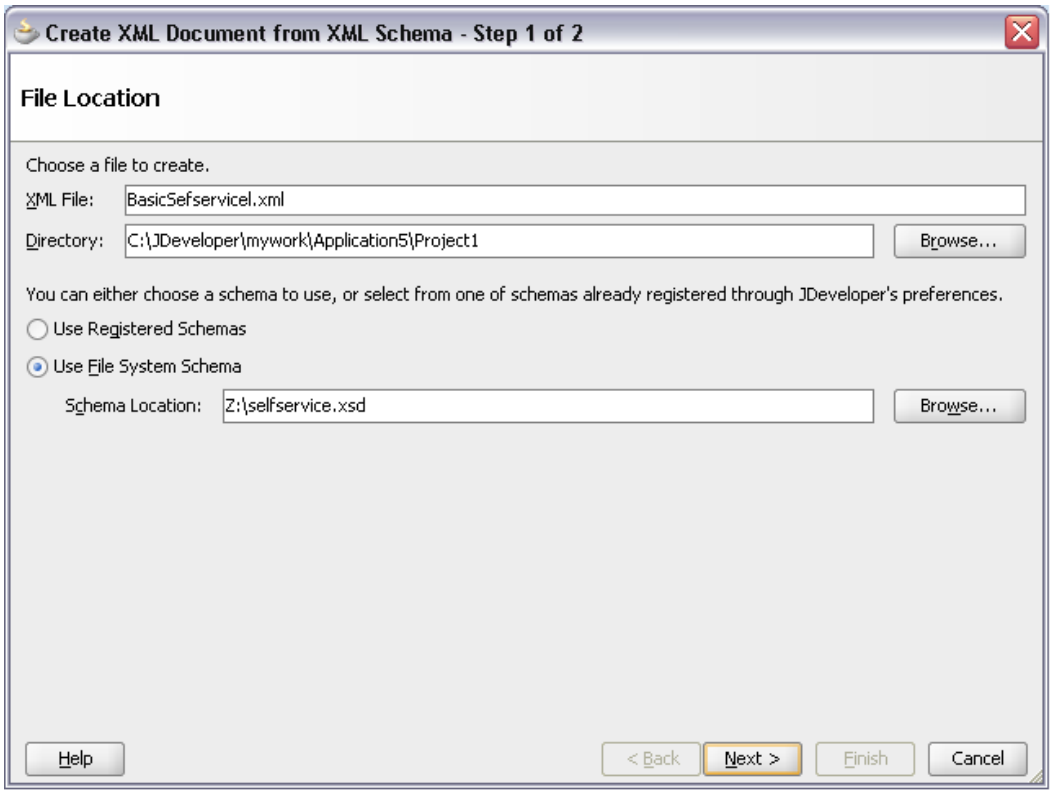
Create blank Self-Service XML from Schema

Using the above created XSD, XML has to be created. Most of the IDEs offers the way to create XML Document from XML Schema. For instance, consider Oracle JDeveloper to create XML.

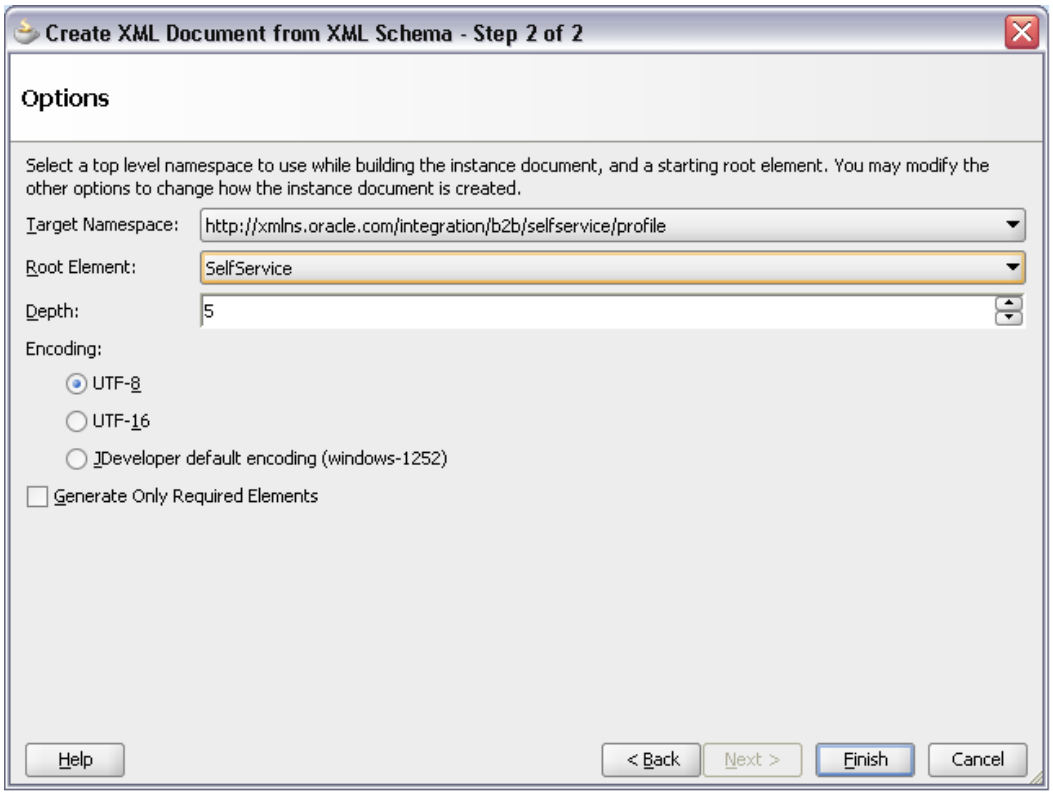
→ Create a Project or use existing project.
→ Right click on Project, and Select New.
→ Choose *XML* in Categories panel.
→ Choose *XML Document from XML Schema* in *Items* panel.



→ Choose ***selfservice.xsd*** and give XML file name.
→ Select Next.



→ Choose Root Element as ***SelfService*** and Depth as **5**.



→ Click Finish.

An XML is created with all possible elements and empty string as value in attributes.

Empty string in this blank XML has to be get filled with right value based on the Appendix in Oracle B2B documentation. In this XML, DocumentPrototocol, TradingPartner, and Agreement elements are created once by IDE. User needs to create these elements based on their business requirement.

Update Document protocol values in XML

In Oracle B2B, Document protocol details are captured in sub-elements like version, type and definition. This helps to reuse the configuration, for instance, once document protocol version defined, it can be reused by different document type.

→ Following document protocols are supported in Self-Service

Table E-1 Document Protocols

Document Protocol Name
EDI_X12
EDI_EDIFACT
HL7
RosettaNet
Custom

→ Update X12 Document protocol name

```
<DocumentProtocol name="EDI_X12">
```

→ Update Document version and type name

```
<DocumentProtocolVersion name="4010">
 <DocumentType name="850">
```

→ Give name to Document definition and location of 850.xsd

```
<DocumentDefinition customFileType="false"
  definitionFileName="schemas/850.xsd"
  name="850def" useDefaultDefinition="true">
```

Make sure, from Document Builder exported file placed in schema/850.xsd location. This schema folder has to be created in which this XML exist.

→ Now, DocumentProtocol element looks below

```
<DocumentProtocol name="EDI_X12">
 <DocumentProtocolVersion name="4010">
  <DocumentType name="850">
   <DocumentDefinition customFileType="false"
  definitionFileName="schemas/850.xsd"
  name="850def" useDefaultDefinition="true">
   </DocumentDefinition>
   <ParameterValue name="" />
  </DocumentType>
  <ParameterValue name="" />
 </DocumentProtocolVersion>
</DocumentProtocol>
```

Update ParameterValue for Document protocol

This document definition has to be customized by updating the ParameterValue details. Not necessary to capture all ParameterValue in XML from Appendix - EDI_X12 Document Protocol Parameter Values. Most of the ParameterValue has default value, which will be created implicitly while importing to Oracle B2B repository.

If business needs the different value, this has to be captured by choosing right ParameterValue with updating value attribute. In this use case, ParameterValue for *TransactionECSFileBlob* and *GroupID* needs to be changed.

→ Create TransactionECSFileBlob ParameterValue in DocumentDefition element

```
<DocumentDefinition customFileType="false"
  definitionFileName="schemas/850.xsd"
  name="850def" useDefaultDefinition="true">
 <ParameterValue name="TransactionECSFileBlob"
value="schemas/850.ecs" />
</DocumentDefinition>
```

→ In DocumentType element, update blank ParameterValue with GroupID

```
<DocumentType name="850">
   <DocumentDefinition customFileType="false"
  definitionFileName="schemas/850.xsd"
  name="850def" useDefaultDefinition="true">
  <ParameterValue name="TransactionECSFileBlob"
value="schemas/850.ecs" />
   </DocumentDefinition>
   <ParameterValue name="GroupID" value="PO" />
  </DocumentType>
```

→ In DocumentProtocolVersion element, remove blank ParameterValue element.

→ Add new document type and document defition for 997

```
<DocumentType name="997">
   <DocumentDefinition customFileType="false"
  definitionFileName="schemas/997.xsd"
  name="997def" useDefaultDefinition="true">
  <ParameterValue name="TransactionECSFileBlob"
   value="schemas/997.ecs" />
   </DocumentDefinition>
   <ParameterValue name="GroupID" value="FA" />
</DocumentType>
```

→ Now, DocumentProtocol updating completed and element looks like below

```
<DocumentProtocol name="EDI_X12">
 <DocumentProtocolVersion name="4010"
<DocumentType name="850">
   <DocumentDefinition customFileType="false"
  definitionFileName="schemas/850.xsd"
  name="850def" useDefaultDefinition="true">
  <ParameterValue name="TransactionECSFileBlob"
value="schemas/850.ecs" />
   </DocumentDefinition>
   <ParameterValue name="GroupID" value="PO" />
  </DocumentType>
  <DocumentType name="997">
   <DocumentDefinition customFileType="false"
  definitionFileName="schemas/997.xsd"
  name="997def" useDefaultDefinition="true">
  <ParameterValue name="TransactionECSFileBlob"
   value="schemas/997.ecs" />
   </DocumentDefinition>
   <ParameterValue name="GroupID" value="FA" />
</DocumentType>
</DocumentProtocolVersion>
</DocumentProtocol>
```

Update trading partner details in XML

Assume, Trading partner Acme receives document 850 and sends document 997 to GlobalChips. Hence, Two TradingPartner elements have to be created in Self-Service XML, 1) store the details of the Host, and 2) store the details of Remote Trading partner.

Update Host details in TradingPartner Element

In Host, details of Identification, listening delivery channel, and supported document definitions need to be captured.

→ In TradingPartner Element, update name and hosted = true.

```
<TradingPartner hosted="true" name="Acme">
```

→ Add Identification Elements, to capture three identifier details- *EDI Interchange ID, EDI Group ID*, and *EDI Interchange ID Qualifier*. List of identifications supported in Self-Service XML is captured in the Appendix of Oracle B2B documentation.

```
<Identification name="InterchangeID" value="Acme" />
<Identification name="GroupID" value="Acme" />
<Identification name="InterchangeIDQualifier" value="ZZ" />
```

Note: By default, Oracle B2B creates *Name* Identifier. Hence, no need of adding Identification element for *Name identifier.*

→ Update DeliveryChannel element to create a listening channel, which will receive message from trading partner.

```
<DeliveryChannel ackMode="None" compressed="false"
internal="false" listening="true" name="Acme_receive_file"
responseMode="None">
```

→ Update ExchangeProtocolRef element, with the name of exchange

```
<ExchangeProtocolRef name="Generic-File" />
```

Few exchange protocol has ParameterValue(s). These details captured in the section of *Exchange Protocols Parameter Values* in the Appendix of Oracle B2B documentation. For Generic-File exchange, no ParameterValue is available in Oracle B2B.

→ Update TransportProtocolRef element, with the name of transport, and update with possible ParameterValues

```
<TransportProtocolRef name="File">
<ParameterValue name="folder" value="/tmp/Acme_File_Inbound/" />
</TransportProtocolRef>
```

In this usecase, one ParameterValue has created to capture folder value. From this folder, 850 documents picked by Host and processed.

Other ParameterValue list is captured in *Transport Protocols ParameterValues* section in the Appendix of Oracle B2B documentation.

→ Complete DeliveryChannel element

```
<DeliveryChannel ackMode="None" compressed="false"
internal="false" listening="true" name="Acme_receive_file"
responseMode="None">
 <ExchangeProtocolRef name="Generic-File" />
 <TransportProtocolRef name="File">
  <ParameterValue name="folder" value="/tmp/Acme_File_Inbound/" />
 </TransportProtocolRef>
</DeliveryChannel>
```

→ Update SupportedDocumentDefinition element, to capture possible documents to send/receive. Document details whichever is captured in DocumentProtocol has to be used in this element's attribute. Two SupportedDocumentDefinition are created- 1) receive 850, and 2) send 997.

```
<SupportedDocumentDefinition docDefName="850def"
docProtocolName="EDI_X12" docProtocolVersion="4010"
docTypeName="850" initiator="false" />
<SupportedDocumentDefinition docDefName="997def"
docProtocolName="EDI_X12" docProtocolVersion="4010"
docTypeName="997" initiator="true" />
```

→ Host details completely captured.

```
<TradingPartner hosted="true" name="Acme">
 <Identification name="InterchangeID" value="Acme" />
 <Identification name="GroupID" value="Acme" />
 <Identification name="InterchangeIDQualifier" value="ZZ" />
 <DeliveryChannel ackMode="None" compressed="false" internal="false"
listening="true" name="Acme_receive_file" responseMode="None">
  <ExchangeProtocolRef name="Generic-File" />
  <TransportProtocolRef name="File">
   <ParameterValue name="folder" value="/tmp/Acme_File_Inbound/" />
  </TransportProtocolRef>
 </DeliveryChannel>
 <SupportedDocumentDefinition docDefName="850def"
docProtocolName="EDI_X12" docProtocolVersion="4010" docTypeName="850"
initiator="false" />
<SupportedDocumentDefinition docDefName="997def"
docProtocolName="EDI_X12" docProtocolVersion="4010" docTypeName="997"
initiator="true" />
</TradingPartner>
```

Add Remote TP details in TradingPartner Element

Similar to Host, in Remote TP, details of Identification, trading partner delivery channel, and supported document definitions need to be captured.

→ In TradingPartner Element, update name and hosted = true.

```
<TradingPartner hosted="false" name="GlobalChips">
```

→ Add Identification Elements, to capture three identifier details- *EDI Interchange ID, EDI Group ID*, and *EDI Interchange ID Qualifier*.

```
<Identification name="InterchangeID" value=" GlobalChips" />
<Identification name="GroupID" value=" GlobalChips" />
<Identification name="InterchangeIDQualifier" value="ZZ" />
```

Note: By default, Oracle B2B creates *Name* Identifier. Hence, no need of adding Identification element for *Name identifier.*

→ Update DeliveryChannel element to create a delivery channel, which will send message to trading partner.

```
<DeliveryChannel ackMode="None" compressed="false"
internal="false" listening="false"
name="GlobalChips_File_Endpoint" responseMode="None">
```

→ Update ExchangeProtocolRef element, with the name of exchange

```
<ExchangeProtocolRef name="Generic-File" />
```

→ Update TransportProtocolRef element, with the name of transport, and update with possible ParameterValues

```
<TransportProtocolRef name="File">
<ParameterValue name="folder"
      value="/tmp/GlobalChips_File_Outbound/" />
</TransportProtocolRef>
```

In this use case, one ParameterValue has created to capture folder value. Processed 997 documents will be placed in this folder by Host.

→ Complete DeliveryChannel element

```
<DeliveryChannel ackMode="None" compressed="false"
internal="false" listening="false"
name="GlobalChips_File_Endpoint" responseMode="None">
 <ExchangeProtocolRef name="Generic-File" />
 <TransportProtocolRef name="File">
  <ParameterValue name="folder"
value="/tmp/GlobalChips_File_Outbound" />
 </TransportProtocolRef>
</DeliveryChannel>
```

→ Update SupportedDocumentDefinition element, to capture possible documents to send/receive. Document details whichever is captured in DocumentProtocol has to be used in this element's attribute. Two SupportedDocumentDefinition are created- 1) send 850, and 2) receive 997.

```
<SupportedDocumentDefinition docDefName="850def"
docProtocolName="EDI_X12" docProtocolVersion="4010"
docTypeName="850" initiator="true" />
<SupportedDocumentDefinition docDefName="997def"
docProtocolName="EDI_X12" docProtocolVersion="4010"
docTypeName="997" initiator="false" />
```

→ Trading partner details completely captured.

```
<TradingPartner hosted="false" name="GlobalChips">
 <Identification name="InterchangeID" value="GlobalChips" />
 <Identification name="InterchangeIDQualifier" value="ZZ" />
 <Identification name="GroupID" value="GlobalChips" />
 <DeliveryChannel ackMode="None" compressed="false" internal="false"
listening="false" name="GlobalChips_File_Endpoint"
responseMode="None">
 <ExchangeProtocolRef name="Generic-File" />
 <TransportProtocolRef name="File">
  <ParameterValue name="folder"
value="/tmp/GlobalChips_File_Outbound" />
 </TransportProtocolRef>
 </DeliveryChannel>
 <SupportedDocumentDefinition docDefName="850def"
docProtocolName="EDI_X12" docProtocolVersion="4010"
docTypeName="850" initiator="true" />
 <SupportedDocumentDefinition docDefName="997def"
docProtocolName="EDI_X12" docProtocolVersion="4010"
docTypeName="997" initiator="false" />
</TradingPartner>
```

Update agreement details in XML

Document protocol and trading partner details are captured in previous sections. Using these details, agreements has to be created.

Update Inbound Agreement to receive 850 Document

In Oracle B2B, Identification, Delivery channel, and Document details from Host/TP used to create an agreement.

→ Update Agreement element with name and agreementID.

```
<Agreement agreementId="GlobalChips_Acme_X12_4010_850_File"
                  name="GlobalChips_Acme_X12_4010_850_File">
```

Name and agreementId need not to be same always. In ebMS usecase, agreementId is used as CPAID.

→ Update SupportedDocumentType element with document details.

```
<SupportedDocumentType docDefName="850def"
docProtocolName="EDI_X12" docProtocolVersion="4010"
docTypeName="850">
```

Name used in DocumentDefinition element, has to be given to `docDefName` attribute.

→ Update InitiatingParticipant and RespondingParticipant.

Acme receives Document 850 from GlobalChips. GlobalChips details have to go on InitiatingParticipant, and Acme details have to go on RespondingParticipant.

→ Update InitiatingParticipant.

```
<InitiatingParticipant name="GlobalChips">
 <Identifications>
  <IdentificationRef name="GroupID" value="GlobalChips" />
  <IdentificationRef name="InterchangeID" value="GlobalChips" />
  <IdentificationRef name="InterchangeIDQualifier" value="ZZ" />
 </Identifications>
</InitiatingParticipant>
```

Note: Listening channels receives the messages from trading partner. Listening channels works agnostic to agreement and trading partner and are not required to refer from agreement.

→ Update RespondingParticipant.

```
<RespondingParticipant name="Acme">
 <Identifications>
  <IdentificationRef name="GroupID" value="Acme" />
  <IdentificationRef name="InterchangeID" value="Acme" />
  <IdentificationRef name="InterchangeIDQualifier" value="ZZ" />
 </Identifications>
</RespondingParticipant>
```

→ Update AgrDocTypeParameterValues, to specify Functional Acknowledgement, Translation, and Validation required or not.

```
<AgrDocTypeParameterValues>
 <ParameterValue name="validate" value="true" />
 <ParameterValue name="translate" value="true" />
 <ParameterValue name="fa" value="true" />
</AgrDocTypeParameterValues>
```

Note: If *FA* (Functional Acknowledgement) or *validate* enabled, then *translate* must to be enabled.

→ Inbound Agreement for document 850 created successfully

```
<Agreement agreementId="GlobalChips_Acme_X12_4010_850_File"
                  name="GlobalChips_Acme_X12_4010_850_File">
<SupportedDocumentType docDefName="850def"
docProtocolName="EDI_X12" docProtocolVersion="4010"
docTypeName="850">
<InitiatingParticipant name="GlobalChips">
 <Identifications>
  <IdentificationRef name="GroupID" value="GlobalChips" />
  <IdentificationRef name="InterchangeID" value="GlobalChips" />
  <IdentificationRef name="InterchangeIDQualifier" value="ZZ" />
 </Identifications>
</InitiatingParticipant>
<RespondingParticipant name="Acme">
 <Identifications>
  <IdentificationRef name="GroupID" value="Acme" />
  <IdentificationRef name="InterchangeID" value="Acme" />
  <IdentificationRef name="InterchangeIDQualifier" value="ZZ" />
 </Identifications>
</RespondingParticipant>

<AgrDocTypeParameterValues>
 <ParameterValue name="validate" value="true" />
 <ParameterValue name="translate" value="true" />
 <ParameterValue name="fa" value="true" />
</AgrDocTypeParameterValues>
</SupportedDocumentType>
</Agreement>
```

Note: If no internal delivery channel found in Host participant, then any messages received using this agreement will send to *Fabric/JMS/AQ* based on the System Configuration set in Oracle B2B UI.

Add Outbound Agreement to send 997 Document

Similar to inbound agreement, details like Identification, delivery channel, and document details used from Host/TP, to create an agreement. In outbound agreement, delivery channel must to be specified in RespondingParticipant element.

→ Update Agreement element with name and agreementID.

```
<Agreement agreementId="Acme_GlobalChips_X12_4010_997_File"
                  name=" Acme_GlobalChips_X12_4010_997_File">
```

→ Update SupportedDocumentType element with document details.

```
<SupportedDocumentType docDefName="997def"
docProtocolName="EDI_X12" docProtocolVersion="4010"
docTypeName="997">
```

→ Update InitiatingParticipant and RespondingParticipant.

Acme sends document 997 to GlobalChips. Acme details have to go on InitiatingParticipant, and GlobalChips details have to go on RespondingParticipant.

→ Update InitiatingParticipant.

```
<InitiatingParticipant name="Acme">
 <Identifications>
  <IdentificationRef name="GroupID" value="Acme" />
  <IdentificationRef name="InterchangeID" value="Acme" />
  <IdentificationRef name="InterchangeIDQualifier" value="ZZ" />
 </Identifications>
</InitiatingParticipant>
```

Note: Listening channels receives the messages from trading partner. Listening channels works agnostic to agreement and trading partner and are not required to refer from agreement.

→ Update RespondingParticipant.

```
<RespondingParticipant name="GlobalChips">
 <Identifications>
  <IdentificationRef name="GroupID" value="GlobalChips" />
  <IdentificationRef name="InterchangeID" value="GlobalChips" />
  <IdentificationRef name="InterchangeIDQualifier" value="ZZ" />
 </Identifications>
 <DeliveryChannels>
  <DeliveryChannelRef name="GlobalChips_File_Endpoint" />
 </DeliveryChannels>
</RespondingParticipant>
```

→ Update AgrDocTypeParameterValues, to specify Functional Acknowledgement, Translation, and Validation required or not.

```
<AgrDocTypeParameterValues>
 <ParameterValue name="validate" value="true" />
 <ParameterValue name="translate" value="true" />
 <ParameterValue name="fa" value="false" />
</AgrDocTypeParameterValues>
```

→ Outbound Agreement for document 997 created successfully

```
<Agreement agreementId="Acme_GlobalChips_X12_4010_997_File"
                  name=" Acme_GlobalChips_X12_4010_997_File">
<SupportedDocumentType docDefName="997def"
docProtocolName="EDI_X12" docProtocolVersion="4010"
docTypeName="997">
<InitiatingParticipant name="Acme">
 <Identifications>
  <IdentificationRef name="GroupID" value="Acme" />
  <IdentificationRef name="InterchangeID" value="Acme" />
  <IdentificationRef name="InterchangeIDQualifier" value="ZZ" />
 </Identifications>
</InitiatingParticipant>
<RespondingParticipant name="GlobalChips">
 <Identifications>
  <IdentificationRef name="GroupID" value="GlobalChips" />
  <IdentificationRef name="InterchangeID" value="GlobalChips" />
  <IdentificationRef name="InterchangeIDQualifier" value="ZZ" />
 </Identifications>
 <DeliveryChannels>
  <DeliveryChannelRef name="GlobalChips_File_Endpoint" />
 </DeliveryChannels>
</RespondingParticipant>

<AgrDocTypeParameterValues>
 <ParameterValue name="validate" value="true" />
 <ParameterValue name="translate" value="true" />
 <ParameterValue name="fa" value="false" />
</AgrDocTypeParameterValues>
</SupportedDocumentType>
</Agreement>
```

Note: IdentificationRef elements in Agreement are subset of Identification in TradingPartner.

Export ECS/XSD files from Oracle B2B Document Editor

Oracle B2B - Document Editor (Powered by EdifecsTM) is an integrated guideline creation and for defining and managing custom document definitions for EDI transactions.

Guidelines was created in Oracle B2B Document Editor has to be exported in XSD and ECS format and placed in schemas folder. In this use case, EDI X12 v4010 850 and EDI X12 v4010 997 are the guidelines needed – 850.ecs, 850.xsd, 997.xsd, and 997.ecs.

Note: Relative path used to locate schemas folder in this use case. This schemas folder has to be created in which folder Self-Service XML exist.

Complete Self-Service XML

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

xmlns="http://xmlns.oracle.com/integration/b2b/selfservice/profile">
 <DocumentProtocols>
 <DocumentProtocol name="EDI_X12">
 <DocumentProtocolVersion name="4010">
 <DocumentType name="850">
 <DocumentDefinition customFileType="false"
definitionFileName="schemas/850.xsd" name="850def"
useDefaultDefinition="true">
 <ParameterValue name="TransactionECSFileBlob"
value="schemas/850.ecs" />
 </DocumentDefinition>
 <ParameterValue name="GroupID" value="PO" />
 </DocumentType>
 <DocumentType name="997">
 <DocumentDefinition customFileType="false"
definitionFileName="schemas/997.xsd" name="997def"
useDefaultDefinition="true">
 <ParameterValue name="TransactionECSFileBlob"
value="schemas/997.ecs" />
 </DocumentDefinition>
 <ParameterValue name="GroupID" value="FA" />
 </DocumentType>
 </DocumentProtocolVersion>
 </DocumentProtocol>
 </DocumentProtocols>
 <TradingPartners>
 <TradingPartner hosted="true" name="Acme">
 <Identification name="InterchangeID" value="Acme" />
 <Identification name="GroupID" value="Acme" />
 <Identification name="InterchangeIDQualifier" value="ZZ" />
 <DeliveryChannel ackMode="None" compressed="false" internal="false"
listening="true" name="Acme_receive_file" responseMode="None">
 <ExchangeProtocolRef name="Generic-File" />
 <TransportProtocolRef name="File">
 <ParameterValue name="folder" value="/tmp/Acme_File_Inbound/" />
 </TransportProtocolRef>
 </DeliveryChannel>
 <SupportedDocumentDefinition docDefName="850def"
docProtocolName="EDI_X12" docProtocolVersion="4010"
docTypeName="850" initiator="false" />
 <SupportedDocumentDefinition docDefName="997def"
docProtocolName="EDI_X12" docProtocolVersion="4010"
docTypeName="997" initiator="true" />
 </TradingPartner>
 <TradingPartner hosted="false" name="GlobalChips">
 <Identification name="InterchangeID" value="GlobalChips" />
 <Identification name="InterchangeIDQualifier" value="ZZ" />
 <Identification name="GroupID" value="GlobalChips" />
 <DeliveryChannel ackMode="None" compressed="false" internal="false"
listening="false" name="GlobalChips_File_Endpoint"
responseMode="None">
 <ExchangeProtocolRef name="Generic-File" />
 <TransportProtocolRef name="File">
 <ParameterValue name="folder"
 value="/tmp/GlobalChips_File_Outbound" />
 </TransportProtocolRef>
 </DeliveryChannel>
 <SupportedDocumentDefinition docDefName="850def"
docProtocolName="EDI_X12" docProtocolVersion="4010"
docTypeName="850" initiator="true" />
 <SupportedDocumentDefinition docDefName="997def"
docProtocolName="EDI_X12" docProtocolVersion="4010"
docTypeName="997" initiator="false" />
 </TradingPartner>
 </TradingPartners>
```

```
<Agreements>
<Agreement agreementId="GlobalChips_Acme_X12_4010_850_File"
name="GlobalChips_Acme_X12_4010_850_File">
<SupportedDocumentType docDefName="850def"
docProtocolName="EDI_X12" docProtocolVersion="4010"
docTypeName="850">
<InitiatingParticipant name="GlobalChips">
<Identifications>
<IdentificationRef name="GroupID" value="GlobalChips" />
<IdentificationRef name="InterchangeID" value="GlobalChips" />
<IdentificationRef name="InterchangeIDQualifier" value="ZZ" />
</Identifications>
</InitiatingParticipant>
<RespondingParticipant name="Acme">
<Identifications>
<IdentificationRef name="GroupID" value="Acme" />
<IdentificationRef name="InterchangeID" value="Acme" />
<IdentificationRef name="InterchangeIDQualifier" value="ZZ" />
</Identifications>
</RespondingParticipant>
<AgrDocTypeParameterValues>
<ParameterValue name="validate" value="true" />
<ParameterValue name="translate" value="true" />
<ParameterValue name="fa" value="true" />
</AgrDocTypeParameterValues>
</SupportedDocumentType>
</Agreement>

<Agreement agreementId="Acme_GlobalChips_X12_4010_997_File"
name="Acme_GlobalChips_X12_4010_997_File">
<SupportedDocumentType docDefName="997def"
docProtocolName="EDI_X12" docProtocolVersion="4010"
docTypeName="997">
<InitiatingParticipant name="Acme">
<Identifications>
<IdentificationRef name="GroupID" value="Acme" />
<IdentificationRef name="InterchangeID" value="Acme" />
<IdentificationRef name="InterchangeIDQualifier" value="ZZ" />
</Identifications>
</InitiatingParticipant>
<RespondingParticipant name="GlobalChips">
<Identifications>
<IdentificationRef name="GroupID" value="GlobalChips" />
<IdentificationRef name="InterchangeID" value="GlobalChips" />
<IdentificationRef name="InterchangeIDQualifier" value="ZZ" />
</Identifications>
<DeliveryChannels>
<DeliveryChannelRef
name="GlobalChips_File_Endpoint" />
</DeliveryChannels>
</RespondingParticipant>
<AgrDocTypeParameterValues>
<ParameterValue name="validate" value="true" />
<ParameterValue name="translate" value="true" />
<ParameterValue name="fa" value="false" />
</AgrDocTypeParameterValues>
</SupportedDocumentType>
</Agreement>
</Agreements>
</SelfService>
```

Create Oracle B2B Metadata using Self-Service Utility

Self-Service XML was created successfully for EDI X12 v4010 850 inbound, EDI X12 v4010 997 outbound. This XML needs to be converted to Oracle B2B metadata format before getting imported to Oracle B2B.

In SOA_HOME location, all the command-line utilities for Oracle B2B are bundled in single ant script. Using this ANT script's target, Self-Service XML will be converted to a ZIP file, which will be the format of Oracle B2B metadata.

Prerequisite: Set environment variables

Variable Name	Value
ORACLE_HOME	/scratch/<env.USER>/fmwhome/AS11gR1SOA
ANT_HOME	$ORACLE_HOME/../modules/org.apache.ant_1.7.0
JAVA_HOME	$ORACLE_HOME/../jdk160_11

→ Run ant command to create Oracle B2B metadata

```
ant -f ant-b2b-util.xml b2bselfservice
-Dinput="/tmp/selfservice1.xml" -Doutput="soa.zip"
```

A file with name soa.zip will get created in the ORACLE_HOME/bin location.

→ Oralce metadata is in importable format for Oracle B2B and can be imported either using Oracle B2B UI or Command-line approach.

Additional features, Tips and Tricks

This section explains few of the advanced topics as well as few of the tips and tricks to handle the some of the complex Self-Service scenarios

Creating MLLP channel

In Oracle B2B, all the MLLP delivery channels (TCP listening/establish) are created under the Host with attribute listening=true. This channel can be used in an agreement for the remote participant (Initiating/responding), if this trading partner name is given in delivery channel attribute *createdBy*.

For instance, Acme receives message from GlobalChips using MLLP channel with local port number 7777, then delivery channel has to create in Acme as below.

```
<DeliveryChannel createdBy="GlobalChips" compressed="false"
responseMode="None" ackMode="None" name="HL7Ext"
internal="false" listening="true">
 <ExchangeProtocolRef name="MLLP">
 <ParameterValue value="None" name="ImmediateACK"/>
 <ParameterValue value="0x0b" name="Start-Block-Char"/>
 <ParameterValue value="0x1c" name="End-Block-Char"/>
 <ParameterValue value="0x0d" name="Carriage-Return-Char"/>
 </ExchangeProtocolRef>
 <TransportProtocolRef name="TCP">
 <ParameterValue value="Server" name="tcp-param-sockettype"/>
 <ParameterValue value="localhost" name="tcp-param-host"/>
 <ParameterValue value="7777" name="tcp-param-port"/>
 <ParameterValue value="true" name="tcp-param-
PermanentConnectionType"/>
 </TransportProtocolRef>
</DeliveryChannel>
```

Using Folder based input approach

In Self-Service utility, complete Self-Service XML is given as a single XML input. While creating 1000+ agreement, it is tedious to prefer an XML with all these details. Instead of pointing to single XML file in input argument, folder can be pointed.

```
ant -f ant-b2b-util.xml b2bselfservice
-Dinput="/tmp/ss_folder" -Doutput="soa.zip"
```

While trying with multiple file approach, user has to make sure that name of the self-service xml files are in following order

1. Document protocols self-service XMLs
2. Trading partners self-service XMLs (*HOST must be first in this list*)
3. Agreements self-service XMLs

Otherwise, there could be a possibility of error thrown with referenced object does not exist. For instances, create file names as below

doc_selfservice.xml to hold document protocols.
tp_selfservice.xml to hold trading partner details.
tpa_selfservice.xml to hold agreements details.

Note: All these files have to have SelfService element as root element.

Typical Self-Service Errors

1. **Transport protocol is not supported for exchange**

   Make sure that chosen transport protocol and exchange protocol are supported combination in Oracle B2B.

2. **File does not exist**

   Make sure that XSD/ECS files located in document protocol exist and accessible in file system.

3. **Identification type and value combination must be unique in trading partner: {0} type: {1} value: {2}**

   Identification's type and value combination must be unique for trading partner. Or May be identification is created with Name identifier in Self-Service XML. Self-Service XML automatically creates Name Identifier implicitly; hence it is not required to type explicitly in Self-Service XML.

4. **Delivery channel name must be unique in trading partner**

   Make sure that delivery channel name is unique in trading partner.

5. **No SupportedDocumentType found for Agreement**

   Make sure that SupportedDocumentType element exist in Agreement element.

6. **Identification type={0} and value={1} specified in agreement {2}, does not exist in trading partner**

   IdentificationRef element's attributes type and value in agreement combination does not exist in Indentification element's attributes of trading partner. Make sure, trading partner has this type-value combination.

7. **ID does not found for Parametervalue Name {0}**

   Make sure that there is no typo error in ParameterValue name. Make Double check with ParameterValue list in Oracle B2B Documentation Appendix.

FAQ

1. **What are the formats can be used to give value for name attribute in Identification element?**

   Three, For instance, consider *EDI Interchange ID,* following are the formats used to create identification.

```
1. <Identification name="InterchangeID" value="Acme" />
2. <Identification name="itype-InterchangeID" value="Acme" />
3. <Identification name="EDI Interchange ID" value="Acme" />
```

2. **Do we need to specify as IdentificationRef in agreement for all the Identification in TradingPartner Element?**

   No. IdentificationRef list is subset of Identification list.

3. **Do we need to specify mandatorily output folder for Self-Service Utility?**

   No.

4. **Does only relative path allowed to specify XSD and ECS file?**

   No. Absolute path also allowed locating XSD and ECS file. Absolute path has to start with forward slash (/). Hence, in windows box, XSD / ECS have to be located in same drive.

5. **Does Self-Service utility validates Self-service XML before converting to Oracle B2B metadata?**

   Yes. All the XSD validation error found in Self-Service XML listed.

6. **What are the elements not possible to create using Self-Service?**

   Callout.xml, types.xml, User element, and system properties are not possible to create using Self-Service. Oracle B2B UI provides easy way to configure this and most of the user experience these objects get created one time and get reused.

7. **Is it possible to create only document protocol details?**

   Yes.

8. **Is it possible to create only trading partner details?**

   No. Trading partner details need document protocol details to create SupportedDocumentDefinition. Hence, Document protocol details have to exist in to create trading partner details. Similarly, agreement creation needs trading partner and document protocol details.

Reference Information

Links

Oracle B2B
http://www.oracle.com/technology/products/integration/b2b/index.html

Oracle B2B Forum
Forum: http://forums.oracle.com/forums/forum.jspa?forumID=242

Fusion Middleware
http://www.oracle.com/technology/products/middleware/index.html

Service-Oriented Architecture
http://www.oracle.com/technologies/soa/index.html

Oracle B2B 11g Self-Service Appendix
http://download.oracle.com/docs/cd/E14571_01/integration.1111/e10229/app_selfservice.htm

This technote has been assembled using excerpts from:

- Oracle B2B User Manual
- Oracle B2B Partner Training Curriculum
- Oracle B2B Technical Notes
- Oracle B2B Tutorials

Chief Editor:
Krishnamoorthy Dharmalingam

Contributors:
Sundararaman Shenbagam
Ramesh Anantharamaiah
Sankar Mani
Jeffrey Hutchins
Ted Hong

Lasted Updated: May 24, 2011

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com

Oracle Corporation provides the software
that powers the Internet.