

***Tutorial #003:
Building ebXML messaging with
Oracle B2B 11g***

Contents

<i>Introduction</i>	3
<i>Building ebXML messaging in Oracle B2B</i>	4
Agreement identification	5
Message Correlation and Conversation.....	6
Acknowledgement Modes in Oracle B2B.....	7
ebMS Security in Oracle B2B	7
<i>Advanced ebMS features</i>	14
Attachment Feature.....	14
Duplicate Elimination	16
Message Header Validation	17
Message Status Service.....	17
Message Service Handler Ping/Pong	19
CPP/CPA	20
Performance Best Practices	24
Large Payload Configurations	24
<i>Typical ebMS Errors</i>	25
<i>FAQ</i>	25
<i>Reference</i> :.....	25

Introduction

ebXML – Electronic Business using Extensible Markup Language (ebXML) is a standard framework created by UN/CEFACT and OASIS. The ebXML is designed to enable a global electronic marketplace in which enterprises of any size, and in any location, could safely and securely transact business through the exchange of XML-based messages.

ebXML has five main areas of focus:

- Business processes
- Register business processes
- Define trading relationships
- Common terms of business data
- Reliably exchange business messages

ebXML Cookbook is to provide an introduction to ebXML standards, Oracle B2B offerings of the same along with some of the best practices.

Purpose

The main purpose of this book is to get basic understanding of the ebXML standards, learn Oracle B2B ebXML offering and user experience while configuring ebXML protocol and its various use cases including error/exception scenario.

After reading this book, users should be able to independently model ebXML use cases such as correlation, conversation, security and end-to-end scenario using ebXML messages.

Audience

ebXML Cookbook is intended for B2B users who want to Exchange ebXML messages using Oracle B2B as an Integration Gateway product and would like to understand various use cases and implementation.

Prerequisite

Basic knowledge of SOAP, XML and ebXML Standards.

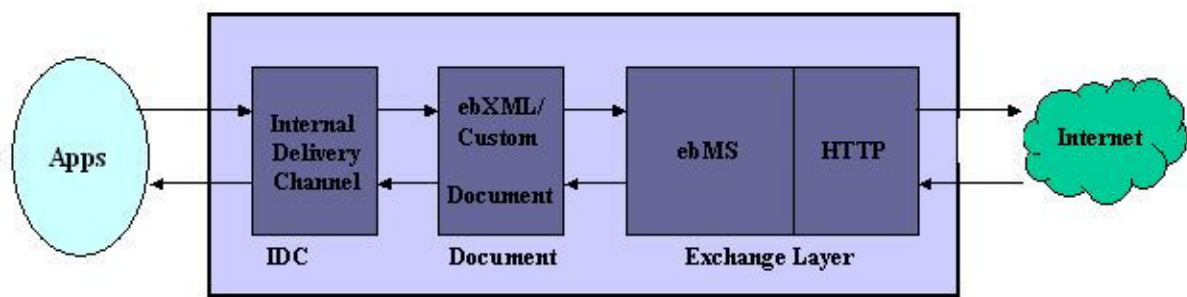
Assumptions

This document is based on Oracle B2B AS11gR1. Most of the Use cases mentioned in this document is based on ebXML standard using Oracle BPEL Process Manager/Mediator as the Back end Application.

Building ebXML messaging in Oracle B2B

As mentioned earlier ebXML document is supported through Custom document plugin and implementation in Oracle B2B follows the below pattern.

Below diagram explains various components in ebMS message processing. It comprises Exchange layer for security and correlation and document layer for validation. Lets learn the components in detail.



ebMS Message flow in Oracle B2B Configuration

Exchange layer:

When HTTP receiver receives a message, Oracle B2B will identify the ebMS exchange using the following http headers

- 1) SOAPAction="ebXML"
- 2) MIME-Version

If the ebMS exchange is identified successfully using the above headers then Oracle B2B check whether the soap message has a security headers like

- 1) EncryptedData (XMLENC)
- 2) Signature (XMLDSIG)

Based on the above security details the soap message will be decrypted using the decryption algorithm specified and if it has a Signature then verify the soap message and its attachments using the certificate attached in the soap message or in the configured key store.

It will also verify the mandatory headers like From, To, CPAlid, ConversationId, Action, Service, and MessageData.

If the message is processed successfully in the ebMS exchange layer then Oracle B2B sends a **positive Acknowledgement** to the sender.

If there is an error while processing the message in the ebMS layer then Oracle B2B sends a **Negative Acknowledgement** with the **Errorlist** to the sender as per the ebMS specification.

Document layer:

In the Document layer document will be validated against the schema configured during design time. It will also convert the payload in case of DTD definition to XSD definitions.

Typically all the ebMS headers from the incoming message will be validated against the configured values.

Internal Delivery Channel:

Then the Internal Delivery channel will send/receive the message from/to the backend applications like BPEL/Mediator.

Agreement identification

Following are the various mechanisms with which the agreement would be identified using ebMS protocol.

Outbound scenario:

a) From, To, Document Type and Document Revision:

In Oracle B2B, it is possible to identify the agreement using these attributes

Eg. AQ header Property

```
doctypeName = ORDERS  
doctypeRevision = 1.0
```

b) Action and Service based Identification

If document Type and Document Revision is not present and if the ebMS action and service is available, which is enqueued using the ACTION attribute, then the ebMS action and service attribute is used to identify the agreement.

e.g.

```
actionName=ACTION:ebMSRequest;Service:FileTransfer;ServiceType:String
```

⚡ **Note:** **actionName** and **eventName** can be used interchangeably.

In case of backend application like fabric then eg.

```
b2b.action=ebMSRequest  
b2b.ebms.Service=FileTransfer  
b2b.ebms.ServiceType =String
```

Inbound Scenario

The Agreement identification mechanism for ebXML in Oracle B2B will follow the below order

a) Action, Service and Service Type based Identification

If the Action and Service matches with the ebMS header Action, Service and Service Type of the incoming message, then the respective Document Type and Document Revision will be used to identify the agreement.

b) From, To, Document Type and Document Revision:

In Oracle B2B, it is possible to identify the agreement using these attributes. Since Oracle B2B supports custom over ebMS Business protocol, applying the Xpath expression on the payload can retrieve the custom Document Type and Document Revision

Message Correlation and Conversation

The acknowledgement messages will be correlated with the actual business messages using RefToMessageId on the received Acknowledgement message.

Oracle b2b also supports multiple conversation based on the conversation Id. One or more request or responses can be grouped under single conversation ID. The request and response messages can be exchanged seamlessly with the same conversation ID. The following properties need to be configure in enterprise manager(em) with respect to ebMS conversations.

```
<property>
  <name>oracle.tip.b2b.ebmsDeliverConvId</name>
  <value>true</value>
</property>
```

The following AQ/JMS header property required for ebXML Conversation,

```
replyToMsgID=<MsgId>:<ConvId>
```

⚡ Note: If the ConvId is empty for the first message of the conversation then B2B will treat the MessageId as conversationID.

Acknowledgement Modes in Oracle B2B

The delivery channels can be configured for Sync/Async/None acknowledgement messages. In case of Sync mode the acknowledgement message will be expected in the sync manner (i.e. on the same session/connection) whereas in case of Async mode the acknowledgement message will be expected in the Async manner (i.e. new session/connection). If the sender is not interested in acknowledgement then the mode “none” can be used.

ebMS Security in Oracle B2B

ebMS Message level Security

Oracle B2B ebMS offers the following security features,

1. Encryption - XMLENC
2. Digital Signature - XMLDSIG

It is necessary to configure the keystore details as a pre-requisite for implementing security features. This will be used while configuring the delivery channel.

Keystore configuration

Provide the java key store (jks) information for Security in the Host trading partner Profile.

→ Click “**Partners**” Tab to add the channel details

→ Click **Profile** to configure keystore information (refer to screen below)



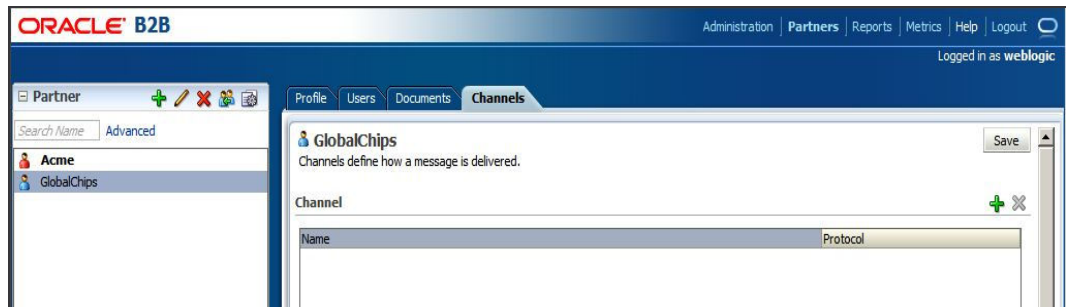
The screenshot shows a 'Key Store' configuration window. It has a title bar 'Key Store'. Below the title bar, there are three input fields: 'Password' (masked with seven dots), 'Confirm Password' (masked with seven dots), and 'Location' (containing the text '/tmp/soa/b2b/acme.jks').

Provide the location of the jks, which contains the host certificate and trading partner certificate.

Provide the keystore password.

Setting up the Trading Partner Delivery Channel

- Click “Channels” Tab to add the channel details
- Click (+) to add Channels for responder (refer to screen below)

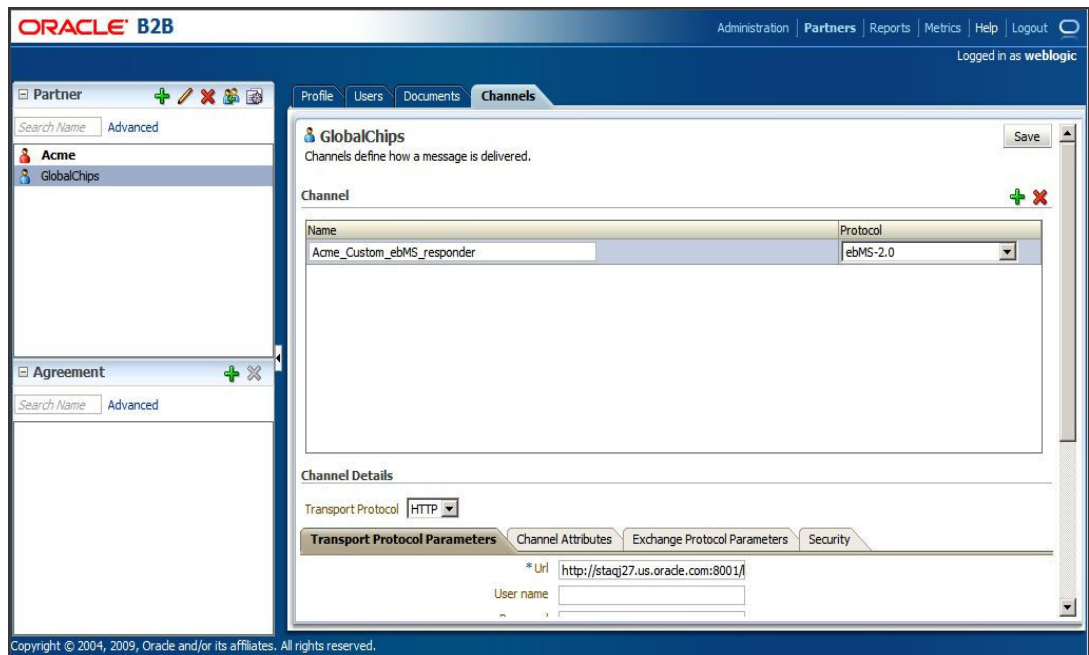


- Provide the Channel Name as “Acme_Custom_ebMS_responder”
- Select the ebMS Channel

Provide the following information

Field	Value
Name	Acme_Custom_ebMS_responder
Protocol	ebMS-2.0

- Provide URL of the remote trading partner
eg. <http://staxx.us.oracle.com:8001/b2b/httpReceiver>



→ Make sure the Ack Mode is “Async” or “Sync”

The screenshot shows the 'Channel Details' dialog box with the 'Channel Attributes' tab selected. The 'Transport Protocol' is set to 'HTTP'. Under 'Channel Attributes', 'Ack Mode' is set to 'Async', 'Response Mode' is set to 'None', and 'Compressed' is unchecked. There are input fields for 'Retry Interval', 'Retry Count', and 'Description'.

→ Select the Security tab to configure the security specifications

The screenshot shows the 'Channel Details' dialog box with the 'Security' tab selected. The 'Transport Protocol' is 'HTTP'. Under 'Security', 'Ack Signed', 'Message Signed', and 'Message Encrypted' are all unchecked. The 'Security Specifications' section is expanded, showing 'Digital Signature' set to 'XMLDSIG with SHA1 - RSA' and 'acme', and 'Encryption' set to 'XMLENC with 3DES - RSA-v1.5' and 'acme'.

Following are the various options

- **Ack Signed**

Enable to receive a signed Acknowledgement.

- **Message Signed**

Enable this feature, to send the signed message and select the appropriate Digital Signature algorithm and host certificate alias.

- **Message Encrypted**

Enable this feature, to send the encrypted message and select the appropriate Encryption algorithm and trading partner certificate alias.

→ Save the newly added Channel

There is a need to select Digital Envelope algorithm depending on whether encryption is enabled.

B2B uses the certificate from the keystore for both signing and encryption and also a lookup to the keystore for Private key, hence there is a need to import the Host/Trading partner certificate into the keystore.

ebMS Security using SSL

This is an enterprise weblogic-SSL configuration using which the ebMS message can also be exchange with the trading partner. In this case the whole channel will be secured(encrypted), as against the message level security discussed in 6.4.1

Configure SSL

- Login to Weblogic Administration Console Page.
- Under Domain Structures, select Servers and click **soa_server1**.

The screenshot shows the Oracle WebLogic Server Administration Console interface. The left sidebar contains a 'Domain Structure' tree with 'Servers' selected. The main content area displays the 'Summary of Servers' page, which includes a table of configured servers. The 'soa_server1' server is highlighted with a checkmark in the selection column.

Name	Cluster	Machine	State	Health	Listen Port
AdminServer(admin)			RUNNING	OK	7001
<input checked="" type="checkbox"/> soa_server1		LocalMachine	RUNNING	OK	8001

c. In the General page, enable the SSL Listen Port, by default the port is 8002.

The screenshot shows the 'Settings for soa_server1' page in the 'General' tab. The 'SSL Listen Port Enabled' checkbox is checked, and the 'SSL Listen Port' is set to 8002. Other settings include 'Listen Port Enabled' (checked) and 'Listen Port' (8001). The left sidebar shows the 'Domain Structure' and 'System Status'.

d. Save the configuration.

e. Select Keystore, Provide the keystore as “Custom Identity and Custom Trust”.

The screenshot shows the 'Keystores' tab in the 'Settings for soa_server1' page. The 'Keystores' dropdown is set to 'Custom Identity and Custom Trust'. The 'Custom Identity Keystore' and 'Custom Trust Keystore' are both set to '/home/hansrini/custom/Gli'. The 'Custom Identity Keystore Type' and 'Custom Trust Keystore Type' are both set to 'JKS'. The left sidebar shows the 'Domain Structure' and 'System Status'.

- f. Select SSL, Provide Private key Alias and Confirm Private key Passphrase.

The screenshot shows the JBoss Management Console interface for configuring SSL on a server instance named 'soa_server1'. The left sidebar contains several panels: 'View changes and restarts', 'Domain Structure' (showing a tree view of the domain), 'How do I...' (with links to configuration guides), and 'System Status' (showing health of running servers). The main content area has a breadcrumb 'Home > Summary of Servers > soa_server1' and a title 'Settings for soa_server1'. Below the title are tabs for 'Configuration', 'Protocols', 'Logging', 'Debug', 'Monitoring', 'Control', 'Deployments', 'Services', 'Security', and 'Notes'. The 'Configuration' tab is active, and within it, the 'SSL' sub-tab is selected. A 'Save' button is visible. The main configuration area contains a text box explaining the purpose of the page. Below this are several sections: 'Identity and Trust Locations' with a dropdown menu set to 'Keystores'; 'Identity' section with 'Private Key Location' (set to 'from Custom Identity Keystore'), 'Private Key Alias' (set to 'gc'), 'Private Key Passphrase' (masked with dots), and 'Confirm Private Key Passphrase' (masked with dots); 'Trust' section with 'Certificate Location' (set to 'from Custom Identity Keystore') and 'Trusted Certificate Authorities' (set to 'from Custom Trust Keystore'). An 'Advanced' link is at the bottom.

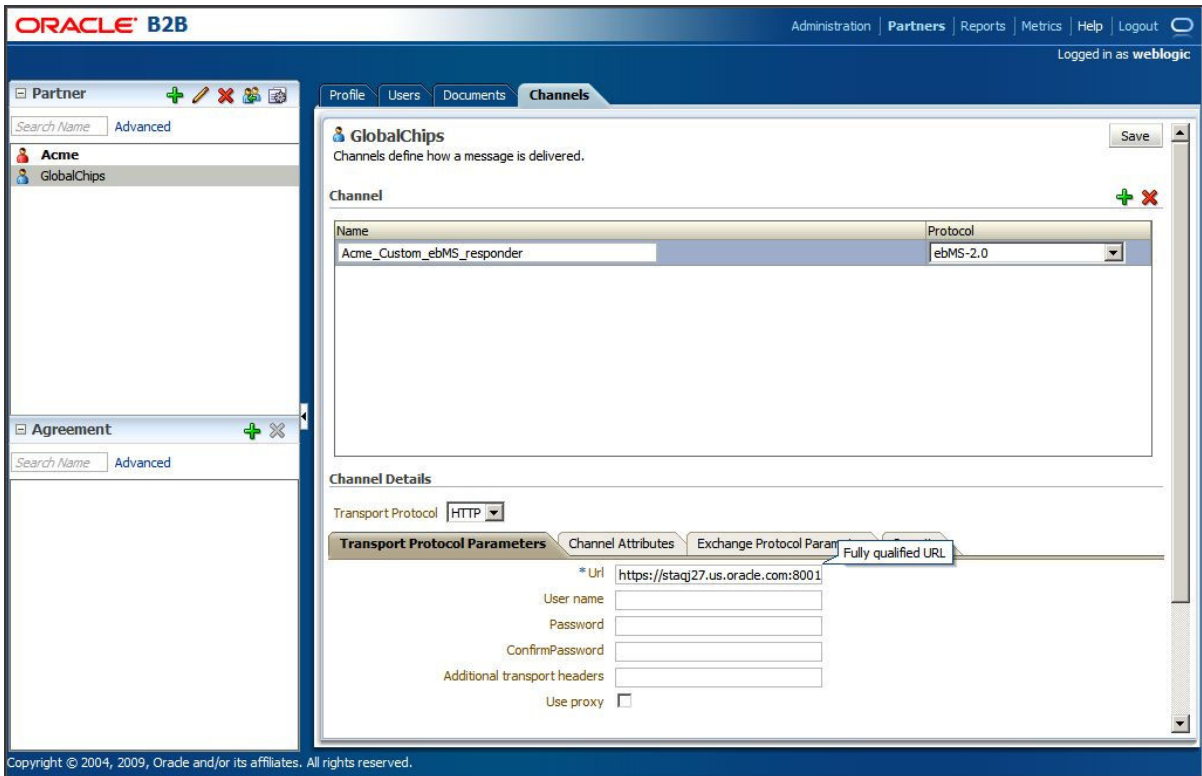
- g. Save the configuration.

Test SSL Configuration

- a. Find the SSL port(default port is 8002)
- b. Access the B2B through the following the URL: <https://hostname:sslport/b2b>

Configure SSL for Trading Partners

- a. Log into the B2B UI Tool.
- b. Click on Partners>>Partner>>Select the Trading Partner (GlobalChips)>>Channels>>Select the Delivery Channel>>Transport Protocol Parameters
- c. Update URL with <https://<host>:port> to enable Transport Security



- d. Repeat the same for all remaining Trading Partners
- e. Save the configuration and Deploy.

Advanced ebMS features

Attachment Feature

EbMS exchange offers attachment feature to send attachment along with business message. These business messages include one or more *attachments* containing XML or non-XML data. Each attachment represents a single payload in ebXML message.

With ebMS, a user can send multiple attachments within one message it is the only certified interoperable, secure protocol with multiple attachment support.

Attachment can be configured as follows.

Outbound :

While enqueue the outbound message provide the file name for the "attachment" attribute as the below file(sample xml).

The below Attach.xml and the schema are specific to Oracle B2B.

```
<?xml version="1.0" encoding="UTF-8"?>
<Attachments xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\Documents and
Settings\hari\Desktop\AttachmentsDescriptor.xsd" version="1.0"
boundary="boundary---">
  <AttachmentPart>
    <Location>file:///tmp/test.jpeg</Location>
    <Content-Type>
      <Top-Level-Type>application</Top-Level-Type>
      <Sub-Type>xml</Sub-Type>
    </Content-Type>
    <Content-Transfer-Encoding>BASE64</Content-Transfer-Encoding>
  </AttachmentPart>
</Attachments>
```

The attach.xml adheres to the Attach.xsd schema definition.

Attachment sample message

```
-----=_Part_11_17658103.1237185226029
Content-Type: text/xml;charset=UTF-8
Content-ID: <ebxheader-8C5701951200DFC64E6000007C86DD00>

<?xml version="1.0" encoding="UTF-8" ?>
<env:Envelope ....
....
....
</env:Envelope>
-----=_Part_11_17658103.1237185226029
Content-Type: text/xml
Content-ID: <8C5701951200DFC64EA000007C86E100>

<...Attachment Payload1....>
-----=_Part_11_17658103.1237185226029
Content-Type: text/xml
Content-ID: <8C5701951200DFC64EA000008C86F100>

<...Attachment Payload2....>
-----=_Part_11_17658103.1237185226029--
```

Inbound :

Set the below property **oracle.tip.b2b.AttachmentsDir** in enterprise manager(em) to store the inbound attachments,

Eg.

```
<property>
  <name>oracle.tip.b2b.AttachmentsDir</name>
  <value>/tmp/</value>
</property>
```

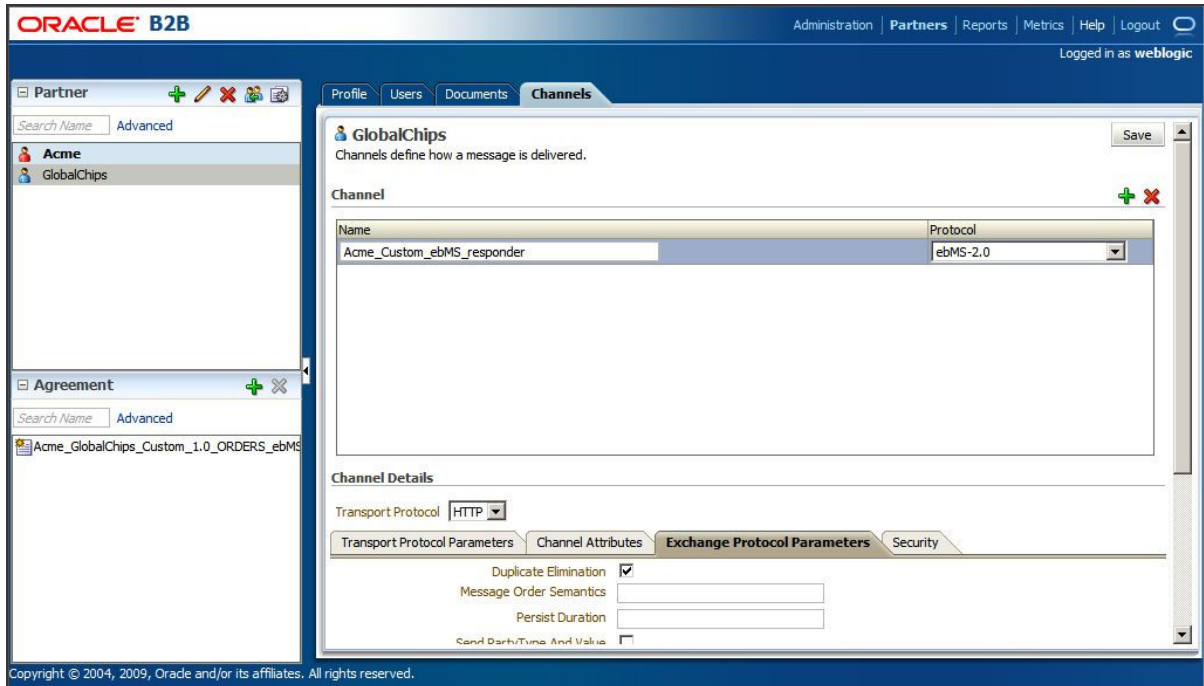
The inbound attachments will be stored in the configured directory with the same file name as attached.

Duplicate Elimination

As part of ebMS reliable messaging Oracle B2B offers Duplicate Elimination feature.

For outbound message it sets the Duplicate Elimination ebMS header which request the receiving trading partner to restrict the duplicate messages.

We can configure the Duplicate elimination as part of the Delivery Channel configuration.

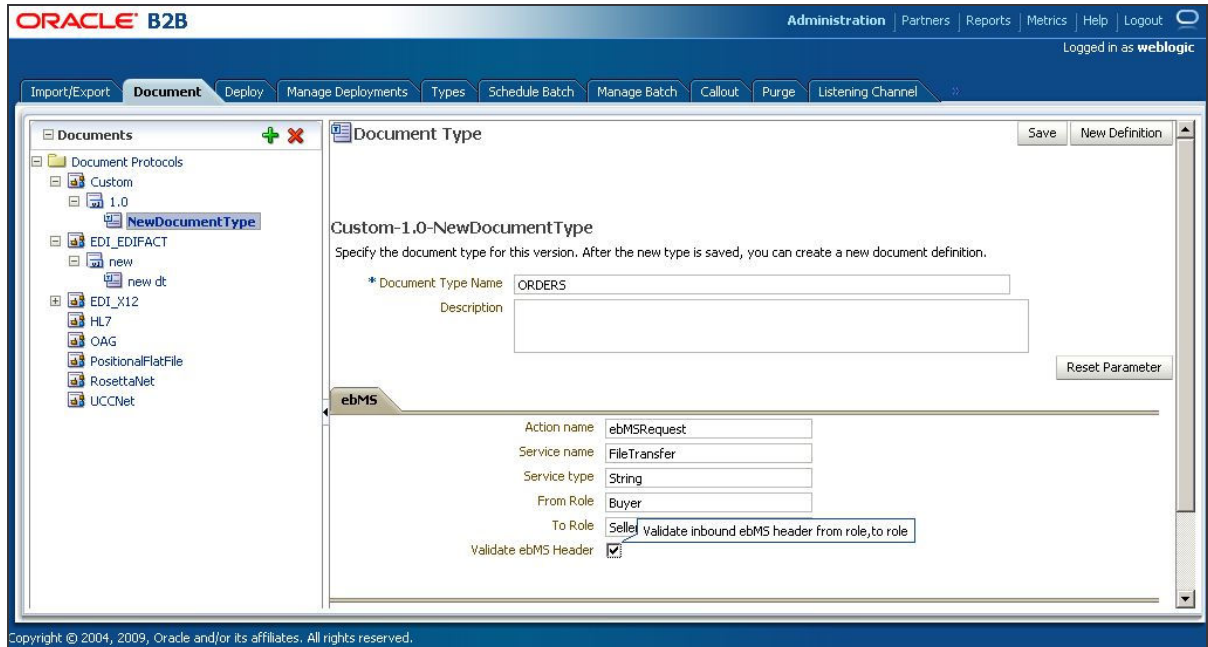


In case of inbound message if the Duplicate Elimination element present in the incoming message then the receiving Application should check for duplicate messages using message Id specified in the Message header and sends an errorlist notification (Negative Acknowledgement) to the sender in case of duplicate message.

Message Header Validation

The message header validation is an excellent feature provided by ebXML, using which the inbound ebMS message header would be validated against the configured data.

In oracle B2B ebMS header validation can be configured as part of document type parameters.



If the ebMS header validation is enabled then the incoming ebMS message headers such as from role, to role gets validated against the configured data.

Message Status Service

The message status service is an excellent feature provided by ebXML, using which the trading partner can query the status (Status Request) of the message. The receiving trading partner should reply with the status of that particular message (Status Response).

The Message status request service consists of the following:

- A Message status Request message containing the details regarding a message previously sent is sent to a Message Service Handler (MSH).
- The Message Service Handler receiving the request response with a Message Status Response message.

Message Status Request / Response

Status Request

A message Status request/response message consists of an ebXML message with no ebXML Payload container and the action element should be **“StatusRequest”** and the service element contains **“urn:oasis:names:tc:ebxml-msg:service”**.

StatusRequest element containing, A RefToMessageId element in StatusRequest element containing the messageId of the message whose status is being queried.

While enqueue the outbound message, set the AQ property action as **StatusRequest**.

```
action=ACTION:StatusRequest;SERVICE:urn:oasis:names:tc:ebxml-msg:service
```

For Fabric

```
b2b.action= StatusRequest  
b2b.ebms.Service=urn:oasis:names:tc:ebxml-msg:service
```

Status Response

The Trading Partner receives a Status request message then should reply with a Status Response message with no ebXML payload container and the Action element should be "StatusResponse" and the service element contains "urn:oasis:names:tc:ebxml-msg: service".

The MessageData element contains a **refToMessageId** that identifies the Message Status Request message.

Implementation of Status Request/Response in Oracle B2B.

Create a business action and Document Type as StatusRequest
While creating the Document type use the following parameter

```
Action: StatusRequest  
Service: urn:oasis:names:tc:ebxml-msg:service  
Identification Expression (XPath) for XML Document: /*[local-name()='StatusRequest']
```

- a. Create an Agreement after creating doctype in admin corresponding to StatusRequest
- b. Enque the message with doc type and revision as StatusRequest and 1.0
- c. The responding StatusResponse message is found in the wiremessage-report

Message Service Handler Ping/Pong

The most popular practice in EBMS world to exchange the ping-pong messages to ensure that the remote Message Handler Service, which is essentially a remote trading partner in B2B world, is up and functioning well. This is done by sending a PING message and the remote MSH responds it with a PONG message.

Ping /Pong

A **Ping message** consists of an ebXML message containing no payload container with the following message headers

The Message Header element MUST contain the following:

- a. From element that identifies the Party creating the MSH Ping message
- b. To element that identifies the Party that is being sent the MSH Ping message
- c. CPAAId element
- d. ConversationId element
- e. Service element that contains: uri:www.ebxml.org/messageService/
- f. an Action element that contains Ping

Signature element is an optional it may be omitted.

The message is then sent to the To Party.

Once the To Party receives the MSH Ping message, it MAY generate a Message Service Handler **Pong** (MSH Pong) message consisting of an ebXML Message containing no ebXML Payload and the following elements in the SOAP Header:

Message Header element
TraceHeaderList element
An Acknowledgment element
An OPTIONAL ds:Signature element

The TraceHeaderList, Acknowledgment and ds:Signature elements MAY be omitted.

The Message Header element MUST contain the following:

- a. From element that identifies the creator of the MSH Ping message
- b. To element that identifies a Party that generated the MSH Ping message
- c. CPAlid element
- d. ConversationId element
- e. Service element that contains the value:uri:www.ebxml.org/messageService/
- f. An Action element that contains the value Ping
- g. RefToMessageId that identifies the MSH Ping message

Parties who receive a MSH Ping message SHOULD always respond to the message. However, there is a risk that some parties might use the MSH Ping message to determine the existence of a Message Service Handler as part of a security attack on that MSH. Therefore, recipients of a MSH Ping MAY ignore the message if they consider that the sender of the message received is unauthorized or part of some attack. The decision process that results in this course of action is implementation dependent.

Implementation of Ping/Pong in Oracle B2B

- a. Create a business action and Document Type as Ping
- b. While creating the Document type use the following parameter.

```
Action:Ping
Service:urn:oasis:names:tc:ebxml-msg:service
Identification Expression (XPath) for XML Document: /*[local-
name()='Ping']
```

- c. Create an Agreement after creating the doctype in admin corresponding to Ping.
- d. Enque the message with doc type and revision as Ping and 1.0.
- e. The responding pong message is found in the wiremessage-report

CPP/CPA

In ebXML, a **Collaboration Protocol Profile** (CPP) will describe message exchange capabilities of a party. A **Collaboration Protocol Agreement** (CPA) will describe the agreement between two parties

The CPP and CPA contains details of transport, messaging, security constrains, and bindings to a Business Process Specification (BPSS) document that contains the definition of the interactions between the two Parties while engaging in a message exchange.

Implementation of CPP/CPA:

The following steps will explain and configure CPP/CPA using Oracle B2B and its runtime behavior.

Salient features:

1. **CPA import with selective BPSS specification:** This utility provides the option to the user to selectively add the BPSS specification while importing the CPA. One or more BPSS documents can also be imported. This is done by adding all the BPSS documents separated by “,”.

```
## Absolute path for BPSS Document. This is optional property.  
oracle.tip.b2b.ebms.BPSSDocument = <c:/tmp/bpss.xml>
```

2. **Auto Creation of Business Action:** This feature automatically creates all the BPSS specification business actions to Oracle B2B business actions. This will facilitate the user to create Business Action in bulk.
4. **Auto Import/Export of security details:** The required security credentials will be taken from CPP/CPA document to Oracle B2B as part of CPA import process, also the security details gets exported from Oracle B2B to CPP/CPA.
5. **CPA Export of Multiple BPSS files from Oracle B2B:** This feature exports the **Oracle B2B** specific metadata to CPP/CPA specific metadata .For every Business Action, it creates a corresponding BPSS during export.
6. **CPA Export of Multiple CPA files from Oracle B2B:** This feature allows the multiple CPA export. Typically it creates a different agreement for every CPA based on the Trading partner.
7. **Log Level for CPP/CPA Utility can be configured as Debug or Info or Error.**
8. **The CPP/CPA utility Log files can be of type text or xml.** These are the mandatory properties for logging

```
#Log files folder location.  
oracle.tip.b2b.ebms.LogDirectory=c:/tmp/cppcpa/  
  
# DEBUG|INFO|ERROR  
oracle.tip.b2b.ebms.LogLevel=DEBUG  
  
# text|xml  
oracle.tip.b2b.ebms.LogType=text
```

Steps to run CPP/CPA Import

- 1) Using Command Line Utility we can convert the CPA file into to oracle b2b specific metadata and then we can import the generated metadata using oracle b2b import option.
- 2) The cppcpa.properties file can be created using the below command.

```
ant b2bcreate-cpaprop
```

Configure all the necessary import values in the cppcpa.properties.

- 3) To convert CPA xml to Oracle B2B Metadata, run the below command,

```
ant b2bcpaimport
```

Steps to run CPP/CPA Export

Using b2b export utility we can export the oracle b2b metadata and then use the oracle b2b metadata to cpa.xml.

- 1) The cppcpa.properties file can be created using the below command.

```
ant b2bcreate-cpaprop
```

- 2) Configure all the necessary export values in the cppcpa.properties.
- 3) To convert Oracle B2B Metadata to CPA xml, run the below command,

```
ant b2bcpaexport
```

Configuration Details

1 oracle.tip.b2b.ebms.BPSSDocument

This property holds the absolute path for BPSS Document. It is an optional property, which will be used to get the BPSS document details to be import into Oracle B2B repository. If the property does not exist then the values will be imported from CPA. Multiple BPSS documents are separated by “;”

2 oracle.tip.b2b.ebms.CPADocument

This mandatory property will be used to get the absolute path of the CPA document to be import into Oracle B2B repository.

3 oracle.tip.b2b.ebms.xsdLocation

This optional property will be used to specify the absolute path of the schema file location. This schema file will be used for document validation.

It will be used only when BPSS document is specified.

4 oracle.tip.b2b.ebms.internalDeliveryChannel.protocol

The default Internal Delivery channel is AQ, if you want to add specific internal delivery channel (JMS/FTP/FILE/SFTP) then this optional property will be used in Oracle B2B Configuration. Specify all the required properties with respect to specific transport. Then this specific channel will be used to send the message to backend applications.

CPA Export Properties:

5 oracle.tip.b2b.ebms.OutputFolder

This mandatory property will be used to place the generated CPP/CPA files in the specified location.

6 oracle.tip.b2b.ebms.Host

This mandatory property will be used to set the Host Trading Partner.

7 oracle.tip.b2b.ebms.HostEndPoint

This mandatory property will be used to set the host endpoint while generating the CPP/CPA export.

8 oracle.tip.b2b.ebms.HostCertificateAlias

In case of secure message transfer, this property will be used to get the host certificate details to the CPP/CPA export.

9 oracle.tip.b2b.ebms.TPCertificateAlias

In case of secure message transfer, this property will be used to get the trading partner certificate details to the CPP/CPA export.

10 oracle.tip.b2b.ebms.BPSSExport

This optional Boolean property will be used to generate the BPSS document.

Common Properties:

11 oracle.tip.b2b.ebms.LogDirectory

This mandatory property will be used to store the log files.

12 oracle.tip.b2b.ebms.LogLevel

This mandatory property will be used to specify the mode of logs like DEBUG|INFO|ERROR.

13 oracle.tip.b2b.ebms.LogType

This mandatory property will be used to specify the log file will be stored as text/xml.

Performance Best Practices

Following are the few of the Best practices for **Oracle B2B** Performance and can be tuned based on the system resources

Outbound processing:

b2b.outboundThreadCount= <no of threads to be spawned for outbound flow>

b2b.outboundSleepTime= 1000 Sleep time (in msec) for thread when there is no message.

Inbound process:

b2b.inboundThreadCount= <no of threads to be spawned for inbound flow>

b2b.inboundSleepTime= 1000

Default threads for process all the events:

b2b.defaultThreadCount= <no of threds to be spawned for all other events such as delivery channel creation/updating, system parameter updation etc >

b2b.defaultSleepTime= 1000

Large Payload Configurations

Following steps need to be configured in case of large payload above 100Mb,

1. As a pre-requisite, the following properties need to be set in B2B
In Admin -> Configuration,
 - Use **JMS Queue as default** property as **true**.
 - **Logpayload** property to be **false**
 - **Large Payload Size** property as <payload size>
 - **Large Payload Directory**. eg. /tmp
2. jvm memory-args
 - Increase max heap size according to machine available RAM
E.g. for 2GB -Xmx2048m and also change min heap size needy basis.
3. Database Configuration changes
 - We can mention the pointer to the performance document
4. Transaction Timeout settings

In Weblogic console
 - Increase JTA transaction timeout from
weblogic console Services -> JTA Timeout Seconds=720 seconds
 - Increase SOADDataSource XA timeout setting to 120-180seconds
weblogic console Services -> JDBC->DataSources->SOADDataSource
5. Fabric settings - For fabric based testing use streaming option incase of file-adaptor listener.

Typical ebMS Errors

1. ebMs header are not matching with the configured value : Check for the action/service/servicetype combination of incoming message is same as configured one.
2. Agreement Not found : Make sure the values send from backend send the configured Action, Service,ServiceType.
3. Decryption failed ; check if the keystore/certificate is configured at the receiving endpoint.
4. Duplicate Protocol Message Error : When the same message has been received more than once. Also check if duplicate elimination is set to true.

FAQ

Reference:

- 1.OASIS-Open.org:
http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0.pdf

Chief Editor:

Ramesh Anantharamaiah

Contributors:

Jeffrey Hutchins

Sundararaman Shenbagam

Ted Hong

Nandagopal Srinivasan

Dheeraj Kumar Madai

Lasted Updated: February 9, 2010

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.

Oracle is a registered trademark of Oracle Corporation. Various
product and service names referenced herein may be trademarks
of Oracle Corporation. All other product and service names
mentioned may be trademarks of their respective owners.

Copyright © 2008 Oracle Corporation
All rights reserved.

ORACLE®