WebSphere software

IBM

Integration Patterns

WebSphere Process Server 6.2
WebSphere Application Server 6.1
WebSphere Message Broker 6.1
WebSphere MQ 7.0

Business Process Choreographer team, Boeblingen, Germany

Disclaimer

The Authors

This document is produced by the Business Process Choreographer team in Böblingen Germany.

Torsten Wilms
IBM Software Group, Application and Integration Middleware Software
BPM Suite Integration Quality Assurance

Werner Führich
IBM Software Group, Application and Integration Middleware Software
BPM Competence Center

Dieter König
IBM Software Group, Application and Integration Middleware Software
Senior Technical Staff Member, Business Process Choreographer

Thanks to the following people for their contributions
Michael Klingler (Daimler AG), Thomas Gehrung (Daimler AG), Marco Scheuermann (Daimler AG), Daniel Fritz (Daimler AG), Jonas Grundler, Anthony O’Dowd.

Table of Contents

1 Introduction **4**
1.1 Overview of the interactions 4
2 Showcase application **6**
2.1 Operational model 6
2.2 Overall sequence diagram of the order process application (showcase) 7
3 Identity propagation **9**
4 Using JAX-WS clients with the BPC- and HTM API **10**
5 Web Service Addressing (WS-A) **11**
6 SSL Configuration **12**
6.1 SSL between WPS/WAS and MQ 12
6.1.1 Create the self-signed certificate for MQ 12
6.1.2 Create the self-signed certificate for WPS 14
6.1.3 Export the self-signed certificate 15
6.1.4 Configure the WebSphere MQ queue manager for SSL 16
6.1.5 Configure the WebSphere Application Server JMS client 19
6.2 SSL between WMB and WAS 22
6.2.1 Create Self-Signed Certficate for WAS 22
6.2.2 Export the WAS self-signed certificate 22
6.2.3 Import to WMB 23
6.3 SSL between WPS and WAS 24
6.3.1 Configure WPS (client) for SSL 24
6.3.2 Configure WAS (server) for SSL 25
7 Patterns (Interactions) **27**
7.1 Sequence of interactions 27
7.1.1 Interaction 1 to 3 - Start Process 28
7.1.1.1 Optional implementation 28
7.1.1.2 Detailed description of the implementation and configuration steps 29
7.1.1.3 Step 1 – WPS: Develop the BPEL Application and define potential process starters 29
7.1.1.4 Step 2 – WPS: Deploy BPEL Application 30
7.1.1.5 Step 3 – WPS: Modify Token consumer settings on the BPC Container 30
7.1.1.6 Step 4 – WAS: Develop Web Service consumer application and define Token Generator – JAX RPC 35
7.1.1.7 Step 4 – WAS: Develop Web Service consumer application and define Token Generator – JAX WS 36
7.1.1.8 Step 5 – WAS: Deploy application 43
7.1.2 Interaction 4a and 4d – Check Stock #1 – WPS to Message Broker 44
7.1.2.1 Step 1 – MQ: Define queues 44
7.1.2.2 Step 2 – WPS: Develop the BPEL application 45
7.1.2.2.1 Propagate the user ID from WPS to Message Broker 45
7.1.2.2.2 Define the ConnectionFactory and queue objects in WID 48
7.1.2.3 Step 3 – WPS: Deploy the BPEL application 48
7.1.2.4 Step 4 – WMB: Develop the message flow and set up a security profile 49
7.1.2.5 Step 5 – WMB: Deploy the message flow application 50
7.1.2.6 Step 6 – SSL configuration between WPS and MQ 50
7.1.3 Interaction 4b and 4c – Check Stock #1 – Message Broker to WAS 51
7.1.3.1 Step 1 – WMQ: Define queues 51
7.1.3.2 Step 2 – WMB: Develop the message flow 52

7.1.3.3 Step 3 – WMB: Deploy the message flow 52
7.1.3.4 Step 4 – WAS: Develop the WAS application 52
7.1.3.5 Step 5 – WAS: Configure the MQ Adapter WAS application 52
7.1.3.6 Step 6 – WAS: Deploy the WAS application 53
7.1.3.7 Step 7 – SSL Configuration 53
7.1.4 Interaction 5a and 5d – Check Stock #2 – WPS to Message Broker 53
7.1.5 Interaction 5b and 5c – Check Stock #2 – Message Broker to WAS 55
7.1.5.1 Step1 – WAS: Develop Web Service provider application and create token consumer 56
7.1.5.2 Step 2 – WAS: Deploy the application 58
7.1.5.3 Step 3 – WMB: Develop the message flow as Web Service consumer 59
7.1.5.4 Step 4 – WMB: Deploy the message flow 62
7.1.5.5 Step 5 – SSL configuration between WMB and WAS 62
7.1.6 Interaction 7 and 8 - SOAP/HTTP from WPS to WAS via Message Broker 63
7.1.6.1 Step 1 – WPS: Develop the BPEL application 64
7.1.6.2 Correlation in BPEL 65
7.1.6.3 Step 2 – WPS: Create a Token Generator 67
7.1.6.4 Step 3 – WPS: Create a Token Consumer 69
7.1.6.5 Step 4 – WPS: Deploy the BPEL application 73
7.1.6.6 Step 5 – WAS: Develop the WAS application 73
7.1.6.7 Step 6 – WAS: Create the Token Request Consumer 73
7.1.6.8 Step 7 – WAS: Create a Token Response Generator 77
7.1.6.9 Step 8 – WAS: Deploy WAS application 80
7.1.6.10 Step 9 – WMB: Develop MessageFlow 81
7.1.6.11 Step 10: Deploy MessageFlow 81
7.1.7 Interaction 11 and 14: Human Task – get Supplier – WAS to WPS 82
7.1.7.1 Step 1 – WPS: Develop the BPEL application and define the potential Human Task Owners 83
7.1.7.2 Step 2 – WPS: Deploy the BPEL application 84
7.1.7.3 Step 3 – WAS: Develop the Web Service client 84
7.1.7.4 Step 4 – WAS: Define the Token Generator – JAX-RPC 84
7.1.7.5 Step 4 – WAS: Define Token Generator – JAX-WS 87
7.1.7.6 Step 5 – WPS: Deploy WAS application 88
7.1.7.7 Step 6 – SSL configuration between WPS and WAS 88
7.1.8 Interaction 13: Web Service Addressing between WAS and WMB 89
7.1.8.1 Step 1 – WMB: Implement the message flow 90
7.1.8.2 Step 2 – WMB: Deploy the message flow 91
7.1.8.3 Step 3 – WAS: Configure and implement the WAS application 91
7.1.8.4 Step 4 – WAS: Deploy the application 93
7.1.9 Interaction 15 and 16 – SOAP/MQ - Identity propagation based on HT owner of preceding activity 94
7.1.9.1 Step 1 – WPS: Identity propagation based on HT owner of the preceding activity 95
7.1.9.2 Step 2 and 3 – WPS: Implement a SOAP/JMS binding 99
7.1.9.3 Step 4 – WPS: Deploy the BPEL application 101
7.1.9.4 Step 5 – WAS: Develop the WAS application 101
7.1.9.5 Step 6 – WAS: Configure MQ Adapter 101
7.1.9.6 Step 7 – WAS: Define Token Generator and Consumer 101
7.1.9.7 Step 8 – WAS: Deploy the WAS application 106
7.1.9.8 Step 9 – WMB: Develop the message flow 106

7.1.9.9 Step 10 – WMB: Deploy th message flow 107
7.1.10 Interaction 17 – RMI between WPS and WAS 107
7.1.10.1 Step 1 – WPS: Develop BPEL application 108
7.1.10.2 Step 2 – WPS: Deploy the BPEL application 110
7.1.10.3 Step 3 – WAS: Develop the application 110
7.1.10.4 Step 4 – WAS: Deploy the application 110
7.1.10.5 Step 5 – SSL configuration 110
7.1.10.6 Step 6 and 7 – Configure CSIv2 authentication 110
7.1.10.6.1 Option 1: Basic Authentication and identity assertion 110
7.1.10.6.2 Option 2: Basic Authentication without identity assertion 114
7.1.10.6.3 When to use LTPA, identity assertion, Basic Authentication and Certificates117
7.1.11 Interaction 18 – Update Global Order DB - SOAP HTTPs between WPS and WAS118
8 Basis setup and application install reference 119
8.1 Configuration of JMS resources 119
8.2 Configuration of JDBC resources 119
8.3 Deployment of the showcase SCA module 119
8.4 WebSphere Message Broker resources 130
8.4.1 Message Flow definition 130
8.4.2 Queue definition 131
8.5 WebSphere Application Server resources 131
8.5.1 JMS Connection Factory resources 131
8.5.2 Configuration of JDBC resources 135
8.5.3 Configuration of DB2 for WAS applications 136
8.5.4 Deployment of the WAS applications 137
9 Terms 138
10 Abbreviations 139
11 Referenced Documents 140

1 Introduction

This paper documents the integration of WebSphere Application Server (WAS), WebSphere Process Server (WPS), WebSphere Message Broker (WMB) and WebSphere MQ (WMQ). It uses a fictive order process to verify secured and reliable communication among these products in a bi-directional way. Throughout this document the terms scenario and showcase are used synonymously. This showcase scenario consists of WPS- , WAS- and WMB applications, MQ queues, two user registries (Tivoli Directory Server, file-based), and DB2 databases. The applications are deployed on several hardware boxes.

The integration considers:

- SSL configuration between WAS, WPS and WMB (refer to chapter SSL configuration)
- Identity propagation and assertion between WAS, WPS and WMB (refer to chapter Identity Propagation)
- JAX-WS and JAX-B clients on WAS for the WPS BPC- and HTM API (refer to chapter using JAX-WS and JAX-B clients)
- Web Service Addressing (WS-A) between WAS and WebSphere Message Broker (refer to chapter Web Service Addressing)
- Integration patterns (refer to chapter Patterns/Interactions)
  - document the detailed implementation steps
  - provide an overview of the interactions from a security point of view in chapter Overview of the interactions

1.1 Overview of the interactions

The showcase, as mentioned before, provided for bi-directional interactions between the servers and registries, consists of several interactions. The following list provides an overview of the interactions:

- Interaction 3 - propagate identity using Username Tokens from WebSphere Application Server to WebSphere Process Server via SOAP/http
- Interaction 4a/d - propagate identity from WPS via MQ to Message Broker. SSL is used for transport level security.
- Interaction 4b/c - propagate identity from WMB via MQ to WAS. SSL is used for transport level security.
- Interaction 5a/d -propagate identity from WPS via MQ to Message Broker. SSL is used for transport level security.
- Interaction 5b/c - identity propagation with identity assertion from WBM to WAS via SOAP/https
- Interaction 7/8 - identity propagation from WPS to WAS via Message Broker. identity is propagated via Username Token in the Web Service Security Header.
- Interaction 11/14 - propagate the identity via LTPA from WAS to WPS with the HTM Web Service API.

- Interaction 13 - Set up WS-A between WAS and WMB. Https will be used as Transport Level Security. Identity propagation will be done using Username Tokens (w/o password).
- Interaction 15/16 - SOAP/MQ; identity propagation not based on process starter identity but on HT owner of preceding activity
- Interaction 17 - SSL with RMI/IIOP; identity propagation between WPS and WAS

Chapter 3 provides an overview about the scenario that we used to demonstrate the integration. It contains the operational model and a UML sequence diagram. After getting an understanding of the process the reader can refer to those concepts and interactions of interest:

- SSL configuration between WAS, WPS and WMB (chapter SSL configuration)
- Identity propagation and assertion between WAS, WPS and WMB (chapter Identity Propagation)
- JAX-WS and JAX-B clients on WAS for the WPS BPC- and HTM API (chapter using JAX-WS and JAX-B clients)
- Web Service Addressing (WS-A) between WAS and WebSphere Message Broker (chapter Web Service Addressing)
- Integration patterns (chapter Patterns/Interactions)
    - Detailed interaction implementations
    - Chapter interaction 15/16 describes identity propagation of the Human Task Owner of the preceding process activity.

Find detailed setup and install information in the Appendix.

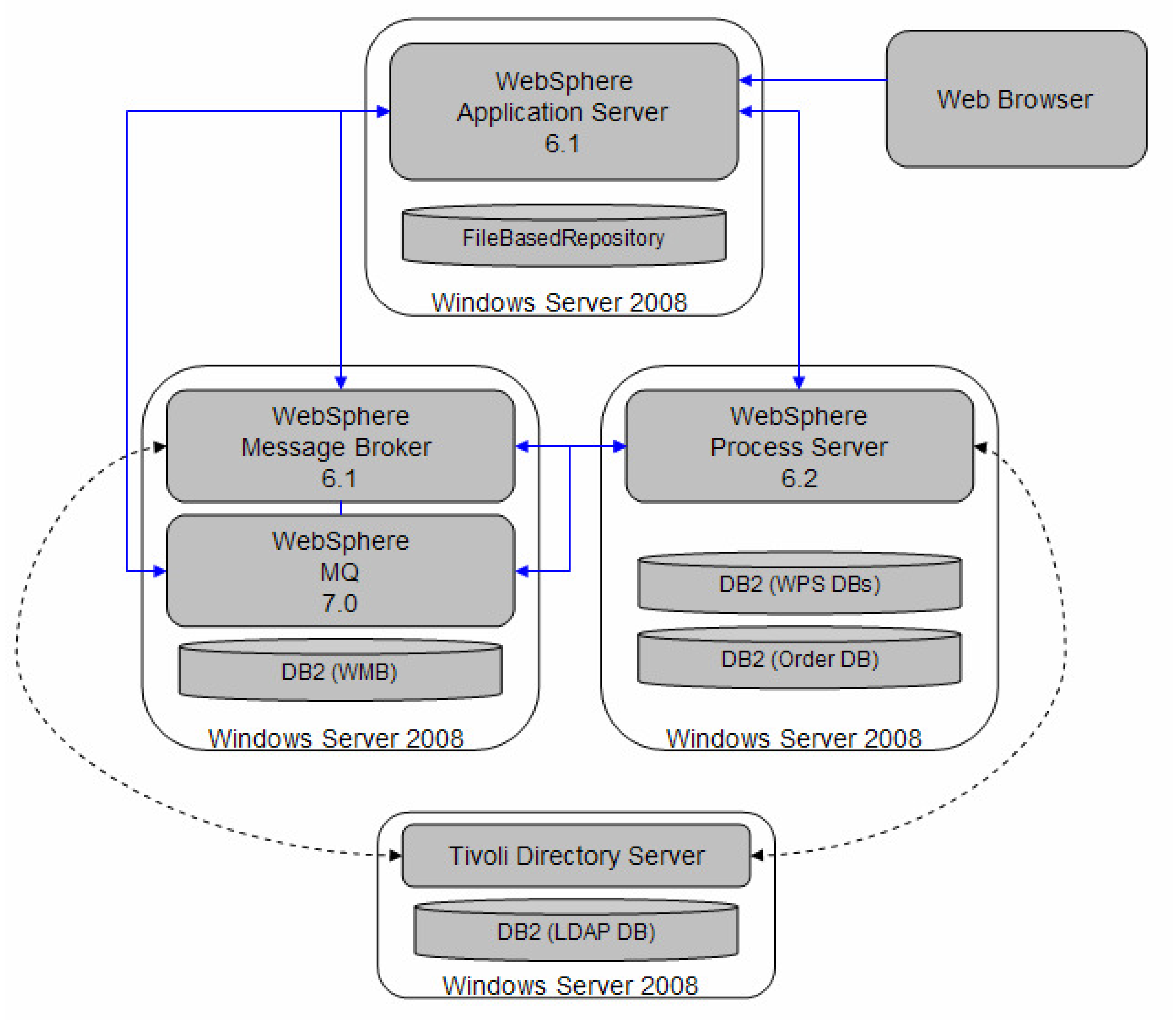
1.2 Scope of the document

This document shows security related integration aspect of WebSphere Application Server (WAS), WebSphere Process Server (WPS), WebSphere Message Broker (WMB) and WebSphere MQ (WMQ). It does not document the basic WebSphere installation, configuration and implementation.

2 Showcase application

2.1 Operational model

The high-level system structure for the “showcase” application is shown in the following diagram. In this document we do not describe how to install the products.

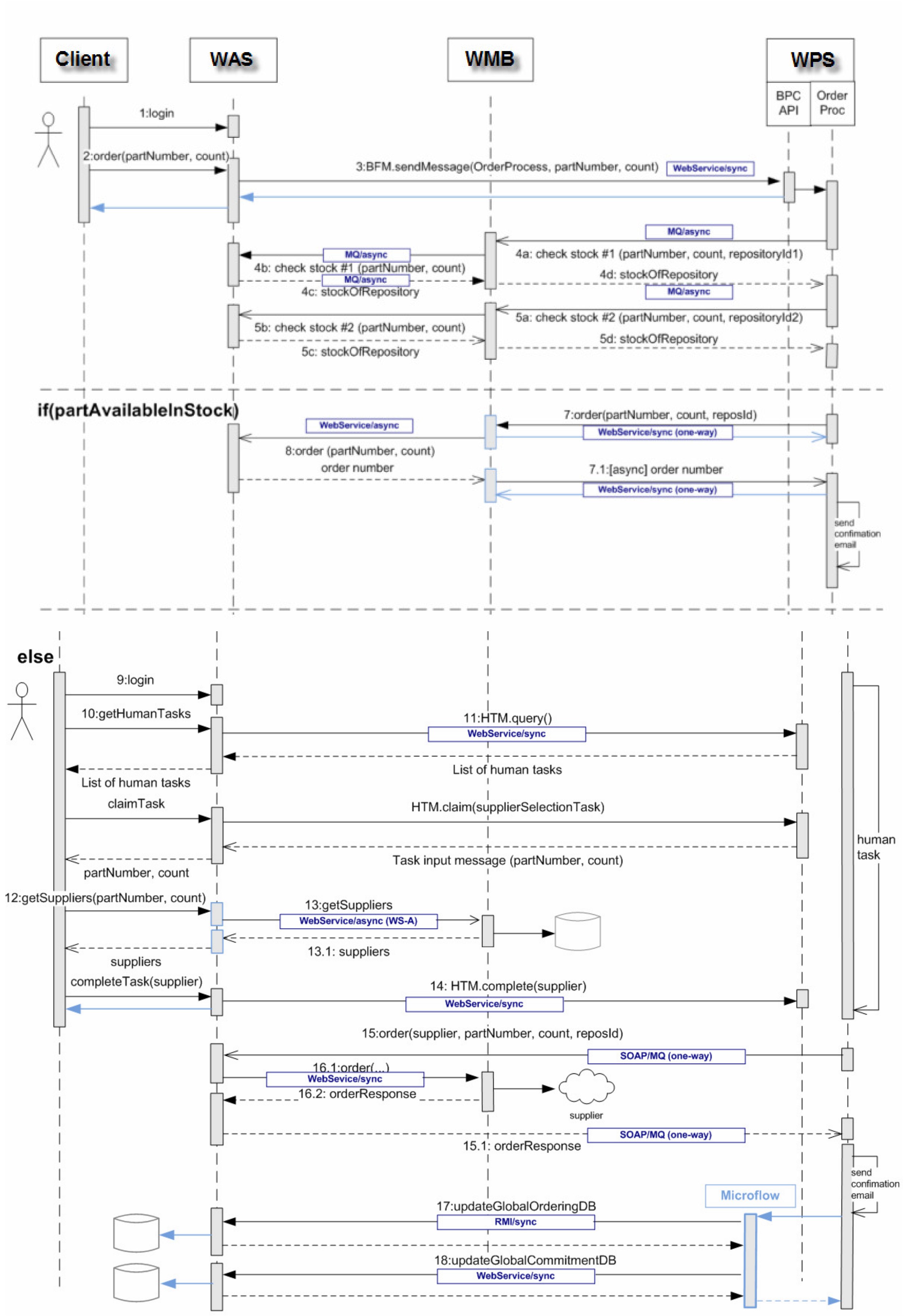


2.2 Overall sequence diagram of the order process application (showcase)

The figure below shows the sequence diagram of the scenario. Each interaction step is documented in detail in the Patterns/Interactions chapter.

The order process is started by a clerk. He uses a web based client to initiate the process. Following this two external systems are used to verify availability of the order item in stock. If so, internal order reservation is executed (interactions 1 to 7.1). Otherwise an internal purchase order is issued. A purchaser will verify the request, select a supplier and submit the external order. Order confirmation automatically updates two order databases at the end (interactions 9 to 18).

The arrows in the sequence diagram indicate the request, the chosen protocol, the message / request name, whether it is synchronous or asynchronous. The kind of processing and protocols also determine the transaction boundaries for the entire scenario.


Client
WAS
WMB
WPS
BPC API
Order Proc
1:login
2:order(partNumber, count)
3:BFM.sendMessage(OrderProcess, partNumber, count)
WebService/sync
MQ/async
4a: check stock #1 (partNumber, count, repositoryId1)
MQ/async
4b: check stock #1 (partNumber, count)
MQ/async
4c: stockOfRepository
4d: stockOfRepository
MQ/async
5a: check stock #2 (partNumber, count, repositoryId2)
5b: check stock #2 (partNumber, count)
5c: stockOfRepository
5d: stockOfRepository
if(partAvailableInStock)
7:order(partNumber, count, reposId)
WebService/async
WebService/sync (one-way)
8:order (partNumber, count)
order number
7.1:[async] order number
WebService/sync (one-way)
send confimation email
else
9:login
10:getHumanTasks
11:HTM.query()
WebService/sync
List of human tasks
List of human tasks
claimTask
HTM.claim(supplierSelectionTask)
Task input message (partNumber, count)
partNumber, count
human task
12:getSuppliers(partNumber, count)
13:getSuppliers
WebService/async (WS-A)
13.1: suppliers
suppliers
completeTask(supplier)
14: HTM.complete(supplier)
WebService/sync
15:order(supplier, partNumber, count, reposId)
SOAP/MQ (one-way)
16.1:order(...)
WebSevice/sync
16.2: orderResponse
supplier
SOAP/MQ (one-way)
15.1: orderResponse
send confimation email
Microflow
17:updateGlobalOrderingDB
RMI/sync
18:updateGlobalCommitmentDB
WebService/sync

3 Identity propagation

This chapter describes how identity propagation can be done between WebSphere Process Server, WebSphere Application Server and WebSphere Message Broker using different kind of transport and communication protocols. With identity propagation we mean that a user identity is carried within a request call from one system to another system.

The following listing provides the interactions described in this document. For the implementation refer to chapter Patterns/Interactions.

From	To	Protocol	Pattern/Interaction
WAS	WPS	Web Services	3, 11, 14
WAS	WMB	Web Service (WS-A)	13
WAS	WMB	Web Services	16.1
WPS	WMB	MQ	4a, 5a
WPS	WMB	Web Services	7
WPS	WAS	SOAP/MQ	15
WPS	WAS	RMI/IIOP	17
WPS	WAS	Web Services	18
WMB	WAS	MQ	4b
WMB	WAS	Web Services	5b, 8

Chapter Interaction 15/16 describes identity propagation of the Human Task Owner of the preceding process activity.

4 Using JAX-WS clients with the BPC- and HTM API

In the showcase application we use JAX-WS and JAX-B based clients on WebSphere Application Server to access the BPC- and HTM API on WebSphere Process Server.

We show how to propagate the user identity from WAS to WPS - both have different user registries - using a programmatic approach and a declarative approach (using JAX-WS policy sets).

By using JAX-B on the client we are able to use strong typed business objects with the APIs instead of generic ones.

- Using BPC API with JAX-WS (startProcess) refer to Interaction 3.
- Using HTM API with JAX-WS (query, claim, complete task) refer to Interaction 11 and 14.

5 Web Service Addressing (WS-A)

We demonstrate how to set up WS-A between WAS and WMB with identity propagation (Username Tokens). For details refer to chapter Interaction 13.

WS-Addressing is a standardized way of including message routing data within the SOAP message. It supports the use of asynchronous interactions by specifying a common SOAP header (wsa:ReplyTo) that contains the endpoint reference (EPR) to which the response is to be sent.

6 SSL Configuration

Some of the interactions of the showcase use a Secure Socket Layer (SSL) connection between the products (WAS, WMQ, WMB and WPS).
The following SSL configurations are described in this chapter.

- SSL between WPS/WAS and MQ
- SSL between WMB and WAS (for http/s between WMB and WAS)
- SSL between WPS and WAS

Refer to *IBM WebSphere Developer Technical Journal: SSL, certificate, and key management enhancements for even stronger security in WebSphere Application Server V6.1*

http://www.ibm.com/developerworks/websphere/techjournal/0612_birk/0612_birk.html

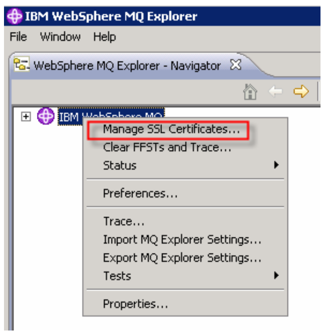
6.1 SSL between WPS/WAS and MQ

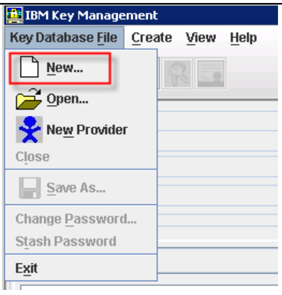
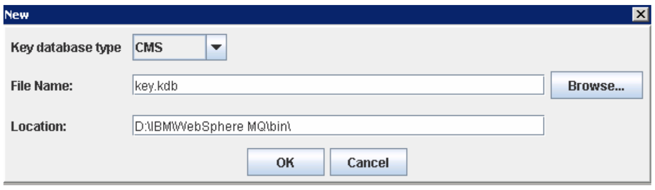
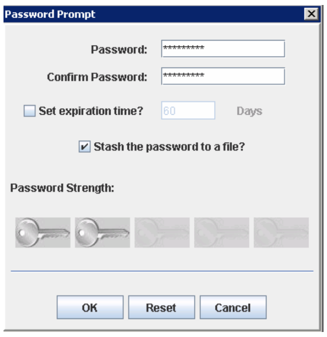
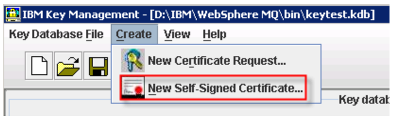
The option for SSL between MQ and WPS/WAS is to use certificates.
Therefore we need to generate and exchange certificates at design time. For the WebSphere MQ server certificate and for the WPS/WAS server certificate, we will use a self-signed certificate.
Find an overview of the SSL handshake at
http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp?topic=/com.ibm.mq.csqzas.doc/sy10660_.htm .

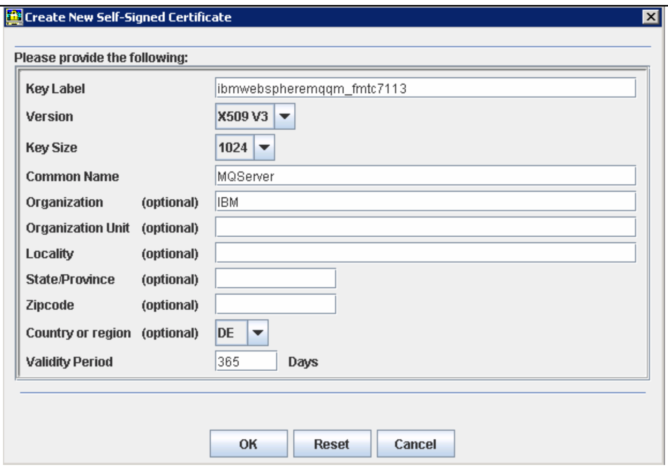
The next steps describe the configuration steps to be performed for **WPS**.
For SSL between WAS and MQ repeat these steps.

Important: MQ stores the client certificates from its trusted peers (WAS, WPS) not in a separate Trust File, but in its keyfile.

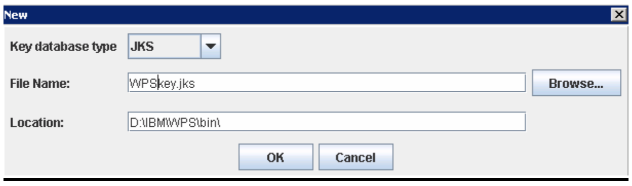
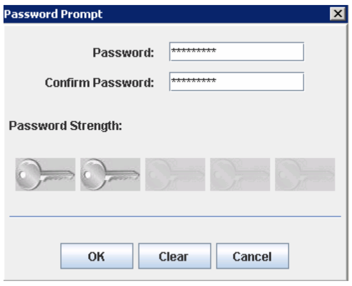
6.1.1 Create the self-signed certificate for MQ

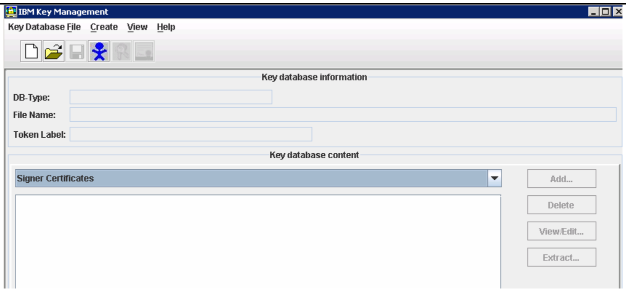
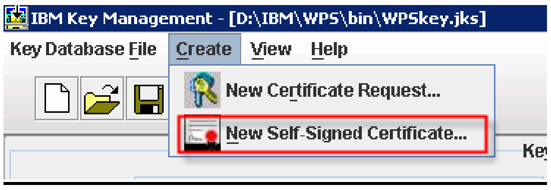
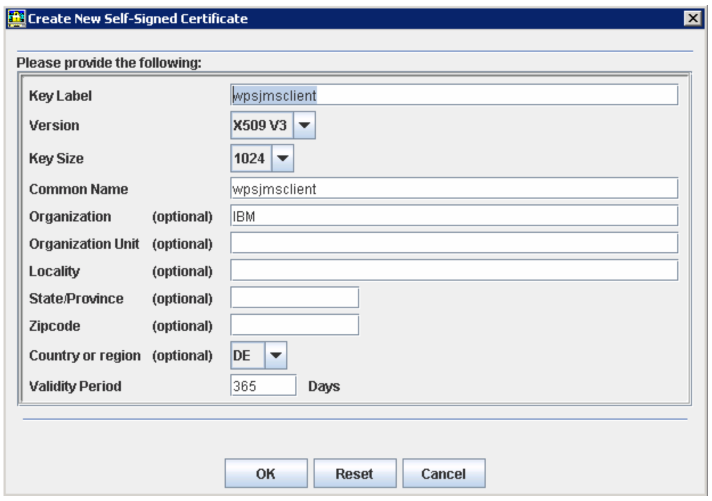
1.	Start key management utility ikeyman by opening WebSphere MQ Explorer and right-clicking on *IBM Websphere MQ*, then select *Manage SSL Certificates*.  IBM WebSphere MQ Explorer File Window Help WebSphere MQ Explorer - Navigator IBM WebSphere MQ Manage SSL Certificates... Clear FFSTs and Trace... Status Preferences... Trace... Import MQ Explorer Settings... Export MQ Explorer Settings... Tests Properties...
2.	Create the key database file by selecting *Key Database File > New*

	 IBM Key Management Key Database File Create View Help New... Open... New Provider Close Save As... Change Password... Stash Password Exit
3.	Accept the default key database type of *CMS*  New Key database type: CMS File Name: key.kdb Location: D:\IBM\WebSphere MQ\bin\
4.	For the File Name, browse to <mqroot>\Qmgrs\<qmgrname>\ssl\ directory and call the file *key.kdb*
5.	When prompted, enter an appropriate password (websphere) Select the option to *stash the password to a file*  Password Prompt Password: ********** Confirm Password: ********** ☐ Set expiration time? 60 Days ☑ Stash the password to a file? Password Strength:
6.	Select *Create* > *New Self Signed Certificate.*  IBM Key Management - [D:\IBM\WebSphere MQ\bin\keytest.kdb] Key Database File Create View Help New Certificate Request... New Self-Signed Certificate... Key datat
7.	Enter a value for the Key Label name as ibmwebspheremq<yourqmgrname_inlowercase>. This will end up being the certificate name. Also enter values for Common Name (e.g MQServer) Organization, and all remaining fields that are labeled optional. You can leave the default Key Size of 1024.

	 Create New Self-Signed Certificate Please provide the following: Key Label: ibmwebspheremqqm_fmtc7113 Version: X509 V3 Key Size: 1024 Common Name: MQServer Organization (optional): IBM Organization Unit (optional): Locality (optional): State/Province (optional): Zipcode (optional): Country or region (optional): DE Validity Period: 365 Days
8.	Enter a filename to store the request or leave the default certreq.arm. The certificate label name must follow this convention if using WebSphere MQ V6, otherwise the queue manager will not know which server certificate to use

6.1.2 Create the self-signed certificate for WPS

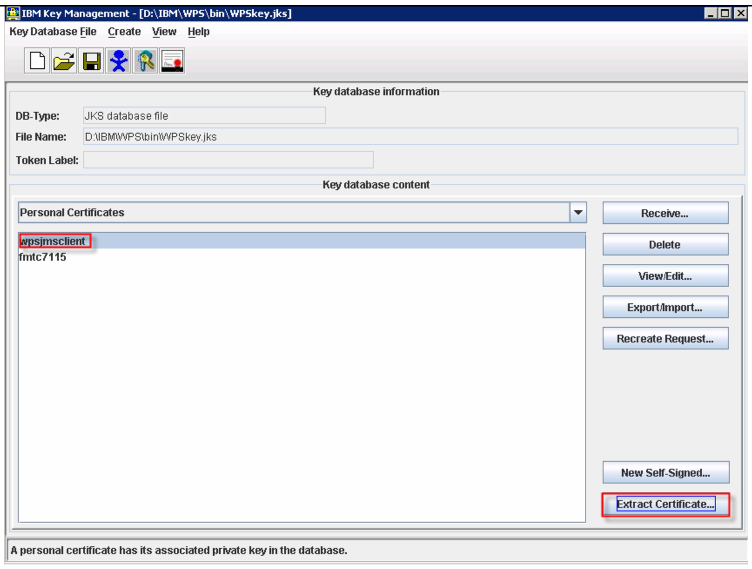
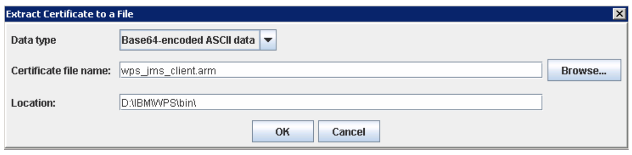
1.	Switch to the version of ikeyman that comes with WebSphere Application Server by launching <wps root>\bin\ikeyman.bat
2.	From the ikeyman menu, *select Key Database File > New*
3.	On the Open dialog, for key database type, accept the default value of JKS (Java™ keystore) Save the file as WPSKey.jks  New Key database type: JKS File Name: WPSkey.jks Location: D:\IBM\WPS\bin\
4.	When prompted to create a keystore password, select a valid password and confirm it (websphere)  Password Prompt Password: ********** Confirm Password: ********** Password Strength:
5.	Optional: Delete all signer certificates from the Signer Certificates tab. Limiting signers limits risk.

	 IBM Key Management Key Database File Create View Help Key database information DB-Type: File Name: Token Label: Key database content Signer Certificates Add... Delete View/Edit... Extract...
6.	Create a new Self-Signed Certificate  IBM Key Management - [D:\IBM\WPS\bin\WPSkey.jks] Key Database File Create View Help New Certificate Request... New Self-Signed Certificate...
7.	Enter data in the Create New Self-Signed certificate dialog with values appropriate to the location of your application server. Set Key Label to a value of your choice. Note that the default Validity Period is set to 365 days. After 365 days you have to renew the certificates. Create New Self-Signed Certificate Please provide the following: Key Label: wpsjmsclient Version: X509 V3 Key Size: 1024 Common Name: wpsjmsclient Organization (optional): IBM Organization Unit (optional): Locality (optional): State/Province (optional): Zipcode (optional): Country or region (optional): DE Validity Period: 365 Days

6.1.3 Export the self-signed certificate

At this point, we have created a self-signed certificate for the WPS MQ client. We now need to extract the jmsclient certificate and place it in the trust file for WebSphere Process Server and WebSphere MQ, so that they can both use it as a signer.

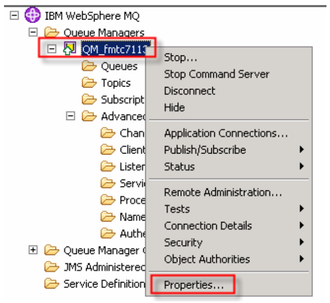
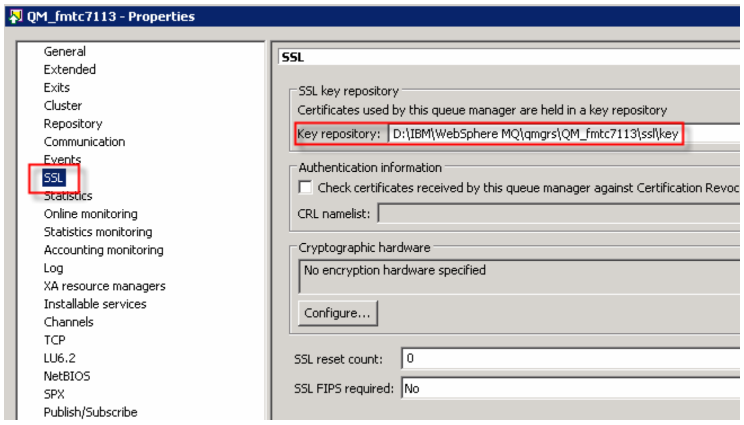
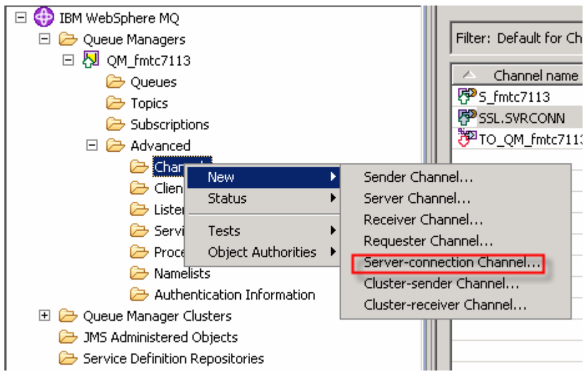
1.	First, we will export the WPS personal certificate. With the ikeyman database open to the WPSKey file, and the jmsclient certificate selected, click Extract Certificate. This exports only the certificate (not the private key).

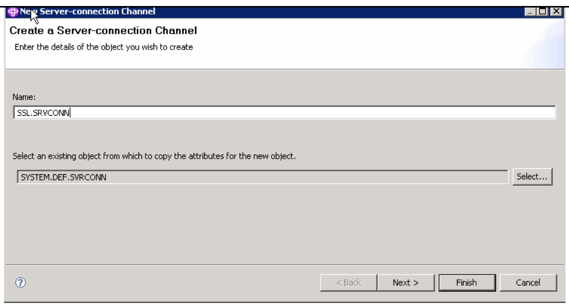
	 IBM Key Management - [D:\IBM\WPS\bin\WPSkey.jks] Key database information DB-Type: JKS database file File Name: D:\IBM\WPS\bin\WPSkey.jks Token Label: Key database content Personal Certificates wpsjmsclient fmtc7115 A personal certificate has its associated private key in the database.
2.	Save the certificate and give it an appropriate name, such as wps_jms_client.arm  Extract Certificate to a File Data type: Base64-encoded ASCII data Certificate file name: wps_jms_client.arm Location: D:\IBM\WPS\bin\
3.	While the WPSKey is used for private keys, we need a trust file which will be will be used for validating signers. We will now create this file and call it WPSServerTrustFile. Using ikeyman, create a new key database by selecting Key Database File => New and call it WPSServerTrustFile.jks
4.	Optional: Switch to the Signer Certificates tab and delete all unnecessary signers
5.	Import the jmsclient certificate into the WebSphere Application Server truststore: switch to the Signer Certificates tab, press the Add button, browse to the location where you saved wps_jms_client.arm, and import the certificate. In later point in time, we will also import the MQServer arm file into the WPS trust store.
6.	**Switch to the MQ keyman** Import the jmsclient certificate into the WebSphere MQ truststore switch to the Signer Certificates tab, press the Add button, browse to was_jms_client.arm, and import the certificate
7.	Remember that we also need to import the MQ Certificate into the WebSphere Application Server truststore, so that the application server can validate the queue manager certificate during the SSL handshake

All the certificates are now in the right places for your application server key and trust files. To verify this, make sure your application server key file contains the jmsclient certificate, and the application server trust file contains the jmsclient certificate and the mqserver certificate.

6.1.4 Configure the WebSphere MQ queue manager for SSL

1.	Make sure all key files are located in *D:\IBM\WebSphere MQ\qmgrs\QM_fmtc7113\ssl\key*

2.	In the MQ Explorer right-click on the queue manager and select *Properties > SSL*  IBM WebSphere MQ Queue Managers QM_fmtc7113 Queues Topics Subscript Advanced Chan Client Lister Servi Proce Name Authe Queue Manager C JMS Administered Service Definition Stop... Stop Command Server Disconnect Hide Application Connections... Publish/Subscribe Status Remote Administration... Tests Connection Details Security Object Authorities Properties...
3.	Verify the Key Repository and click *OK* QM_fmtc7113 - Properties General Extended Exits Cluster Repository Communication Events SSL Statistics Online monitoring Statistics monitoring Accounting monitoring Log XA resource managers Installable services Channels TCP LU6.2 NetBIOS SPX Publish/Subscribe SSL SSL key repository Certificates used by this queue manager are held in a key repository Key repository: D:\IBM\WebSphere MQ\qmgrs\QM_fmtc7113\ssl\key Authentication information ☐ Check certificates received by this queue manager against Certification Revoc CRL namelist: Cryptographic hardware No encryption hardware specified Configure... SSL reset count: 0 SSL FIPS required: No
4.	Next, we will configure the channel with which the JMS client will communicate with the queue manager for SSL: Note that "CN", "OU", "O", and so on, must be uppercase. Also note that PC (postal code) is not an accepted part of the DN in WebSphere MQ. Finally, although some areas of the documentation may mention that the DN values need to be in quotes, we found in our testing that quoted values such as CN='jmsclient' did not work in WebSphere MQ V6 for Windows.
5.	In MQ Explorer, select your queue manager, then select the *Advanced* folder, then the *Channels* folder, and right-click. Select *New > Server Connection Channel*  IBM WebSphere MQ Queue Managers QM_fmtc7113 Queues Topics Subscriptions Advanced Chan Clien Lister Servi Proce Namelists Authentication Information Queue Manager Clusters JMS Administered Objects Service Definition Repositories New Status Tests Object Authorities Sender Channel... Server Channel... Receiver Channel... Requester Channel... Server-connection Channel... Cluster-sender Channel... Cluster-receiver Channel... Filter: Default for Ch Channel name S_fmtc7113 SSL.SVRCONN TO_QM_fmtc711
6.	On the next dialog, enter a name for the channel (we use *SSL.SVRCONN*), then click *Next*.



New Server-connection Channel

Create a Server-connection Channel

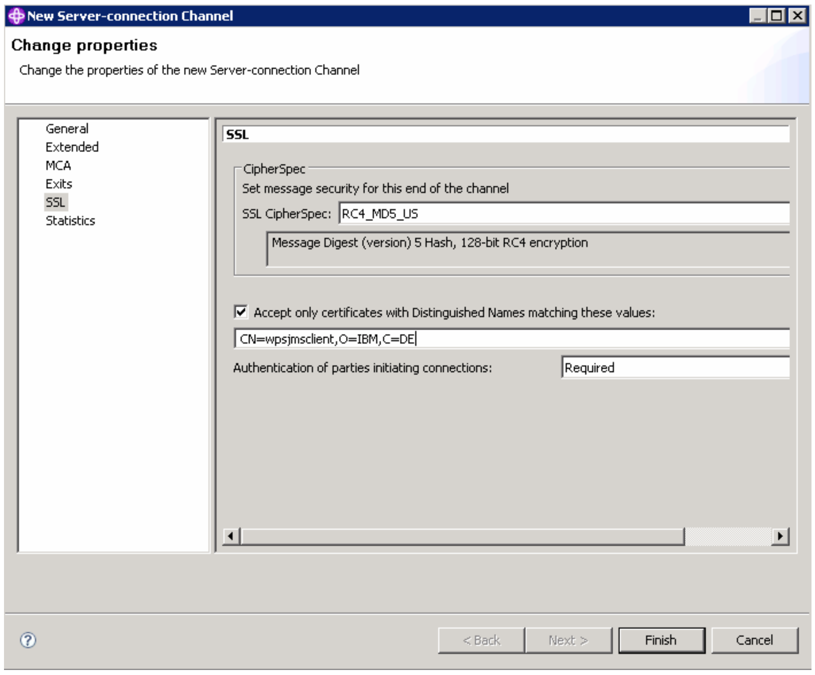
Enter the details of the object you wish to create

Name: SSL.SRVCONN

Select an existing object from which to copy the attributes for the new object.

SYSTEM.DEF.SVRCONN

7. Switch to the *SSL* tab view, and specify a cipher specification. For this example, we will use *RC4_MD5_US*, but you should evaluate your organization's security needs and consider alternative, stronger ciphers if necessary. Notice that the default setting for Authentication of parties initiating connection is *Required*



New Server-connection Channel

Change properties

Change the properties of the new Server-connection Channel

SSL

CipherSpec

Set message security for this end of the channel

SSL CipherSpec: RC4_MD5_US

Message Digest (version) 5 Hash, 128-bit RC4 encryption

☑ Accept only certificates with Distinguished Names matching these values:

CN=wpsjmsclient,O=IBM,C=DE

Authentication of parties initiating connections: Required

8. We need to prevent the queue manager from accepting a certificate from simply any client that has a certificate issued by one of the CAs in the queue manager's keystore. To do so, we need to set the SSLPEER parameter on the channel. This parameter is used to check the Distinguished Name (DN) of the certificate from the client at the other end of a WebSphere MQ channel. If the DN received from the client does not match the SSLPEER value, the channel will not start. Set this by checking *Only accept certificates with Distinguished Names* matching these values, and enter the DN value that matches the client certificate. In our case, this would be: CN=jmsclient,OU=issw,O=ibm,C=US (based on how we generated the self-signed client certificate).

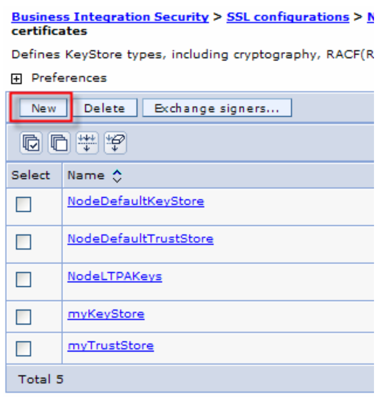
We have now configured the server connection channel that the **WPS** JMS client will use to communicate with the queue manager. If you have not yet done so, you should tighten all channels to require SSL (or remove the channel), including channels such as SYSTEM.DEF.SVRCONN

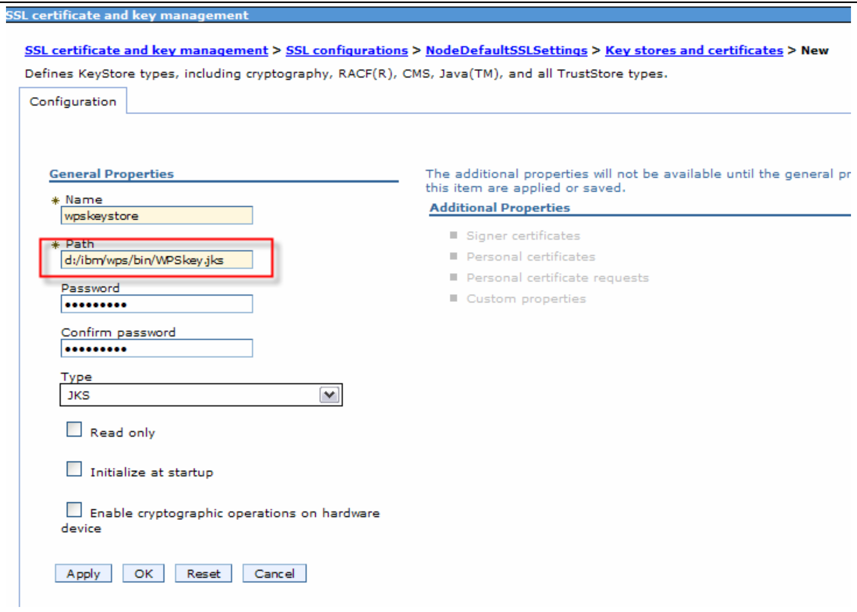
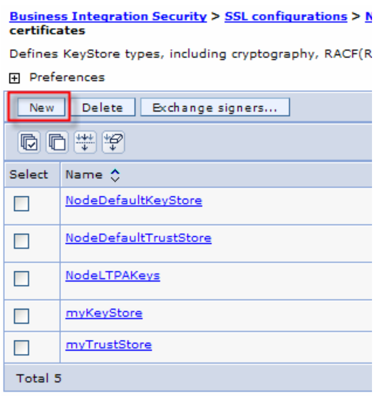
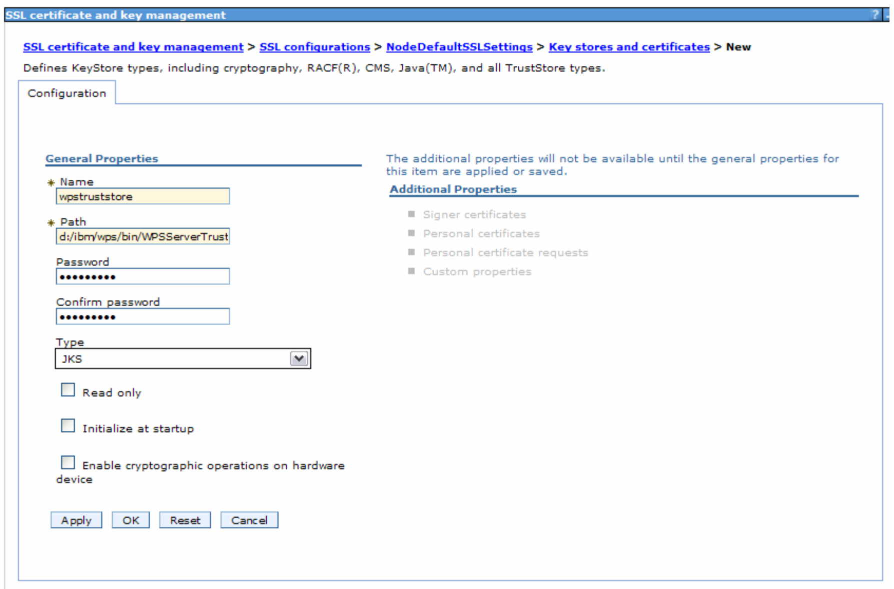
If you have more than one SSL client (as we have in the showcase) and you want to only accept request from DNs matching specific values, you have to create additional channels.
In the showcase we have two SSL channels:

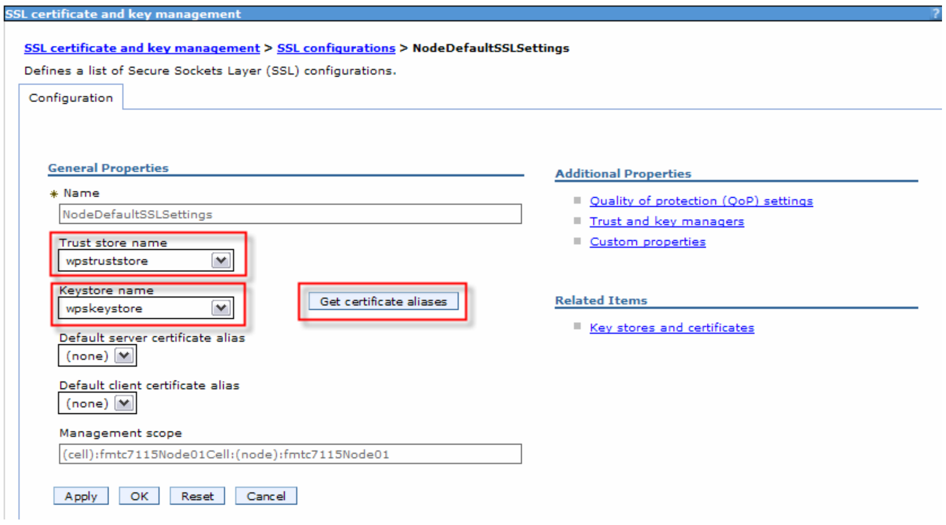
- **SSL.SRVCONN for WPS**
- **SSLWAS for WAS**

Certificate security warning
As you configure certificate keystores for WebSphere MQ, remember that each signing certificate in the keystore represents trust between you and that signer (typically a Certificate Authority, CA). In the most basic case, placing any signing certificate in the WMQ Server keystore without DN verification means that WebSphere MQ should accept all connects from any party that has a certificate from that CA. Unless you are using self-signed certificates or have a dedicated CA just for WebSphere MQ, that is almost completely insecure. Thus, we restrict the certificates to those with the matching DN value that we specify. That ensures that the identity in the certificate is really the identity that we expect. However, there is a catch. If two CAs were to issue certificates with the same DN, our security would again be compromised. That should not happen since a reputable CA would not do such a thing, but two different CAs might issue certificates with the same subject, which is why you need to remove all of the certificates except for the certificate from the CA you expect.

6.1.5 Configure the WebSphere Application Server JMS client

1.	In the WebSphere administrative console, navigate to *Security > SSL certificate and key management > SSL configurations*
2.	Select *NodeDefaultSSLSettings*
3.	Select *Key stores and certificates*
4.	Create a new KeyStore by clicking *New*  Business Integration Security > SSL configurations > N certificates Defines KeyStore types, including cryptography, RACF(R ⊞ Preferences New Delete Exchange signers... Select | Name ☐ NodeDefaultKeyStore ☐ NodeDefaultTrustStore ☐ NodeLTPAKeys ☐ myKeyStore ☐ myTrustStore Total 5
5.	Name the new keystore, for example, *wpskeystore* Change path to *WPS_INSTALL_ROOT/bin/WPSkey.jks* Enter a password (e.g websphere) Select as Type *JKS* Click *OK* and Save

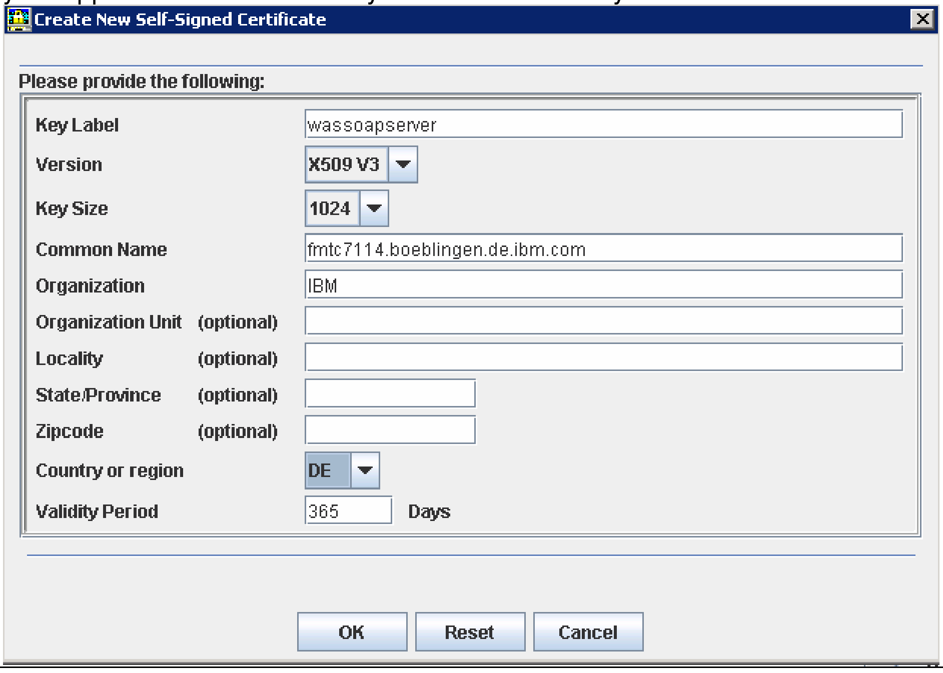
	 SSL certificate and key management SSL certificate and key management > SSL configurations > NodeDefaultSSLSettings > Key stores and certificates > New Defines KeyStore types, including cryptography, RACF(R), CMS, Java(TM), and all TrustStore types. Configuration General Properties * Name: wpskeystore * Path: d:/ibm/wps/bin/WPSkey.jks Password: ••••••••• Confirm password: ••••••••• Type: JKS ☐ Read only ☐ Initialize at startup ☐ Enable cryptographic operations on hardware device Apply OK Reset Cancel The additional properties will not be available until the general pr this item are applied or saved. Additional Properties Signer certificates Personal certificates Personal certificate requests Custom properties
6.	Create a new TrustStore by clicking *New*  Business Integration Security > SSL configurations > N certificates Defines KeyStore types, including cryptography, RACF(R ⊞ Preferences New Delete Exchange signers... Select | Name ☐ NodeDefaultKeyStore ☐ NodeDefaultTrustStore ☐ NodeLTPAKeys ☐ myKeyStore ☐ myTrustStore Total 5
7.	Name the new truststore, for example, *wpstruststore* Change path to *WPS_INSTALL_ROOT/bin/WPSServerTrustFile.jks* Select as Type *JKS* Click *OK* and Save  SSL certificate and key management SSL certificate and key management > SSL configurations > NodeDefaultSSLSettings > Key stores and certificates > New Defines KeyStore types, including cryptography, RACF(R), CMS, Java(TM), and all TrustStore types. Configuration General Properties * Name: wpstruststore * Path: d:/ibm/wps/bin/WPSServerTrust Password: ••••••••• Confirm password: ••••••••• Type: JKS ☐ Read only ☐ Initialize at startup ☐ Enable cryptographic operations on hardware device Apply OK Reset Cancel The additional properties will not be available until the general properties for this item are applied or saved. Additional Properties Signer certificates Personal certificates Personal certificate requests Custom properties

8.	Navigate to *Business Integration Security > SSL configurations > NodeDefaultSSLSettings* Select as Trust store name *wpstruststore* Select as keystore name *wpskeystore* Click *OK* and save  SSL certificate and key management SSL certificate and key management > SSL configurations > NodeDefaultSSLSettings Defines a list of Secure Sockets Layer (SSL) configurations. Configuration General Properties * Name NodeDefaultSSLSettings Trust store name wpstruststore Keystore name wpskeystore Get certificate aliases Default server certificate alias (none) Default client certificate alias (none) Management scope (cell):fmtc7115Node01Cell:(node):fmtc7115Node01 Apply OK Reset Cancel Additional Properties Quality of protection (QoP) settings Trust and key managers Custom properties Related Items Key stores and certificates

6.2 SSL between WMB and WAS

This chapter describes how to setup SSL between WMB and WAS for SOAP/HTTPs.

6.2.1 Create Self-Signed Certficate for WAS

1.	Switch to the version of ikeyman that comes with WebSphere Application Server by launching <was root>\bin\ikeyman.bat
2.	From the ikeyman menu, select Key Database File => New
3.	On the Open dialog, for key database type, accept the default value of JKS (Java™ keystore).
4.	Save the file as WASKey.jks
5.	When prompted to create a keystore password, select a valid password and confirm it (websphere)
6.	Delete all signer certificates from the Signer Certificates tab. As mentioned earlier, limiting signers limits risk
7.	Switch to Personal Certificates, and click New Self-Signed
8.	Enter data in the Create New Self-Signed certificate dialog with values appropriate to the location of your application server. Set Key Label to a value of your choice  Create New Self-Signed Certificate Please provide the following: Key Label: wassoapserver Version: X509 V3 Key Size: 1024 Common Name: fmtc7114.boeblingen.de.ibm.com Organization: IBM Organization Unit (optional): Locality (optional): State/Province (optional): Zipcode (optional): Country or region: DE Validity Period: 365 Days

6.2.2 Export the WAS self-signed certificate

At this point, we have created a self-signed certificate for the WebSphere Application Server. We now need to extract the certificate and place it in the trust file for WebSphere Application Server and WebSphere MB, so that they can both use it as a signer:
To export the certificate:

a. With the ikeyman database open to the WASKey file, and the certificate selected, click Extract Certificate. This exports only the certificate (not the private key).
b. Save the certificate and give it an appropriate name, such as was_soap_server.arm.

6.2.3 Import to WMB

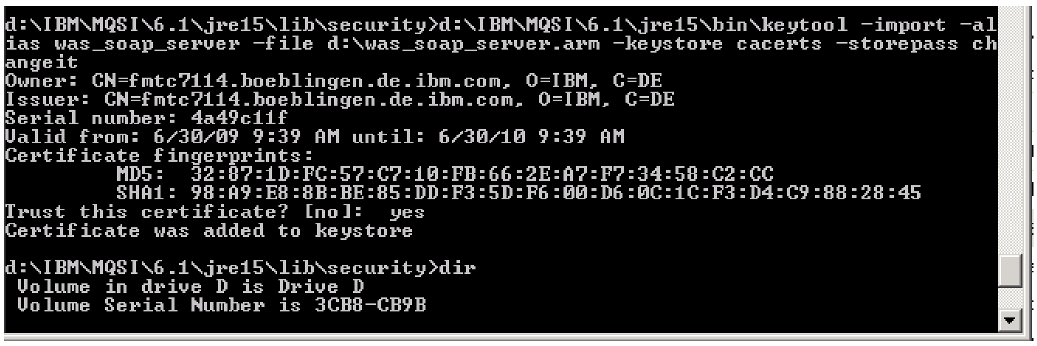
Refer also to the Info center at
http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r1m0/index.jsp?topic=/com.ibm.etools.mft.doc/ap12235_.htm

1. Adding certificates to the cacerts file
You must add the certificate for the WAS web service to the cacerts file for Message Broker. This file is the default trust store for the broker and is located in the broker's JRE security directory.
The cacerts file is located in the "%MQSI_FILEPATH%\jre\lib\security"

2. Importing a certificate into the cacerts file

Use the keytool command to modify the cacerts file:

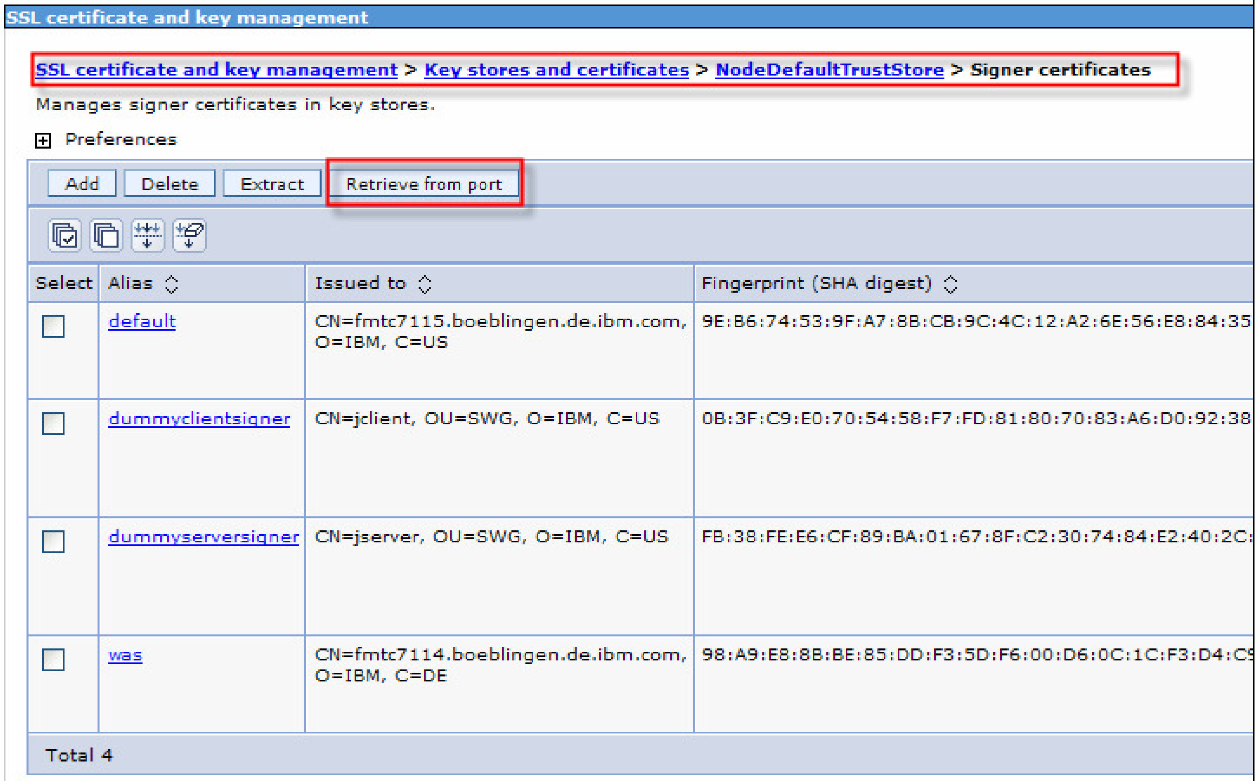
1.	Click Start > IBM WebSphere Message Broker 6.1 > Command Console to open a broker command console

2.	In the command console, type the following command:
	"%MQSI_FILEPATH%\jre\bin\keytool" -import -alias mykey -file name of certificate file -keystore cacerts -storepass changeit
	
	where:
	• *name of certificate file* is the fully qualified name of the certificates file. This file is typically found in the message broker user's home directory.
	• *changeit* is the default password for the cacerts file. You can use the keytool command to change the password, but, because it is not a configurable property of the broker, the broker always attempts to access the cacerts file using the default password changeit.
	
	
	`d:\IBM\MQSI\6.1\jre15\lib\security>d:\IBM\MQSI\6.1\jre15\bin\keytool -import -al`
	`ias was_soap_server -file d:\was_soap_server.arm -keystore cacerts -storepass ch`
	`angeit`
	`Owner: CN=fmtc7114.boeblingen.de.ibm.com, O=IBM, C=DE`
	`Issuer: CN=fmtc7114.boeblingen.de.ibm.com, O=IBM, C=DE`
	`Serial number: 4a49c11f`
	`Valid from: 6/30/09 9:39 AM until: 6/30/10 9:39 AM`
	`Certificate fingerprints:`
	`         MD5:  32:87:1D:FC:57:C7:10:FB:66:2E:A7:F7:34:58:C2:CC`
	`         SHA1: 98:A9:E8:8B:BE:85:DD:F3:5D:F6:00:D6:0C:1C:F3:D4:C9:88:28:45`
	`Trust this certificate? [no]:  yes`
	`Certificate was added to keystore`
	
	`d:\IBM\MQSI\6.1\jre15\lib\security>dir`
	` Volume in drive D is Drive D`
	` Volume Serial Number is 3CB8-CB9B`

3.	Verify that the cacerts file was updated by looking at the change date of the cacerts file.
4.	Restart WMB

6.3 SSL between WPS and WAS

6.3.1 Configure WPS (client) for SSL

1.	From the administrative console, follow *Security > SSL certificate and key management > key stores and certificates > NodeDefaultTrustStore > Signer certificates > Retrieve from port* 
2.	Enter the remote machine name in the Host field of the WAS server (see screenshot below)
3.	Enter *CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS 9402* of the remote machine as port (see screenshot below)
4.	Enter *Alias* for reference (see screenshot below)
5.	Click *Retrieve signer information* to retrieve the keys from WAS

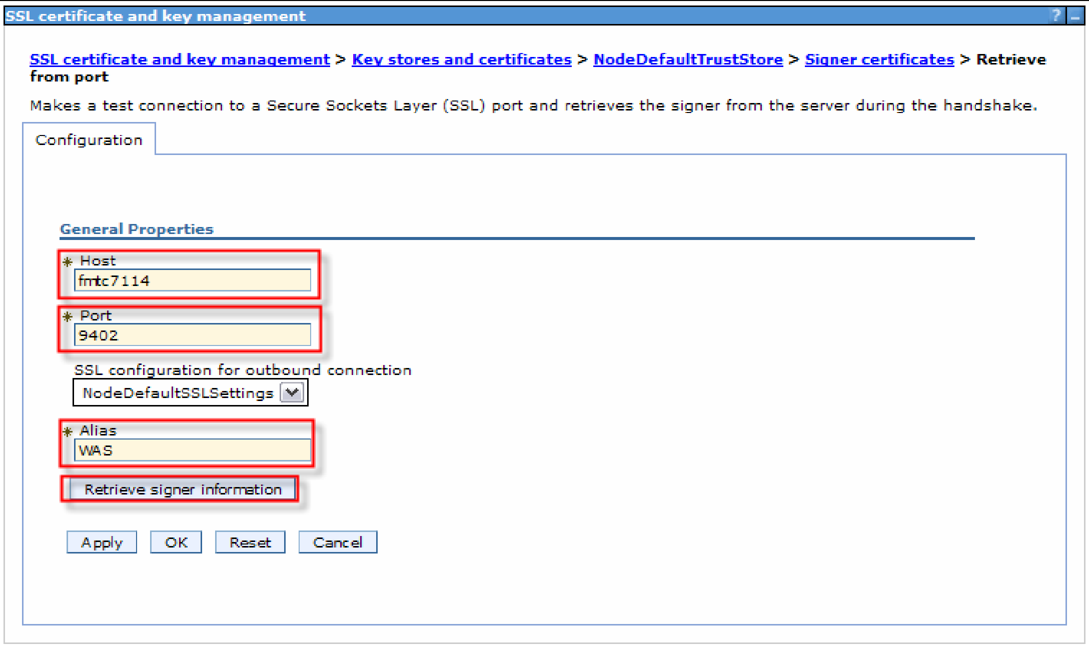
SSL certificate and key management

SSL certificate and key management > Key stores and certificates > NodeDefaultTrustStore > Signer certificates

Manages signer certificates in key stores.

Select	Alias	Issued to	Fingerprint (SHA digest)
☐	default	CN=fmtc7115.boeblingen.de.ibm.com, O=IBM, C=US	9E:B6:74:53:9F:A7:8B:CB:9C:4C:12:A2:6E:56:E8:84:35
☐	dummyclientsigner	CN=jclient, OU=SWG, O=IBM, C=US	0B:3F:C9:E0:70:54:58:F7:FD:81:80:70:83:A6:D0:92:38
☐	dummyserversigner	CN=jserver, OU=SWG, O=IBM, C=US	FB:38:FE:E6:CF:89:BA:01:67:8F:C2:30:74:84:E2:40:2C:
☐	was	CN=fmtc7114.boeblingen.de.ibm.com, O=IBM, C=DE	98:A9:E8:8B:BE:85:DD:F3:5D:F6:00:D6:0C:1C:F3:D4:C

Total 4



SSL certificate and key management

SSL certificate and key management > Key stores and certificates > NodeDefaultTrustStore > Signer certificates > Retrieve from port

Makes a test connection to a Secure Sockets Layer (SSL) port and retrieves the signer from the server during the handshake.

Configuration

General Properties

* Host: fmtc7114

* Port: 9402

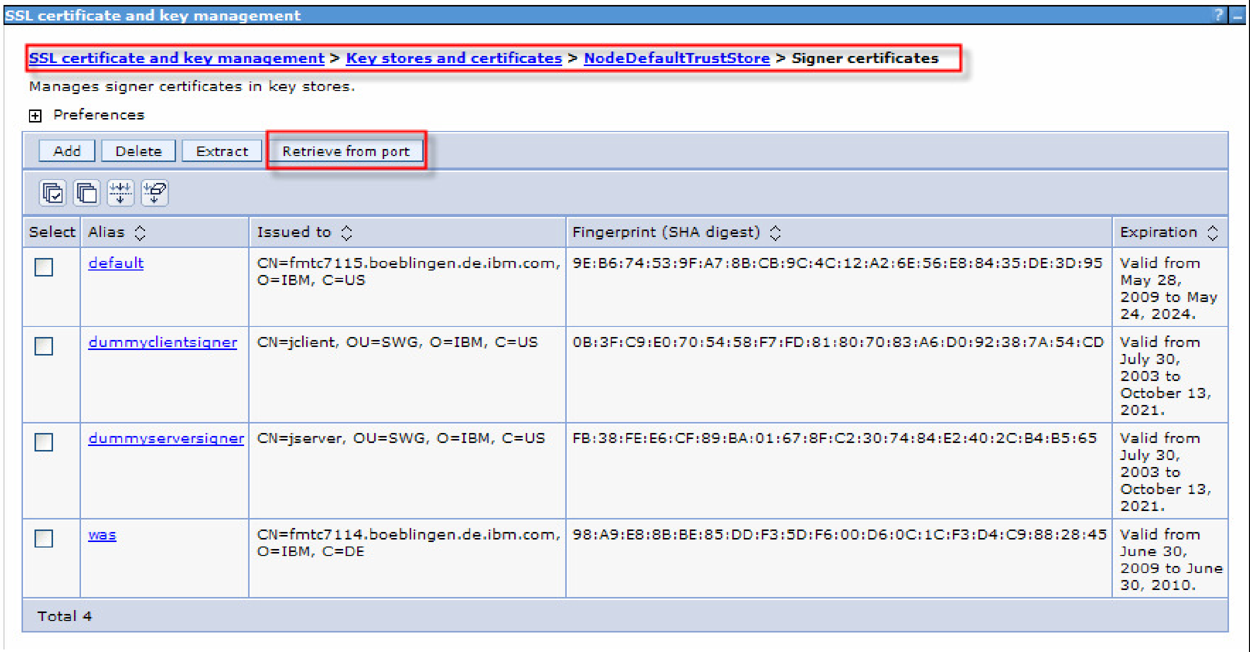
SSL configuration for outbound connection: NodeDefaultSSLSettings

* Alias: WAS

6. Apply and save the changes

6.3.2 Configure WAS (server) for SSL

1. From the administrative console, follow
Security > SSL certificate and key management > key stores and certificates > NodeDefaultTrustStore -> Signer certificates > Retrieve from port



SSL certificate and key management

SSL certificate and key management > Key stores and certificates > NodeDefaultTrustStore > Signer certificates

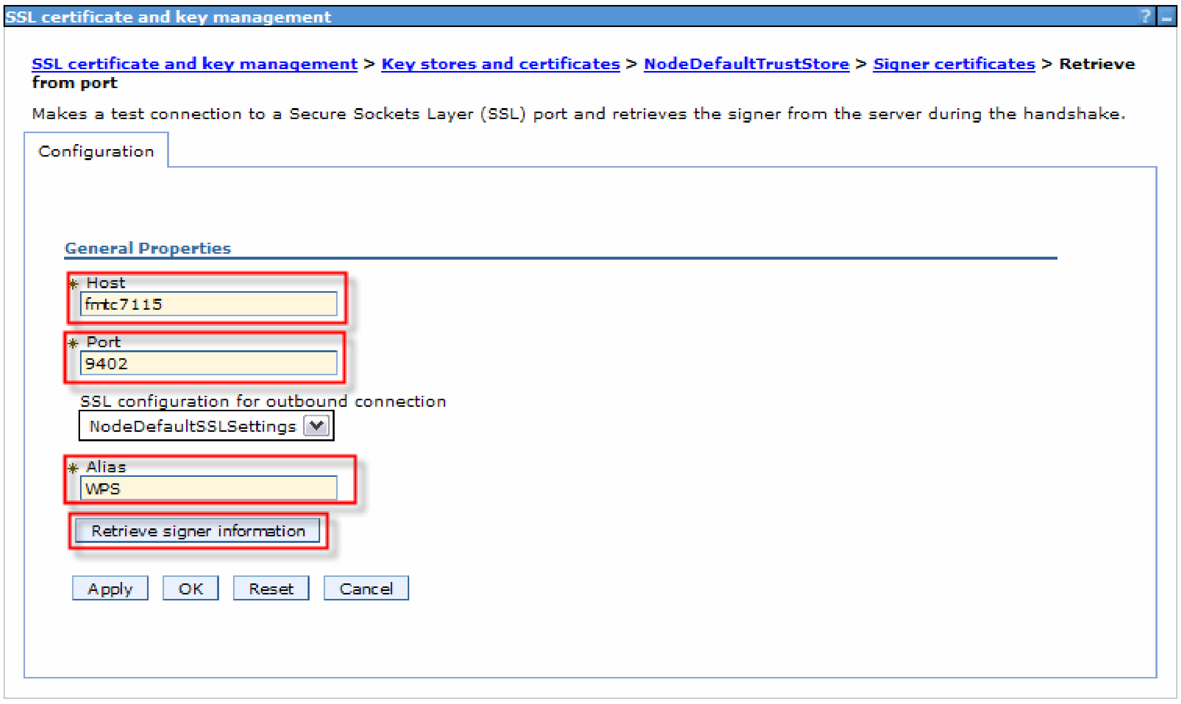
Manages signer certificates in key stores.

Select	Alias	Issued to	Fingerprint (SHA digest)	Expiration
☐	default	CN=fmtc7115.boeblingen.de.ibm.com, O=IBM, C=US	9E:B6:74:53:9F:A7:8B:CB:9C:4C:12:A2:6E:56:E8:84:35:DE:3D:95	Valid from May 28, 2009 to May 24, 2024.
☐	dummyclientsigner	CN=jclient, OU=SWG, O=IBM, C=US	0B:3F:C9:E0:70:54:58:F7:FD:81:80:70:83:A6:D0:92:38:7A:54:CD	Valid from July 30, 2003 to October 13, 2021.
☐	dummyserversigner	CN=jserver, OU=SWG, O=IBM, C=US	FB:38:FE:E6:CF:89:BA:01:67:8F:C2:30:74:84:E2:40:2C:B4:B5:65	Valid from July 30, 2003 to October 13, 2021.
☐	was	CN=fmtc7114.boeblingen.de.ibm.com, O=IBM, C=DE	98:A9:E8:8B:BE:85:DD:F3:5D:F6:00:D6:0C:1C:F3:D4:C9:88:28:45	Valid from June 30, 2009 to June 30, 2010.

Total 4

2. Enter remote machine name in the Host field(see screenshot below)

3. Enter *CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS 9402* of the remote machine at Port (see screenshot below)

4.	Enter *Alias* for reference (see screenshot below)
5.	Click *Retrieve signer information*  SSL certificate and key management SSL certificate and key management > Key stores and certificates > NodeDefaultTrustStore > Signer certificates > Retrieve from port Makes a test connection to a Secure Sockets Layer (SSL) port and retrieves the signer from the server during the handshake. Configuration General Properties * Host fmtc7115 * Port 9402 SSL configuration for outbound connection NodeDefaultSSLSettings * Alias WPS
6.	Apply and save the changes

7 Patterns (Interactions)

7.1 Sequence of interactions

In this chapter we describe each step of the showcase. We describe

- how to **configure** the specifications for security – on consumer and provider side
- the **implementation** of the step in WAS, in WMB, and WPS.

How to read the configuration steps

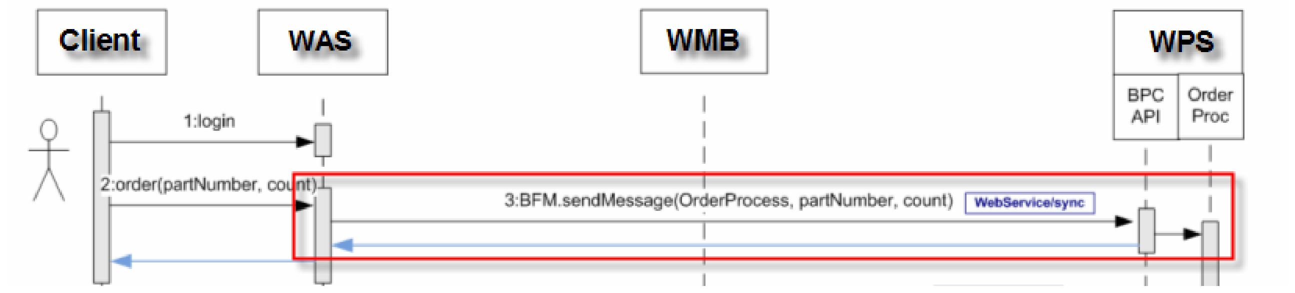
- We have chosen the approach which describes specifications in the message flow or BPEL application directly with server configuration to show its dependencies.
- The configuration is described from the consumer side first and the corresponding settings on the provider side. In some steps it is described the opposite way as the settings are driven by the provider side.

7.1.1 Interaction 1 to 3 - Start Process

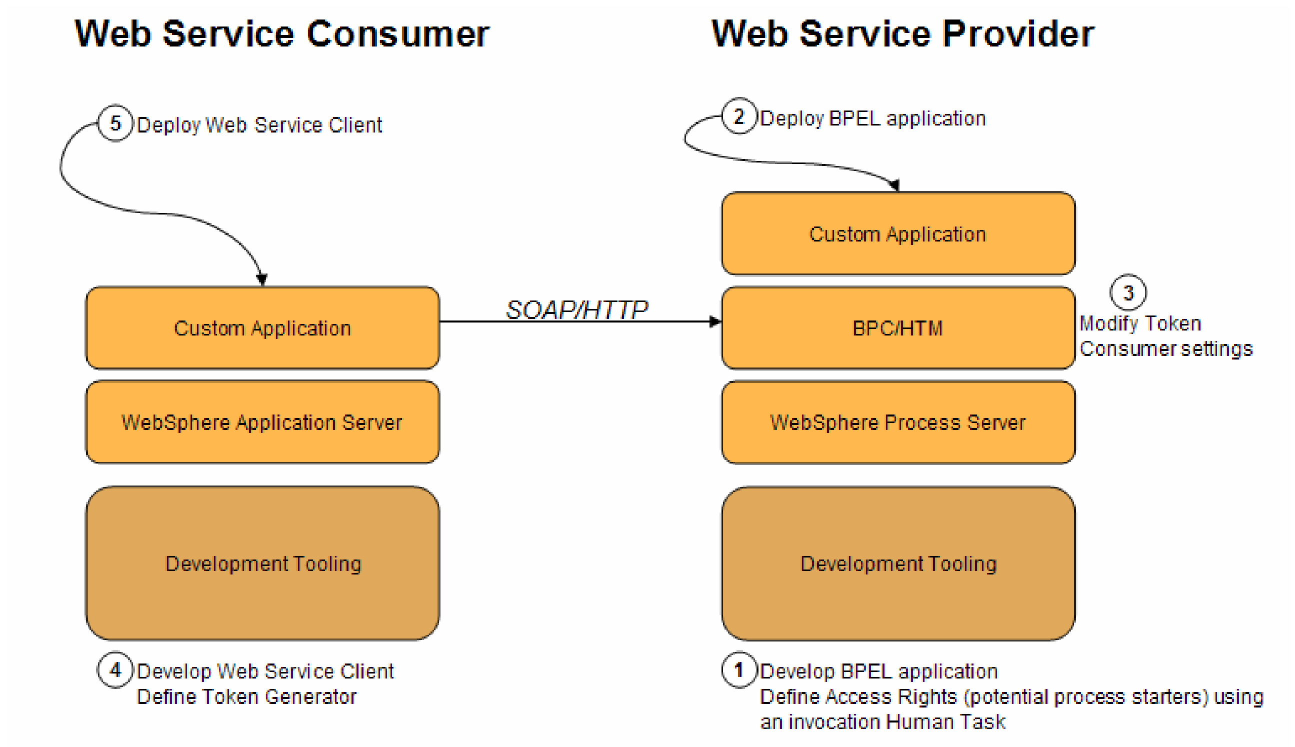
This section describes how to propagate the identity using Username Tokens from WebSphere Application Server to WebSphere Process Server via SOAP/http.

Client Application	Server Application
StartProcessEAR_3 (WAS)	ShowcaseApp (WPS, BPC Web Service API)

The figure below shows the relevant part in the sequence diagram:


Client
WAS
WMB
WPS
BPC API
Order Proc
1:login
2:order(partNumber, count)
3:BFM.sendMessage(OrderProcess, partNumber, count)
WebService/sync

The figure below shows the high-level implementation and configuration steps that have to be performed. Find details of the steps in the next sections.


Web Service Consumer
Web Service Provider
5 Deploy Web Service Client
2 Deploy BPEL application
Custom Application
SOAP/HTTP
Custom Application
BPC/HTM
3 Modify Token Consumer settings
WebSphere Application Server
WebSphere Process Server
Development Tooling
Development Tooling
4 Develop Web Service Client
Define Token Generator
1 Develop BPEL application
Define Access Rights (potential process starters) using an invocation Human Task

7.1.1.1 Optional implementation

In this scenario we use Username tokens to propagate the identity. Another option would be to use an LTPA token instead the Username Token. Following table lists pros and cons using LTPA- and Username tokens:

	LTPA Token	**Username Token**
Pro	+ No SSL required, LTPA Key exchange is sufficient	+ No Realm in token: No Realm mapping required, if using different realms
Con	- If different Realms: Realm mapping required	- Identity assertion must be configured, if password is not known - SSL must be configured

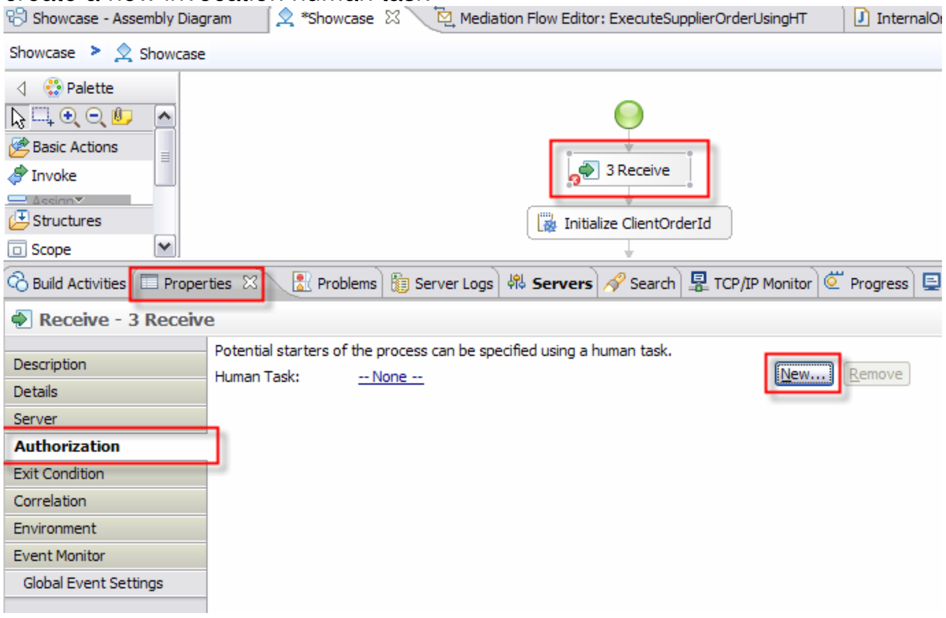
7.1.1.2 Detailed description of the implementation and configuration steps

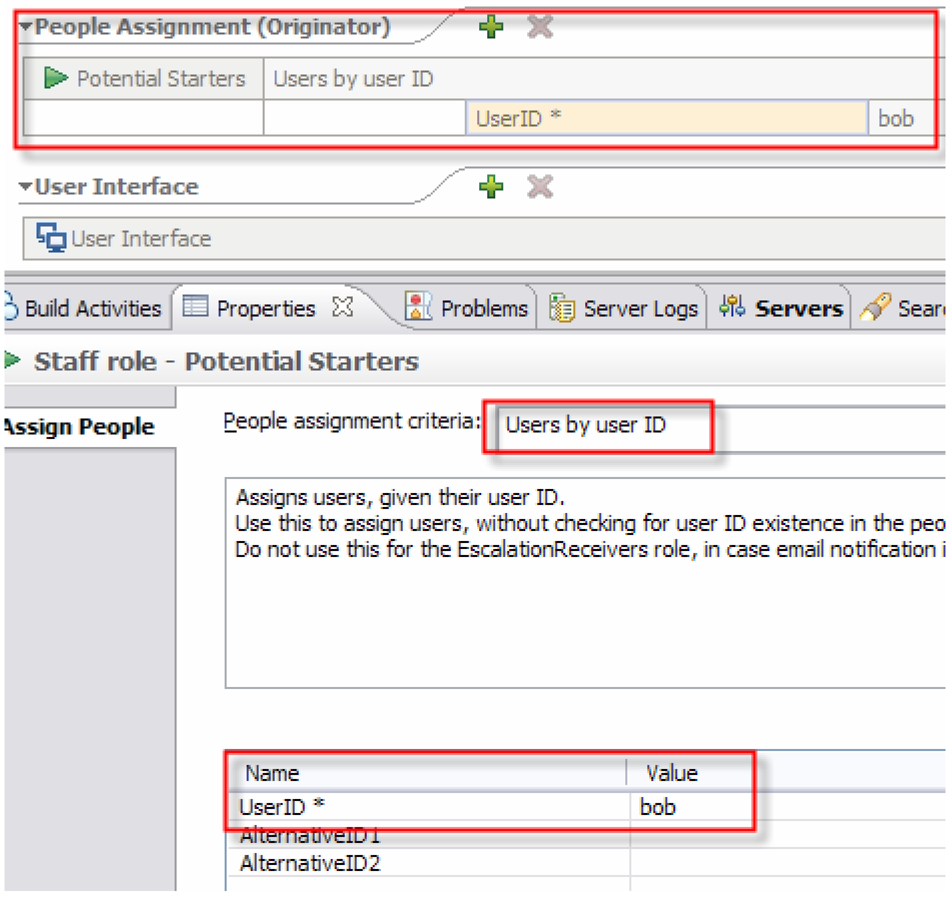
To configure the Username Token with identity assertion and SSL between WebSphere Application Server and WebSphere Process Server, follow the next steps.

7.1.1.3 Step 1 – WPS: Develop the BPEL Application and define potential process starters

Detailed implementation steps, which are not security relevant, are not described. Refer to the WID artefacts to see how the BPEL application is developed.

To define who (users, groups, dynamic staff assignments) is allowed to start a process, an invocation human task must be defined on the receive activity of the BPEL process.

1.	In the BPEL editor, click on the *Receive* activity and select under *Properties > Authorization > New* to create a new invocation human task  Showcase - Assembly Diagram *Showcase Mediation Flow Editor: ExecuteSupplierOrderUsingHT Showcase > Showcase Palette Basic Actions Invoke Structures Scope 3 Receive Initialize ClientOrderId Build Activities Properties Problems Server Logs Servers Search TCP/IP Monitor Progress Receive - 3 Receive Description Details Server Authorization Exit Condition Correlation Environment Event Monitor Global Event Settings New... Remove Potential starters of the process can be specified using a human task. Human Task: -- None --
2.	Click on *Potential starters* and select under *Properties* the *People assignment criteria*. In our case it is just a User ID. It could be also a Group of users, or a dynamic staff assignment.

People Assignment (Originator)

Potential Starters	Users by user ID	
	UserID *	bob

User Interface

User Interface

Build Activities | Properties | Problems | Server Logs | Servers | Sear

Staff role - Potential Starters

ssign People

People assignment criteria: Users by user ID

Assigns users, given their user ID.
Use this to assign users, without checking for user ID existence in the peo
Do not use this for the EscalationReceivers role, in case email notification i

Name	Value
UserID *	bob
AlternativeID1	
AlternativeID2	

7.1.1.4 Step 2 – WPS: Deploy BPEL Application

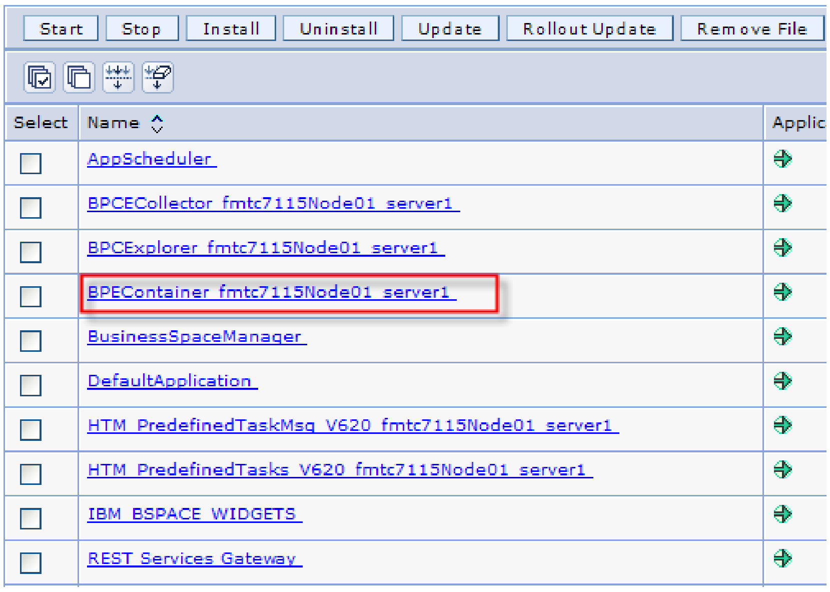
Detailed deployment steps are described in the appendix.

7.1.1.5 Step 3 – WPS: Modify Token consumer settings on the BPC Container

By default, the Business Process Container application accepts LTPA- and Username Tokens.
We have to modify the Web Service security bindings of the Username Token consumer to use User Id assertion as we do not send the User's password from WAS to WPS.

The following sequence describes the detailed configuration steps how to modify the Web Service security bindings of the Username Token consumer to use User ID assertion for the BPCContainer application.

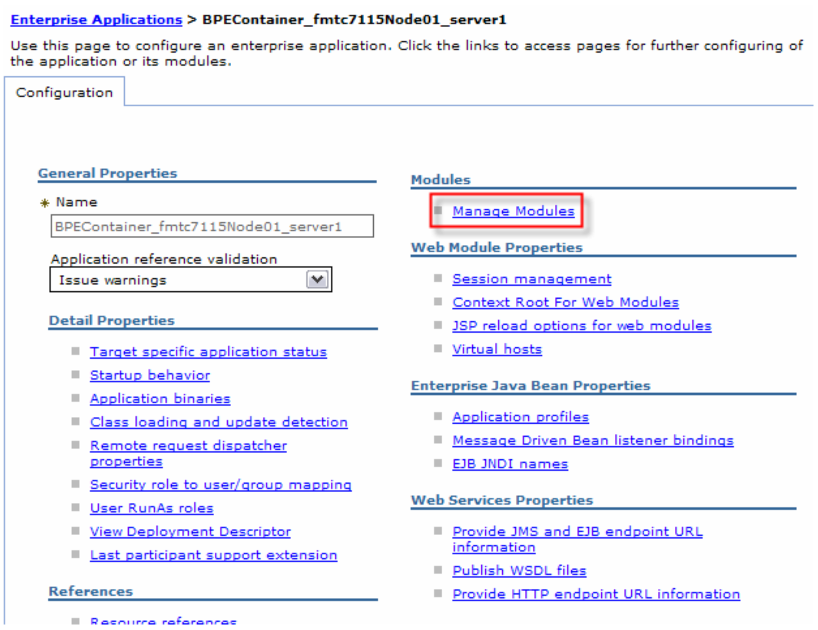
3.	To modify the security binding click in the Admin Console on *Applications->Enterprise Applications->BPCContainer_<yourDeploymentTarget>*



Start | Stop | Install | Uninstall | Update | Rollout Update | Remove File

Select	Name	Applic
☐	AppScheduler	➜
☐	BPCECollector_fmtc7115Node01_server1	➜
☐	BPCExplorer_fmtc7115Node01_server1	➜
☐	BPEContainer_fmtc7115Node01_server1	➜
☐	BusinessSpaceManager	➜
☐	DefaultApplication	➜
☐	HTM_PredefinedTaskMsg_V620_fmtc7115Node01_server1	➜
☐	HTM_PredefinedTasks_V620_fmtc7115Node01_server1	➜
☐	IBM_BSPACE_WIDGETS	➜
☐	REST Services Gateway	➜

4. Click *Manage Modules*



Enterprise Applications > BPEContainer_fmtc7115Node01_server1

Use this page to configure an enterprise application. Click the links to access pages for further configuring of the application or its modules.

Configuration

General Properties

* Name

BPEContainer_fmtc7115Node01_server1

Application reference validation

Issue warnings

Detail Properties

- Target specific application status
- Startup behavior
- Application binaries
- Class loading and update detection
- Remote request dispatcher properties
- Security role to user/group mapping
- User RunAs roles
- View Deployment Descriptor
- Last participant support extension

References

- Resource references

Modules

- Manage Modules

Web Module Properties

- Session management
- Context Root For Web Modules
- JSP reload options for web modules
- Virtual hosts

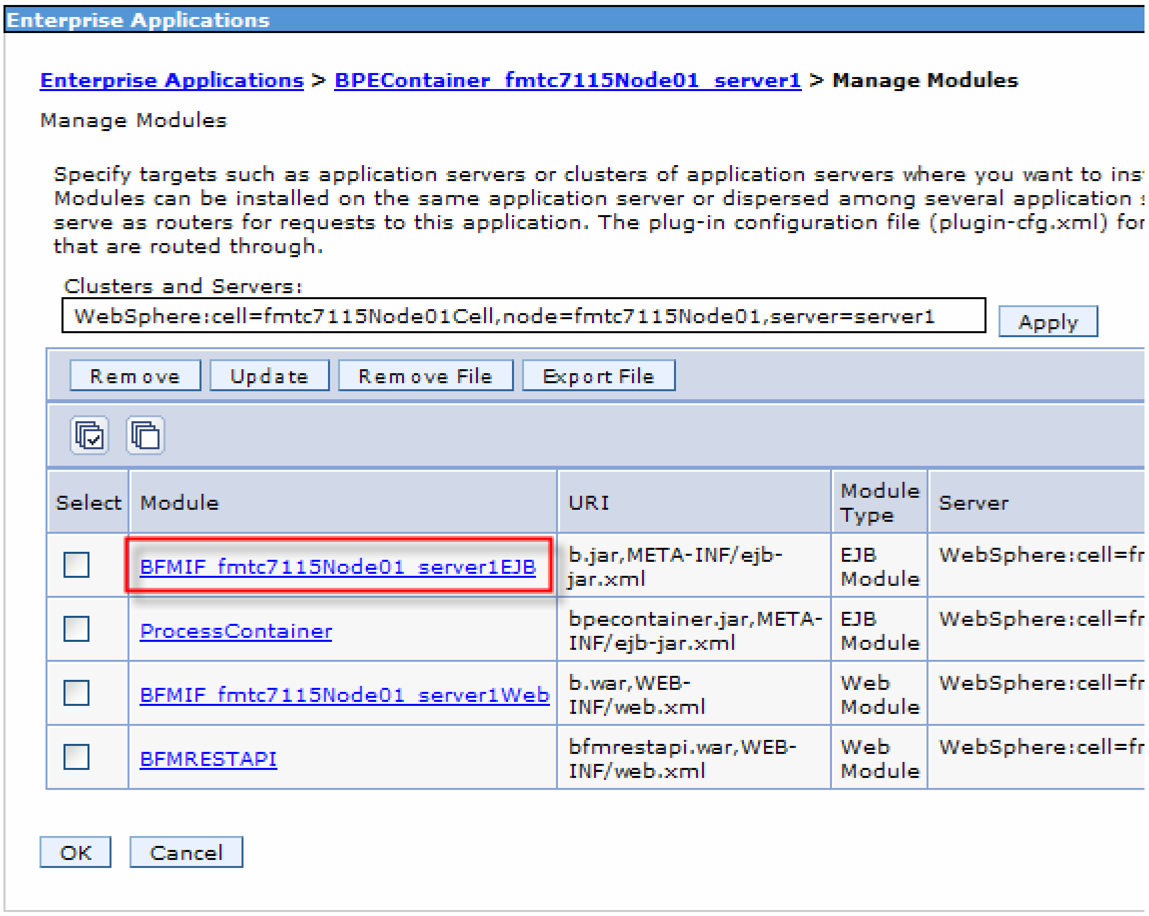
Enterprise Java Bean Properties

- Application profiles
- Message Driven Bean listener bindings
- EJB JNDI names

Web Services Properties

- Provide JMS and EJB endpoint URL information
- Publish WSDL files
- Provide HTTP endpoint URL information

5. Click on the module *BFIM_<yourDeploymentTarget>*



Enterprise Applications

Enterprise Applications > BPEContainer_fmtc7115Node01_server1 > **Manage Modules**

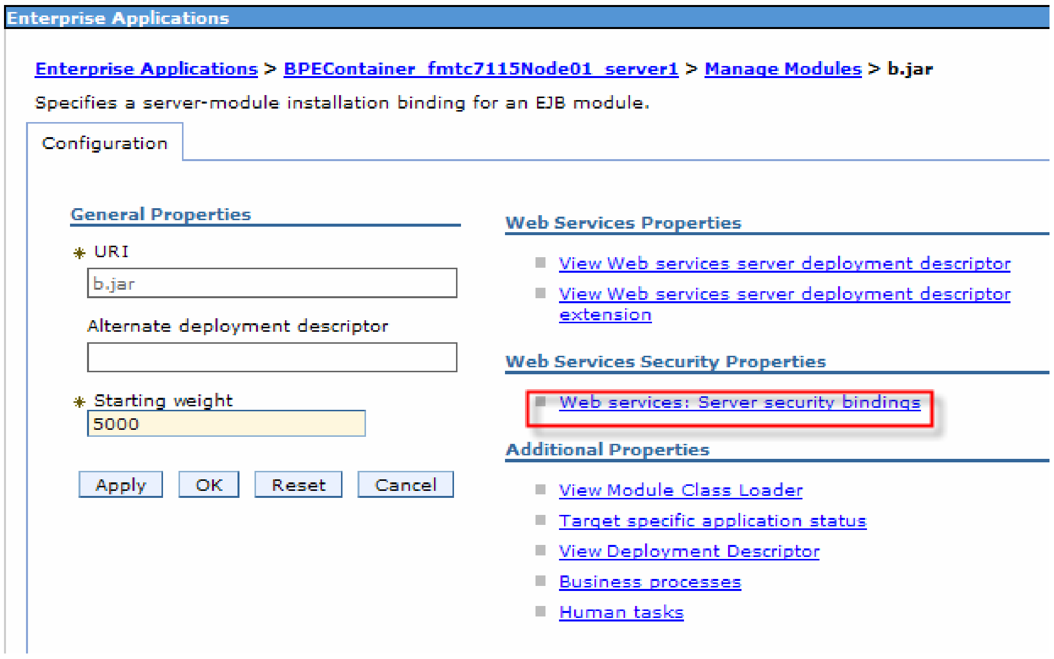
Manage Modules

Specify targets such as application servers or clusters of application servers where you want to ins
Modules can be installed on the same application server or dispersed among several application s
serve as routers for requests to this application. The plug-in configuration file (plugin-cfg.xml) for
that are routed through.

Clusters and Servers:
WebSphere:cell=fmtc7115Node01Cell,node=fmtc7115Node01,server=server1

Select	Module	URI	Module Type	Server
☐	BFMIF_fmtc7115Node01_server1EJB	b.jar,META-INF/ejb-jar.xml	EJB Module	WebSphere:cell=fr
☐	ProcessContainer	bpecontainer.jar,META-INF/ejb-jar.xml	EJB Module	WebSphere:cell=fr
☐	BFMIF_fmtc7115Node01_server1Web	b.war,WEB-INF/web.xml	Web Module	WebSphere:cell=fr
☐	BFMRESTAPI	bfmrestapi.war,WEB-INF/web.xml	Web Module	WebSphere:cell=fr

6. Click *Web services: Server security bindings*



Enterprise Applications

Enterprise Applications > BPEContainer_fmtc7115Node01_server1 > Manage Modules > **b.jar**

Specifies a server-module installation binding for an EJB module.

Configuration

General Properties

* URI
b.jar

Alternate deployment descriptor

* Starting weight
5000

Web Services Properties

- View Web services server deployment descriptor
- View Web services server deployment descriptor extension

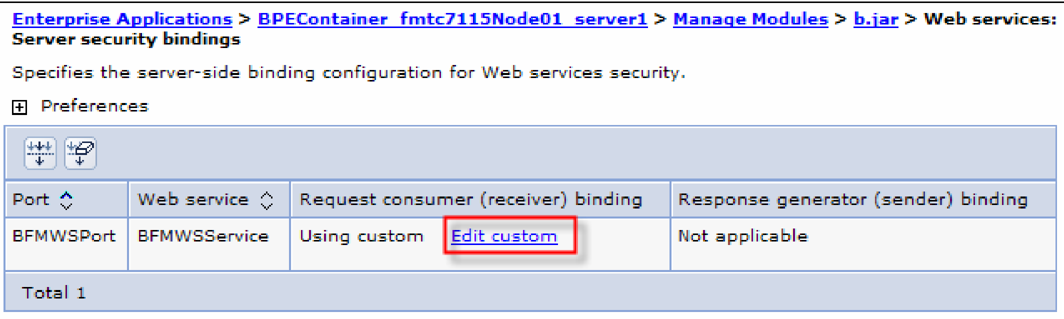
Web Services Security Properties

- Web services: Server security bindings

Additional Properties

- View Module Class Loader
- Target specific application status
- View Deployment Descriptor
- Business processes
- Human tasks

7. Click *Edit custom*



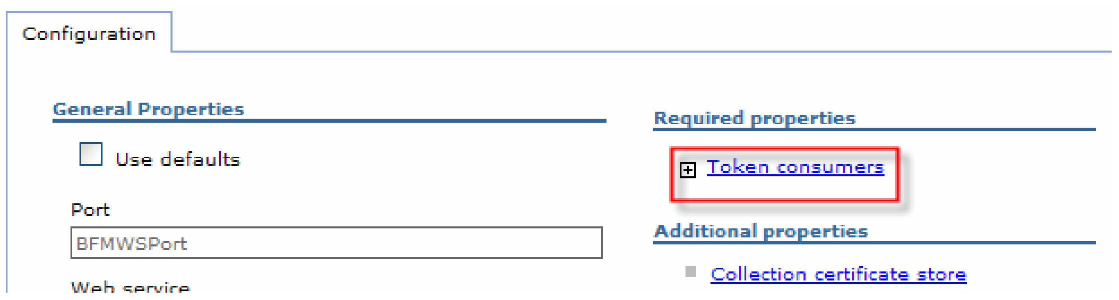
Enterprise Applications > BPEContainer_fmtc7115Node01_server1 > Manage Modules > b.jar > Web services: Server security bindings

Specifies the server-side binding configuration for Web services security.

⊞ Preferences

Port	Web service	Request consumer (receiver) binding	Response generator (sender) binding
BFMWSPort	BFMWSService	Using custom Edit custom	Not applicable
Total 1			

8. Click *Token consumers*



Configuration

General Properties

☐ Use defaults

Port

BFMWSPort

Web service

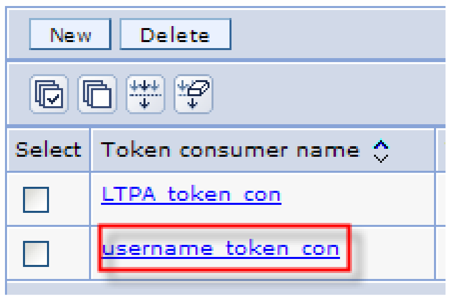
Required properties

⊞ Token consumers

Additional properties

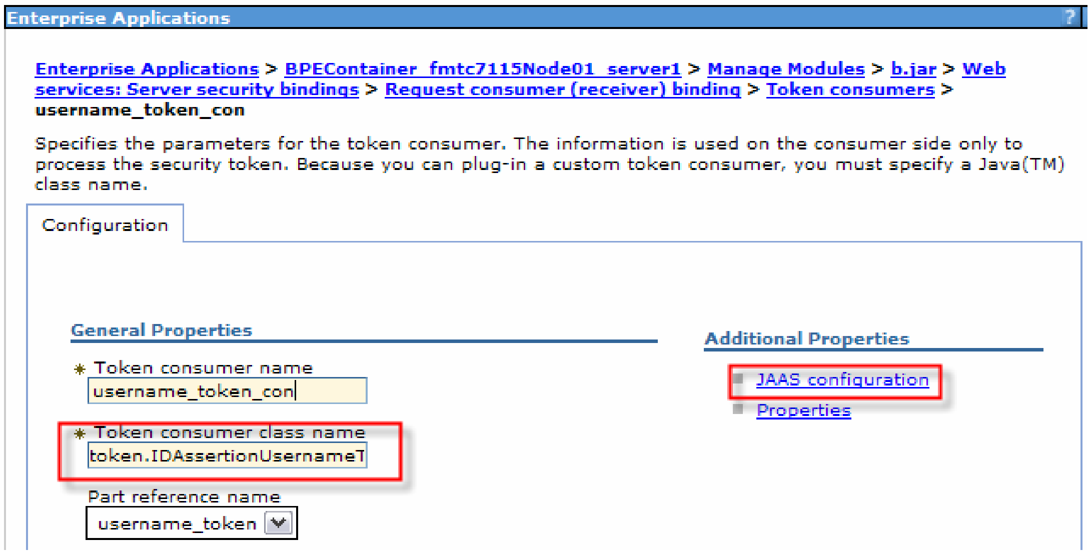
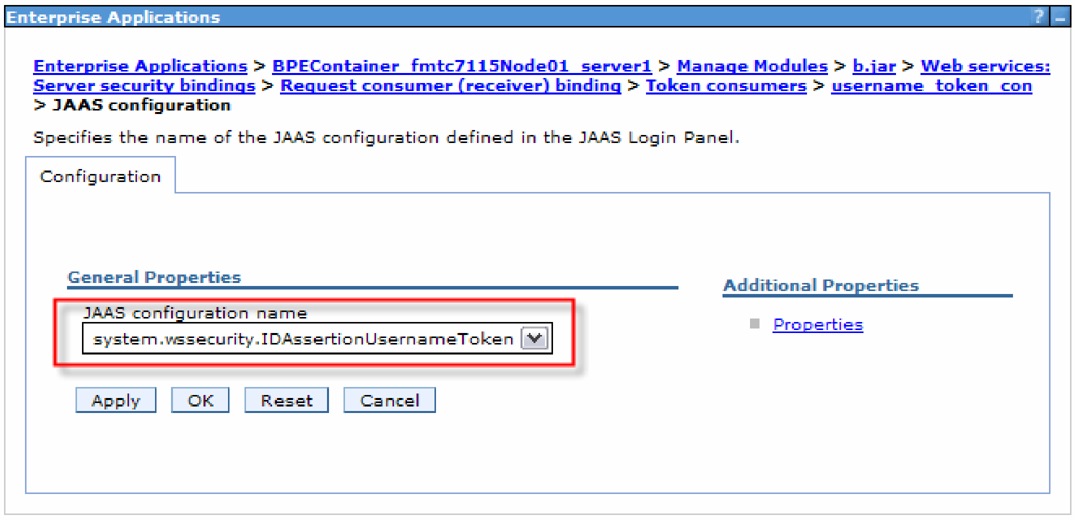
Collection certificate store

9. Click *username_token_con*



Select	Token consumer name
☐	LTPA_token_con
☐	username_token_con

10. Modify the Token consumer class name: Replace the existing entry
com.ibm.wsspi.wssecurity.token.UsernameTokenConsumer
with:
com.ibm.wsspi.wssecurity.token.IDAssertionUsernameTokenConsumer
and click *Apply* and *JAAS configuration*

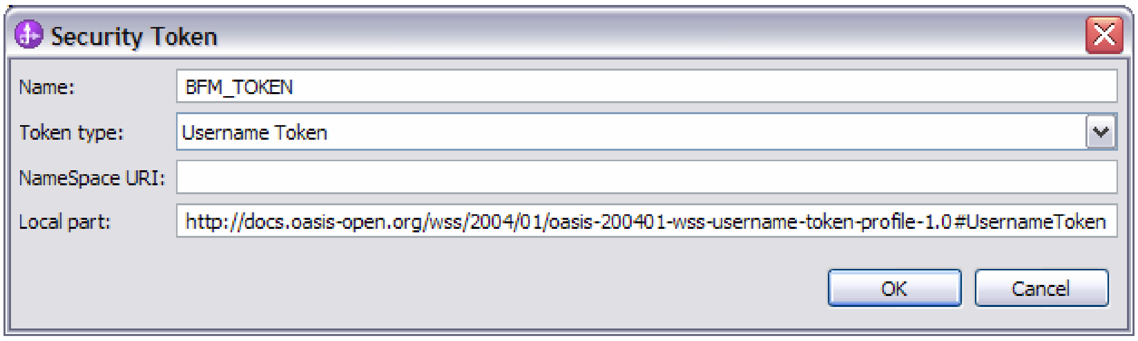
	 Enterprise Applications Enterprise Applications > BPEContainer_fmtc7115Node01_server1 > Manage Modules > b.jar > Web services: Server security bindings > Request consumer (receiver) binding > Token consumers > **username_token_con** Specifies the parameters for the token consumer. The information is used on the consumer side only to process the security token. Because you can plug-in a custom token consumer, you must specify a Java(TM) class name. Configuration General Properties * Token consumer name: username_token_con * Token consumer class name: token.IDAssertionUsernameT Part reference name: username_token Additional Properties JAAS configuration Properties
11.	Change the JAAS configuration name to ***system.wssecurity.IDAssertionUsernameToken*** Click *OK* and save  Enterprise Applications Enterprise Applications > BPEContainer_fmtc7115Node01_server1 > Manage Modules > b.jar > Web services: Server security bindings > Request consumer (receiver) binding > Token consumers > username_token_con > **JAAS configuration** Specifies the name of the JAAS configuration defined in the JAAS Login Panel. Configuration General Properties JAAS configuration name: system.wssecurity.IDAssertionUsernameToken Apply OK Reset Cancel Additional Properties Properties

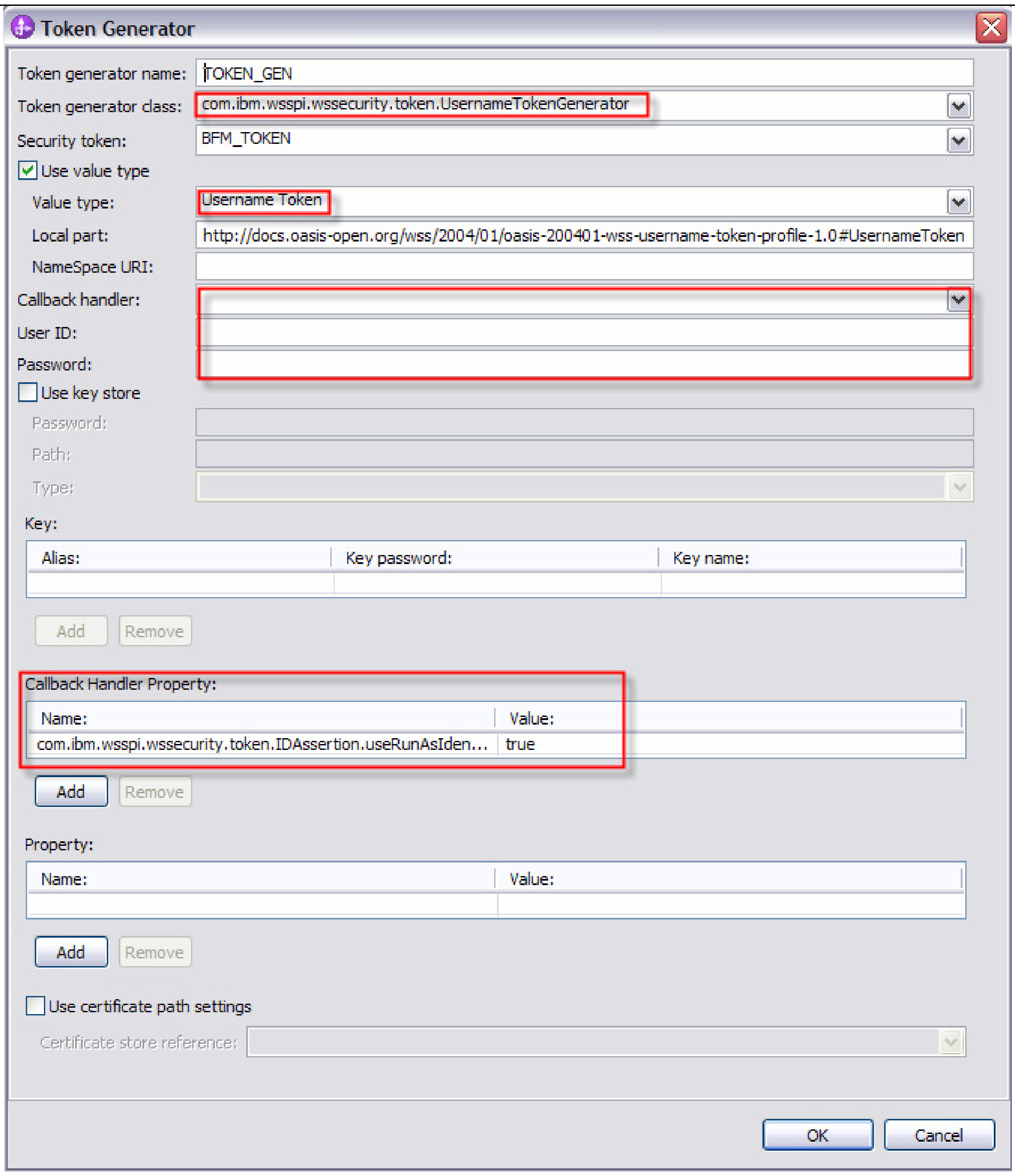
You have now modified the Web Service security bindings of the Username Token consumer to use User Id assertion.

7.1.1.6 Step 4 – WAS: Develop Web Service consumer application and define Token Generator – JAX RPC

Detailed implementation steps, which are not security relevant, are not described. Refer to the WID artefacts to see how the application is developed.

The consumer (WAS) has to send an Asserted Username Token to the BPC Web Service. This is a Username Token without password.

1.	To create a Username Token, open the Deployment Descriptor of the StartProcess_Web application
2.	Click the *WS Extension tab* Add a Security Token under *Request Generator Configuration* Select as Token type *Username Token* Local part is filled automatically. Click *OK*  Security Token Name: BFM_TOKEN Token type: Username Token NameSpace URI: Local part: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken
3.	Click the tab *WS-Security Bindings* Add a Token Generator under *Security Request Generator Configuration* Token Generator Name: TOKEN_GEN Token generator class: com.ibm.wsspi.wssecurity.token.UsernameTokenGenerator Security Token: BFM_TOKEN Use value type: Checked Callback handler: Blank UserID Blank Password Blank Callback handler Properties: com.ibm.wsspi.wssecurity.token.IDAssertion.isUsed=true com.ibm.wsspi.wssecurity.token.IDAssertion.useRunAsIdentity=true *You'll need to click the Add button to add a row and then select name and value fields to type over.*

	 Token Generator Token generator name: TOKEN_GEN Token generator class: com.ibm.wsspi.wssecurity.token.UsernameTokenGenerator Security token: BFM_TOKEN ☑ Use value type Value type: Username Token Local part: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken NameSpace URI: Callback handler: User ID: Password: ☐ Use key store Password: Path: Type: Key: Alias: / Key password: / Key name: Callback Handler Property: Name: com.ibm.wsspi.wssecurity.token.IDAssertion.useRunAsIden... Value: true Property: Name: / Value: ☐ Use certificate path settings Certificate store reference: Click OK
4.	Save the deployment descriptor

7.1.1.7 Step 4 – WAS: Develop Web Service consumer application and define Token Generator – JAX WS

When using JAX-WS you have two options to generate a token:

- By configuration using policy sets
- By implementation

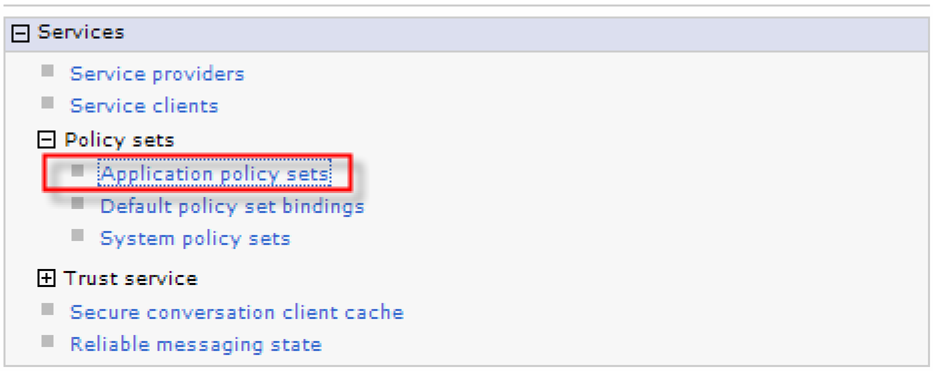
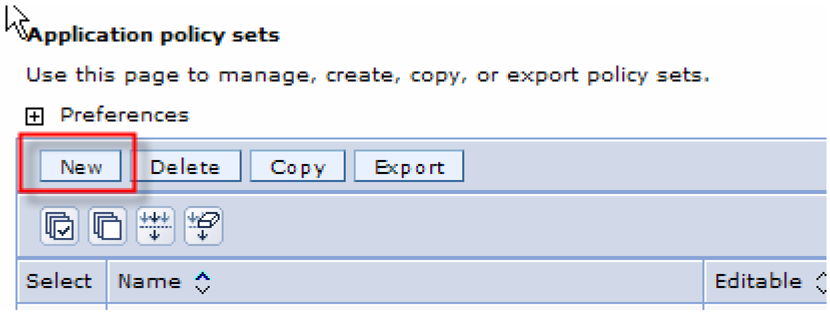
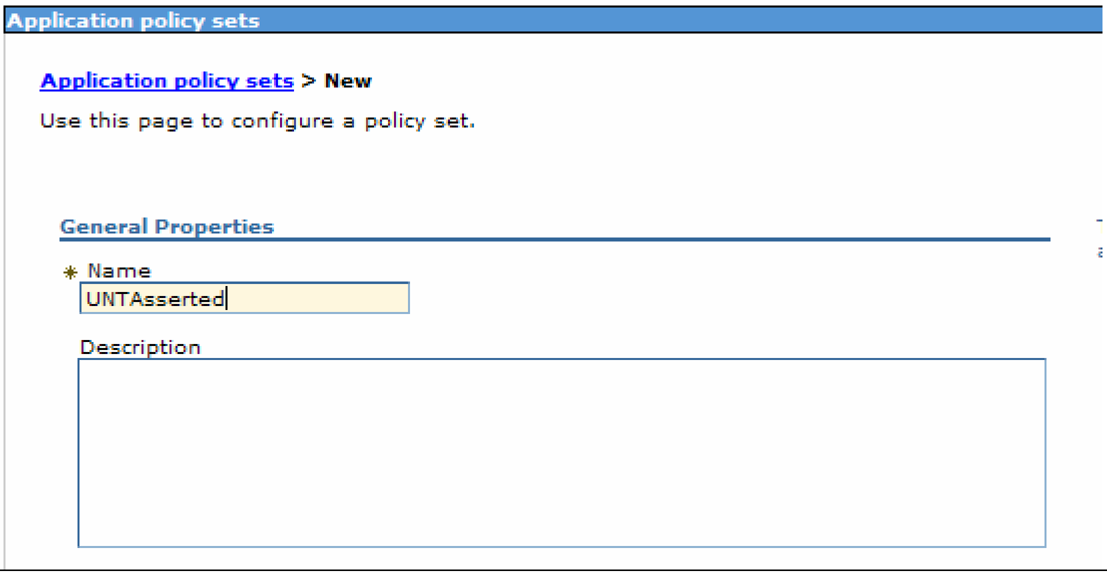
<u>Generate the token by programming (this option is implemented in the showcase):</u>

This section describes how to use the programmatic approach:

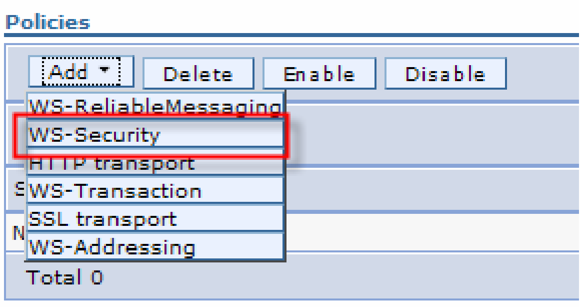
```
public void startProcess() {

        BFMWSService service = new BFMWSService();
        BFMIF bfm = service.getBFMWSPort();

        try {
            enhanceSecurity(bfm,
            com.ibm.websphere.security.auth.WSSubject.getCallerPrincipal(), "");

        } catch (WSSException e1) {
            e1.printStackTrace();
        }


        Order order = new Order();
        order.setClientEmail(getClientEmail());
        order.setPartNumber(getPartNumber());
        order.setPartCount(new Integer(getPartCount()));

        Start start = new Start();
        start.setOrder(order);

        com.ibm.xmlns.prod.websphere.business_process.services._6.SendMessage
        sendMessage = new ObjectFactory().createSendMessage();
        sendMessage.setProcessTemplateName("Showcase");
        sendMessage.setPortType(new QName("http://Showcase/Order", "Order"));
        sendMessage.setOperation("start");
        sendMessage.setAny(getElement(start));


    com.ibm.xmlns.prod.websphere.business_process.services._6.SendMessageRespons
    e response;
        try {
            response = bfm.sendMessage(sendMessage);
            setPiid(response.getPIID());
        } catch (ProcessFaultMsg e) {
            e.printStackTrace();
        }

    }




private void enhanceSecurity(BFMIF port, String user, String password) throws
WSSException {
        BindingProvider binding = (BindingProvider) port;
        Map requestContext = binding.getRequestContext();

        WSSFactory wssFactory = WSSFactory.getInstance();
        WSSGenerationContext genContext = wssFactory.newWSSGenerationContext();
        //UNTGenerateCallbackHandler untCallbackHandler = new
        UNTGenerateCallbackHandler(user, password, true, true);
        UNTGenerateCallbackHandler untCallbackHandler = new
        UNTGenerateCallbackHandler(user, null, true, true);
        SecurityToken secToken = wssFactory.newSecurityToken(UsernameToken.class,
        untCallbackHandler);

        genContext.add(secToken);
        genContext.process(requestContext);
    }
```

Generate the token by configuration

Create a new policy set for Username Tokens. We will not use the default Username Policy set, because it will also encrypt the message:

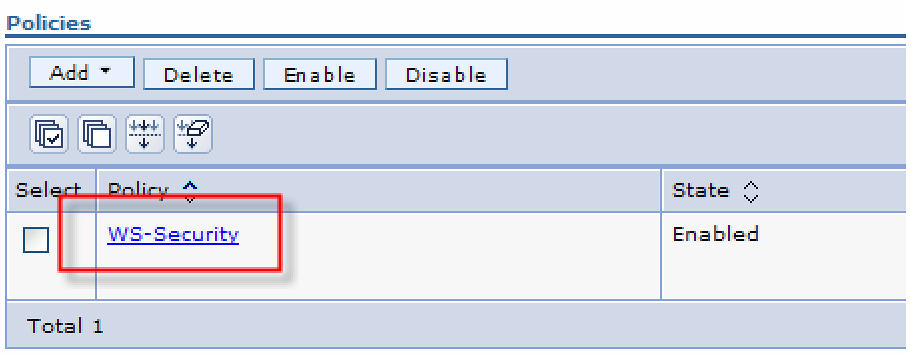
	In the administrative console click Services – Policy Sets – Application policy sets  Services · Service providers · Service clients · Policy sets · Application policy sets · Default policy set bindings · System policy sets · Trust service · Secure conversation client cache · Reliable messaging state
	Click *New*  **Application policy sets** Use this page to manage, create, copy, or export policy sets. Preferences · New · Delete · Copy · Export · Select · Name · Editable
	Enter a name and click Apply  **Application policy sets** Application policy sets > **New** Use this page to configure a policy set. General Properties · * Name: UNTAsserted · Description:

Click on Add and select WS-Security



Total 0

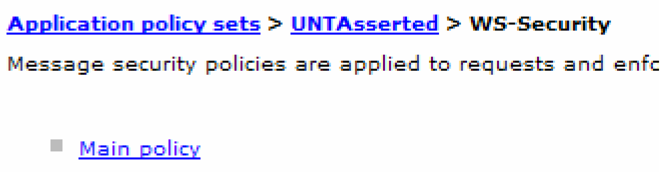
Click Apply and click on WS-Security



Select	Policy	State
☐	WS-Security	Enabled

Total 1

Click on Main policy

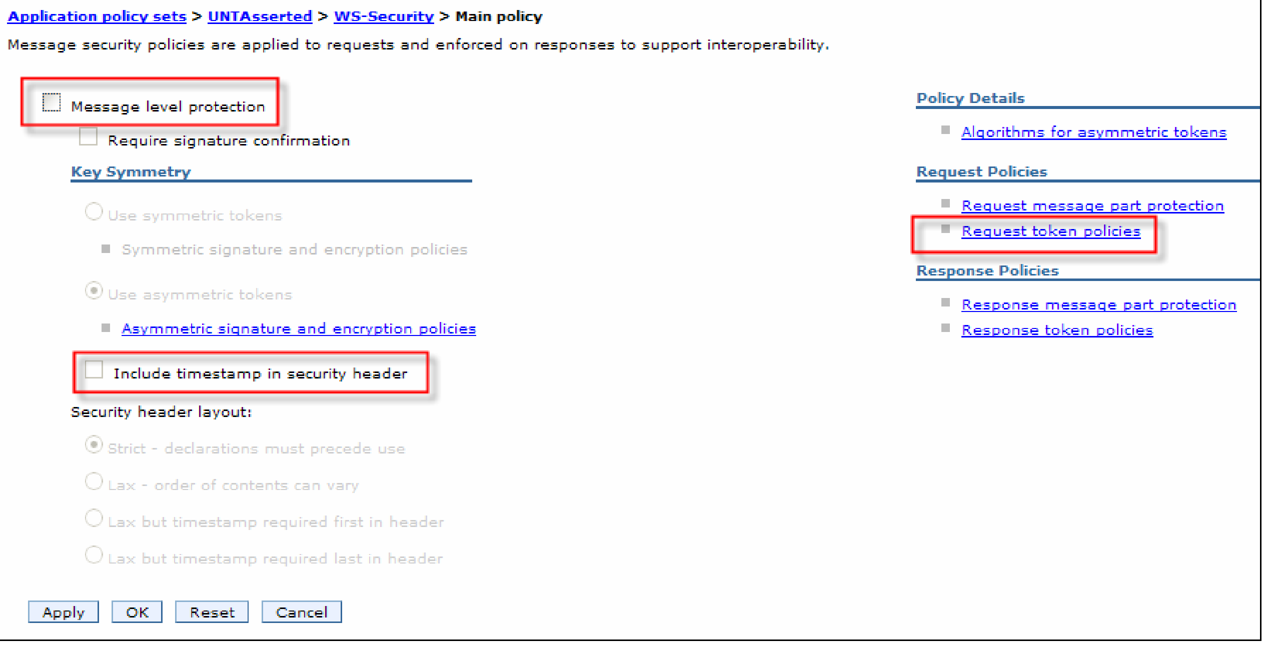


Application policy sets > UNTAsserted > WS-Security

Message security policies are applied to requests and enfc

- Main policy

De-select Include timestamp…
De-select Message Level protection
Click Apply
Click Request Token policies



Application policy sets > UNTAsserted > WS-Security > Main policy

Message security policies are applied to requests and enforced on responses to support interoperability.

☐ Message level protection

☐ Require signature confirmation

Key Symmetry

○ Use symmetric tokens

- Symmetric signature and encryption policies

● Use asymmetric tokens

- Asymmetric signature and encryption policies

☐ Include timestamp in security header

Security header layout:

● Strict - declarations must precede use

○ Lax - order of contents can vary

○ Lax but timestamp required first in header

○ Lax but timestamp required last in header

Policy Details

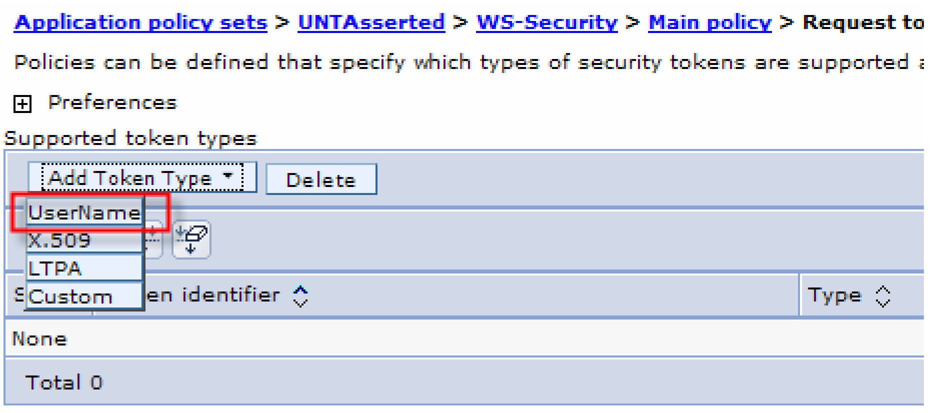
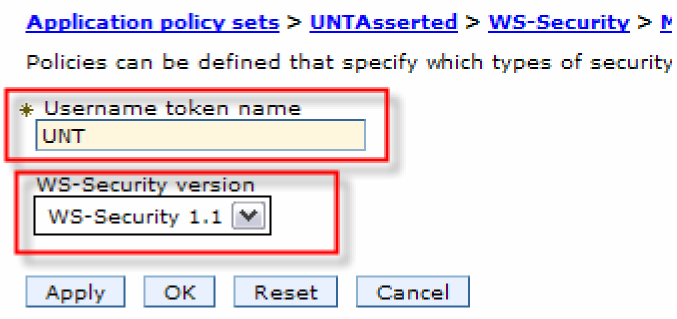
- Algorithms for asymmetric tokens

Request Policies

- Request message part protection
- Request token policies

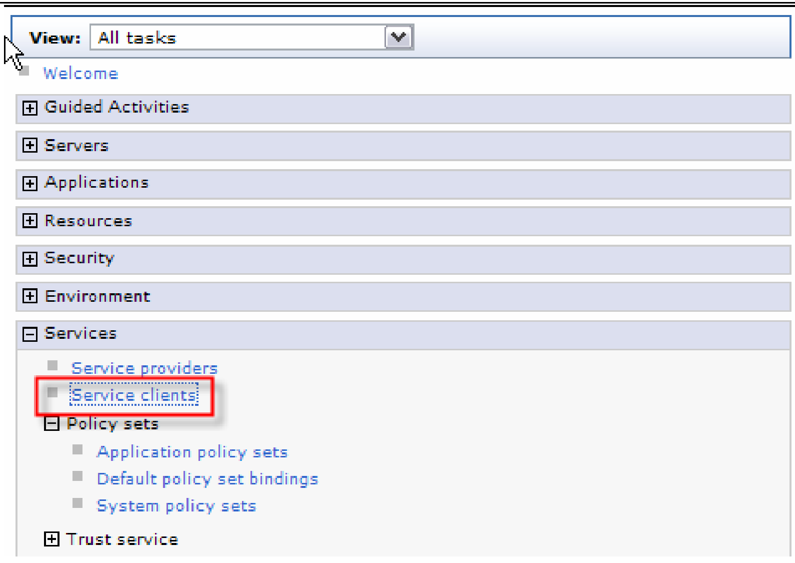
Response Policies

- Response message part protection
- Response token policies

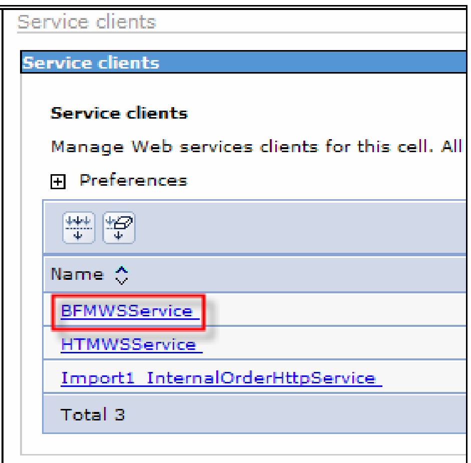
	Click on Add Token Type Click Username  Application policy sets > UNTAsserted > WS-Security > Main policy > Request to Policies can be defined that specify which types of security tokens are supported a Preferences Supported token types Add Token Type Delete UserName X.509 LTPA Custom en identifier Type None Total 0
	Enter a Token name Select as WS-Security version 1.1 Click Apply Save  Application policy sets > UNTAsserted > WS-Security > M Policies can be defined that specify which types of security * Username token name UNT WS-Security version WS-Security 1.1

Bind the policy set to the service client:

	Click on Service clients Click on BFMWSService


View: All tasks

- Welcome
- ⊞ Guided Activities
- ⊞ Servers
- ⊞ Applications
- ⊞ Resources
- ⊞ Security
- ⊞ Environment
- ⊟ Services
  - Service providers
  - Service clients
  - ⊟ Policy sets
    - Application policy sets
    - Default policy set bindings
    - System policy sets
  - ⊞ Trust service


Service clients

Service clients

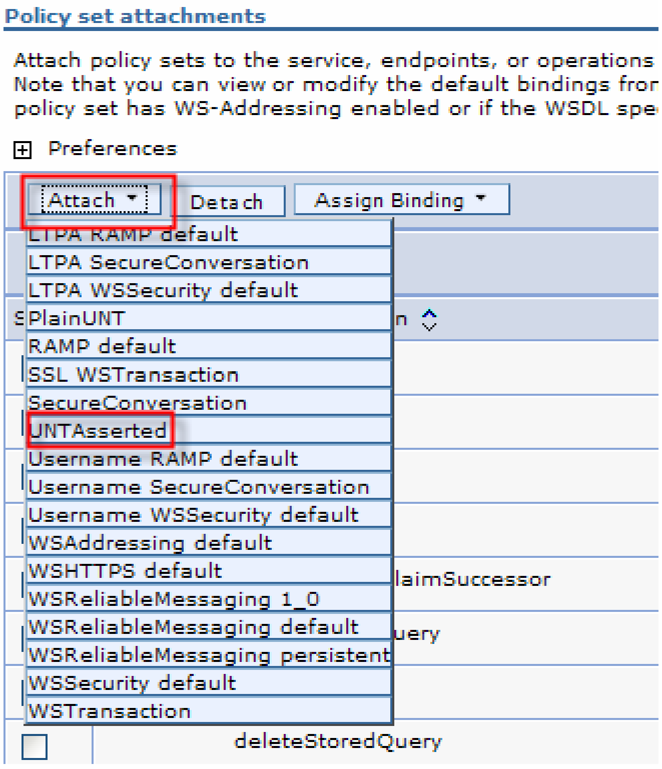
Service clients

Manage Web services clients for this cell. All

⊞ Preferences

Name ⇕
BFMWSService
HTMWSService
Import1_InternalOrderHttpService
Total 3

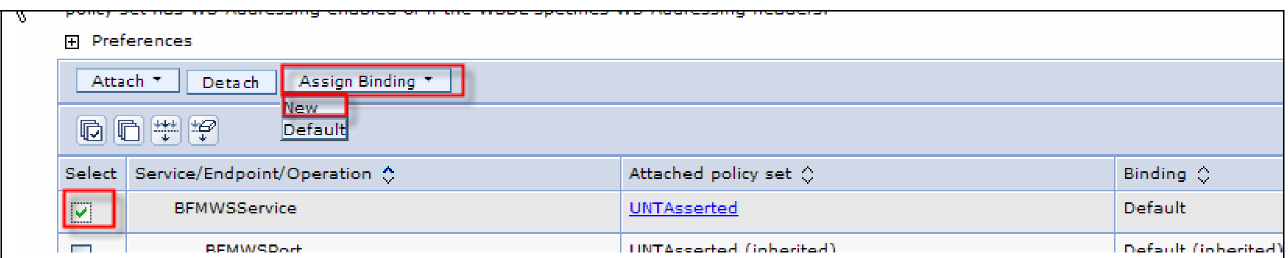
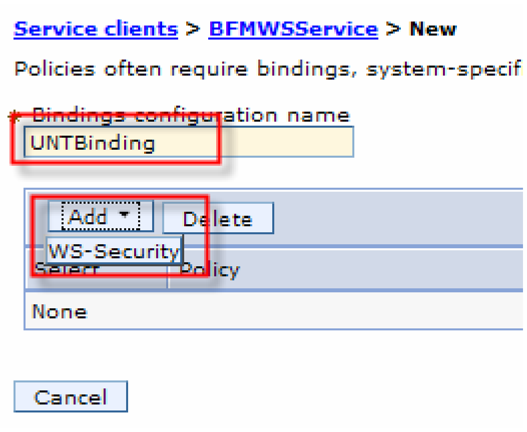
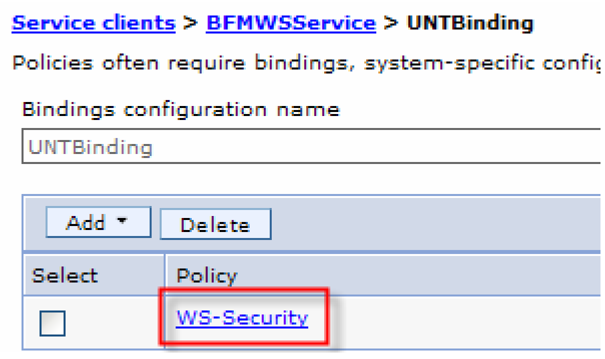
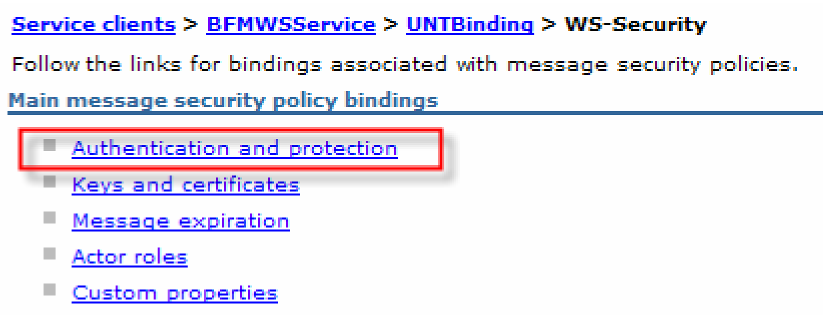
Select BFMService
Click Attach and click UNTAsserted

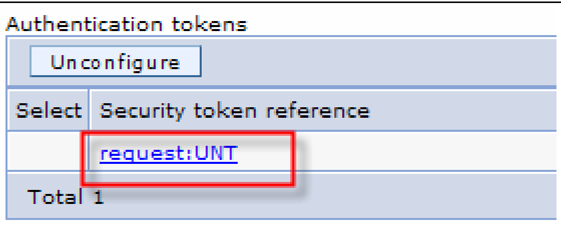
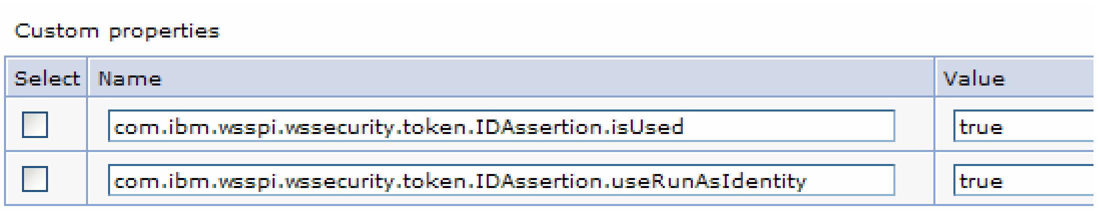

Policy set attachments

Attach policy sets to the service, endpoints, or operations
Note that you can view or modify the default bindings fror
policy set has WS-Addressing enabled or if the WSDL spe

⊞ Preferences

Select BFMService
Click Assign Binding
Click New

	 Preferences Attach Detach Assign Binding New Default Select | Service/Endpoint/Operation | Attached policy set | Binding ✓ | BFMWSService | UNTAsserted | Default | BFMWSPort | UNTAsserted (inherited) | Default (inherited)
	Enter a Name and click WS-Security  Service clients > BFMWSService > New Policies often require bindings, system-specif * Bindings configuration name UNTBinding Add Delete WS-Security Select | Policy None Cancel
	Click on WS-Security  Service clients > BFMWSService > UNTBinding Policies often require bindings, system-specific confi Bindings configuration name UNTBinding Add Delete Select | Policy WS-Security
	Click on Authentication and protection  Service clients > BFMWSService > UNTBinding > WS-Security Follow the links for bindings associated with message security policies. Main message security policy bindings Authentication and protection Keys and certificates Message expiration Actor roles Custom properties
	Click on request:UNT

	 Authentication tokens Unconfigure Select | Security token reference request:UNT Total 1
	Add the custom properties: com.ibm.wsspi.wssecurity.token.IDAssertion.isUsed to sent the username without password com.ibm.wsspi.wssecurity.token.IDAssertion.useRunAsIdentity = true to sent the RunAs identity as Username  Custom properties Select | Name | Value ☐ | com.ibm.wsspi.wssecurity.token.IDAssertion.isUsed | true ☐ | com.ibm.wsspi.wssecurity.token.IDAssertion.useRunAsIdentity | true
	Save

7.1.1.8 Step 5 – WAS: Deploy application

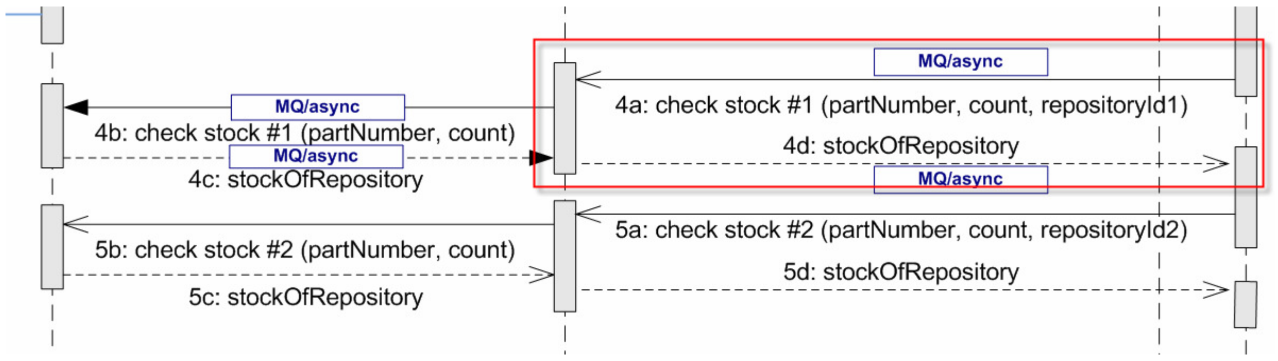
Detailed deployment steps are not described.

7.1.2 Interaction 4a and 4d – Check Stock #1 – WPS to Message Broker

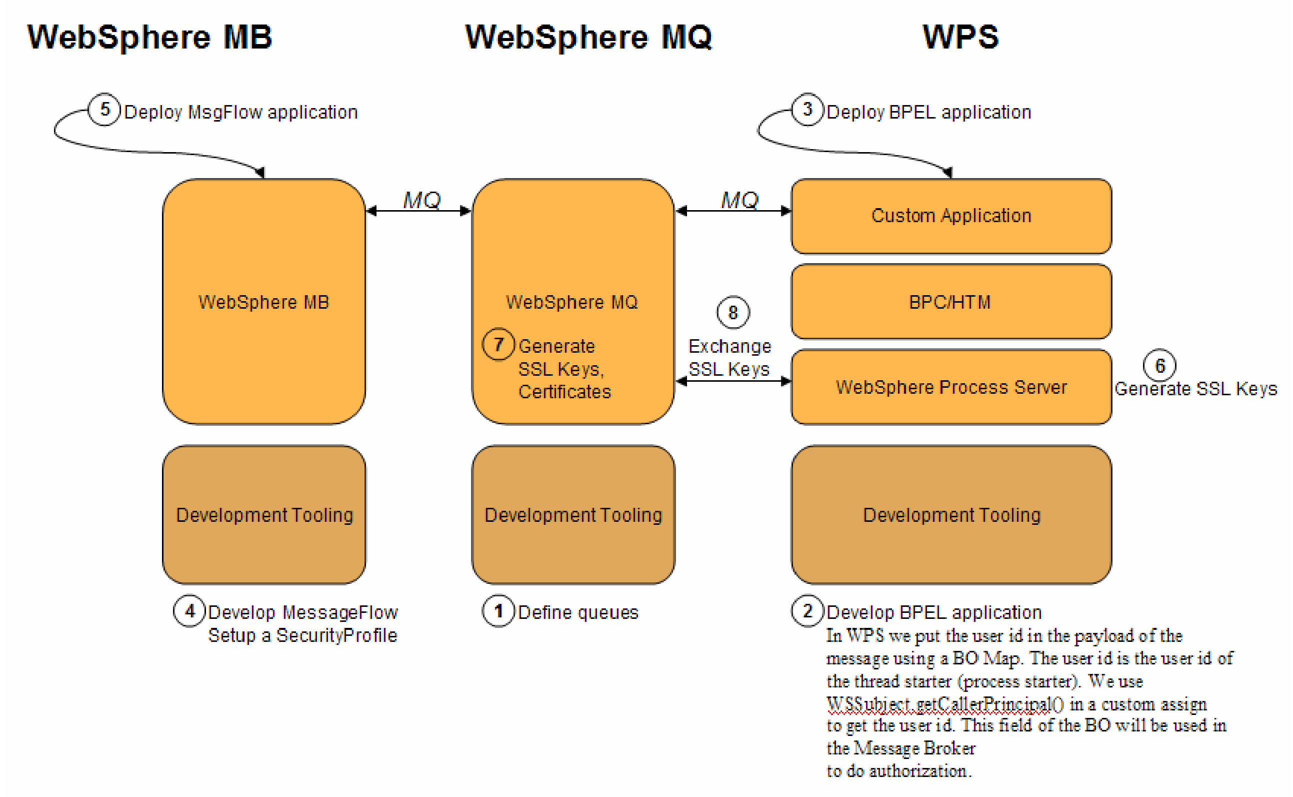
This chapter describes how to propagate a user ID from WPS via MQ to Message Broker.
SSL is used for transport level security.

Client Application	Server Application
ShowcaseApp (WPS, SCA Import - MQ Binding)	CheckStockMQ.mgsflow (WMB)

The figure below shows the relevant part in the sequence diagram:


MQ/async
4a: check stock #1 (partNumber, count, repositoryId1)
4d: stockOfRepository
MQ/async
MQ/async
4b: check stock #1 (partNumber, count)
MQ/async
4c: stockOfRepository
5a: check stock #2 (partNumber, count, repositoryId2)
5d: stockOfRepository
5b: check stock #2 (partNumber, count)
5c: stockOfRepository

The figure below shows the high-level implementation and configuration steps that have to be performed. Find details of the steps in the next sections.



In WPS we put the user id in the payload of the message using a BO Map. The user id is the user id of the thread starter (process starter). We use WSSubject.getCallerPrincipal() in a custom assign to get the user id. This field of the BO will be used in the Message Broker to do authorization.

7.1.2.1 Step 1 – MQ: Define queues

Make sure that following queues are defined in MQ:

STOCK_5_INPUT_J2EE
STOCK_5_INPUT_WPS
STOCK_5_OUTPUT_J2EE
STOCK_5_OUTPUT_WPS

7.1.2.2 Step 2 – WPS: Develop the BPEL application

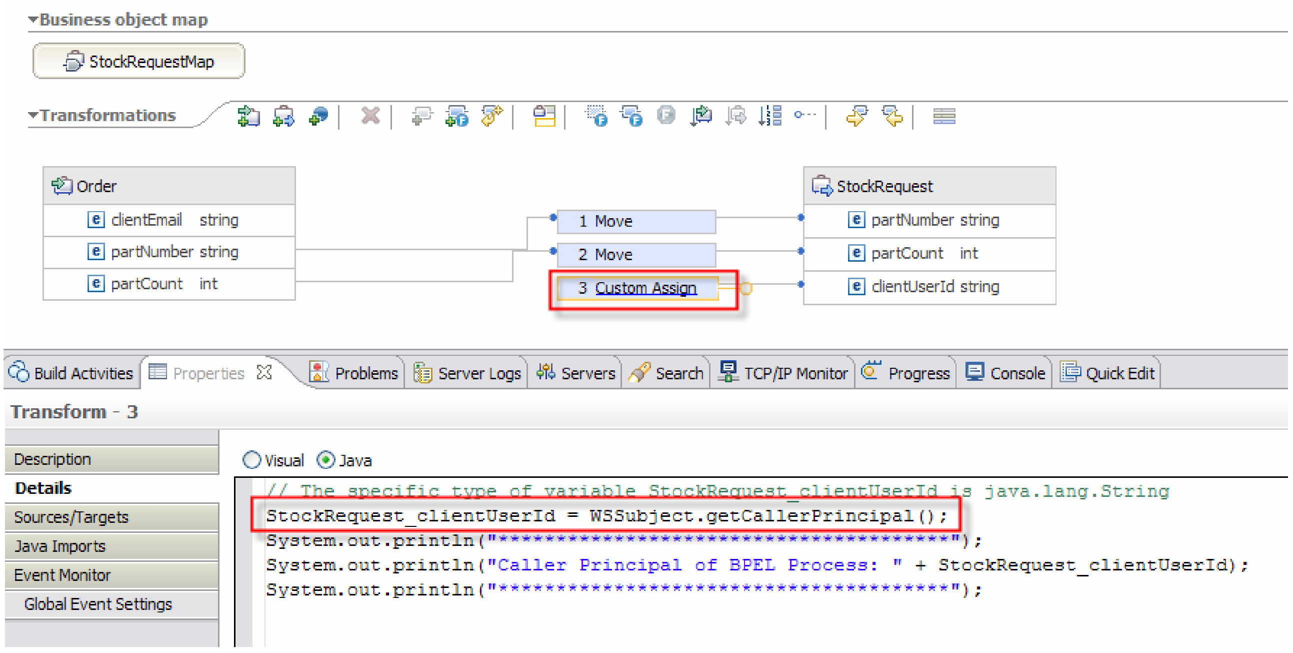
7.1.2.2.1 Propagate the user ID from WPS to Message Broker

There are three options to propagate the user ID from WPS to Message Broker.

1. **Only Option 1 is implemented in the showcase.**
2. Option 2 is feasible and documented, but not implemented in the showcase.
3. Option 3 is not feasible and therefore not implemented in the showcase.

7.1.2.2.1.1 Option 1 - User propagation via payload (This is the implemented option in the showcase)

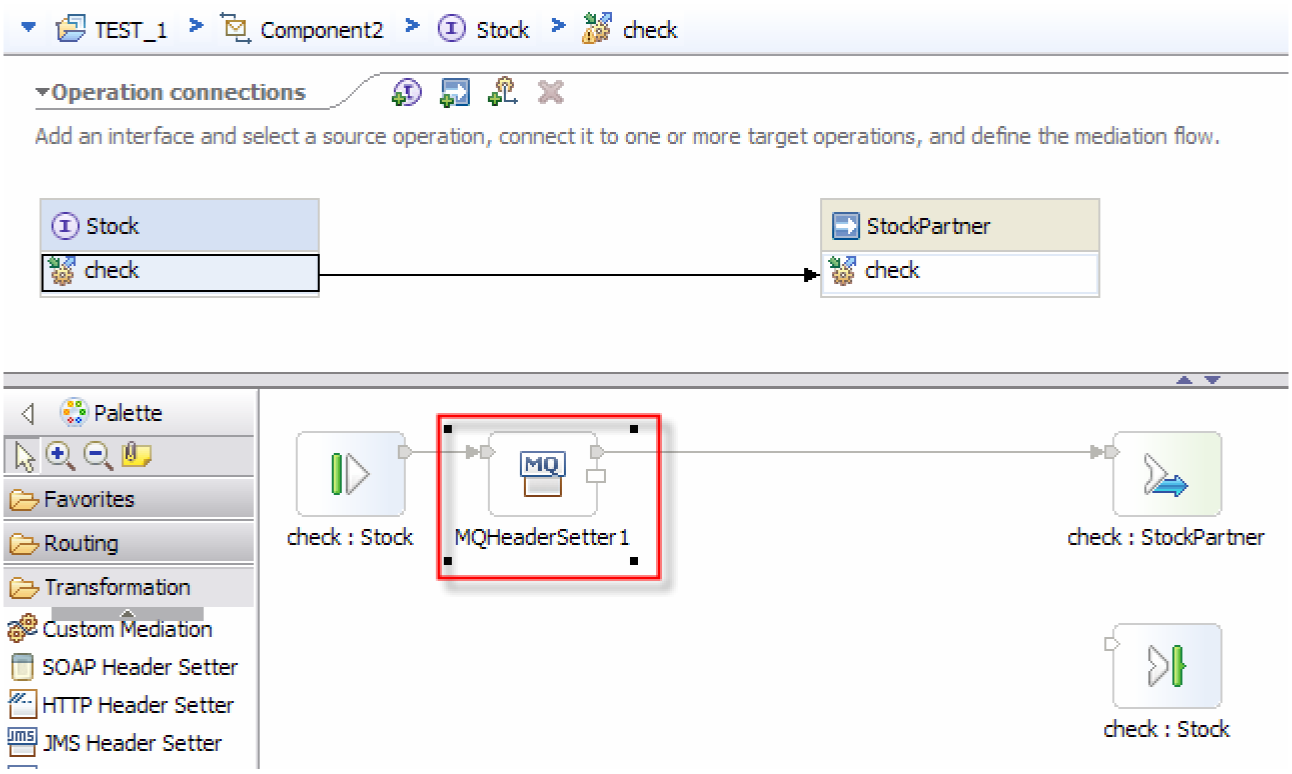
In WPS we put the user ID in the payload of the message using a BO Map.
The user ID is the user ID of the thread starter (process starter). We use `WSSubject.getCallerPrincipal()` in a custom assign to get the user ID under which the thread runs.
This field of the BO will be used in the Message Broker to do authorization.

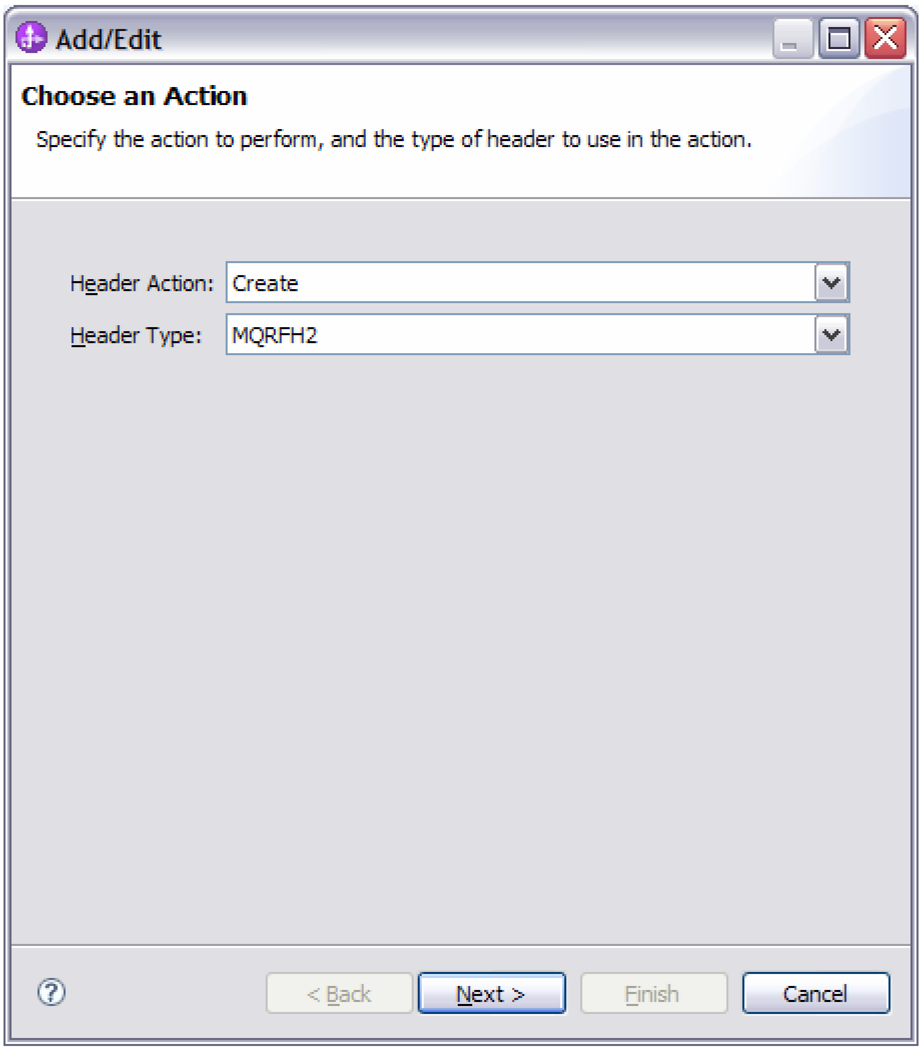


```
// The specific type of variable StockRequest_clientUserId is java.lang.String
StockRequest_clientUserId = WSSubject.getCallerPrincipal();
System.out.println("***************************************");
System.out.println("Caller Principal of BPEL Process: " + StockRequest_clientUserId);
System.out.println("***************************************");
```

7.1.2.2.1.2 Option 2 - User propagation via Custom MQ Header (not implemented)

In WPS MQ Headers can be produced and modified using mediation components. A username custom header (e.g MQRFH2) can be passed via MQ to the Message Broker.
To create an MQRFH2 in WPS you have to create a mediation module. Within the mediation module an MQHeaderSetter node sets the MQRFH2:




Add/Edit

Choose an Action

Specify the action to perform, and the type of header to use in the action.

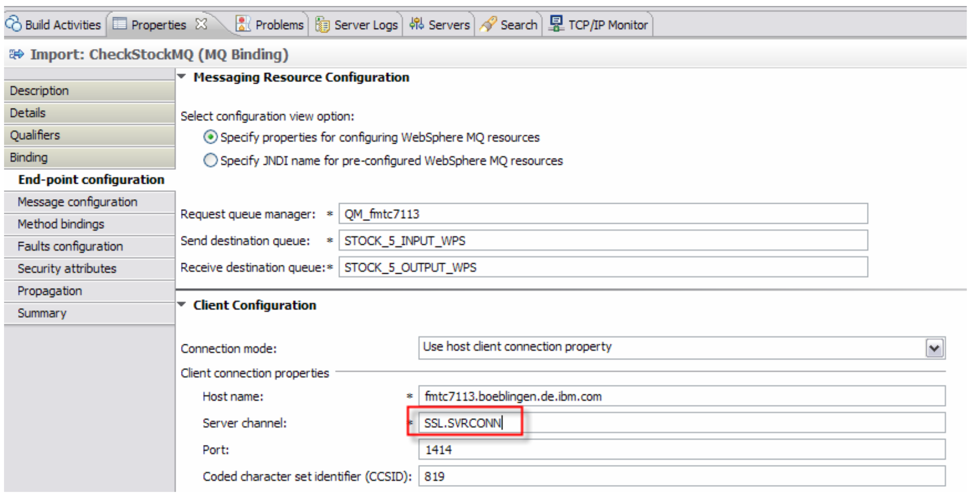
Header Action:	Create
Header Type:	MQRFH2

On the Broker site, the message flow can access and extract the user ID from the MQRFH2 Header field.

7.1.2.2.2 Define the ConnectionFactory and queue objects in WID

After the SSL configuration is done (which is described here) configure the MQ import in the SCA Assembly Diagram to match the SSL settings.

1. Open the Assembly Diagram in WID and click the MQ Import *CheckStockMQ*
   At *Properties > End-point configuration* make sure you use as Server channel the SSL server connection channel.



Import: CheckStockMQ (MQ Binding)

Messaging Resource Configuration

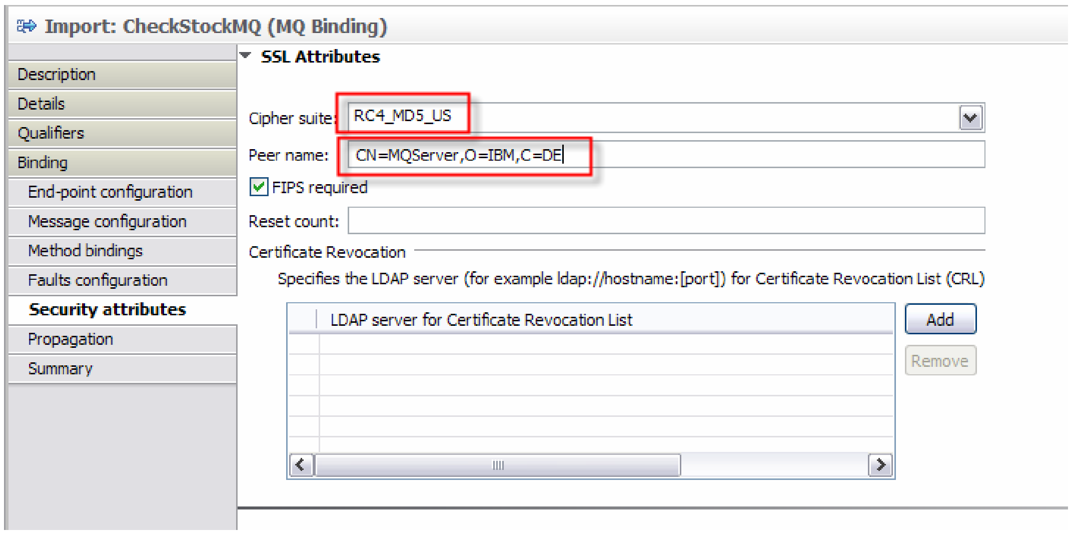
Select configuration view option:
(●) Specify properties for configuring WebSphere MQ resources
( ) Specify JNDI name for pre-configured WebSphere MQ resources

Request queue manager: * QM_fmtc7113
Send destination queue: * STOCK_5_INPUT_WPS
Receive destination queue: * STOCK_5_OUTPUT_WPS

Client Configuration

Connection mode: Use host client connection property
Client connection properties
Host name: * fmtc7113.boeblingen.de.ibm.com
Server channel: * SSL.SVRCONN
Port: 1414
Coded character set identifier (CCSID): 819

2. At *Security attributes* select as *Cipher Suite* the one you have selected during the MQ SSL configuration.
   Enter as *Peer name* the DN which you have defined during the creation of the certificates.



Import: CheckStockMQ (MQ Binding)

SSL Attributes

Cipher suite: RC4_MD5_US
Peer name: CN=MQServer,O=IBM,C=DE
[✓] FIPS required
Reset count:
Certificate Revocation
Specifies the LDAP server (for example ldap://hostname:[port]) for Certificate Revocation List (CRL)
LDAP server for Certificate Revocation List

7.1.2.3 Step 3 – WPS: Deploy the BPEL application

Detailed deployment steps are not described.

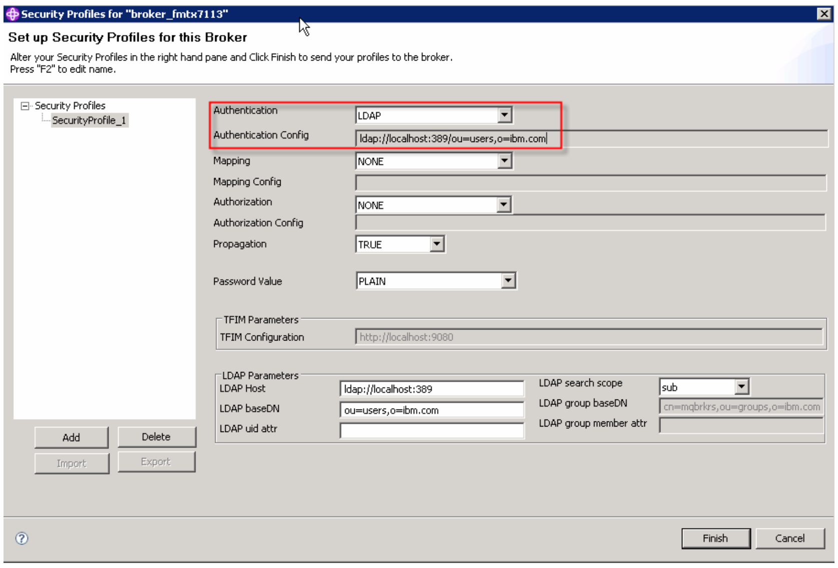
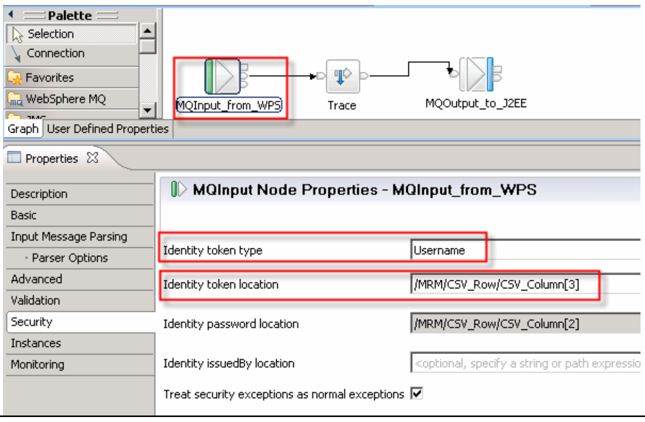
7.1.2.4 Step 4 – WMB: Develop the message flow and set up a security profile

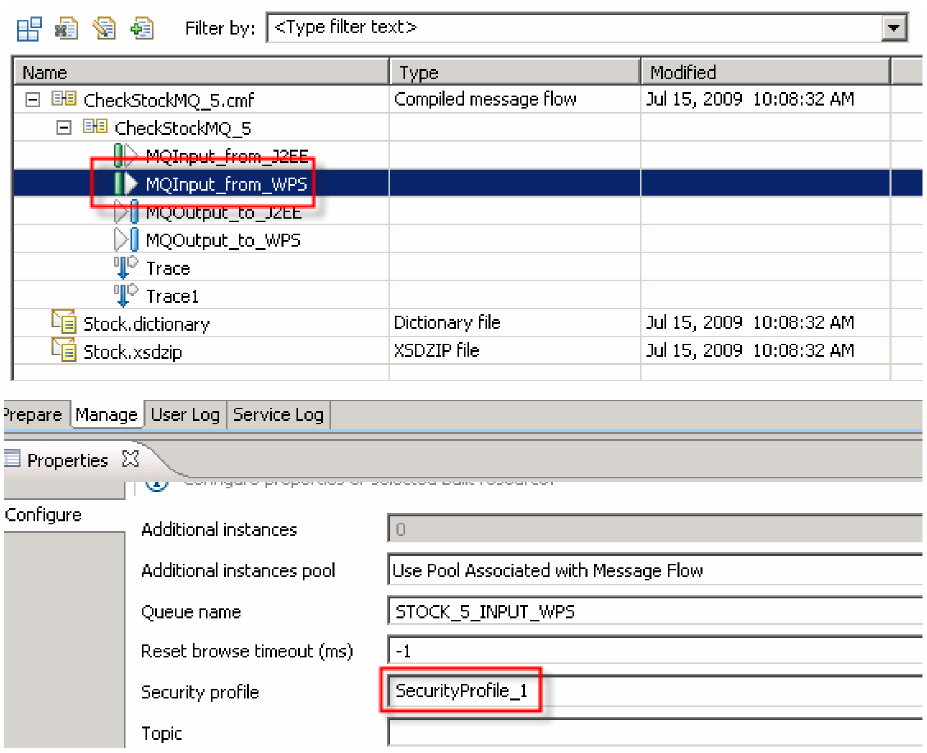
Detailed implementation steps, which are not security relevant, are not described. Refer to the WMB Toolkit artefacts to see how the message flow application is developed.

If the User Id is provided with the input message, HTTPInput, SOAPInput, or MQInput nodes can be examined for an identity field. The identity is used as is, or can be mapped to an alternate identity. This identity is used to ensure that the client is authorized to access the message flow.
Authentication and authorization are performed using an LDAP. The type of security actions to be taken (authentication, authorization, and mapping) and the external provider to use are controlled by security profiles defined for the broker.

Reference Material:
Using the New Features in WebSphere Message Broker V6.1
http://www.redbooks.ibm.com/abstracts/redp4458.html

In the showcase we can do authentication on the message flow:

1	The input message (csv format) contains the user Id in column 3 of the message
2	A *Security Profile* on the broker must exist and *Authentication* must be set to *LDAP*  Security Profiles for "broker_fmtx7113" Set up Security Profiles for this Broker Alter your Security Profiles in the right hand pane and Click Finish to send your profiles to the broker. Press "F2" to edit name. Security Profiles SecurityProfile_1 Authentication: LDAP Authentication Config: ldap://localhost:389/ou=users,o=ibm.com Mapping: NONE Mapping Config: Authorization: NONE Authorization Config: Propagation: TRUE Password Value: PLAIN TFIM Parameters TFIM Configuration: http://localhost:9080 LDAP Parameters LDAP Host: ldap://localhost:389 LDAP baseDN: ou=users,o=ibm.com LDAP uid attr: LDAP search scope: sub LDAP group baseDN: cn=mqbrkrs,ou=groups,o=ibm.com LDAP group member attr:
3	Specify in the MQInput Node: identity token type = *Username* identity token location = path to the user ID field  Palette Selection Connection Favorites WebSphere MQ MQInput_from_WPS Trace MQOutput_to_J2EE Graph User Defined Properties Properties Description Basic Input Message Parsing Parser Options Advanced Validation Security Instances Monitoring MQInput Node Properties - MQInput_from_WPS Identity token type: Username Identity token location: /MRM/CSV_Row/CSV_Column[3] Identity password location: /MRM/CSV_Row/CSV_Column[2] Identity issuedBy location: <optional, specify a string or path expressio Treat security exceptions as normal exceptions ✓

4	The Security Profile must be added to the MQ input node of the message flow in the Broker archive. 

Filter by: <Type filter text>

Name	Type	Modified
CheckStockMQ_5.cmf	Compiled message flow	Jul 15, 2009 10:08:32 AM
CheckStockMQ_5		
MQInput_from_J2EE		
MQInput_from_WPS		
MQOutput_to_J2EE		
MQOutput_to_WPS		
Trace		
Trace1		
Stock.dictionary	Dictionary file	Jul 15, 2009 10:08:32 AM
Stock.xsdzip	XSDZIP file	Jul 15, 2009 10:08:32 AM

Prepare | Manage | User Log | Service Log

Properties

Configure

Additional instances	0
Additional instances pool	Use Pool Associated with Message Flow
Queue name	STOCK_5_INPUT_WPS
Reset browse timeout (ms)	-1
Security profile	SecurityProfile_1
Topic	

7.1.2.5 Step 5 – WMB: Deploy the message flow application

Detailed deployment steps are not described.

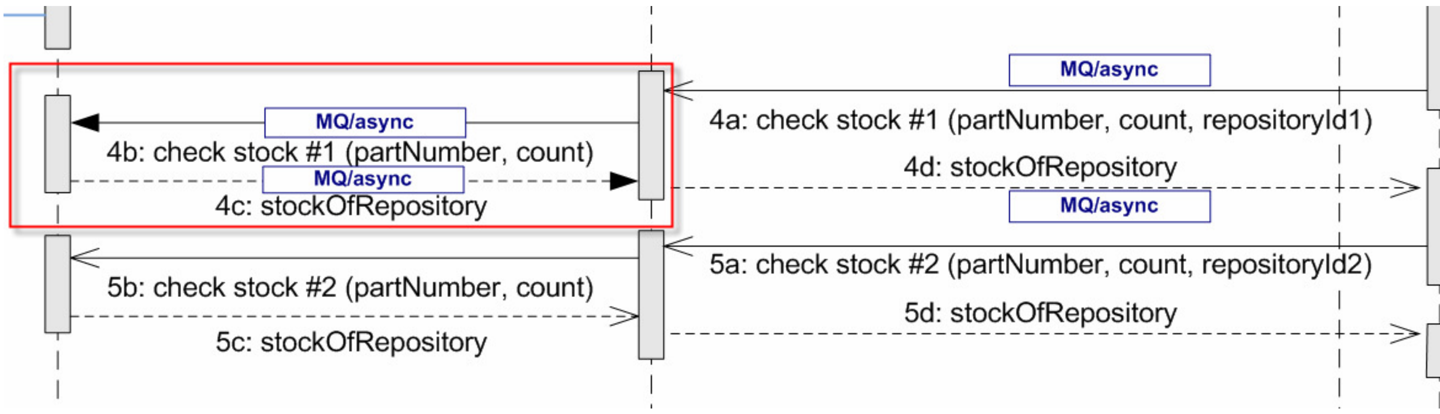
7.1.2.6 Step 6 – SSL configuration between WPS and MQ

Refer to chapter "SSL between WPS and MQ"

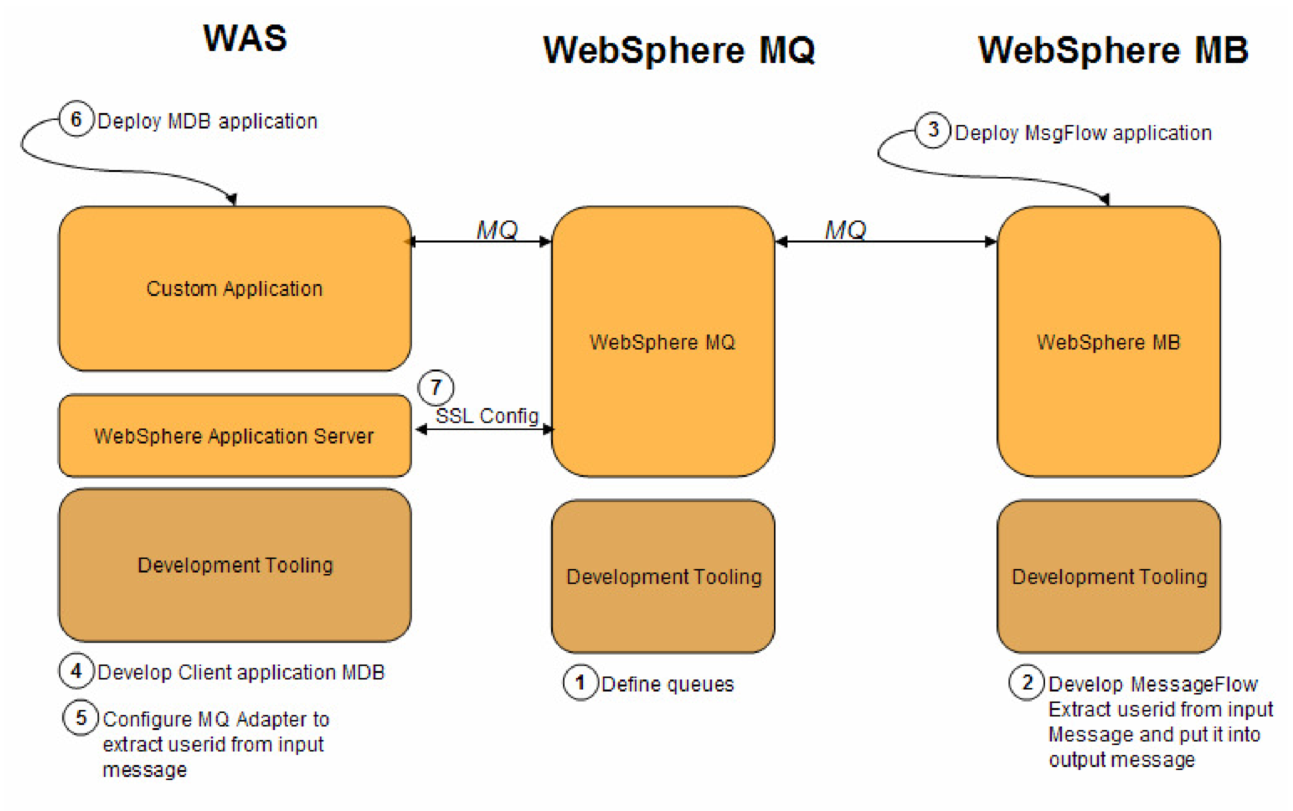
7.1.3 Interaction 4b and 4c – Check Stock #1 – Message Broker to WAS

This scenario describes how to propagate a user ID from WMB via MQ to WAS.
SSL is used for transport layer security.

The figure below shows the relevant part in the sequence diagram:


MQ/async
4a: check stock #1 (partNumber, count, repositoryId1)
4d: stockOfRepository
MQ/async
4b: check stock #1 (partNumber, count)
MQ/async
4c: stockOfRepository
MQ/async
5a: check stock #2 (partNumber, count, repositoryId2)
5d: stockOfRepository
5b: check stock #2 (partNumber, count)
5c: stockOfRepository

The figure below shows the high-level implementation and configuration steps that have to be performed. Find details of the steps in the next sections.


WAS
WebSphere MQ
WebSphere MB
6 Deploy MDB application
3 Deploy MsgFlow application
MQ
MQ
Custom Application
WebSphere MQ
WebSphere MB
7 SSL Config
WebSphere Application Server
Development Tooling
Development Tooling
Development Tooling
4 Develop Client application MDB
5 Configure MQ Adapter to extract userid from input message
1 Define queues
2 Develop MessageFlow Extract userid from input Message and put it into output message

7.1.3.1 Step 1 – WMQ: Define queues

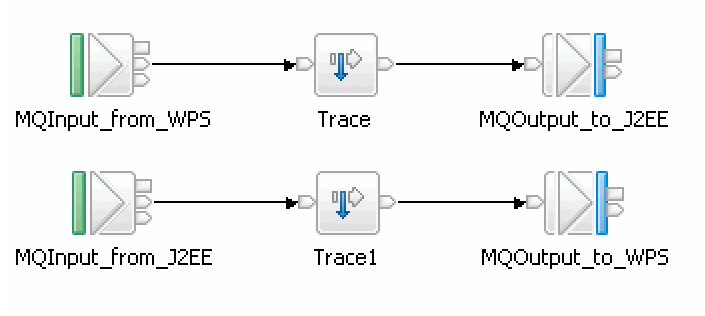
Make sure that the following queues exist on MQ:

```
STOCK_5_INPUT_J2EE
```

STOCK_5_INPUT_WPS
STOCK_5_OUTPUT_J2EE
STOCK_5_OUTPUT_WPS

7.1.3.2 Step 2 – WMB: Develop the message flow

Detailed implementation steps, which are not security relevant, are not described. Refer to the WMB artefacts to see how the application is developed.
The message flow is just a pass-through flow.



7.1.3.3 Step 3 – WMB: Deploy the message flow

Detailed deployment steps are not described.

7.1.3.4 Step 4 – WAS: Develop the WAS application

Detailed implementation steps, which are not security relevant, are not described. Refer to the WID artefacts to see how the application is developed.

7.1.3.5 Step 5 – WAS: Configure the MQ Adapter WAS application

The term “MQ Adapter” means here that we switch the user context of the thread under which the Java MDB runs. To do so, the following must be implemented:

1. Parse the MQ input message
2. Get the user ID from the payload
3. Switch the user context

Parse the MQ input message

```
BytesMessage bytesMsg = (BytesMessage) msg;
byte[] payloadba = new byte[(int) bytesMsg.getBodyLength()];
int datalen = bytesMsg.readBytes(payloadba);
if (datalen != bytesMsg.getBodyLength()) {
    System.out.println("BodyLength = " + bytesMsg.getBodyLength()
            + ", but returned data lengt = " + datalen);
    return;
}
String payload = new String(payloadba);

// handle payload
String reply = handlePayload(payload);
```

Get the user ID from the payload

```
            String[] results = payload.split(","); // number,count,user
            partNumber = results[0];
            partCount = results[1];
            userId = results[2];
```

Switch the User Context

```
        AuthenticationHandler result = null;
        result = new AuthenticationHandler();
        realm = "defaultWIMFileBasedRealm";
        try {

            result.setSubject(com.ibm.ws.security.core.ContextManagerFactory
                    .getInstance().login(realm, userId));

            WSSubject.setRunAsSubject(result.getSubject());

        } catch (WSLoginFailedException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        } catch (WSSecurityException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

7.1.3.6 Step 6 – WAS: Deploy the WAS application

Detailed deployment steps are not described.

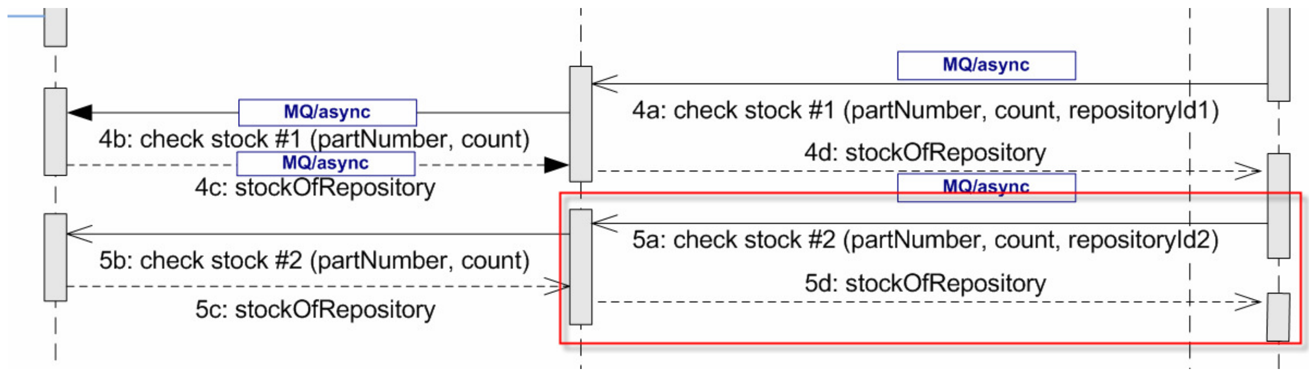
7.1.3.7 Step 7 – SSL Configuration

Refer to the chapter SSL Configuration.

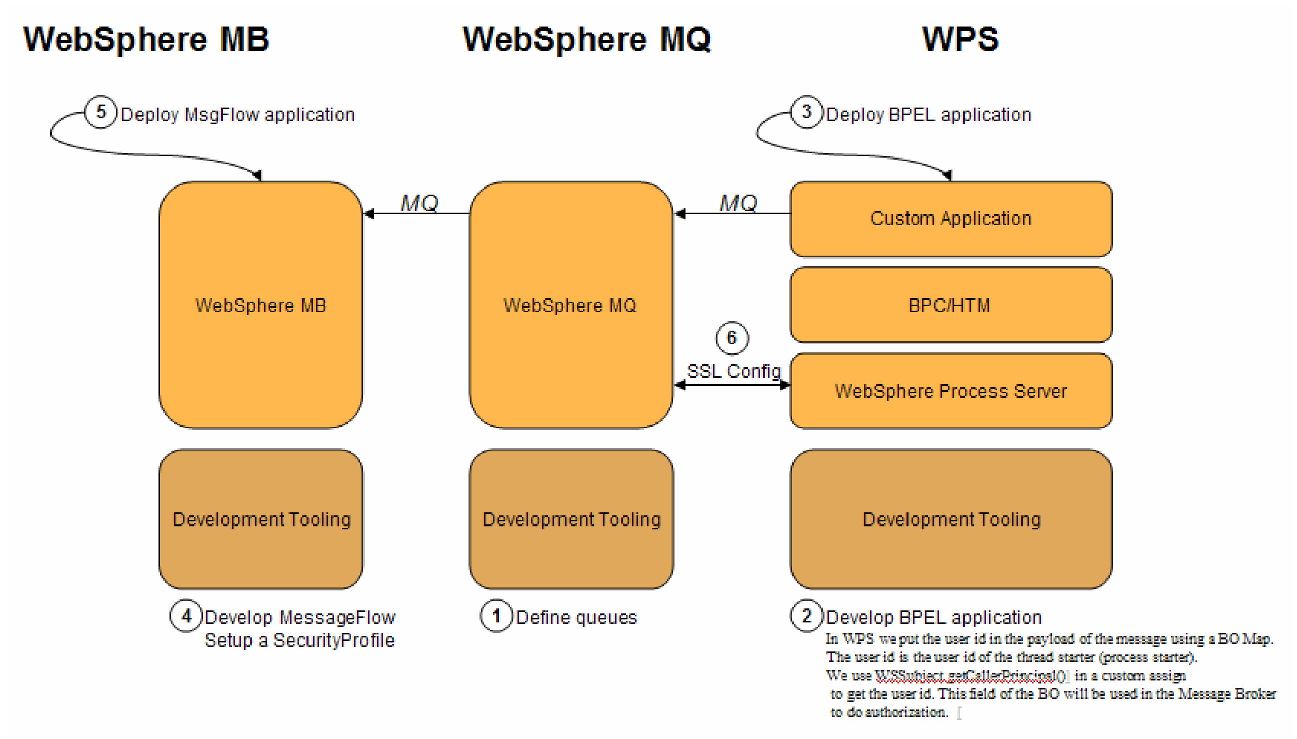
7.1.4 Interaction 5a and 5d – Check Stock #2 – WPS to Message Broker

Client Application	Server Application
ShowcaseApp (WPS, SCA Import - MQ Binding)	CheckStockMQ.mgsflow (WMB)

The figure below shows the relevant part in the sequence diagram:


MQ/async
4a: check stock #1 (partNumber, count, repositoryId1)
4d: stockOfRepository
MQ/async
4b: check stock #1 (partNumber, count)
MQ/async
4c: stockOfRepository
MQ/async
5a: check stock #2 (partNumber, count, repositoryId2)
5d: stockOfRepository
5b: check stock #2 (partNumber, count)
5c: stockOfRepository

The figure below shows the high-level implementation and configuration steps that have to be performed. Find details of the steps in the next sections.



In WPS we put the user id in the payload of the message using a BO Map.
The user id is the user id of the thread starter (process starter).
We use WSSubject.getCallerPrincipal() in a custom assign
to get the user id. This field of the BO will be used in the Message Broker
to do authorization.

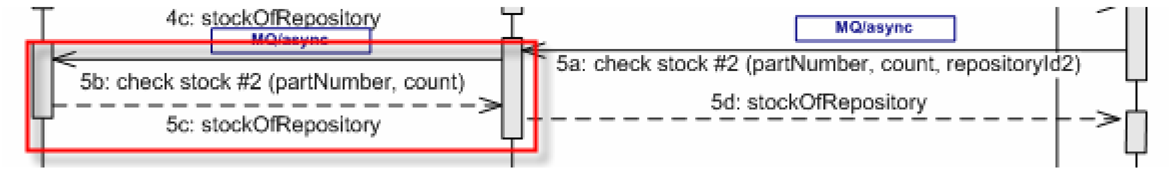
Refer to the *Chapter Interaction 4a and 4d – Check Stock #1 – WPS to Message Broker*

7.1.5 Interaction 5b and 5c – Check Stock #2 – Message Broker to WAS

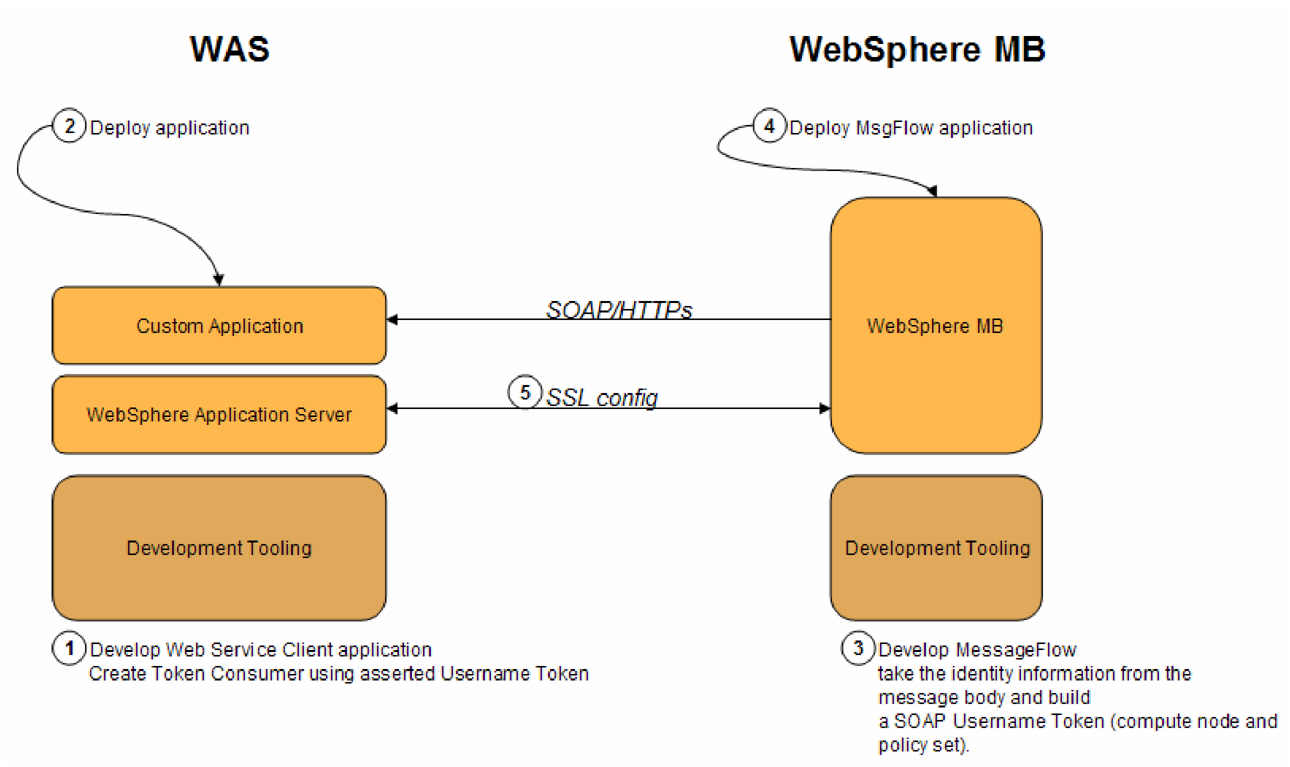
This chapter describes identity propagation with identity assertion from WBM to WAS via SOAP/HTTP.

Client Application	Server Application
CheckStockMQ.mgsflow (WMB)	CheckStock2EAR (WAS)

The figure below shows the relevant part in the sequence diagram:


4c: stockOfRepository
MQ/async
5b: check stock #2 (partNumber, count)
5c: stockOfRepository
MQ/async
5a: check stock #2 (partNumber, count, repositoryId2)
5d: stockOfRepository

The figure below shows the high-level implementation and configuration steps that have to be performed. Find details of the steps in the next sections.

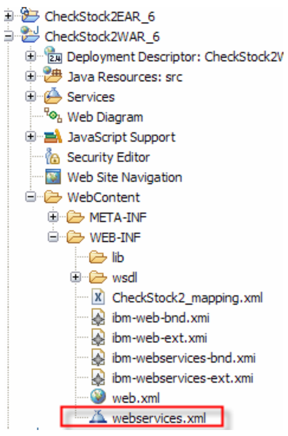
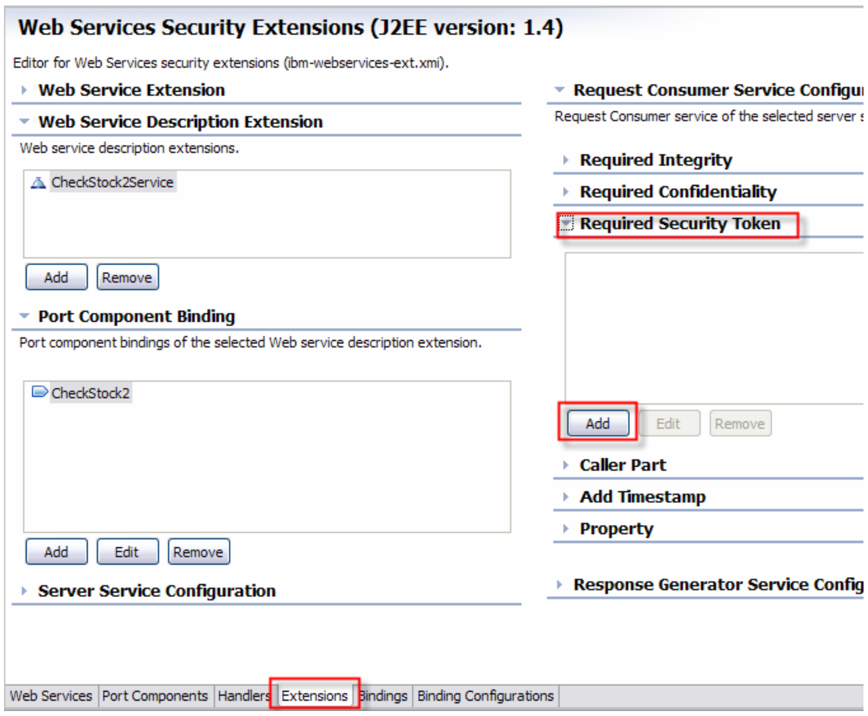

WAS
WebSphere MB
2 Deploy application
4 Deploy MsgFlow application
Custom Application
SOAP/HTTPs
WebSphere MB
WebSphere Application Server
5 SSL config
Development Tooling
Development Tooling
1 Develop Web Service Client application
Create Token Consumer using asserted Username Token
3 Develop MessageFlow
take the identity information from the message body and build
a SOAP Username Token (compute node and policy set).

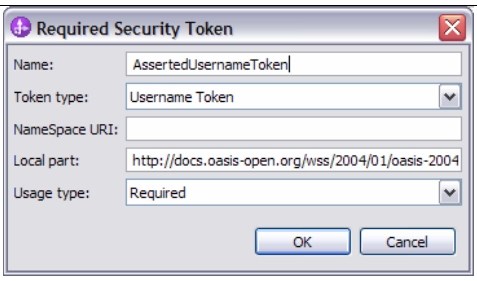
In this scenario a trusted user vouches for the end-user. WebSphere Application Server provides functionality that you can use to configure identity assertion and there are different ways in which it can be configured. This chapter documents one such way to achieve identity assertion by using a combination of transport-level basic authentication and message level Username token where:

- Transport-level basic authentication will be used to carry the credential of the trusted caller and
- Username Token will be used to carry the identity of the asserted user.

7.1.5.1 Step1 – WAS: Develop Web Service provider application and create token consumer

WAS, the service provider, expects from the Web Service consumer an asserted Username Token. Therefore, we have to configure the deployment descriptor of WAS accordingly.

1.	To create a *Request Consumer Security Token* open the webservice.xml and go to the tab *Extension*  CheckStock2EAR_6 CheckStock2WAR_6 Deployment Descriptor: CheckStock2V Java Resources: src Services Web Diagram JavaScript Support Security Editor Web Site Navigation WebContent META-INF WEB-INF lib wsdl CheckStock2_mapping.xml ibm-web-bnd.xmi ibm-web-ext.xmi ibm-webservices-bnd.xmi ibm-webservices-ext.xmi web.xml webservices.xml
2.	Open *Request Consumer Service Configuration Details > Required Security Token* and Click *Add*  Web Services Security Extensions (J2EE version: 1.4) Editor for Web Services security extensions (ibm-webservices-ext.xmi). Web Service Extension Web Service Description Extension Web service description extensions. CheckStock2Service Add Remove Port Component Binding Port component bindings of the selected Web service description extension. CheckStock2 Add Edit Remove Server Service Configuration Request Consumer Service Configu Request Consumer service of the selected server s Required Integrity Required Confidentiality Required Security Token Add Edit Remove Caller Part Add Timestamp Property Response Generator Service Config Web Services Port Components Handlers Extensions Bindings Binding Configurations
3.	Name the token for example *AssertedUsernameToken* Select as Token type *Username Token* Local Part is set automatically when choosing *Username Token* Usage type is *Required* Click *OK*


Required Security Token

Name: AssertedUsernameToken

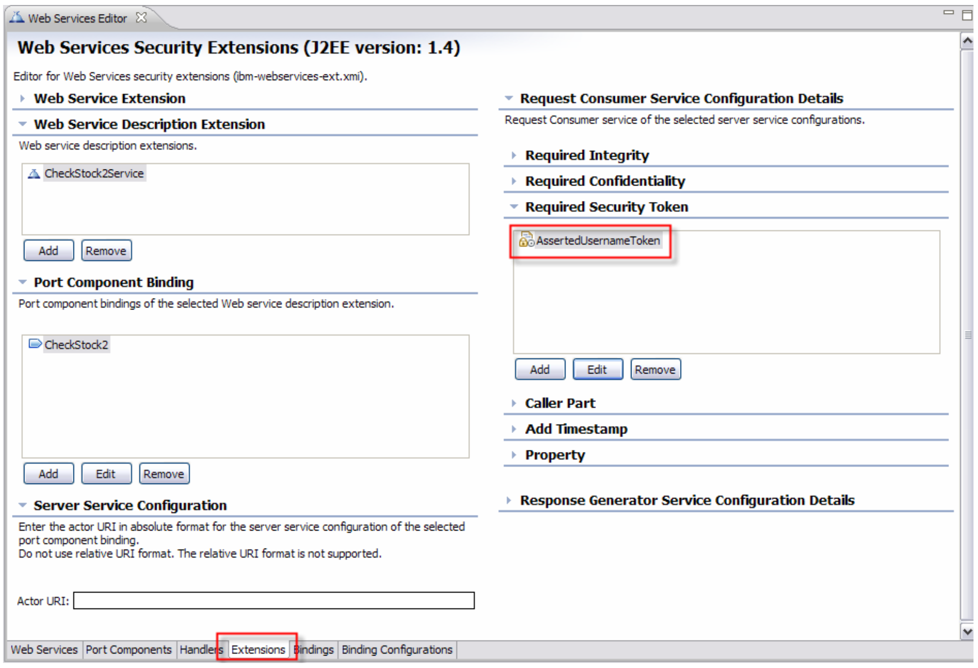
Token type: Username Token

NameSpace URI:

Local part: http://docs.oasis-open.org/wss/2004/01/oasis-2004

Usage type: Required

4. The token is now available in the *Required Security Token* section


Web Services Editor

Web Services Security Extensions (J2EE version: 1.4)

Editor for Web Services security extensions (ibm-webservices-ext.xmi).

Web Service Extension

Web Service Description Extension

Web service description extensions.

CheckStock2Service

Port Component Binding

Port component bindings of the selected Web service description extension.

CheckStock2

Server Service Configuration

Enter the actor URI in absolute format for the server service configuration of the selected port component binding.
Do not use relative URI format. The relative URI format is not supported.

Actor URI:

Request Consumer Service Configuration Details

Request Consumer service of the selected server service configurations.

Required Integrity

Required Confidentiality

Required Security Token

AssertedUsernameToken

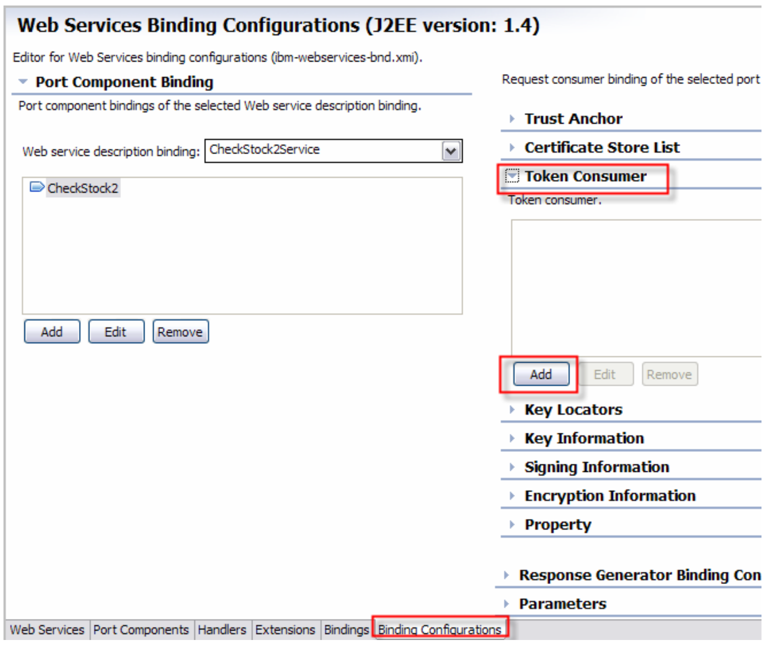
Caller Part

Add Timestamp

Property

Response Generator Service Configuration Details

5. Open the *Binding Configurations Tab*
Open *Request Consumer Binding Configuration Details > Token Consumer*
Click *Add*


Web Services Binding Configurations (J2EE version: 1.4)

Editor for Web Services binding configurations (ibm-webservices-bnd.xmi).

Port Component Binding

Port component bindings of the selected Web service description binding.

Web service description binding: CheckStock2Service

CheckStock2

Request consumer binding of the selected port

Trust Anchor

Certificate Store List

Token Consumer

Token consumer.

Key Locators

Key Information

Signing Information

Encryption Information

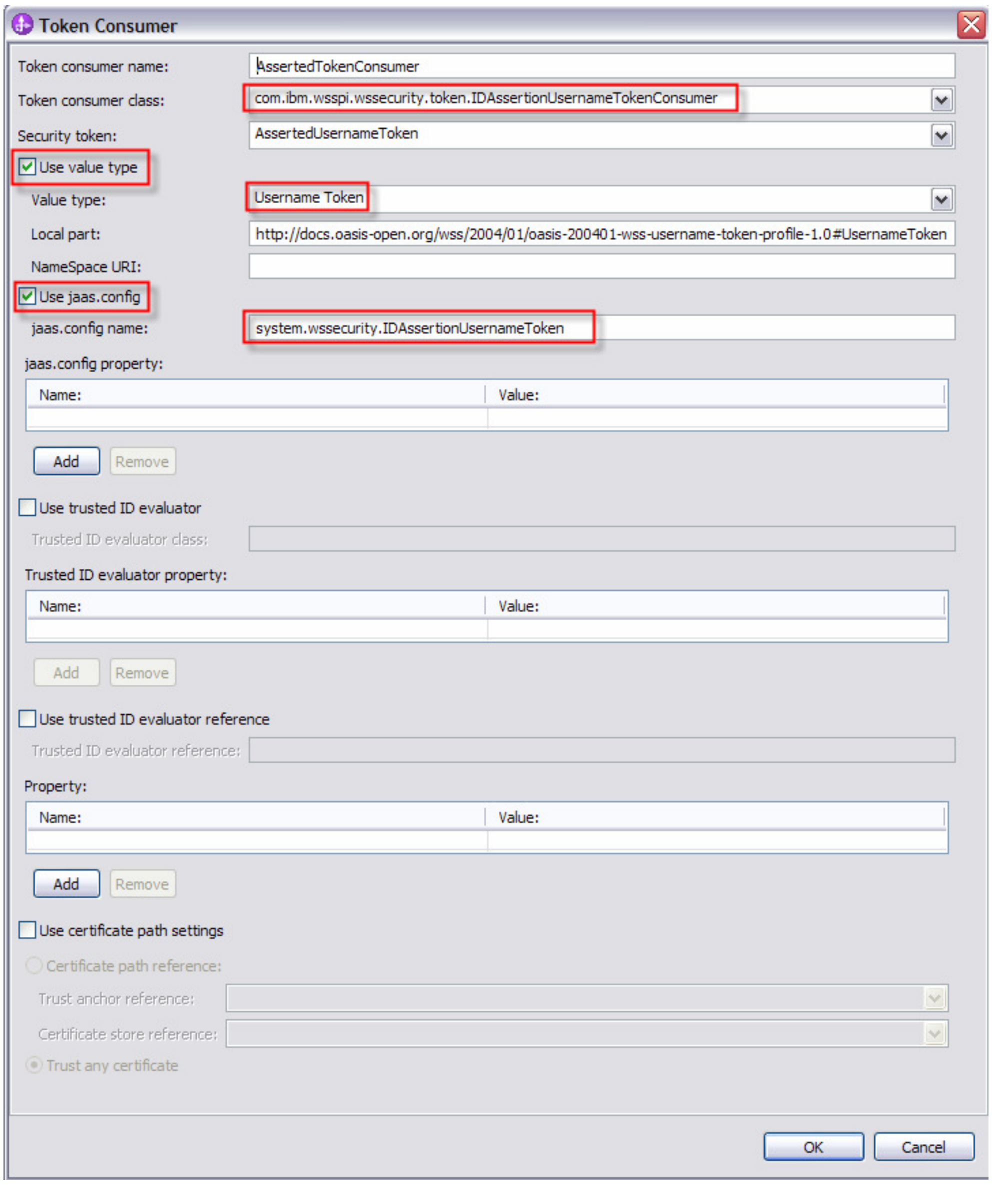
Property

Response Generator Binding Con

Parameters

6.

- In the Token Consumer dialog box enter a consumer name, e.g AssertedTokenConsumer
- Select as Token consumer class
  com.ibm.wsspi.wssecurity.token.IDAssertionUsernameTokenConsumer
- As Security Token select *AssertedUsernameToken*
- Check *Use value type*
- Select as Value type: *Username Token*
- Local Part is generated automatically
- Check *Use.jaas.config*
- Enter as jaas.config name: *system.wssecurity.IDAssertionUsernameToken*
  by selecting the *IDAssertionUsernameToken* we define that we just need the user ID, and no password
- Click *OK*


Token Consumer

Token consumer name: AssertedTokenConsumer
Token consumer class: com.ibm.wsspi.wssecurity.token.IDAssertionUsernameTokenConsumer
Security token: AssertedUsernameToken
[✓] Use value type
Value type: Username Token
Local part: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken
NameSpace URI:
[✓] Use jaas.config
jaas.config name: system.wssecurity.IDAssertionUsernameToken
jaas.config property:

Name:	Value:

[ ] Use trusted ID evaluator
Trusted ID evaluator class:
Trusted ID evaluator property:

Name:	Value:

[ ] Use trusted ID evaluator reference
Trusted ID evaluator reference:
Property:

Name:	Value:

[ ] Use certificate path settings
Certificate path reference:
Trust anchor reference:
Certificate store reference:
Trust any certificate

7.1.5.2 Step 2 – WAS: Deploy the application

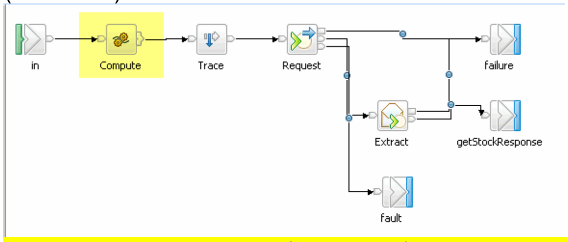
Detailed deployment steps are described in the appendix.

7.1.5.3 Step 3 – WMB: Develop the message flow as Web Service consumer

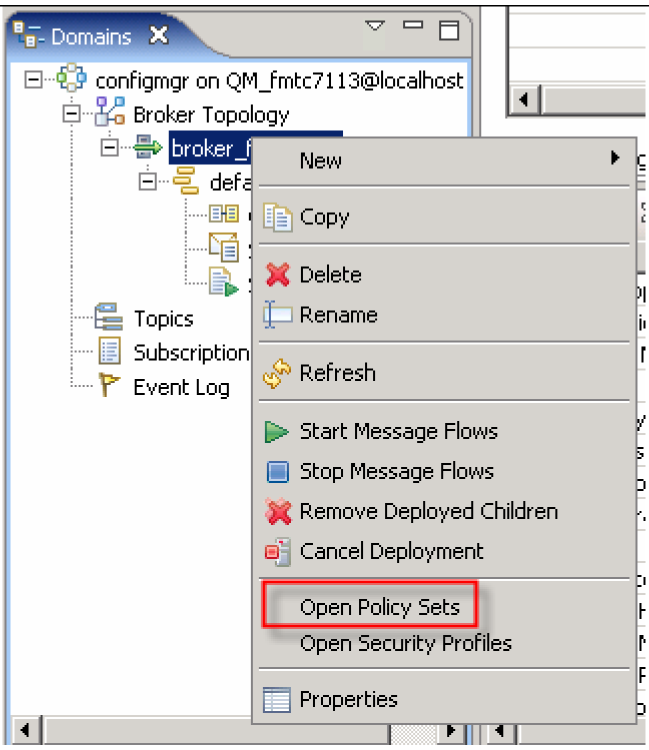
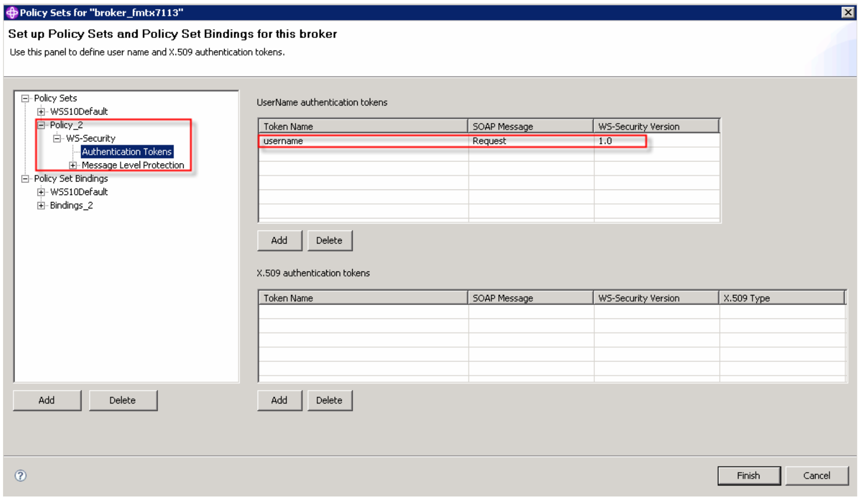
This section describes how to take the identity information from the message body and build a SOAP Username Token. The objective is to get a SOAP Message as shown below, which is build in WMB and sent to WAS.

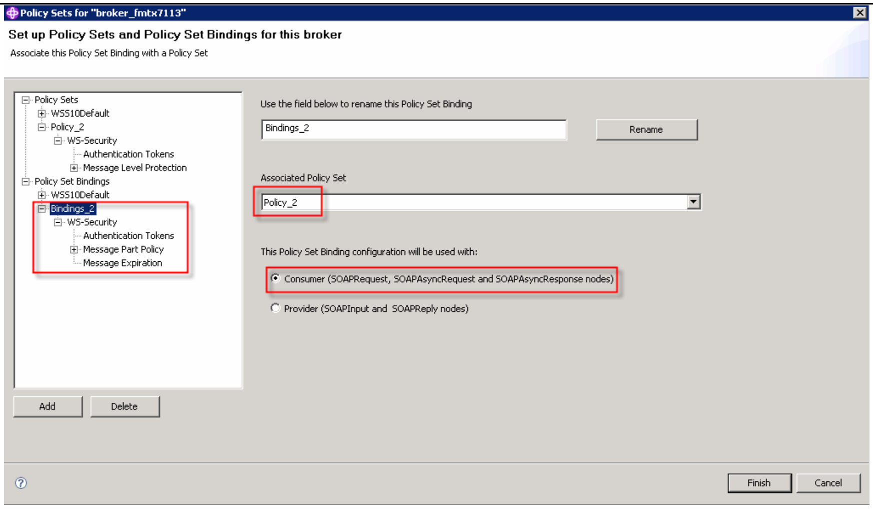
```
<soapenv:Envelope xmlns:show="http://showcase.ibm.com" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <wsse:Security  soapenv:mustUnderstand="1"  xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <wsse:UsernameToken  wsu:Id="UsernameToken-31775739"  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
        <wsse:Username>bob</wsse:Username>
         </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body>
    <show:getStock>
      <partNumber>0</partNumber>
      <count>0</count>
    </show:getStock>
  </soapenv:Body>
</soapenv:Envelope>
```

To build such SOAP messages with the Username Token, perform the following steps.

1.	First, create a Compute Node. The Compute Node extracts the user ID from the input message and assigns it to an `IdentityMappedToken` . The Compute Node will also set the `IdentityMappedType` to `username`
	Based on both properties, the SOAP request node will later build the SOAP Security Header with a (asserted) Username Token.
	 in
	Compute
	Trace
	Request
	failure
	Extract
	getStockResponse
	fault
	`SET OutputRoot.Properties.IdentityMappedType='username';`
	`SET OutputRoot.Properties.IdentityMappedToken= InputRoot.MRM.CSV_Row.CSV_Column[3];`

2.	To propagate the IdentityMappedType and IdentityMappedTokenType to the SOAP Security Header a **Policy Set** must be created and added to the bar file. Also, a **Security Profile** must be added to the RequestNode: • Create a Policy Set • Right-clickon the broker and select Open Policy Sets

	 Domains configmgr on QM_fmtc7113@localhost Broker Topology broker_f defa Topics Subscription Event Log New Copy Delete Rename Refresh Start Message Flows Stop Message Flows Remove Deployed Children Cancel Deployment Open Policy Sets Open Security Profiles Properties
3.	Add a new *Policy Set* (for example, Policy_2) and add a new *Authentication Token* (for example, username)  Policy Sets for "broker_fmtx7113" **Set up Policy Sets and Policy Set Bindings for this broker** Use this panel to define user name and X.509 authentication tokens. Policy Sets WSS10Default Policy_2 WS-Security Authentication Tokens Message Level Protection Policy Set Bindings WSS10Default Bindings_2 UserName authentication tokens Token Name / SOAP Message / WS-Security Version username / Request / 1.0 X.509 authentication tokens Token Name / SOAP Message / WS-Security Version / X.509 Type
4.	Add a *New Policy Set Binding*



Policy Sets for "broker_fmtx7113"

Set up Policy Sets and Policy Set Bindings for this broker

Associate this Policy Set Binding with a Policy Set

```
⊟ Policy Sets
  ⊞ WSS10Default
  ⊟ Policy_2
    ⊟ WS-Security
        Authentication Tokens
      ⊞ Message Level Protection
⊟ Policy Set Bindings
  ⊞ WSS10Default
  ⊟ Bindings_2
    ⊟ WS-Security
        Authentication Tokens
      ⊞ Message Part Policy
        Message Expiration
```

Use the field below to rename this Policy Set Binding: Bindings_2

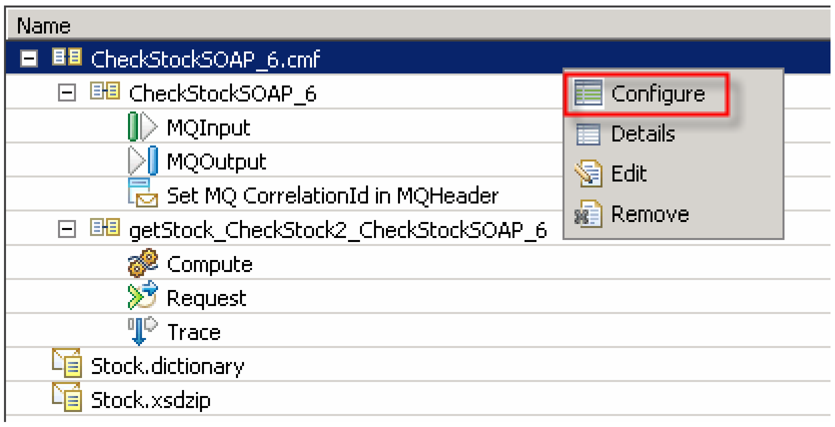
Associated Policy Set: Policy_2

This Policy Set Binding configuration will be used with:

[✓ Consumer (SOAPRequest, SOAPAsyncRequest and SOAPAsyncResponse nodes)]

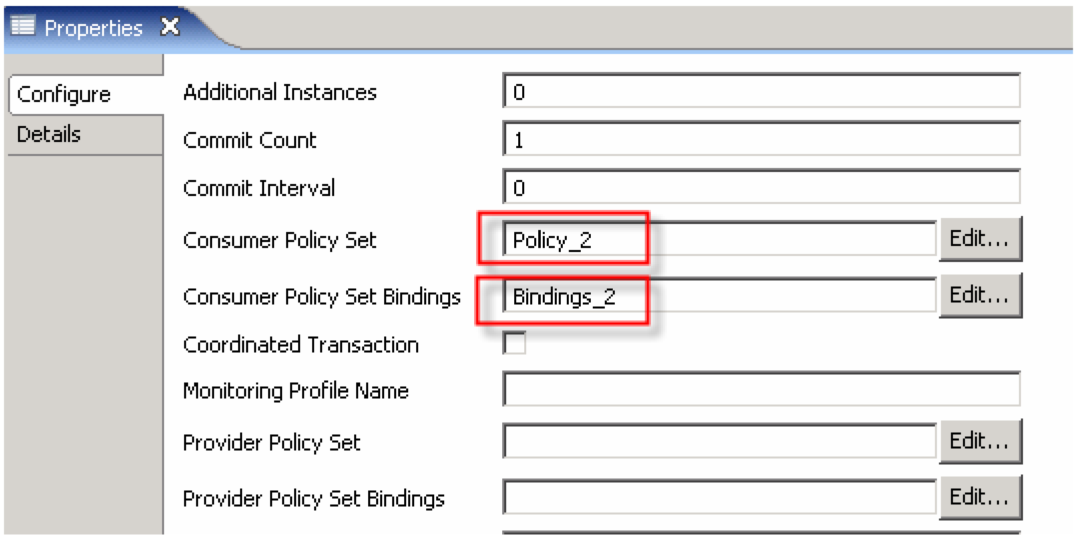
Provider (SOAPInput and SOAPReply nodes)

5. Add the Policy set to the BAR:
Right-click on the *cmf* file and select *Configure*



```
Name
⊟ CheckStockSOAP_6.cmf
    ⊟ CheckStockSOAP_6
          MQInput
          MQOutput
          Set MQ CorrelationId in MQHeader
    ⊟ getStock_CheckStock2_CheckStockSOAP_6
          Compute
          Request
          Trace
  Stock.dictionary
  Stock.xsdzip
```

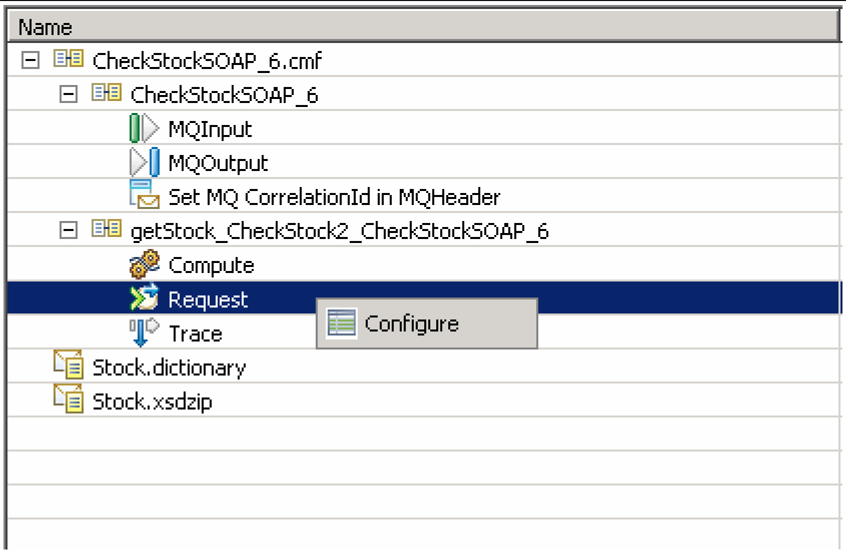
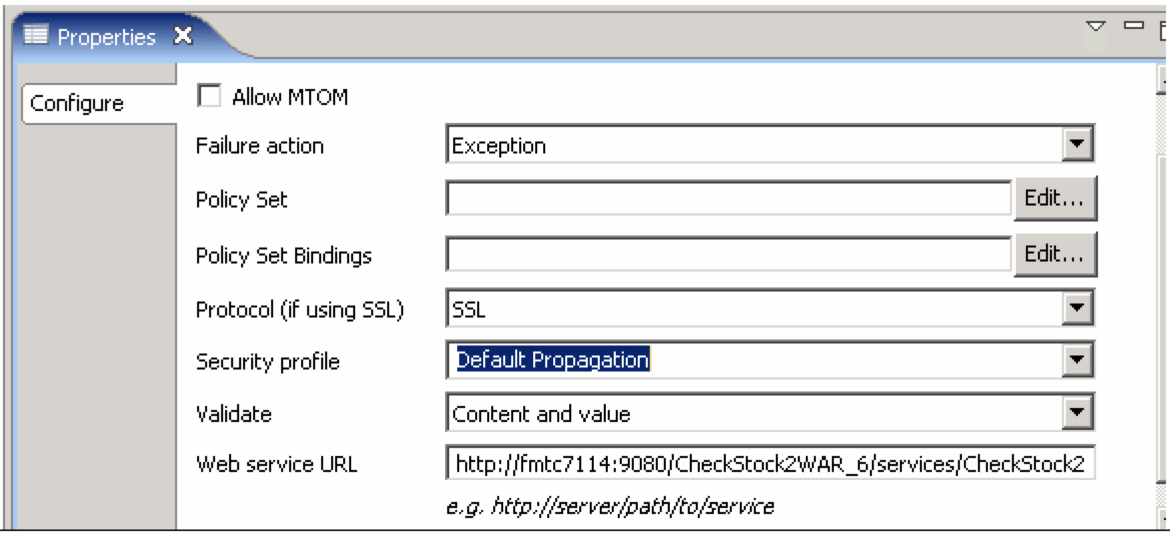
6. Add the Policy set to the *Consumer Policy Set* and the Binding to the *Consumer Binding*



Properties

Additional Instances	0
Commit Count	1
Commit Interval	0
Consumer Policy Set	Policy_2
Consumer Policy Set Bindings	Bindings_2
Coordinated Transaction	☐
Monitoring Profile Name	
Provider Policy Set	
Provider Policy Set Bindings	

7. Add the *defaultSecurity* profile to the RequestNode (the defaultSecurity profile is configured for identity propagation):
In the BAR file Right-click on *the Request Node* and select *Configure*

	 Name CheckStockSOAP_6.cmf CheckStockSOAP_6 MQInput MQOutput Set MQ CorrelationId in MQHeader getStock_CheckStock2_CheckStockSOAP_6 Compute Request Trace Stock.dictionary Stock.xsdzip Configure
8.	As Security profile, select *Default propagation*  Properties Configure ☐ Allow MTOM Failure action: Exception Policy Set: Edit... Policy Set Bindings: Edit... Protocol (if using SSL): SSL Security profile: Default Propagation Validate: Content and value Web service URL: http://fmtc7114:9080/CheckStock2WAR_6/services/CheckStock2 *e.g. http://server/path/to/service*

7.1.5.4 Step 4 – WMB: Deploy the message flow

Detailed deployment steps are described in the appendix.

7.1.5.5 Step 5 – SSL configuration between WMB and WAS

Refer to chapter “SSL between WMB and WAS”

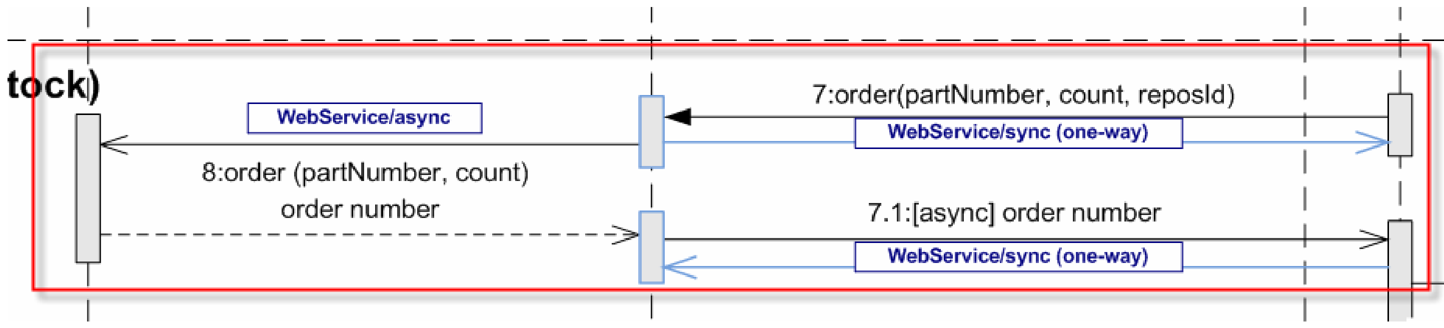
7.1.6 Interaction 7 and 8 - SOAP/HTTP from WPS to WAS via Message Broker

This section describes identity propagation from WPS to WAS via Message Broker. identity is propagated via Username Token in the Web Service Security Header.
This section also describes the implementation of an asynchronous SOAP Request from WMB to WAS.

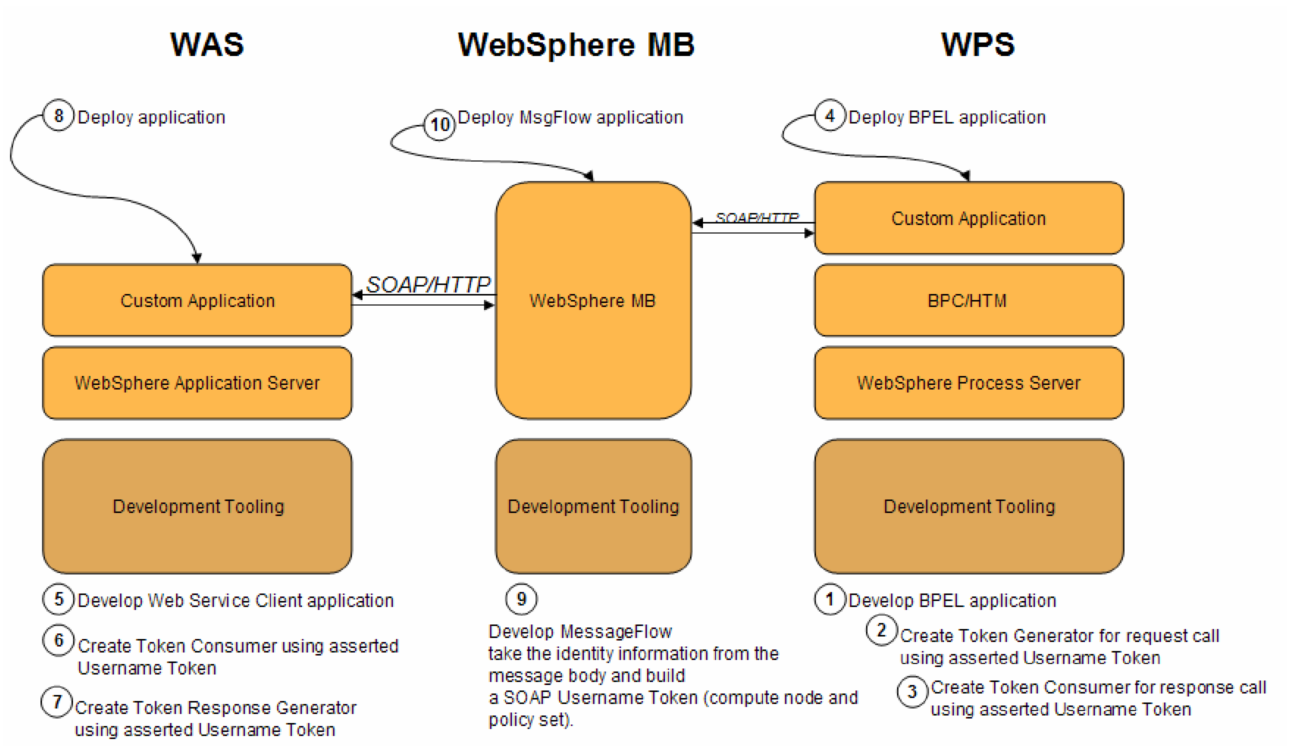
- WPS makes a one-way call with a Username Token to WMB
- WMB copies the message header (Username Token) and calls WAS
- WAS sends a response message, which is received by WMB
- WMB forwards it using a one-way call to WPS

	Client Application	Server Application
1	ShowcaseApp (WPS, SCA Import – SOAP/HTTP Binding)	InternalOrder_7Flow.mgsflow (WMB)
2	InternalOrder_7Flow.mgsflow (WMB)	InternalOrder8 (WAS)
3	InternalOrder_7Flow.mgsflow (WMB)	ShowcaseApp (WPS, SCA Export – SOAP/HTTP Binding)

The figure below shows the relevant part in the sequence diagram:


tock)
WebService/async
7:order(partNumber, count, reposId)
WebService/sync (one-way)
8:order (partNumber, count)
order number
7.1:[async] order number
WebService/sync (one-way)

The figure below shows the high-level implementation and configuration steps that have to be performed. Find details of the steps in the next sections.

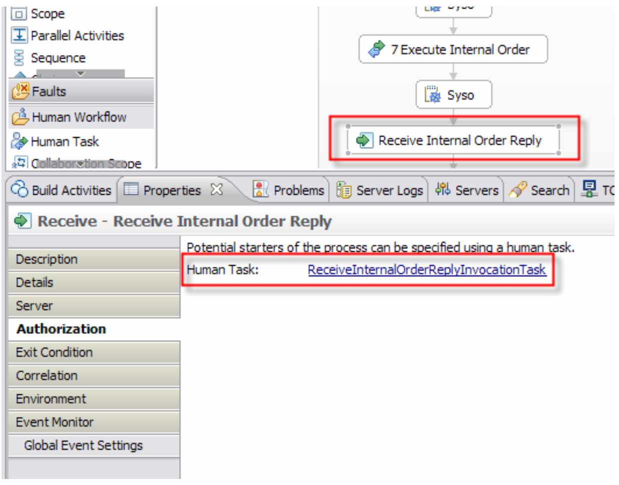


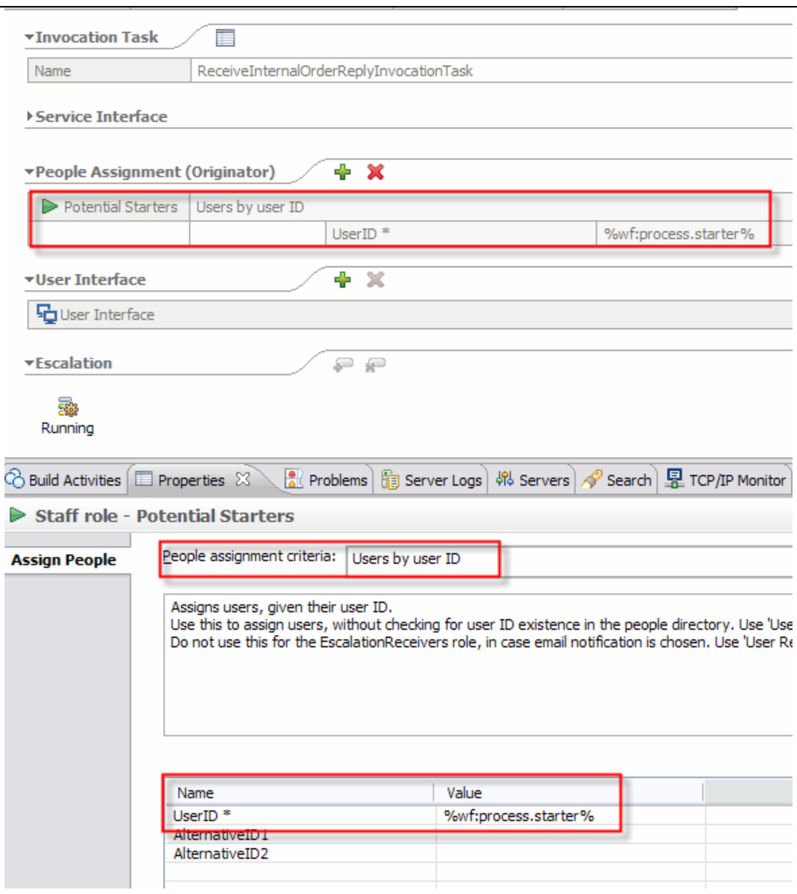
WAS
WebSphere MB
WPS
8 Deploy application
10 Deploy MsgFlow application
4 Deploy BPEL application
Custom Application
SOAP/HTTP
WebSphere MB
SOAP/HTTP
Custom Application
BPC/HTM
WebSphere Application Server
WebSphere Process Server
Development Tooling
Development Tooling
Development Tooling
5 Develop Web Service Client application
6 Create Token Consumer using asserted Username Token
7 Create Token Response Generator using asserted Username Token
9 Develop MessageFlow take the identity information from the message body and build a SOAP Username Token (compute node and policy set).
1 Develop BPEL application
2 Create Token Generator for request call using asserted Username Token
3 Create Token Consumer for response call using asserted Username Token

7.1.6.1 Step 1 – WPS: Develop the BPEL application

Refer to the WID artefacts to see how the full BPEL application is developed.

The process must be configured so that it only accepts responses with the identity of the process starter:

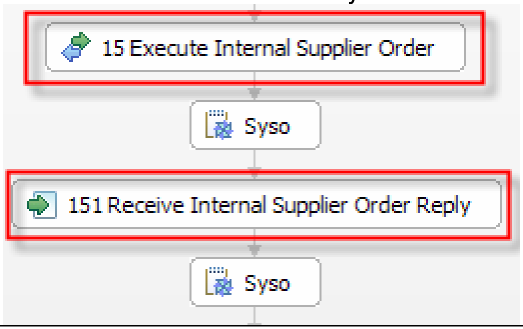
1.	Add a Authorization Human task to the Receive activity  Receive - Receive Internal Order Reply; Potential starters of the process can be specified using a human task.; Human Task: ReceiveInternalOrderReplyInvocationTask
2.	Open the Human Task and select as Potential Starters *Users by user ID* and as value *%wf:process.starter%*

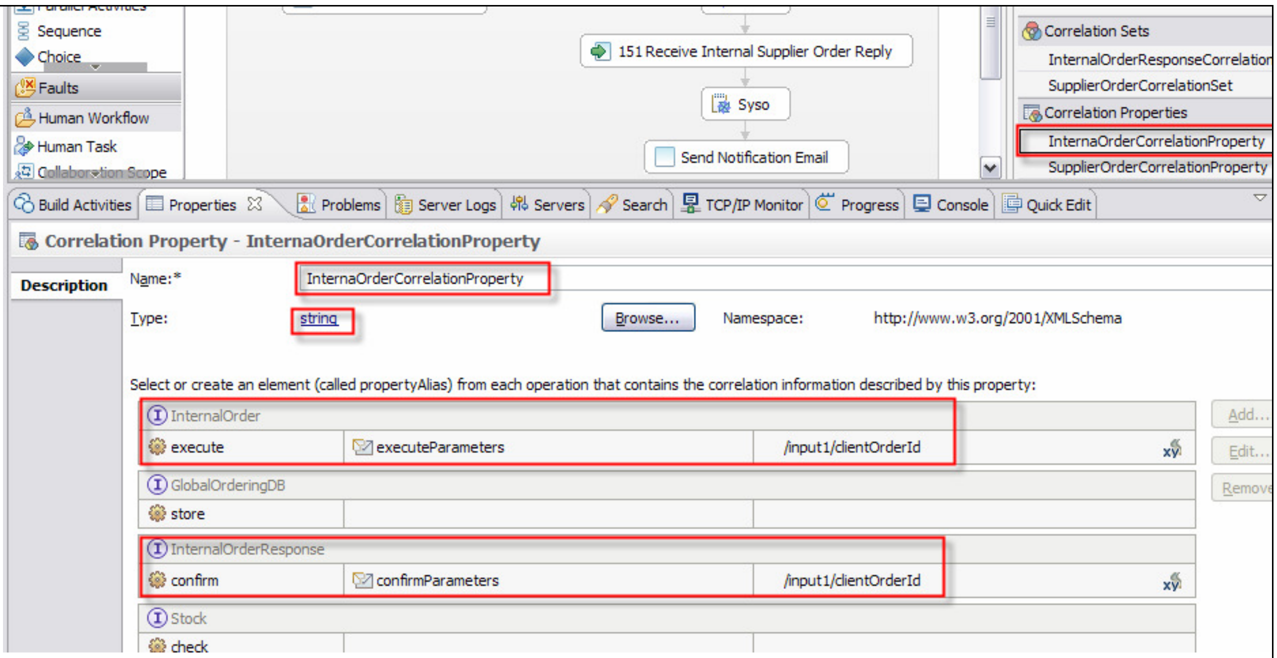
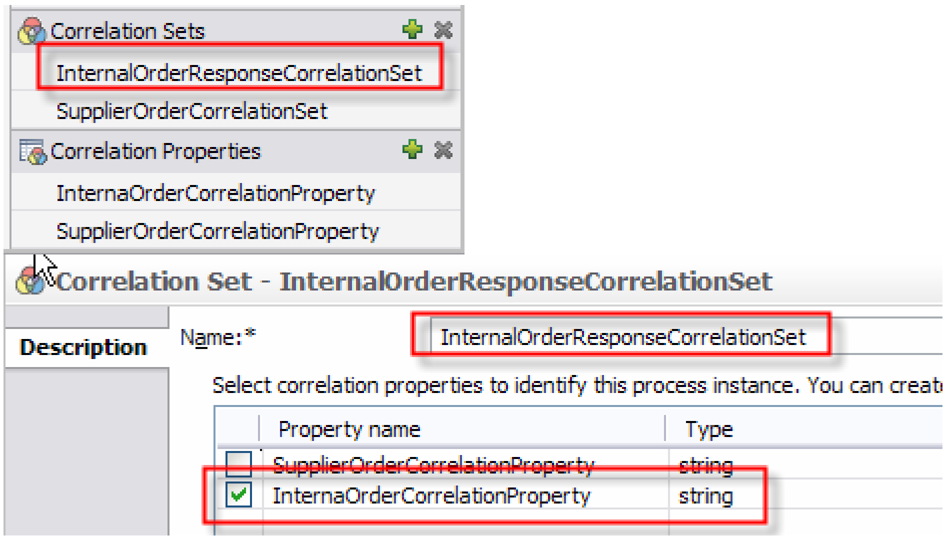
	 Invocation Task Name ReceiveInternalOrderReplyInvocationTask Service Interface People Assignment (Originator) Potential Starters Users by user ID UserID * %wf:process.starter% User Interface User Interface Escalation Running Build Activities Properties Problems Server Logs Servers Search TCP/IP Monitor Staff role - Potential Starters Assign People People assignment criteria: Users by user ID Assigns users, given their user ID. Use this to assign users, without checking for user ID existence in the people directory. Use 'Use Do not use this for the EscalationReceivers role, in case email notification is chosen. Use 'User Re Name Value UserID * %wf:process.starter% AlternativeID1 AlternativeID2
3.	Save the Human Task

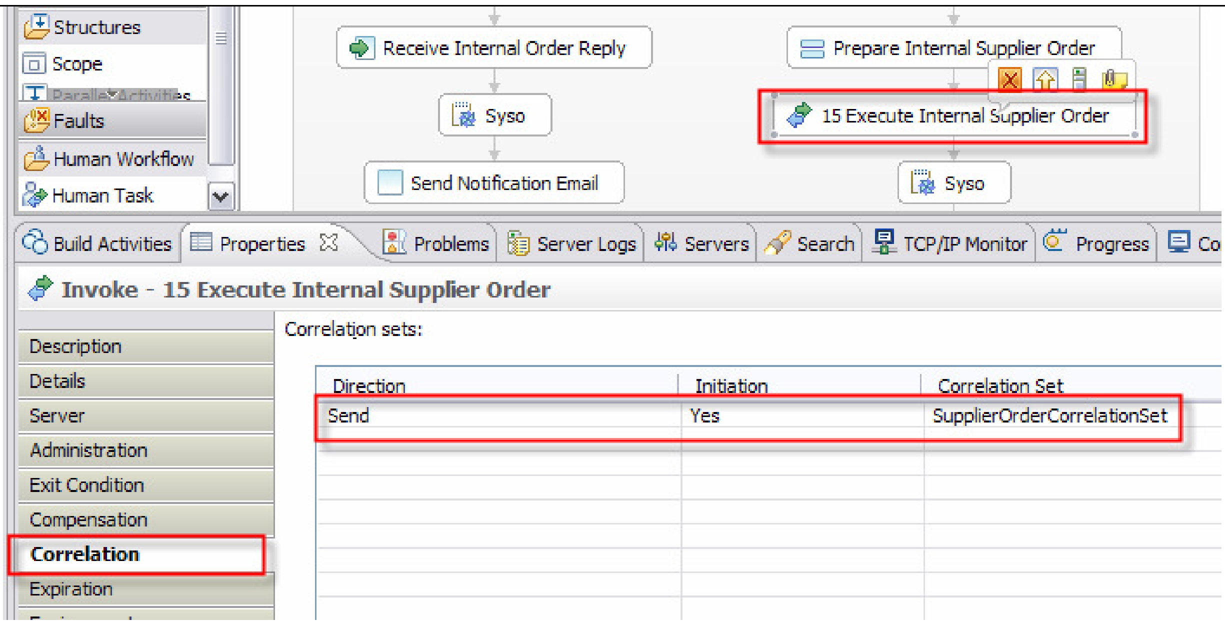
7.1.6.2 Correlation in BPEL

In this interaction step we use BPEL correlation set to correlate response messages. This section describes the implementation of a correlation set in the showcase application.

Reference Material BPC Samples Page http://publib.boulder.ibm.com/bpcsamp/advancedProcessFeatures/correlation.html

1	The BPEL contains a one way invoke and a Receive activity.  15 Execute Internal Supplier Order Syso 151 Receive Internal Supplier Order Reply Syso
2	To correlate the response message in the Receive activity, a Correlation Property must be created. The Correlation Property specifies the correlation parameters of the request interface and of the response interface.

	 **Correlation Property - InternaOrderCorrelationProperty** Name:* InternaOrderCorrelationProperty Type: string Browse... Namespace: http://www.w3.org/2001/XMLSchema Select or create an element (called propertyAlias) from each operation that contains the correlation information described by this property: InternalOrder — execute — executeParameters — /input1/clientOrderId GlobalOrderingDB — store InternalOrderResponse — confirm — confirmParameters — /input1/clientOrderId Stock — check Correlation Sets: InternalOrderResponseCorrelation, SupplierOrderCorrelationSet Correlation Properties: InternaOrderCorrelationProperty, SupplierOrderCorrelationProperty
3	Also a Correlation Set must be defined, which contains a link to the InternalOrderCorrelationProperty.  Correlation Sets: InternalOrderResponseCorrelationSet, SupplierOrderCorrelationSet Correlation Properties: InternaOrderCorrelationProperty, SupplierOrderCorrelationProperty **Correlation Set - InternalOrderResponseCorrelationSet** Name:* InternalOrderResponseCorrelationSet Select correlation properties to identify this process instance. You can creat Property name / Type ☐ SupplierOrderCorrelationProperty — string ☑ InternaOrderCorrelationProperty — string
4	First, the correlation set must be initialized at the invoke activity

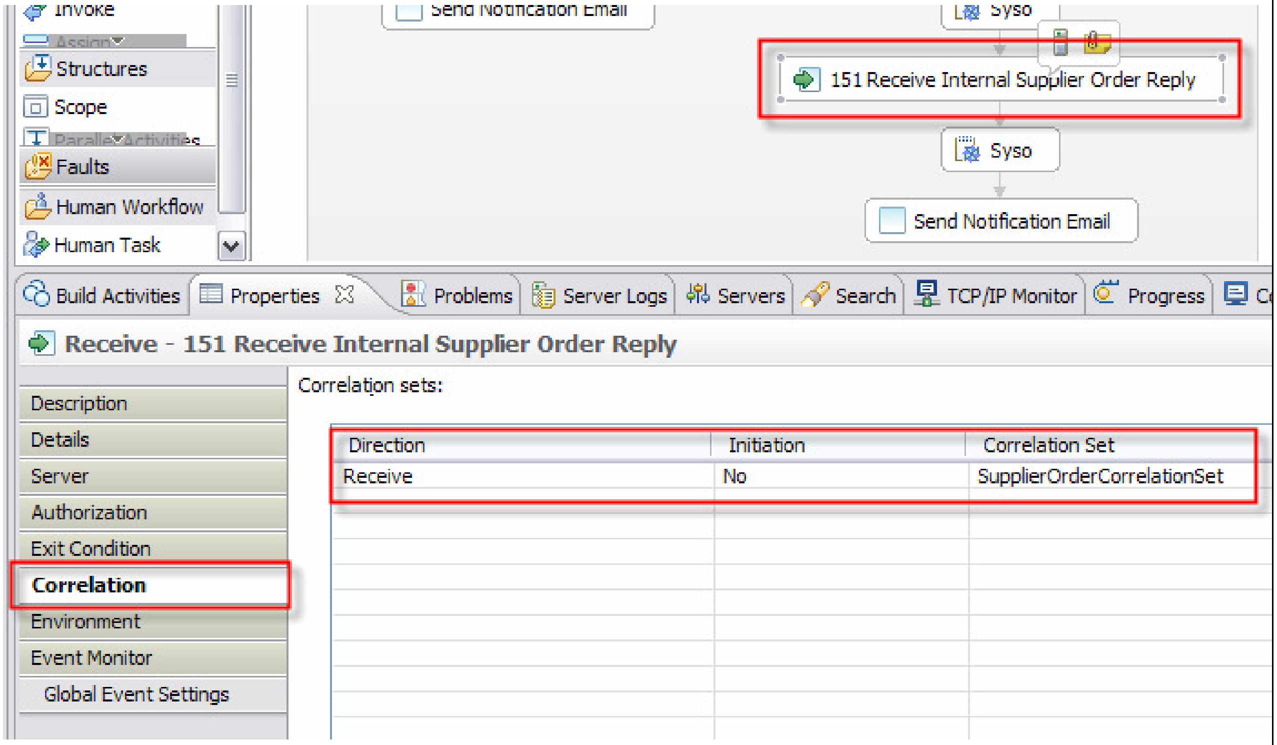


Invoke - 15 Execute Internal Supplier Order

Correlation sets:

Direction	Initiation	Correlation Set
Send	Yes	SupplierOrderCorrelationSet

5 Second, the correlation set must be used at the receive activity



Receive - 151 Receive Internal Supplier Order Reply

Correlation sets:

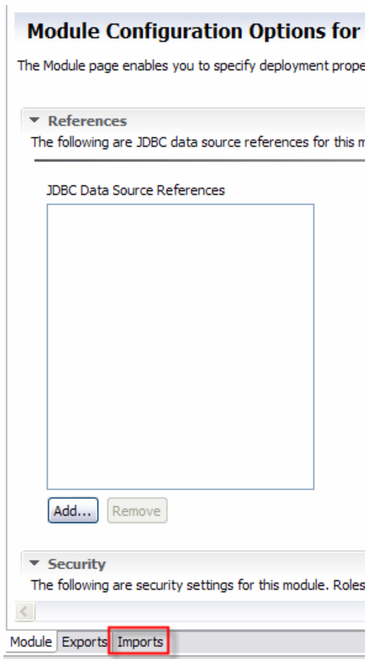
Direction	Initiation	Correlation Set
Receive	No	SupplierOrderCorrelationSet

6 Now, the BPEL flow is able to correlate the response message to the appropriate process instance.

7.1.6.3 Step 2 – WPS: Create a Token Generator

WMB as service provider expects from the WPS client an asserted Username Token. Therefore, we have to configure the deployment descriptor of WPS accordingly.

1.	*Right-click* the showcase SCA Module and select *Open Deployment Editor*
2.	Click the *Imports* tab


Module Configuration Options for

The Module page enables you to specify deployment prope

References

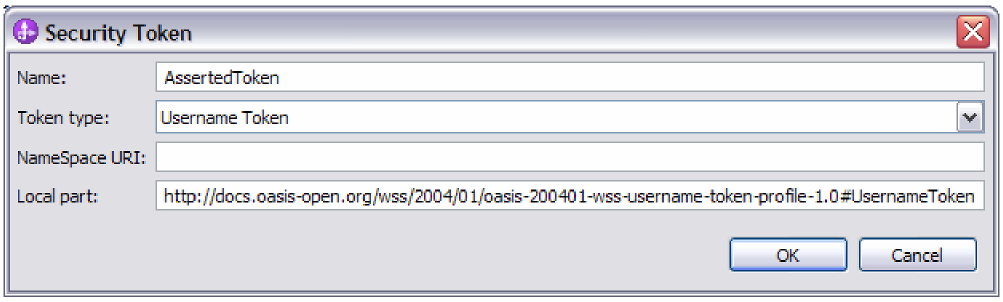
The following are JDBC data source references for this m

JDBC Data Source References

Security

The following are security settings for this module. Roles

3. Click *the WS-Security Extension* tab
Add a Security Token *under Request Generator Configuration*
Select as Token type *Username Token*
Local part is filled automatically.


Security Token

Name: AssertedToken

Token type: Username Token

NameSpace URI:

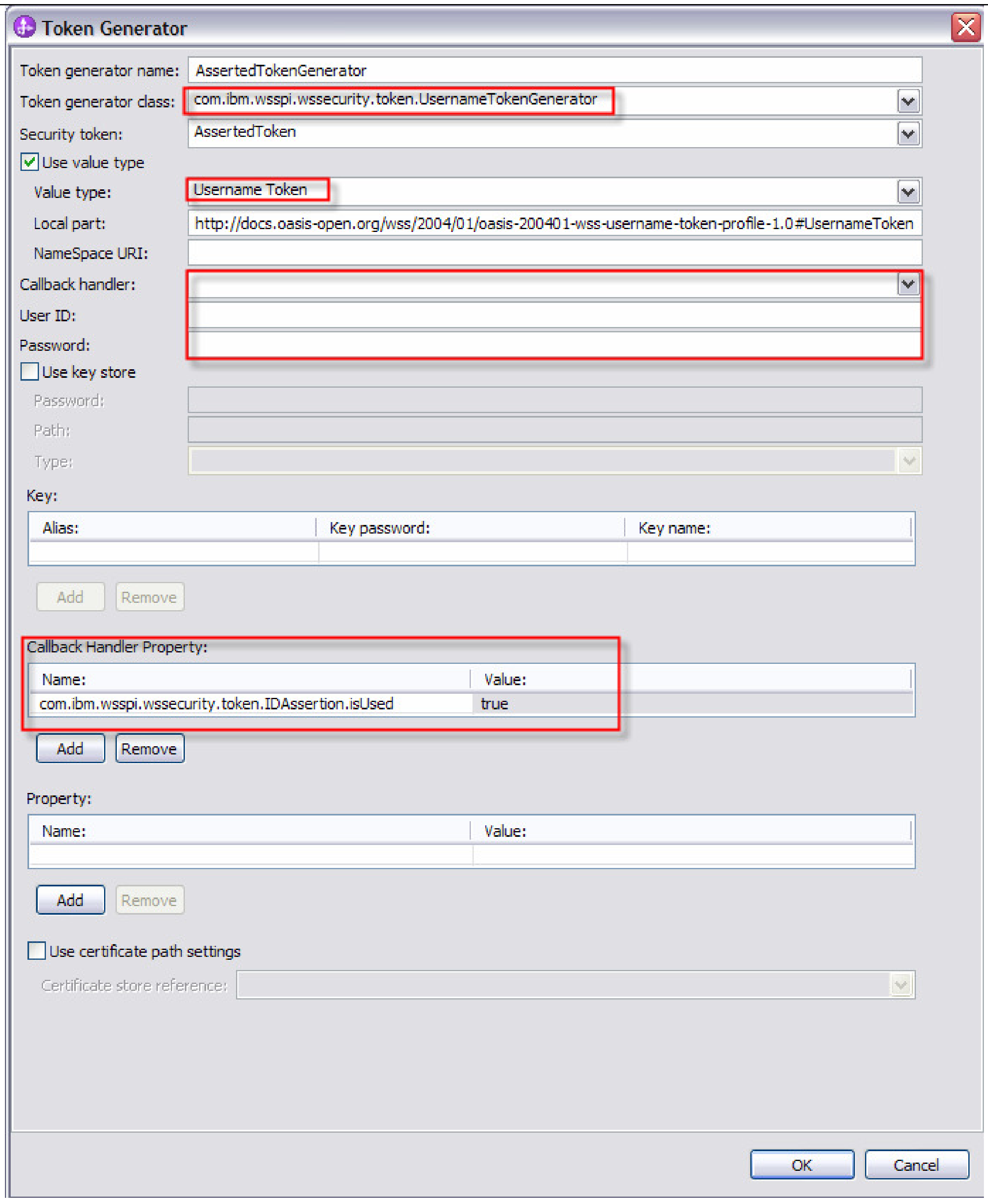
Local part: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken

4. Click *WS-Security Bindings*
Add a *Token Generator* under *Security Request Generator Configuration*

Token Generator Name:	AssertedTokenGenerator
Token generator class:	com.ibm.wsspi.wssecurity.token.UsernameTokenGenerator
Security Token:	AssertedToken
Use value type:	Checked
Callback handler:	Blank
UserID	Blank
Password	Blank

Callback handler Properties:

com.ibm.wsspi.wssecurity.token.IDAssertion.isUsed=true
com.ibm.wsspi.wssecurity.token.IDAssertion.useRunAsIdentity=true
You'll need to click the Add button to add a row and then select name and value fields to type over.



Token Generator

Token generator name: AssertedTokenGenerator

Token generator class: com.ibm.wsspi.wssecurity.token.UsernameTokenGenerator

Security token: AssertedToken

[✓] Use value type

Value type: Username Token

Local part: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken

NameSpace URI:

Callback handler:

User ID:

Password:

[ ] Use key store

Password:

Path:

Type:

Key:

Alias:	Key password:	Key name:

Callback Handler Property:

Name:	Value:
com.ibm.wsspi.wssecurity.token.IDAssertion.isUsed	true

Property:

Name:	Value:

[ ] Use certificate path settings

Certificate store reference:

7.1.6.4 Step 3 – WPS: Create a Token Consumer

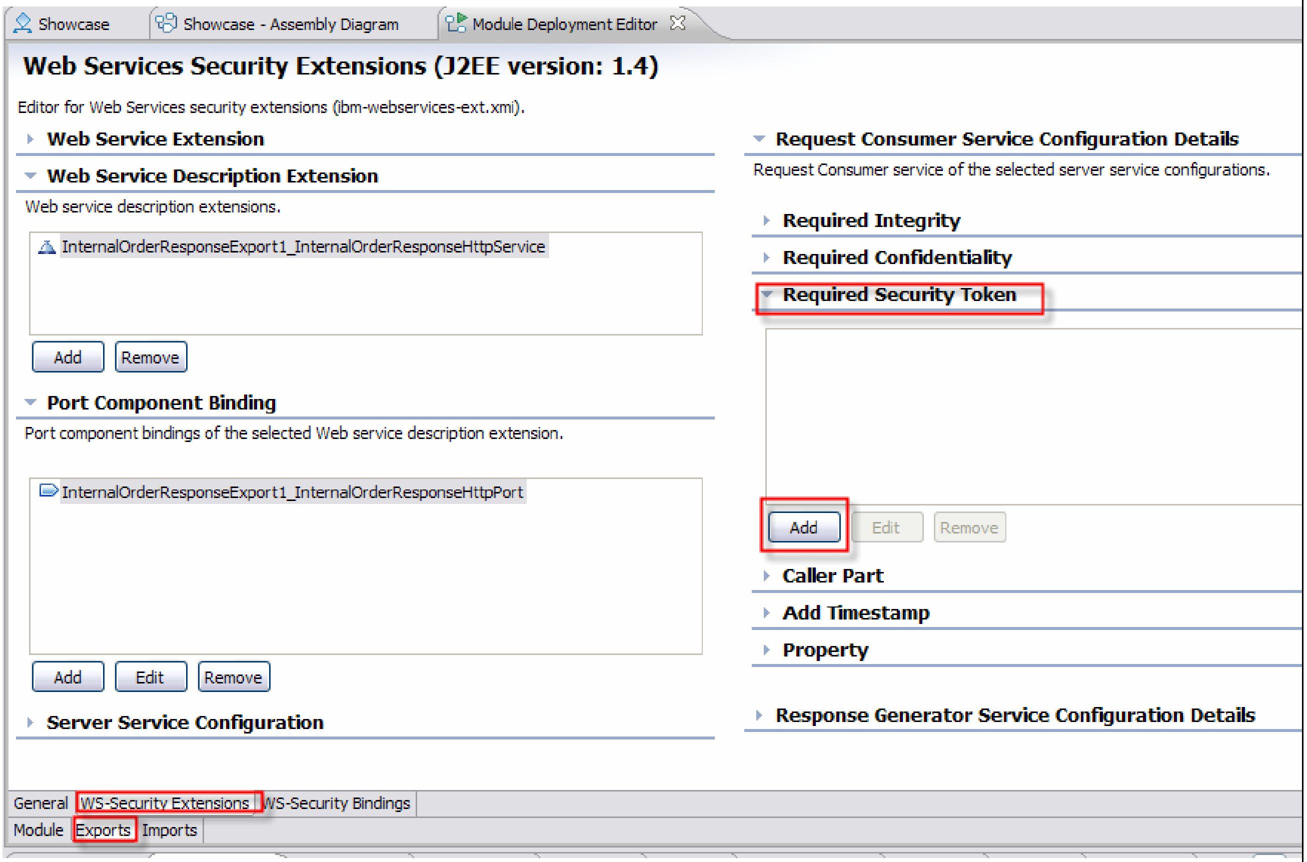
The response to WPS is delivered as Web Service call from WMB to WPS. This means:

- WPS is Web Service provider
- WMB is Web Service client

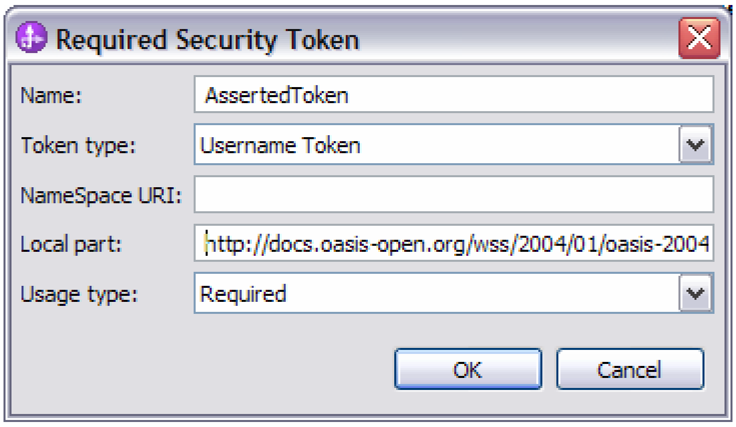
We need to deliver in the identity in form of a Username Token from (WAS to) WMB to WPS.

Right-click the showcase SCA Module and select *Open Deployment Editor*

Click the *Exports* tab > *WS-Security-Extensions* and add a *Required Security Token*

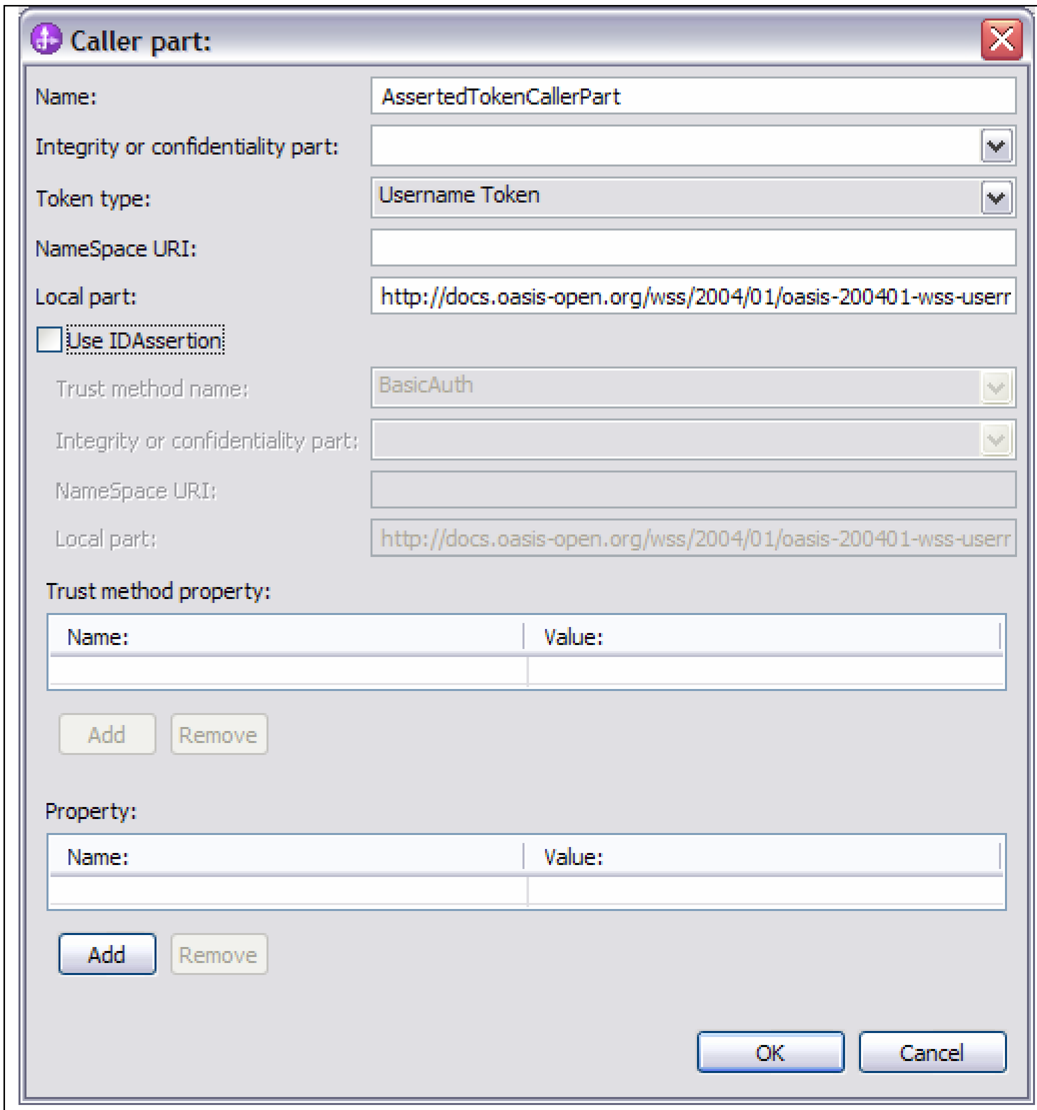

Showcase
Showcase - Assembly Diagram
Module Deployment Editor
Web Services Security Extensions (J2EE version: 1.4)
Editor for Web Services security extensions (ibm-webservices-ext.xmi).
Web Service Extension
Web Service Description Extension
Web service description extensions.
InternalOrderResponseExport1_InternalOrderResponseHttpService
Add
Remove
Port Component Binding
Port component bindings of the selected Web service description extension.
InternalOrderResponseExport1_InternalOrderResponseHttpPort
Add
Edit
Remove
Server Service Configuration
Request Consumer Service Configuration Details
Request Consumer service of the selected server service configurations.
Required Integrity
Required Confidentiality
Required Security Token
Add
Edit
Remove
Caller Part
Add Timestamp
Property
Response Generator Service Configuration Details
General
WS-Security Extensions
WS-Security Bindings
Module
Exports
Imports

Click *the WS-Security Extension* tab
Add a Security Token *under Request Generator Configuration*
Select as Token type *Username Token*
Local part is filled automatically.



Name:	AssertedToken
Token type:	Username Token
NameSpace URI:	
Local part:	http://docs.oasis-open.org/wss/2004/01/oasis-2004
Usage type:	Required

Open *Request Consumer Binding Configuration Details > Caller Part*
Click Add



Caller part:

Name:	AssertedTokenCallerPart
Integrity or confidentiality part:	
Token type:	Username Token
NameSpace URI:	
Local part:	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-userr

[ ] Use IDAssertion

Trust method name:	BasicAuth
Integrity or confidentiality part:	
NameSpace URI:	
Local part:	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-userr

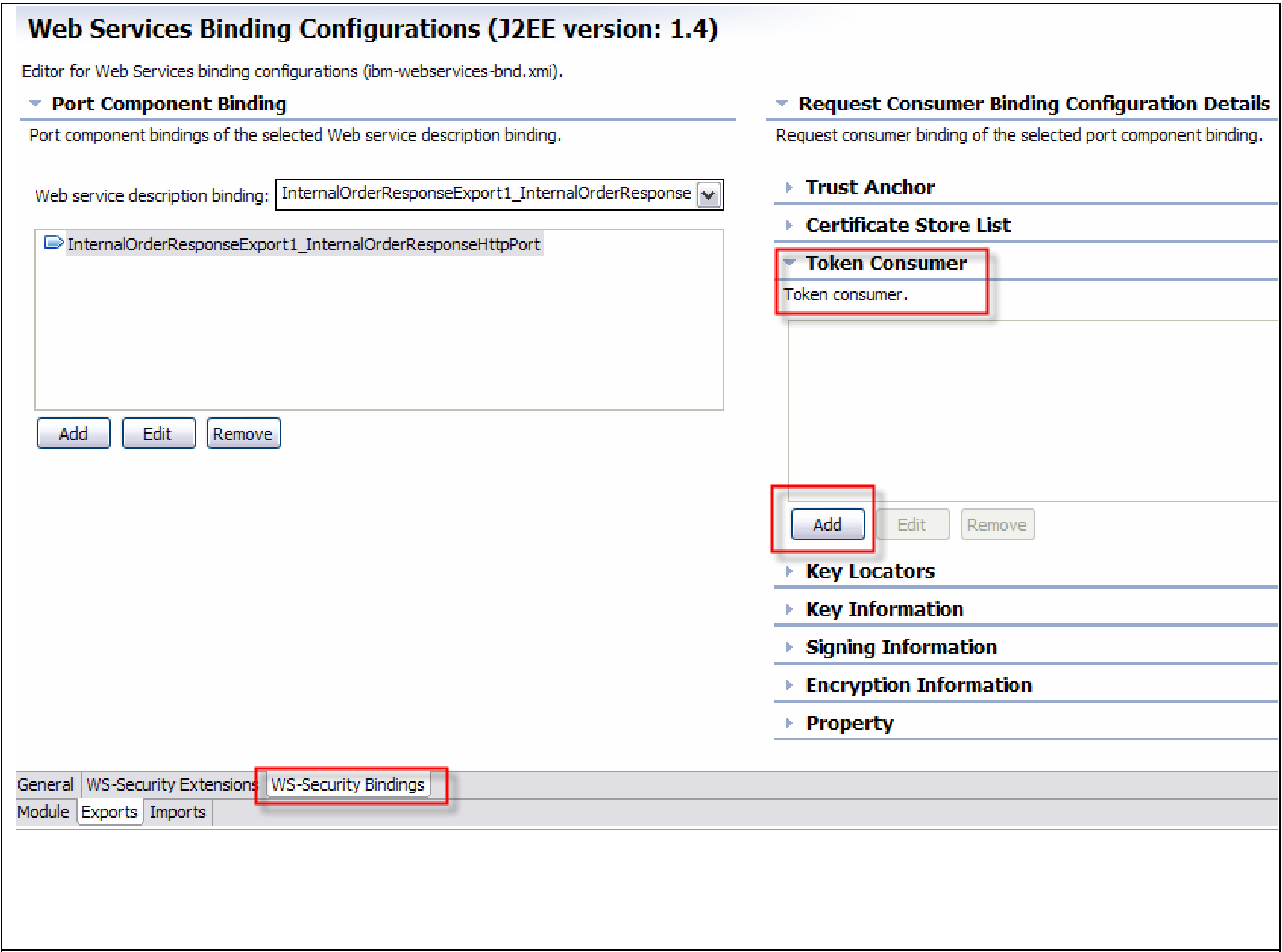
Trust method property:

Name:	Value:

Property:

Name:	Value:

Switch to the *WS-Security Bindings* tab and add a new *Token Consumer*



Web Services Binding Configurations (J2EE version: 1.4)

Editor for Web Services binding configurations (ibm-webservices-bnd.xmi).

Port Component Binding

Port component bindings of the selected Web service description binding.

Web service description binding: InternalOrderResponseExport1_InternalOrderResponse

InternalOrderResponseExport1_InternalOrderResponseHttpPort

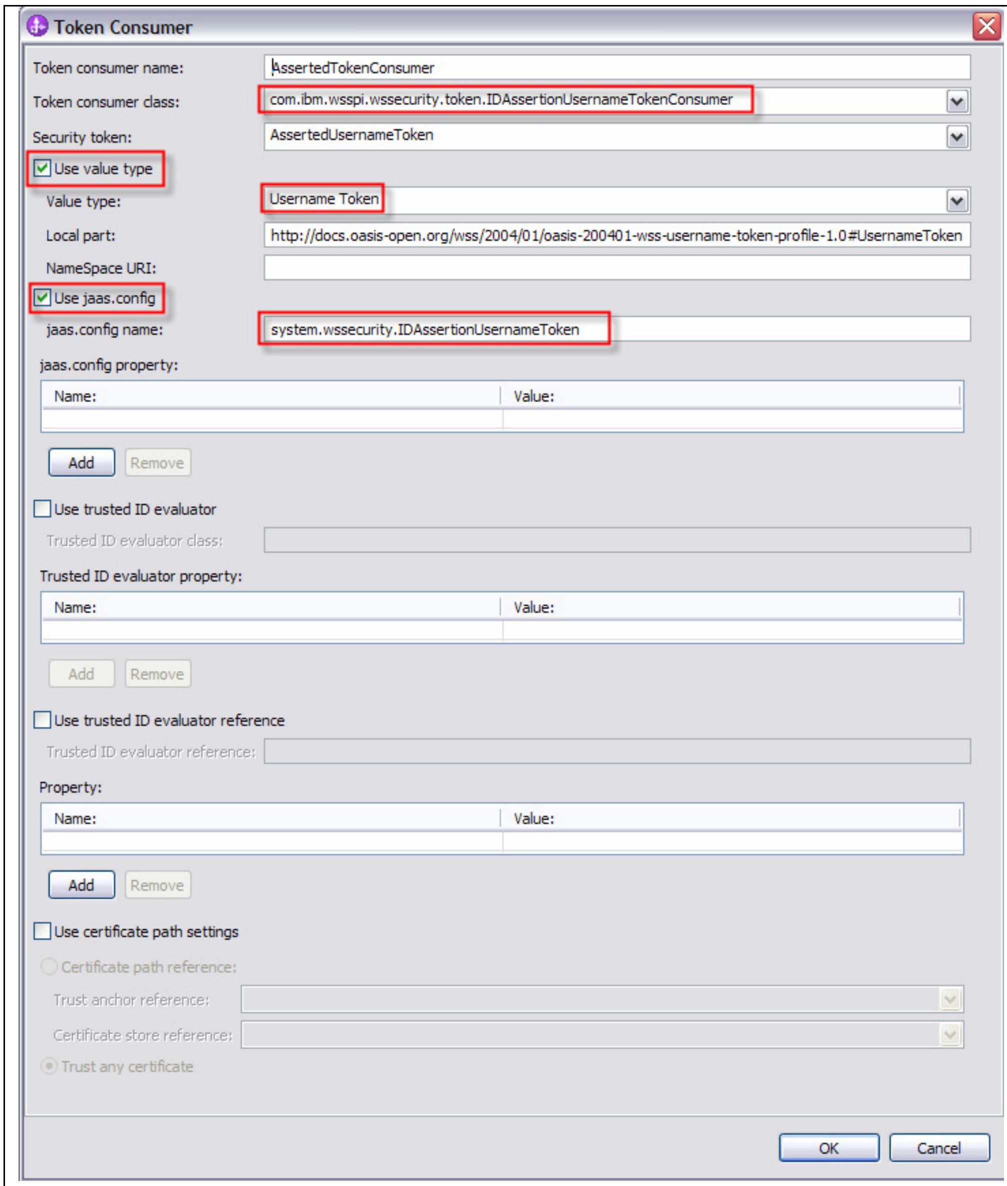
Request Consumer Binding Configuration Details

Request consumer binding of the selected port component binding.

Token Consumer

Token consumer.

- In the Token Consumer dialog box enter a consumer name, e.g AssertedTokenConsumer
- Select as Token consumer class ***com.ibm.wsspi.wssecurity.token.IDAssertionUsernameTokenConsumer***
- As Security Token select *AssertedUsernameToken*
- Check *Use value type*
- Select as Value type: *Username Token*
- Local Part is generated automatically
- Check *Use.jaas.config*
- Enter as jaas.config name: *system.wssecurity.IDAssertionUsernameToken*
  by selecting the *IDAssertionUsernameToken* we define that we just need the user ID, and no password
- Click *OK*


Token Consumer

Token consumer name:	AssertedTokenConsumer
Token consumer class:	com.ibm.wsspi.wssecurity.token.IDAssertionUsernameTokenConsumer
Security token:	AssertedUsernameToken
☑ Use value type	
Value type:	Username Token
Local part:	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken
NameSpace URI:	
☑ Use jaas.config	
jaas.config name:	system.wssecurity.IDAssertionUsernameToken

jaas.config property:

Name:	Value:

☐ Use trusted ID evaluator

Trusted ID evaluator class:

Trusted ID evaluator property:

Name:	Value:

☐ Use trusted ID evaluator reference

Trusted ID evaluator reference:

Property:

Name:	Value:

☐ Use certificate path settings

○ Certificate path reference:

Trust anchor reference:

Certificate store reference:

● Trust any certificate

7.1.6.5 Step 4 – WPS: Deploy the BPEL application

Detailed deployment steps are described in the appendix.

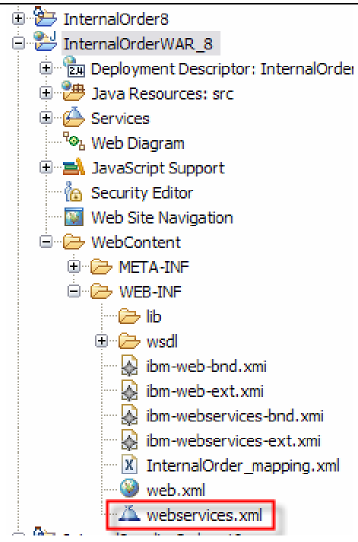
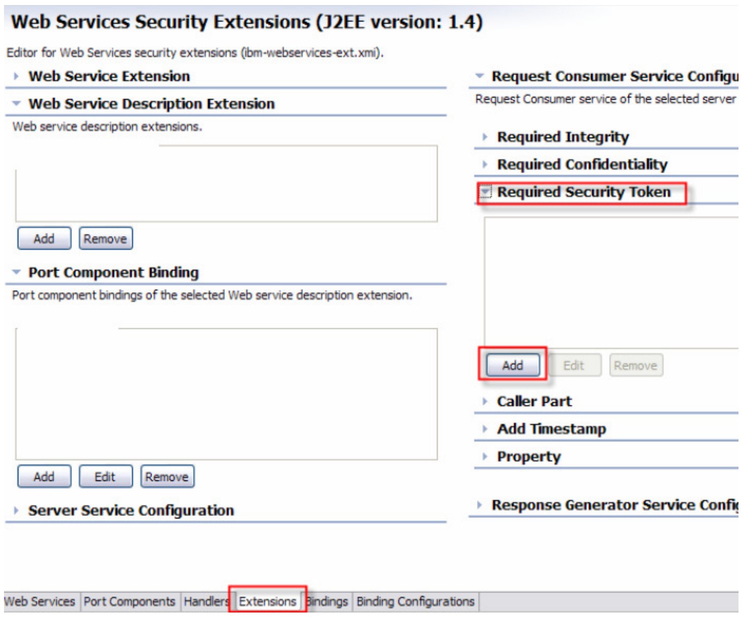
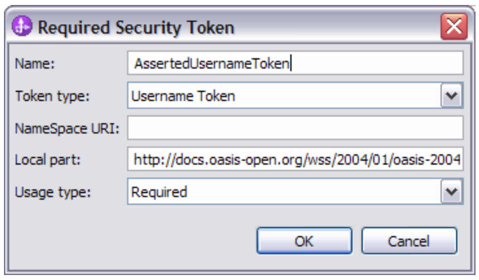
7.1.6.6 Step 5 – WAS: Develop the WAS application

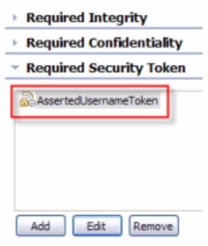
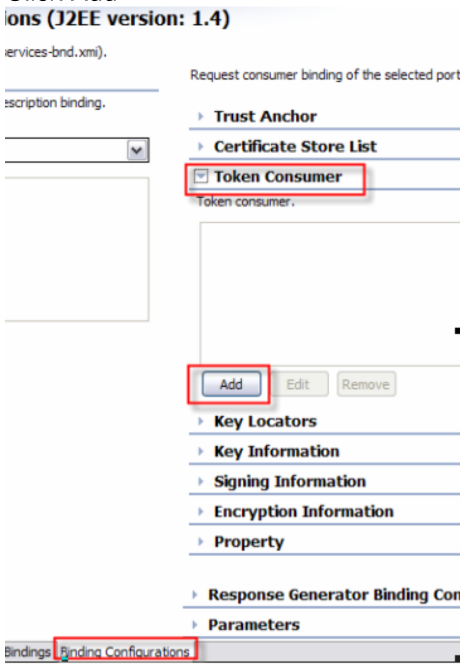
Detailed implementation steps, which are not security relevant, are not described. Refer to the WID artefacts to see how the application is developed.

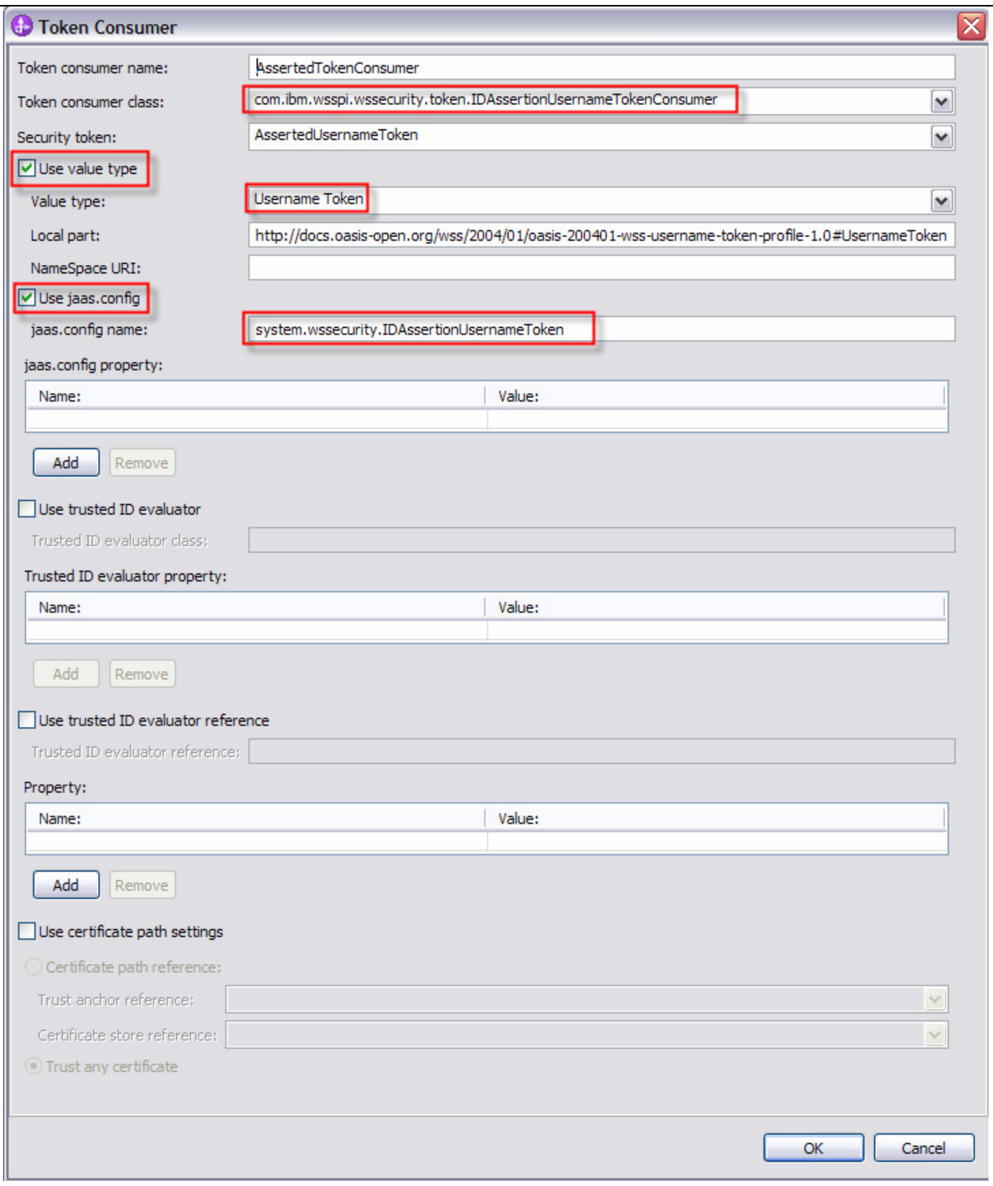
7.1.6.7 Step 6 – WAS: Create the Token Request Consumer

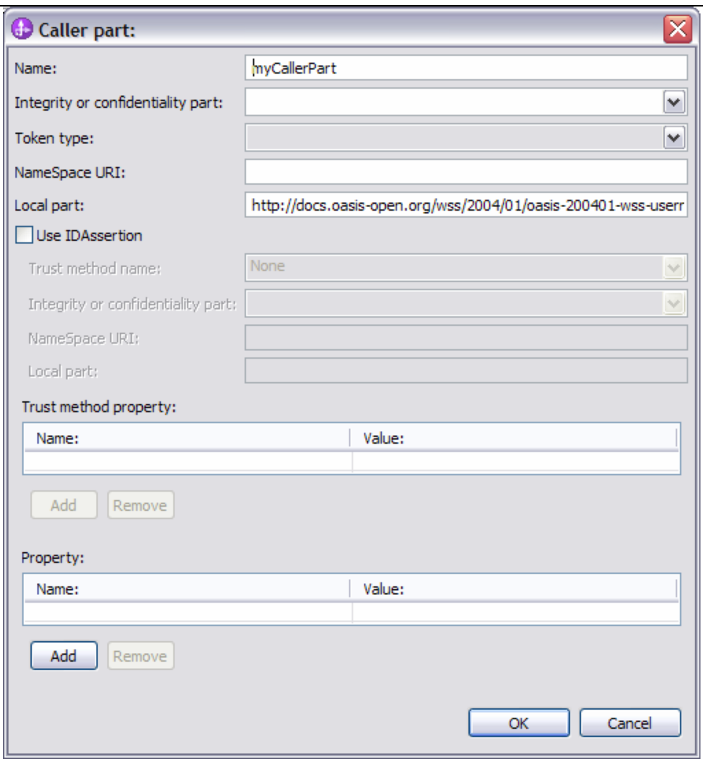
WAS as service provider expects an asserted Username Token from the client. Therefore, we have to configure the deployment descriptor of WAS accordingly.

1.	To create a *Request Consumer Security Token o*pen the webservice.xml and goto the tab Extension

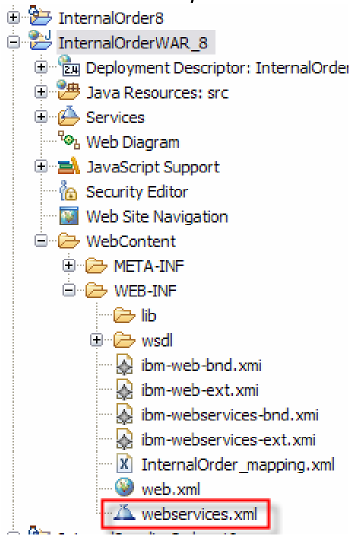
	 InternalOrder8 InternalOrderWAR_8 Deployment Descriptor: InternalOrde Java Resources: src Services Web Diagram JavaScript Support Security Editor Web Site Navigation WebContent META-INF WEB-INF lib wsdl ibm-web-bnd.xmi ibm-web-ext.xmi ibm-webservices-bnd.xmi ibm-webservices-ext.xmi InternalOrder_mapping.xml web.xml webservices.xml
2.	Open *Request Consumer Service Configuration Details > Required Security Token* and Click *Add*  **Web Services Security Extensions (J2EE version: 1.4)** Editor for Web Services security extensions (ibm-webservices-ext.xmi). **Web Service Extension** **Web Service Description Extension** Web service description extensions. Add Remove **Port Component Binding** Port component bindings of the selected Web service description extension. Add Edit Remove **Server Service Configuration** **Request Consumer Service Configu** Request Consumer service of the selected server **Required Integrity** **Required Confidentiality** **Required Security Token** Add Edit Remove **Caller Part** **Add Timestamp** **Property** **Response Generator Service Confi** Web Services | Port Components | Handlers | Extensions | Bindings | Binding Configurations
3.	Name the token for example *AssertedUsernameToken* Select as Token type *Username Token* Local Part is set automatically when choosing *Username Token* Usage type is *Required* Click *OK*  Required Security Token Name: AssertedUsernameToken Token type: Username Token NameSpace URI: Local part: http://docs.oasis-open.org/wss/2004/01/oasis-2004 Usage type: Required OK Cancel
4.	The token is now available in the Required Security Token section

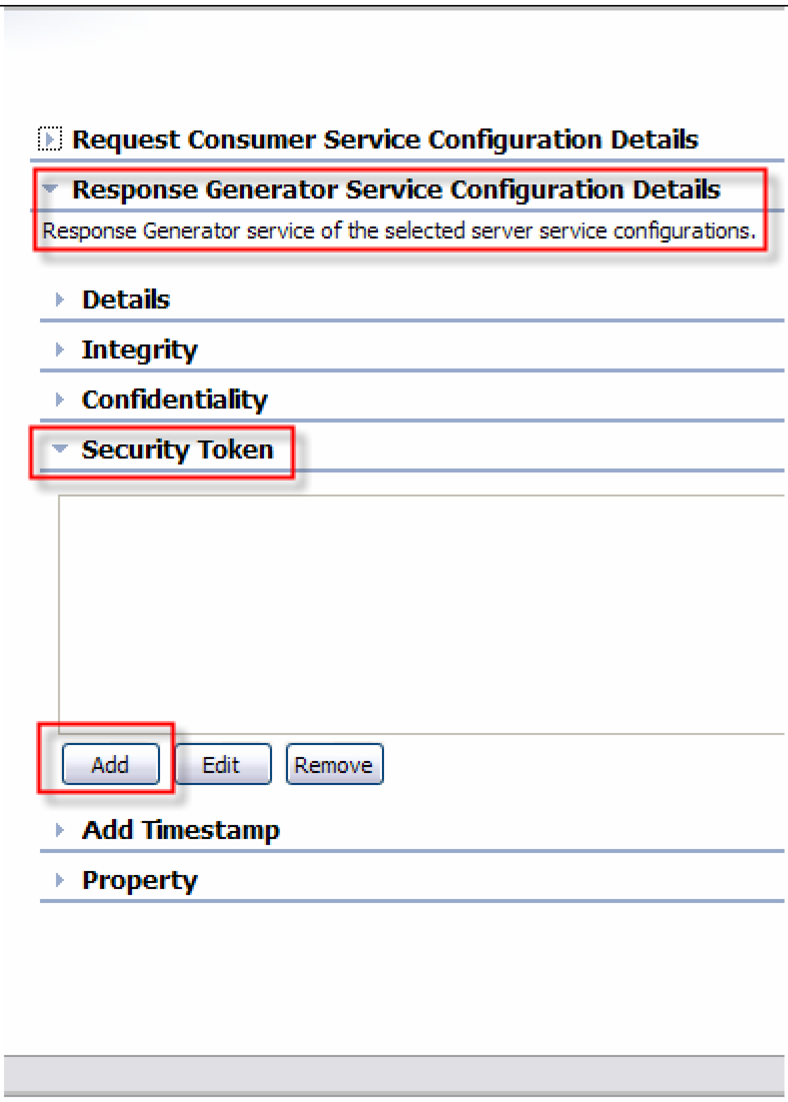
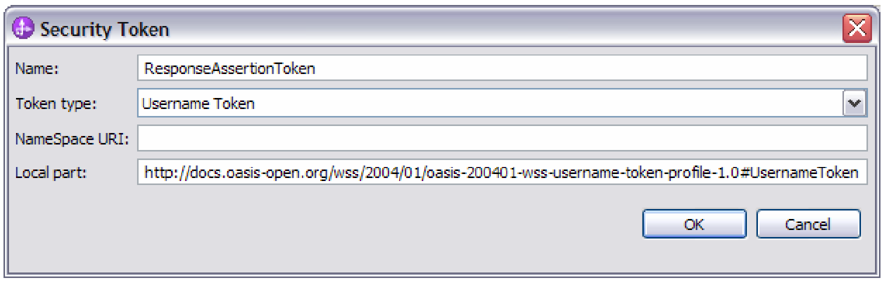
	 Required Integrity Required Confidentiality Required Security Token AssertedUsernameToken Add Edit Remove
5.	Open the *Binding Configurations Tab* Open *Request Consumer Binding Configuration Details > Token Consumer* Click *Add*  ions (J2EE version: 1.4) ervices-bnd.xmi). scription binding. Request consumer binding of the selected port Trust Anchor Certificate Store List Token Consumer Token consumer. Add Edit Remove Key Locators Key Information Signing Information Encryption Information Property Response Generator Binding Con Parameters Bindings Binding Configurations
6.	• In the Token Consumer dialog box enter a consumer name, e.g AssertedTokenConsumer • Select as Token consumer class ***com.ibm.wsspi.wssecurity.token.IDAssertionUsernameTokenConsumer*** • As Security Token select *AssertedUsernameToken* • Check *Use value type* • Select as Value type: *Username Token* • Local Part is generated automatically • Check *Use.jaas.config* • Enter as jaas.config name: *system.wssecurity.IDAssertionUsernameToken* by selecting the *IDAssertionUsernameToken* we define that we just need the user ID, and no password • Click *OK*

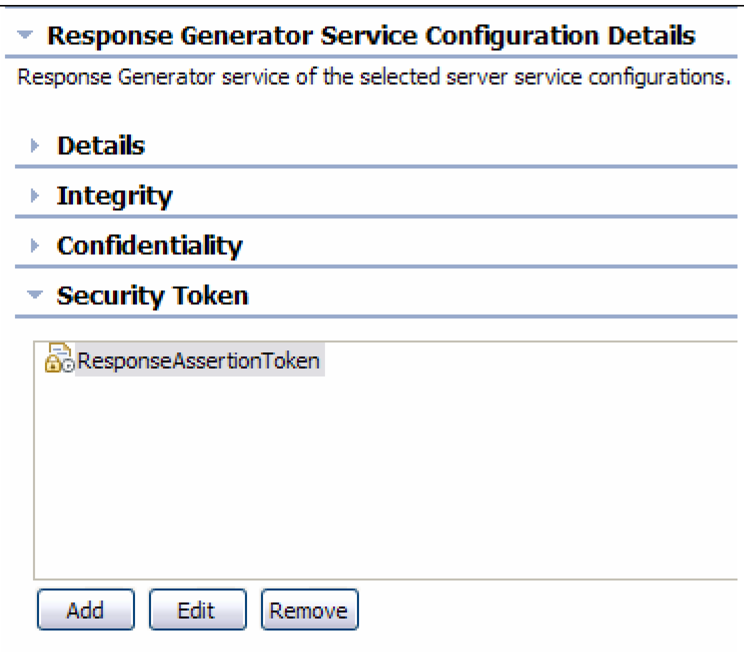
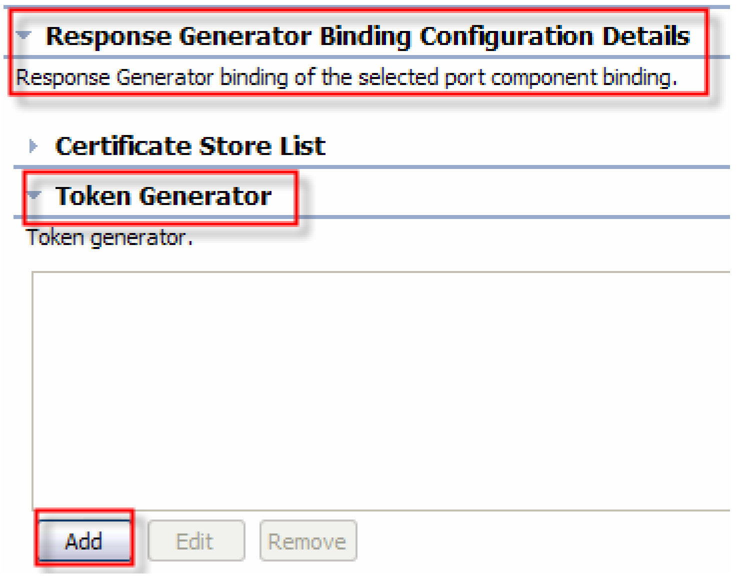
	 Token Consumer Token consumer name: AssertedTokenConsumer Token consumer class: com.ibm.wsspi.wssecurity.token.IDAssertionUsernameTokenConsumer Security token: AssertedUsernameToken [✓] Use value type Value type: Username Token Local part: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken NameSpace URI: [✓] Use jaas.config jaas.config name: system.wssecurity.IDAssertionUsernameToken jaas.config property: Name: Value: Add Remove [ ] Use trusted ID evaluator Trusted ID evaluator class: Trusted ID evaluator property: Name: Value: Add Remove [ ] Use trusted ID evaluator reference Trusted ID evaluator reference: Property: Name: Value: Add Remove [ ] Use certificate path settings Certificate path reference: Trust anchor reference: Certificate store reference: Trust any certificate OK Cancel
7.	Open the Extension Tab Open Request Consumer Binding Configuration Details > Caller Part Click Add

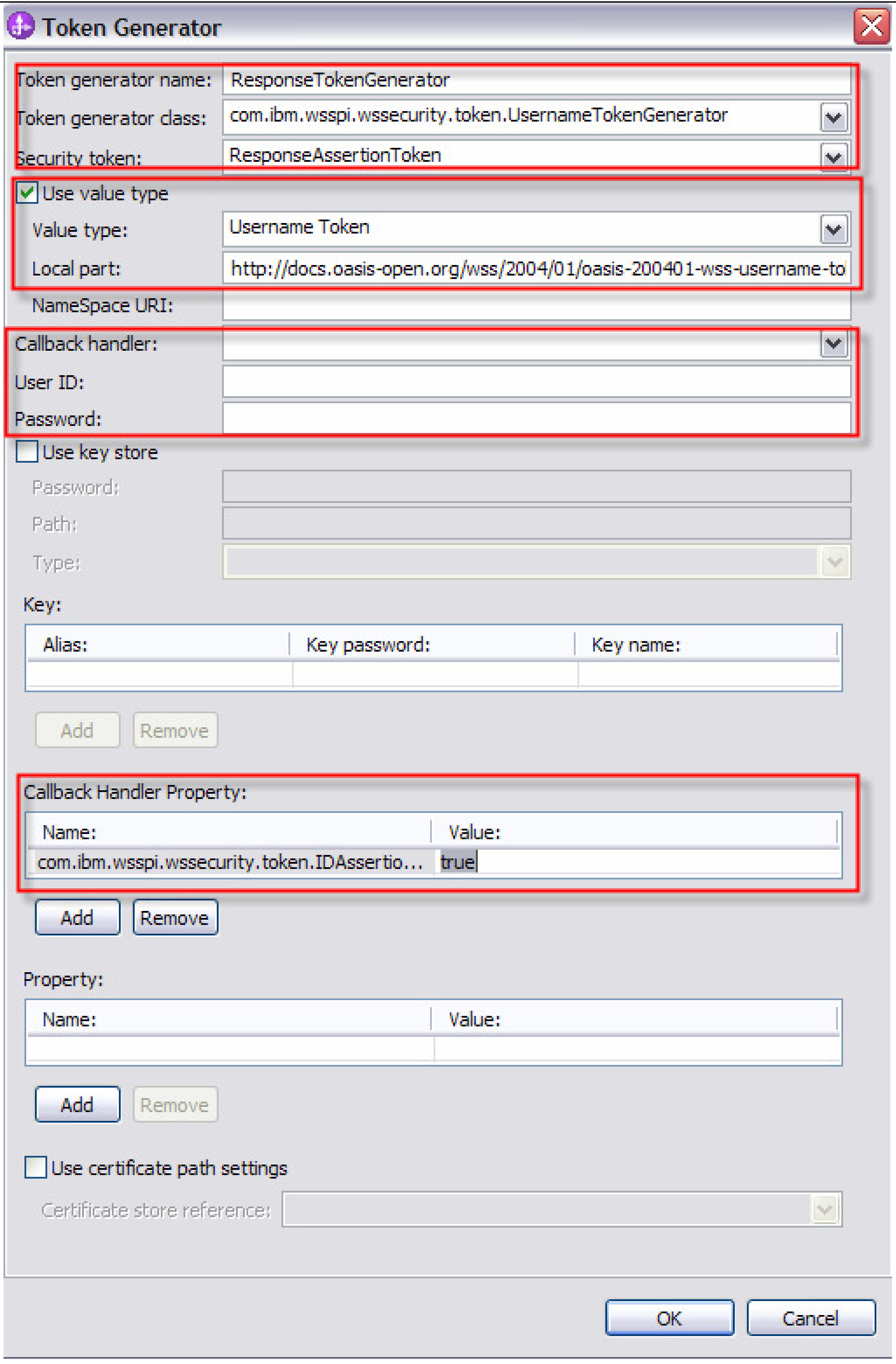
	 Caller part: Name: myCallerPart Integrity or confidentiality part: Token type: NameSpace URI: Local part: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-userr Use IDAssertion Trust method name: None Integrity or confidentiality part: NameSpace URI: Local part: Trust method property: Name: Value: Property: Name: Value:

7.1.6.8 Step 7 – WAS: Create a Token Response Generator

1.	To create a *Response Generator Security Token* open the webservice.xml and goto the tab Extension  InternalOrder8 InternalOrderWAR_8 Deployment Descriptor: InternalOrde Java Resources: src Services Web Diagram JavaScript Support Security Editor Web Site Navigation WebContent META-INF WEB-INF lib wsdl ibm-web-bnd.xmi ibm-web-ext.xmi ibm-webservices-bnd.xmi ibm-webservices-ext.xmi InternalOrder_mapping.xml web.xml webservices.xml
2.	Open *Request Consumer Service Configuration Details > Required Security Token* and Click *Add*

	 Request Consumer Service Configuration Details Response Generator Service Configuration Details Response Generator service of the selected server service configurations. Details Integrity Confidentiality Security Token Add Edit Remove Add Timestamp Property
3.	Name the token for example Response*AssertionToken* Select as Token type *Username Token* Local Part is set automatically when choosing *Username Token* Click *OK*  Security Token Name: ResponseAssertionToken Token type: Username Token NameSpace URI: Local part: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken
4.	The token is now available in the Required Security Token section

	 Response Generator Service Configuration Details Response Generator service of the selected server service configurations. Details Integrity Confidentiality Security Token ResponseAssertionToken
5.	Open the *Binding Configurations Tab* Open *Response Generator Binding Configuration Details* > *Token Consumer* Click *Add*  Response Generator Binding Configuration Details Response Generator binding of the selected port component binding. Certificate Store List Token Generator Token generator.
6.	Token Generator Name: ResponseTokenGenerator Token generator class: com.ibm.wsspi.wssecurity.token.UsernameTokenGenerator Security Token: ResponseAssertionToken Use value type: Checked Callback handler: Blank UserID Blank Password Blank Callback handler Properties: com.ibm.wsspi.wssecurity.token.IDAssertion.isUsed=true com.ibm.wsspi.wssecurity.token.IDAssertion.useRunAsIdentity=true *You'll need to click the Add button to add a row and then select name and value fields to type over.*


Token Generator

Token generator name: ResponseTokenGenerator

Token generator class: com.ibm.wsspi.wssecurity.token.UsernameTokenGenerator

Security token: ResponseAssertionToken

☑ Use value type

Value type: Username Token

Local part: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-to

NameSpace URI:

Callback handler:

User ID:

Password:

☐ Use key store

Password:

Path:

Type:

Key:

Alias:	Key password:	Key name:

Callback Handler Property:

Name:	Value:
com.ibm.wsspi.wssecurity.token.IDAssertio...	true

Property:

Name:	Value:

☐ Use certificate path settings

Certificate store reference:

7.1.6.9 Step 8 – WAS: Deploy WAS application

Detailed deployment steps are described in the appendix.

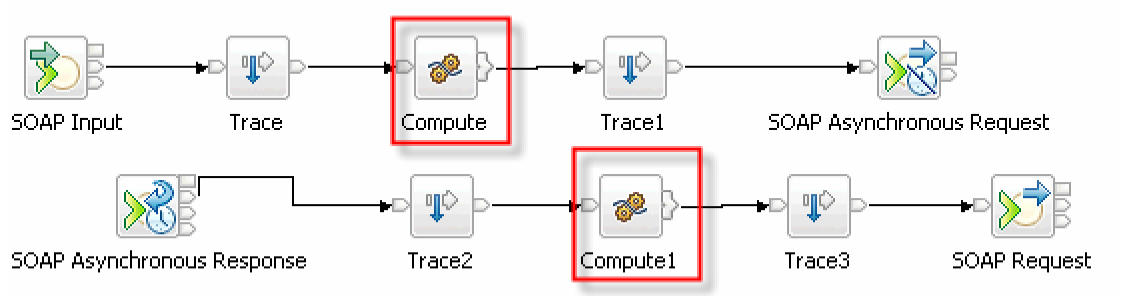
7.1.6.10 Step 9 – WMB: Develop MessageFlow

Message Broker makes an asynchronous SOAP Request to WAS.

The objective is to build a SOAP Message like below, which is sent to WAS. To achieve this, we copy just the WS-Security Header from the input node to the output node.

```
<soapenv:Envelope xmlns:int="http://Showcase/InternalOrder" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <wsse:Security  soapenv:mustUnderstand="1"  xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <wsse:UsernameToken  wsu:Id="UsernameToken-20140850"  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
        <wsse:Username>admin</wsse:Username>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body>
    <int:execute>
      <input1>
        <!--Optional:-->
        <partNumber>3</partNumber>
        <!--Optional:-->
        <partCount>1</partCount>
        <!--Optional:-->
        <repositoryId>1</repositoryId>
        <!--Optional:-->
        <clientOrderId>1</clientOrderId>
      </input1>
    </int:execute>
  </soapenv:Body>
</soapenv:Envelope>
```

In the Compute nodes for the request and response flows copy the SOAP input Header into the SOAP output header



```
--COPY SOAP UsernameToken
SET OutputRoot.SOAP.Header = InputRoot.SOAP.Header;
```

7.1.6.11 Step 10: Deploy MessageFlow

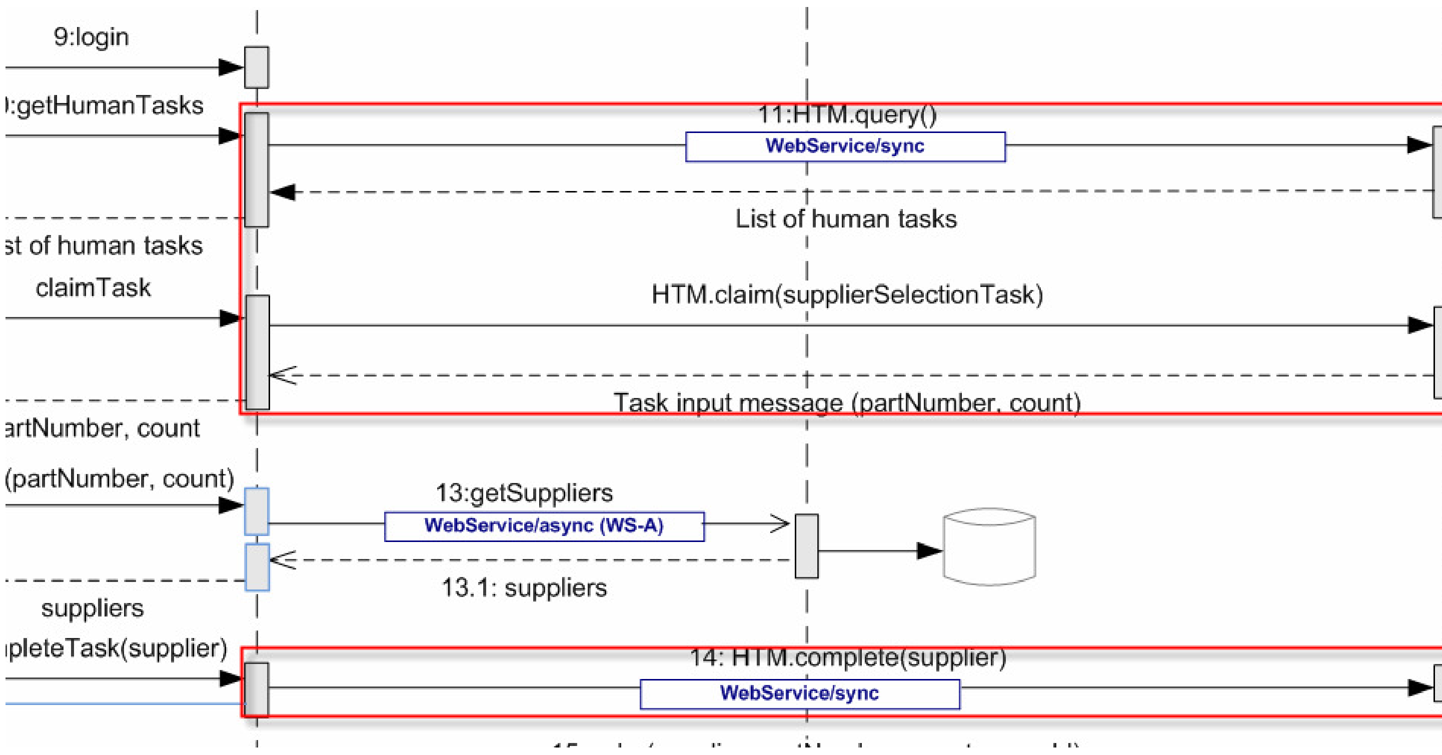
Detailed deployment steps are not described

7.1.7 Interaction 11 and 14: Human Task – get Supplier – WAS to WPS

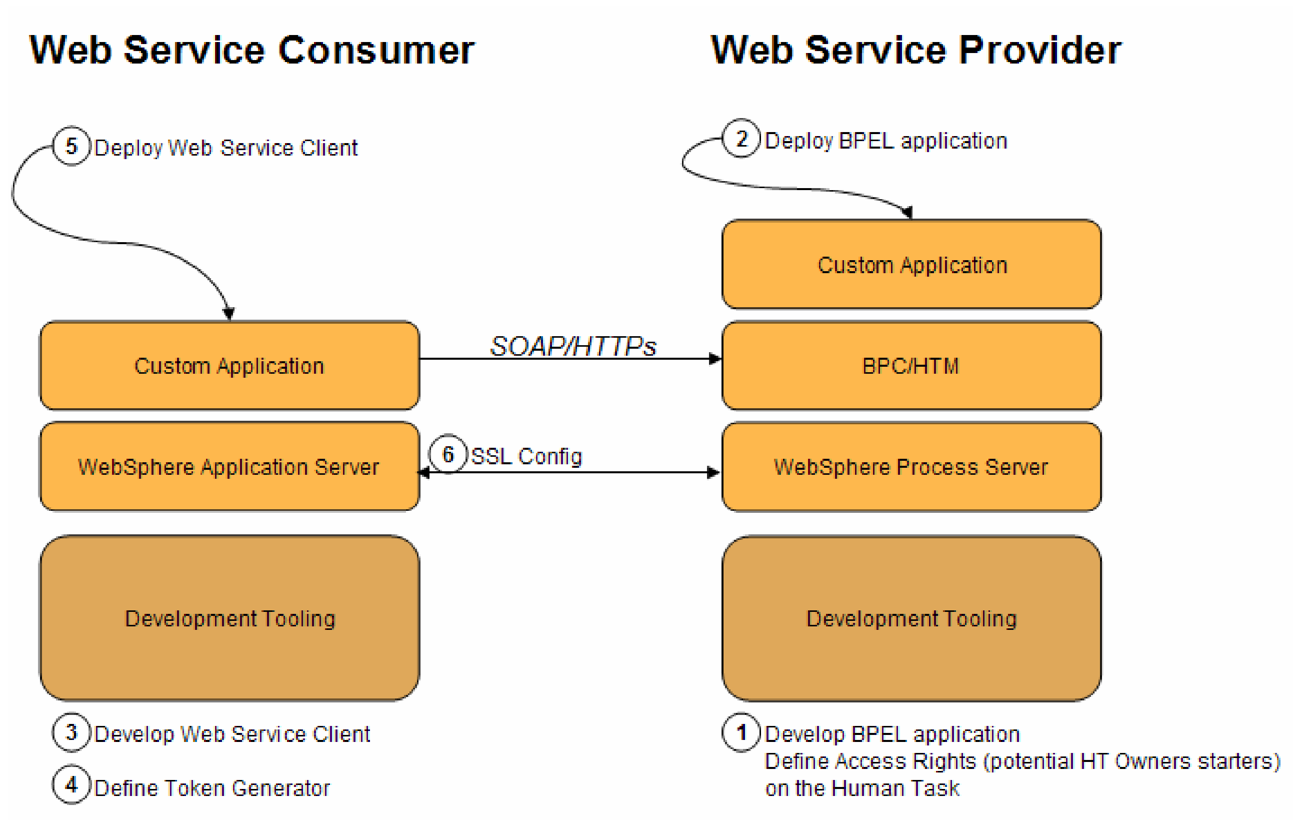
This scenario shows how to propagate the user identity via LTPA from WAS to WPS with the HTM Web Service API.

Client Application	Server Application
HumanTaskInterface (WAS)	ShowcaseApp (HTM Web Service API)

The figure below shows the relevant part in the sequence diagram:



The figure below shows the high-level implementation and configuration steps that have to be performed. Find details of the steps in the next sections.


Web Service Consumer
Web Service Provider
5 Deploy Web Service Client
2 Deploy BPEL application
Custom Application
SOAP/HTTPs
Custom Application
BPC/HTM
WebSphere Application Server
6 SSL Config
WebSphere Process Server
Development Tooling
Development Tooling
3 Develop Web Service Client
4 Define Token Generator
1 Develop BPEL application
Define Access Rights (potential HT Owners starters) on the Human Task

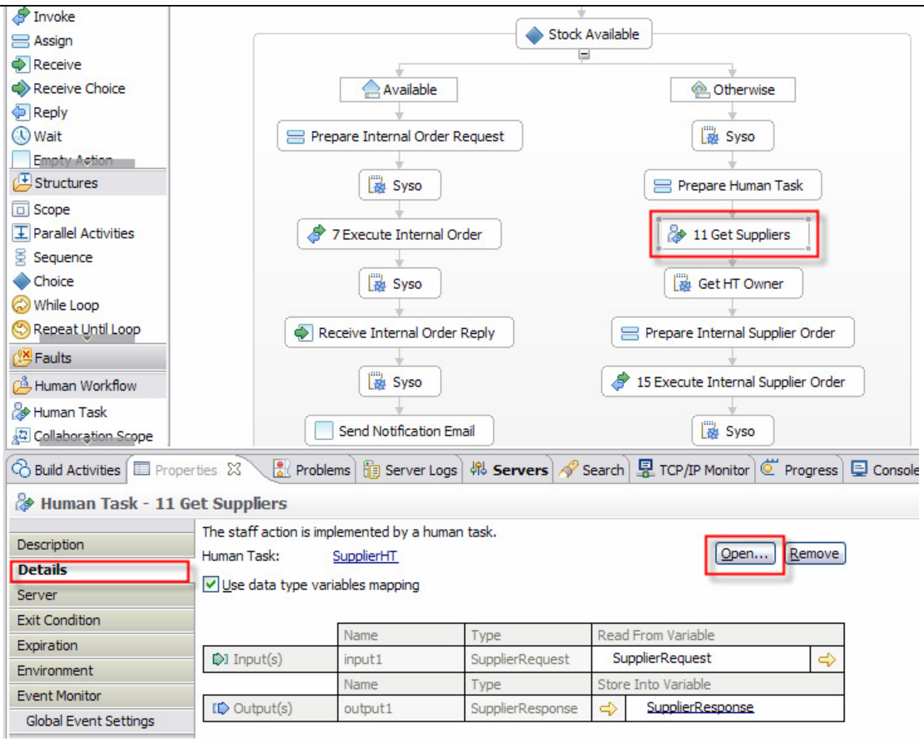
For all three calls (query, claim, complete) the generic HTM Web Service API is used.
By default, the HTM Web Service API supports LTPA- and Username Tokens. Therefore, no security-specific configuration is necessary on WPS side.

7.1.7.1 Step 1 – WPS: Develop the BPEL application and define the potential Human Task Owners

Detailed implementation steps, which are not security relevant, are not described. Refer to the WID artefacts to see how the BPEL application is developed.

To define potential HT owners follow the next steps:

	Open the Human task



Human Task - 11 Get Suppliers

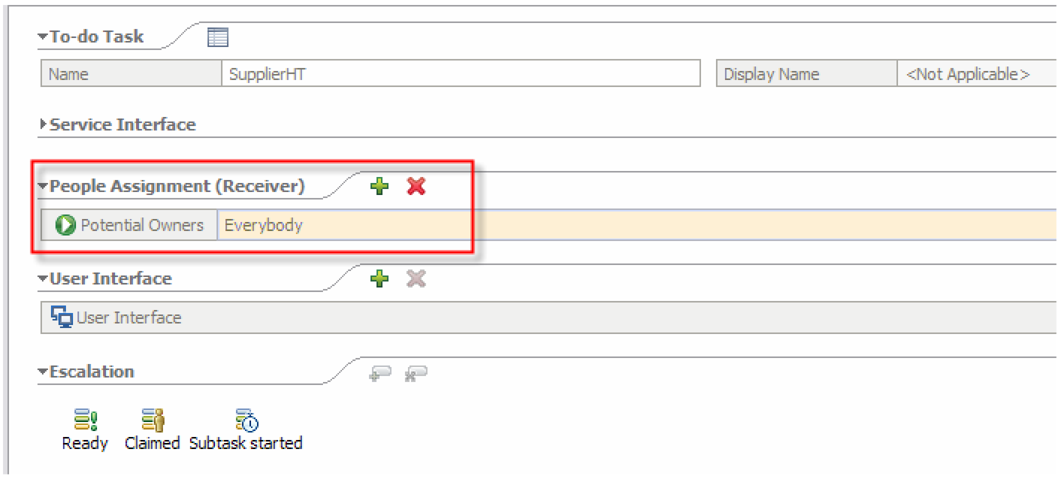
The staff action is implemented by a human task.

Human Task: SupplierHT

☑ Use data type variables mapping

	Name	Type	Read From Variable
Input(s)	input1	SupplierRequest	SupplierRequest
	Name	Type	Store Into Variable
Output(s)	output1	SupplierResponse	SupplierResponse

In the Task you can define Potential Owners. In our case, everybody who is authenticated can claim the task.



Name	SupplierHT
Display Name	<Not Applicable>
Potential Owners	Everybody

7.1.7.2 Step 2 – WPS: Deploy the BPEL application

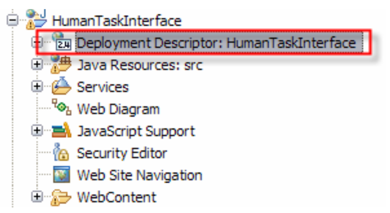
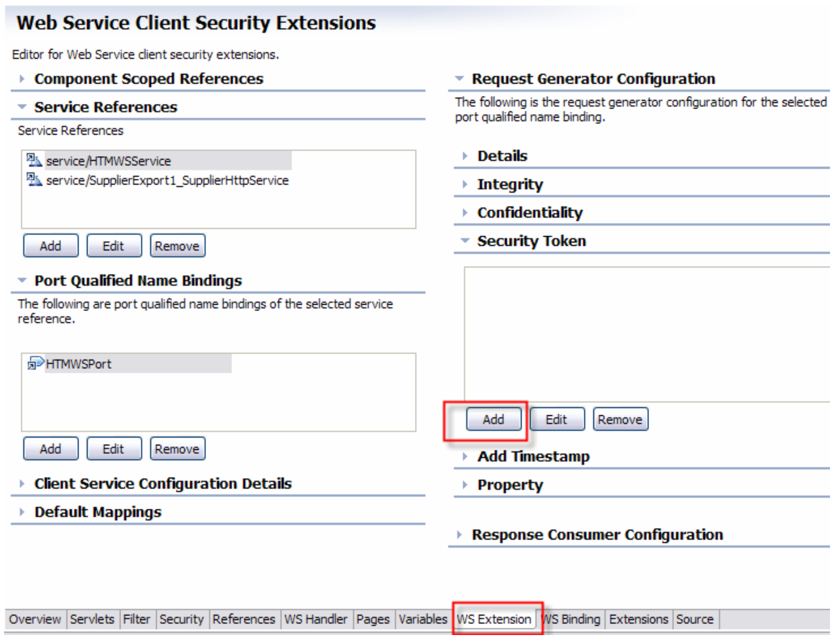
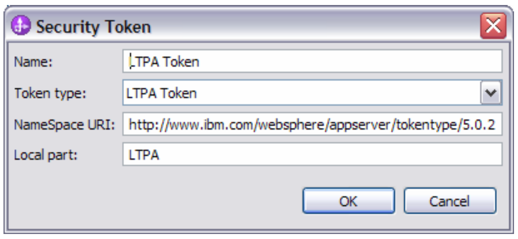
Detailed deployment steps are described in the appendix.

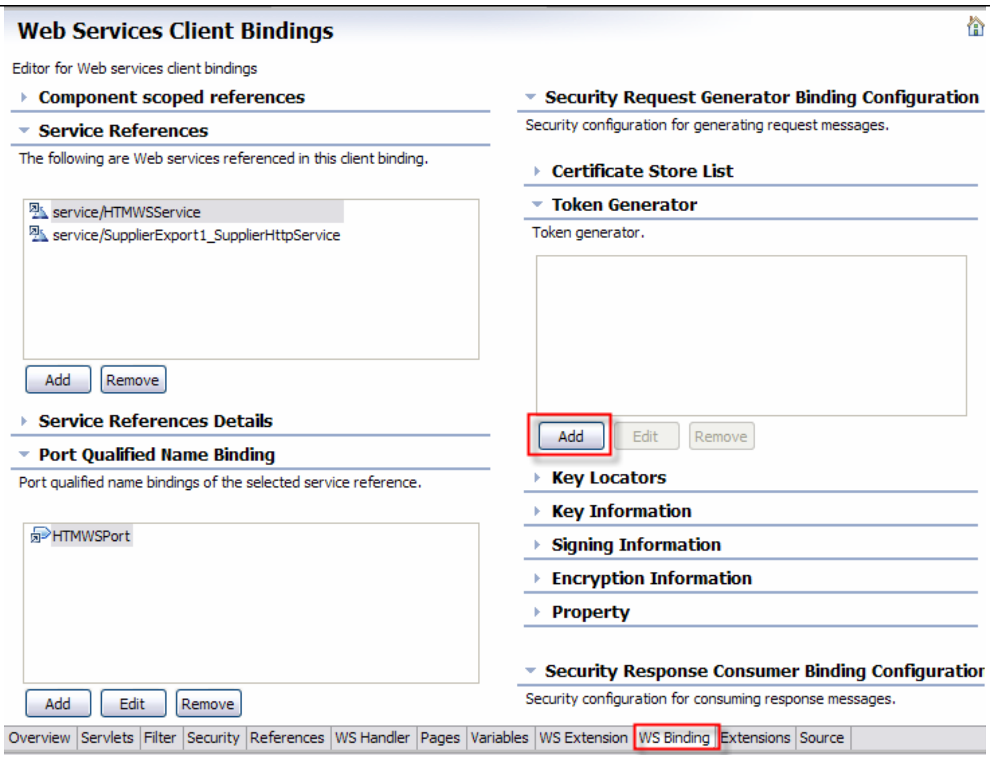
7.1.7.3 Step 3 – WAS: Develop the Web Service client

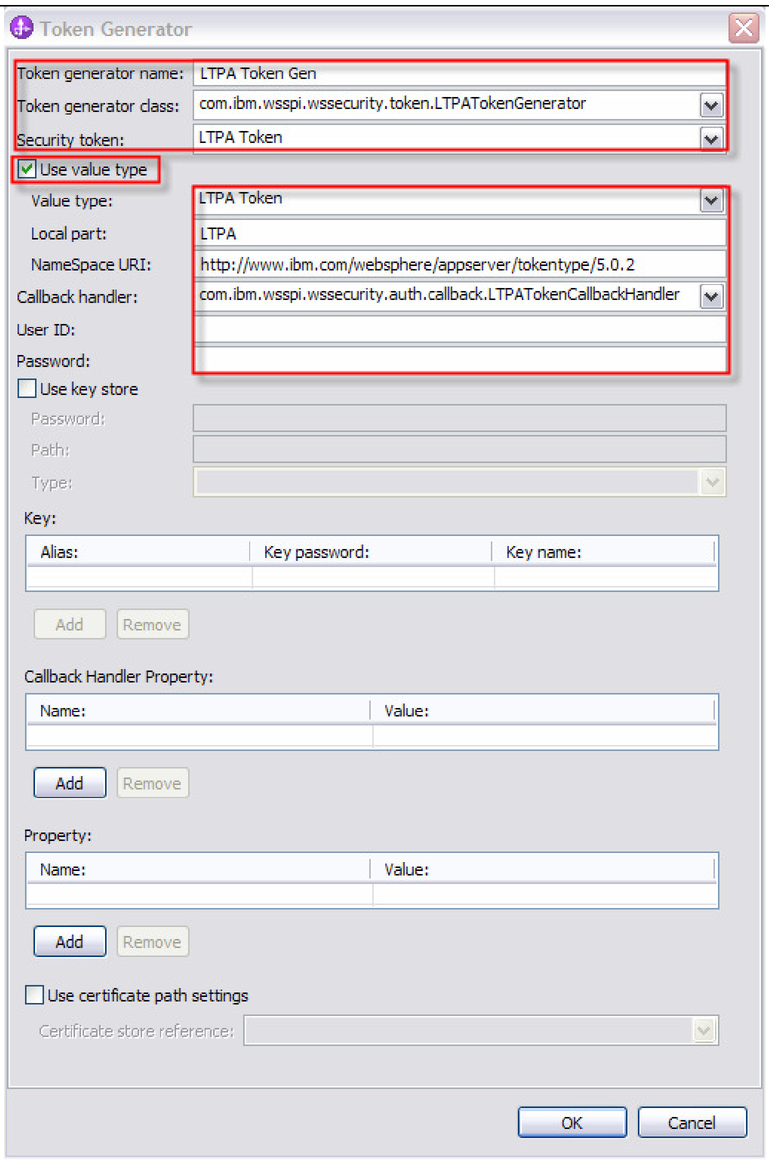
Detailed implementation steps, which are not security relevant, are not described. Refer to the WID artefacts to see how the BPEL application is developed.

7.1.7.4 Step 4 – WAS: Define the Token Generator – JAX-RPC

On the client side (WAS) a Security Token and a Token Generator must be configured. The next steps describe how to configure the client application to send an LTPA Token.

	Open the deployment descriptor on the client application  HumanTaskInterface Deployment Descriptor: HumanTaskInterface Java Resources: src Services Web Diagram JavaScript Support Security Editor Web Site Navigation WebContent
	In the deployment descriptor open the WS Extension tab, and add a new Security Token  Web Service Client Security Extensions Editor for Web Service client security extensions. Component Scoped References Service References Service References service/HTMWSService service/SupplierExport1_SupplierHttpService Add Edit Remove Port Qualified Name Bindings The following are port qualified name bindings of the selected service reference. HTMWSPort Add Edit Remove Client Service Configuration Details Default Mappings Request Generator Configuration The following is the request generator configuration for the selected port qualified name binding. Details Integrity Confidentiality Security Token Add Edit Remove Add Timestamp Property Response Consumer Configuration Overview Servlets Filter Security References WS Handler Pages Variables WS Extension WS Binding Extensions Source
	Add a new LTPA token Security Token Name: LTPA Token Token type: LTPA Token NameSpace URI: http://www.ibm.com/websphere/appserver/tokentype/5.0.2 Local part: LTPA
	Go to the *WS Binding* tab and add a new Token Generator

	 **Web Services Client Bindings** Editor for Web services client bindings Component scoped references Service References The following are Web services referenced in this client binding. service/HTMWSService service/SupplierExport1_SupplierHttpService Service References Details Port Qualified Name Binding Port qualified name bindings of the selected service reference. HTMWSPort Security Request Generator Binding Configuration Security configuration for generating request messages. Certificate Store List Token Generator Token generator. Key Locators Key Information Signing Information Encryption Information Property Security Response Consumer Binding Configuratior Security configuration for consuming response messages. Overview | Servlets | Filter | Security | References | WS Handler | Pages | Variables | WS Extension | WS Binding | Extensions | Source
	Name the Token generator for example *LTPA Token Gen* Select as class *com.ibm.wsspi.wssecurity.token.LTPATokenGenerator* Select as Security Token the *LTPA Token* Enable *Use value type* Select as Value type *LTPA Token*


Token Generator

Token generator name: LTPA Token Gen
Token generator class: com.ibm.wsspi.wssecurity.token.LTPATokenGenerator
Security token: LTPA Token
[✓] Use value type
Value type: LTPA Token
Local part: LTPA
NameSpace URI: http://www.ibm.com/websphere/appserver/tokentype/5.0.2
Callback handler: com.ibm.wsspi.wssecurity.auth.callback.LTPATokenCallbackHandler
User ID:
Password:
[ ] Use key store
Password:
Path:
Type:
Key:

Alias:	Key password:	Key name:

Add Remove

Callback Handler Property:

Name:	Value:

Add Remove

Property:

Name:	Value:

Add Remove

[ ] Use certificate path settings
Certificate store reference:

OK Cancel

7.1.7.5 Step 4 – WAS: Define Token Generator – JAX-WS

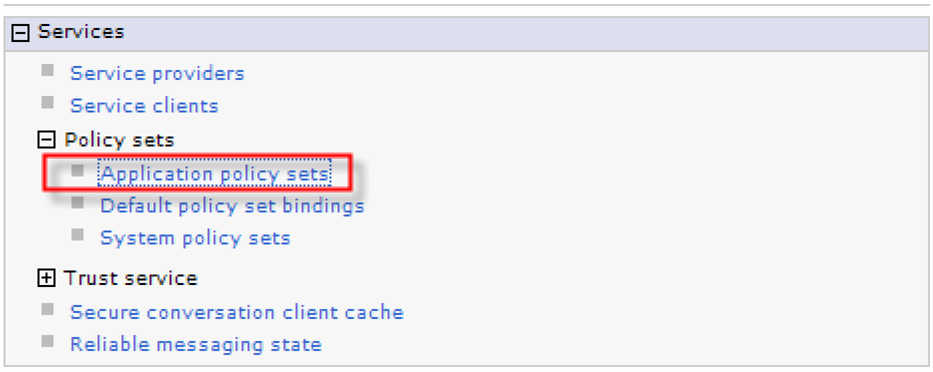
If using JAX-WS you have following two options to generate a LTPA Token:

- By coding
- By declaration

This section describes how to use the programmatic approach:

```
HTMWSService service = new HTMWSService();
htm = service.getHTMWSPort();

try {
        enhanceSecurity(htm,
        com.ibm.websphere.security.auth.WSSubject.getCallerPrincipal(), "");

        } catch (WSSException e1) {
            e1.printStackTrace();
        }
```

```
...

private void enhanceSecurity(HTMIF port, String user, String password) throws
WSSException {
        BindingProvider binding = (BindingProvider) port;
        Map requestContext = binding.getRequestContext();

        WSSFactory wssFactory = WSSFactory.getInstance();
        WSSGenerationContext genContext = wssFactory.newWSSGenerationContext();
        //UNTGenerateCallbackHandler untCallbackHandler = new
        UNTGenerateCallbackHandler(user, password, true, true);
        LTPAGenerateCallbackHandler   ltpaCallbackHandler = new
        LTPAGenerateCallbackHandler(user, null);

        SecurityToken secToken = wssFactory.newSecurityToken(LTPAToken.class,
        ltpaCallbackHandler);


        genContext.add(secToken);
        genContext.process(requestContext);
    }
```

	In the administrative console, select Policy sets > Application policy sets  Services Service providers Service clients Policy sets Application policy sets Default policy set bindings System policy sets Trust service Secure conversation client cache Reliable messaging state

7.1.7.6 Step 5 – WPS: Deploy WAS application

Detailed deployment steps are described in the appendix.

7.1.7.7 Step 6 – SSL configuration between WPS and WAS

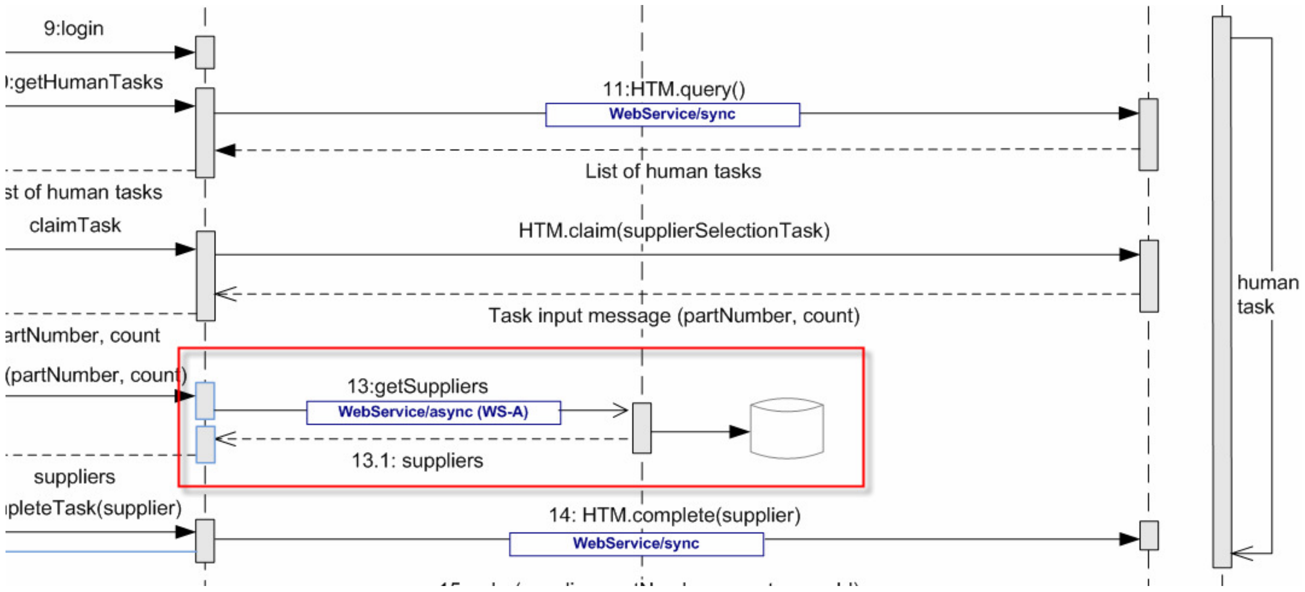
Refer to chapter “SSL between WPS and WAS”

7.1.8 Interaction 13: Web Service Addressing between WAS and WMB

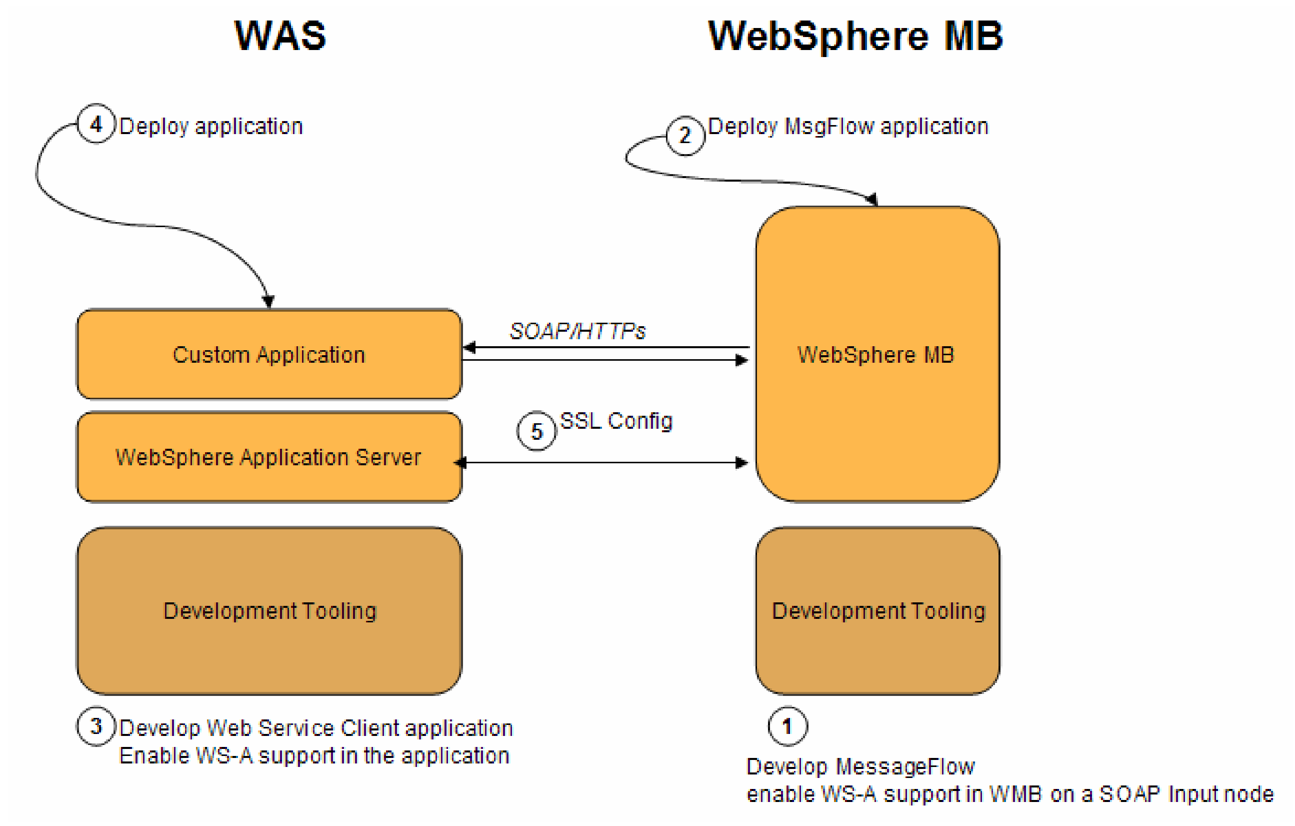
This section describes how to set up WS-A between WAS and WMB. HTTPs will be used as Transport Level Security. Identity propagation will be done using Username Tokens (w/o password).

Client Application	Server Application
HumanTaskInterface (WAS)	Supplier.msgflow (WMB)

The figure below shows the relevant part in the sequence diagram:



The figure below shows the high-level implementation and configuration steps that have to be performed. Find details of the steps in the next sections.


WAS
WebSphere MB
4 Deploy application
2 Deploy MsgFlow application
Custom Application
SOAP/HTTPs
WebSphere MB
5 SSL Config
WebSphere Application Server
Development Tooling
Development Tooling
3 Develop Web Service Client application
Enable WS-A support in the application
1
Develop MessageFlow
enable WS-A support in WMB on a SOAP Input node

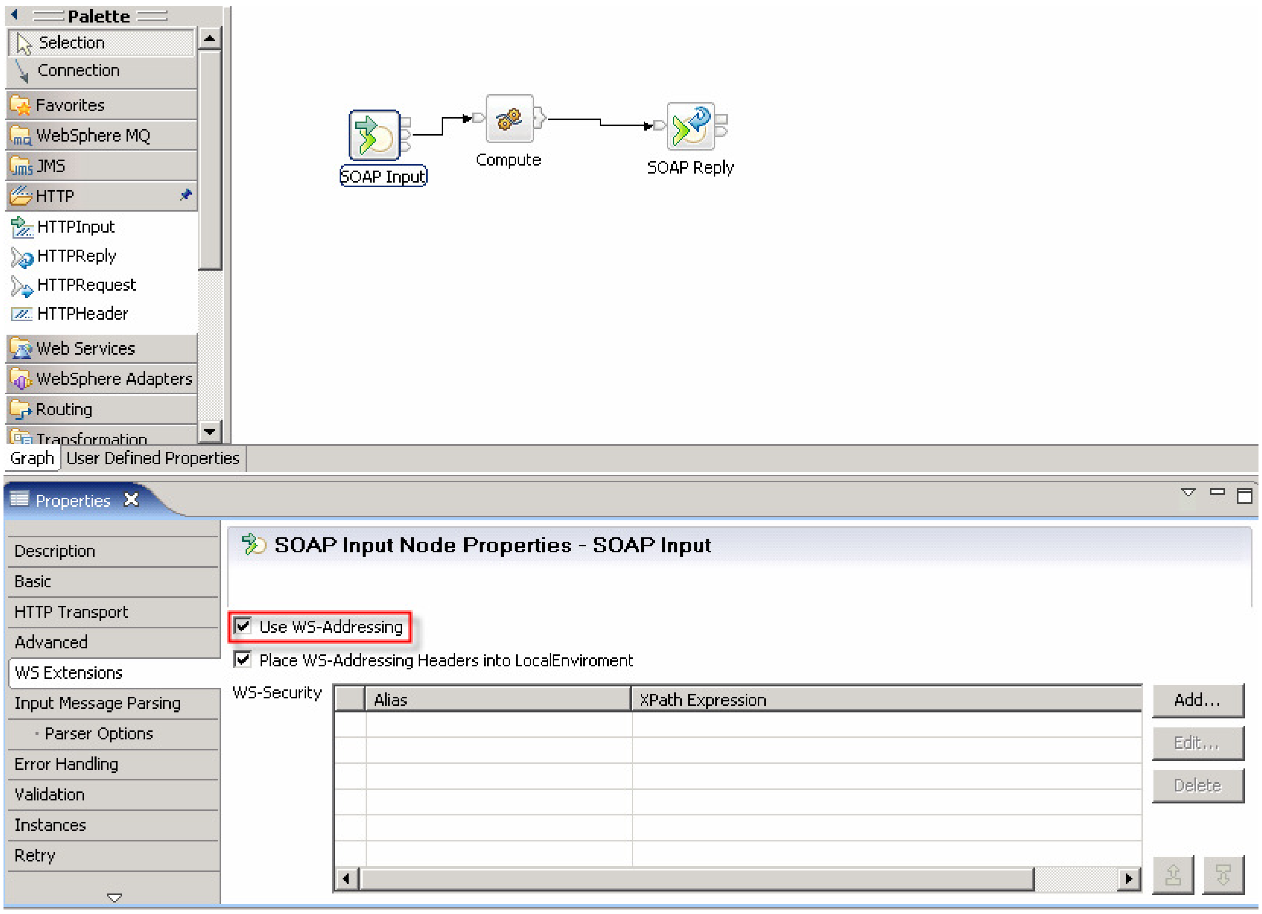
Reference Material

- Web Services Handbook for WebSphere Application Server Version 6.1, Chapter 19 "WS-Addressing and WS-Resource", SG24-7257
- DeveloperWorks Article "Driving WS-Addressing in WebSphere Application Server Version 6.1" at http://www.ibm.com/developerworks/webservices/library/ws-soa-wsawsa/

7.1.8.1 Step 1 – WMB: Implement the message flow

The SOAPInput node has a property for processing WS-Addressing information present in the incoming message called *Use WS-Addressing*. If you select this property, the WS-Addressing information is processed and the process itself is called engaging WS-Addressing. The default is that WS-Addressing is not engaged. To enable WS-A support in WMB on a SOAP Input node, open the *Properties* tab of the SOAP input node and select *WS Extensions* -> *Use WS-Addressing* (see figure below).
There is also the option to specify this property in the WSDL. Refer to http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r1m0/index.jsp?topic=/com.ibm.etools.mft.doc/ac64500_.htm.

The SOAPReply node uses WS-Addressing if WS-Addressing is engaged on the SOAPInput node that is referenced by the reply identifier of the message entering the reply node.
The SOAPReply node uses addressing information in the *Destination.SOAP.Reply.WSA* folder of the local environment to determine where to send the reply and with what Message Addressing Properties (MAPs).
If the *Destination.SOAP.Reply.WSA* does not exist, or is completely empty when inspected by the SOAPReply node, the node uses the default addressing headers that were part of the incoming message. Therefore, you do not have to propagate the local environment in the default case, and addressing still works as expected. Refer to
http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r1m0/index.jsp?topic=/com.ibm.etools.mft.doc/ac64510_.htm



SOAP Input Node Properties - SOAP Input

☑ Use WS-Addressing

☑ Place WS-Addressing Headers into LocalEnviroment

WS-Security

Alias	XPath Expression

7.1.8.2 Step 2 – WMB: Deploy the message flow

Detailed deployment steps are described in the appendix.

7.1.8.3 Step 3 – WAS: Configure and implement the WAS application

To make a WS-A call in a Java application, the following basic steps have to be performed:

1.	Generate a JAX-RPC client based on the WSDL
2.	Set the Input parameters of the SOAP request message
3.	Create the SOAP Proxy `SupplierProxy sp = new SupplierProxy();`
4.	**Create an Endpoint Reference Object** `EndpointReference epr = null;`

5.	Create **a URI object holding the endpoint**
	`URI uri = new URI(`
	`    "http://fmtc7113.boeblingen.de.ibm.com:7800/testwsdlWeb/sca/SupplierExpor`
	`t1");`

6.	**Add the URI to the EndpointReference object** `epr = com.ibm.wsspi.wsaddressing.EndpointReferenceManager`

```
                        .createEndpointReference(uri);
```

7. **Create a Stub object**

```
javax.xml.rpc.Stub stub = ((javax.xml.rpc.Stub) sp.getSupplier());
```

8. **Set the EPR object as WS-Addressing destination property to the stub**

```
stub.setProperty(
    com.ibm.websphere.wsaddressing.WSAConstants.WSADDRESSING_DESTINATION_EPR,
epr);
```

9. Do the SOAP call and get the response. No extra configuration is needed for the Web services provider on WebSphere Application Server 6.1. The application server automatically inserts WS-Addressing headers in the response.

10. Find details for 7 to 9 in the code snippet below (step 12, `GetSupplierBean.java`)

11. The WSDL binding information can specify that WS-Addressing is mandatory or optional:
<wsdl:binding name="xyzBinding" type="intf:xyzBean">
<wsaw:UsingAddressing wsdl:required="false"
xmlns:wsaw="http://www.w3.org/2006/02/addressing/wsdl"/>
......
When specifying **wsdl:required="true"** the Web service returns a fault if WS-Addressing information is missing in the client request message. If a WebSphere Application Server client sends a message without specifying addressing properties the message automatically contains the mandatory WS-Addressing information. Therefore WebSphere clients do not have to worry about WS-Addressing.

12. `GetSupplierBean.java:`

```
package com.ibm.wsapitest;

import java.rmi.RemoteException;

import Showcase.Get;
import Showcase.GetResponse;
import Showcase.SupplierProxy;
import Showcase.SupplierRequest;

import com.ibm.websphere.wsaddressing.EndpointReference;
import com.ibm.websphere.wsaddressing.EndpointReferenceCreationException;
import java.net.URI;
import java.net.URISyntaxException;

public class GetSupplierBean {

    String partNumber, partCount, supplierId;

    public String getSupplier() {

        System.out.println("create SOAP Request");
        SupplierRequest sr = new SupplierRequest();
        sr.setPartCount(1);
        sr.setPartNumber("123");
        Get g = new Get();
        g.setInput1(sr);
        GetResponse res = null;

        System.out.println("create SOAP Proxy");
        SupplierProxy sp = new SupplierProxy();

        EndpointReference epr = null;
        try {

            URI uri = new URI(

    "http://fmtc7113.boeblingen.de.ibm.com:7800/testwsdlWeb/sca/SupplierExport1");
            epr = com.ibm.wsspi.wsaddressing.EndpointReferenceManager
                        .createEndpointReference(uri);

        } catch (EndpointReferenceCreationException e) {
            // TODO Auto-generated catch block
            System.out.println("********************** Error creating erp");
            e.printStackTrace();
        } catch (URISyntaxException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        System.out.println("create stub");
        javax.xml.rpc.Stub stub = ((javax.xml.rpc.Stub) sp.getSupplier());

        System.out.println("set stub property");
        stub
                ._setProperty(

    com.ibm.websphere.wsaddressing.WSAConstants.WSADDRESSING_DESTINATION_EPR,
                        epr);

        System.out.println("do SOAP call");
        try {
            res = sp.getSupplier().get(g);
        } catch (RemoteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        String[] response = res.getOutput1();
        setSupplierId(response[0]);
        System.out.println(response[0]);
        return response[0];

    }

    public String getPartNumber() {
        return partNumber;
    }

    public void setPartNumber(String partNumber) {
        this.partNumber = partNumber;
    }

    public String getPartCount() {
        return partCount;
    }

    public void setPartCount(String partCount) {
        this.partCount = partCount;
    }

    public String getSupplierId() {
        return supplierId;
    }

    public void setSupplierId(String supplierId) {
        this.supplierId = supplierId;
    }

}
```

7.1.8.4 Step 4 – WAS: Deploy the application

Detailed deployment steps are described in the appendix.

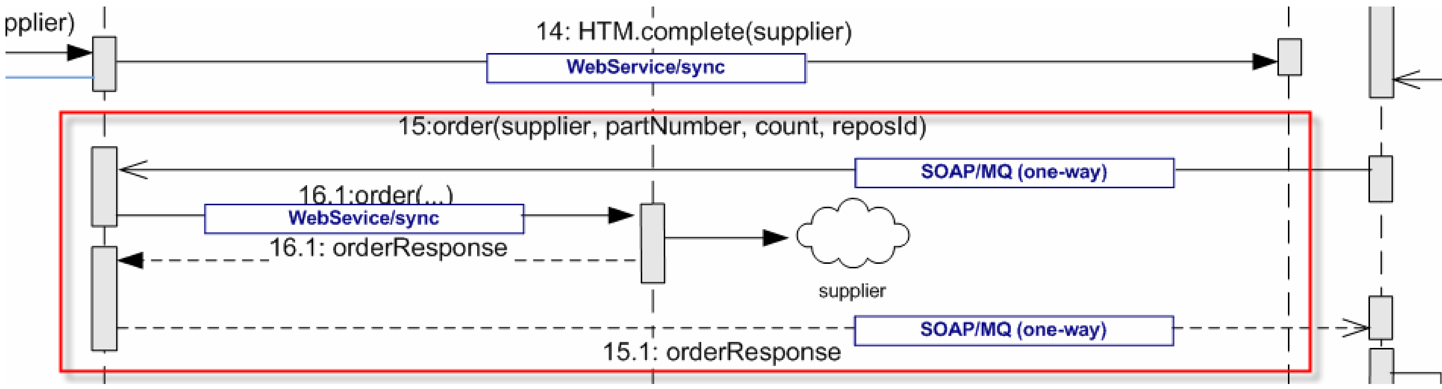
7.1.9 Interaction 15 and 16 – SOAP/MQ - Identity propagation based on HT owner of preceding activity

This step documents:

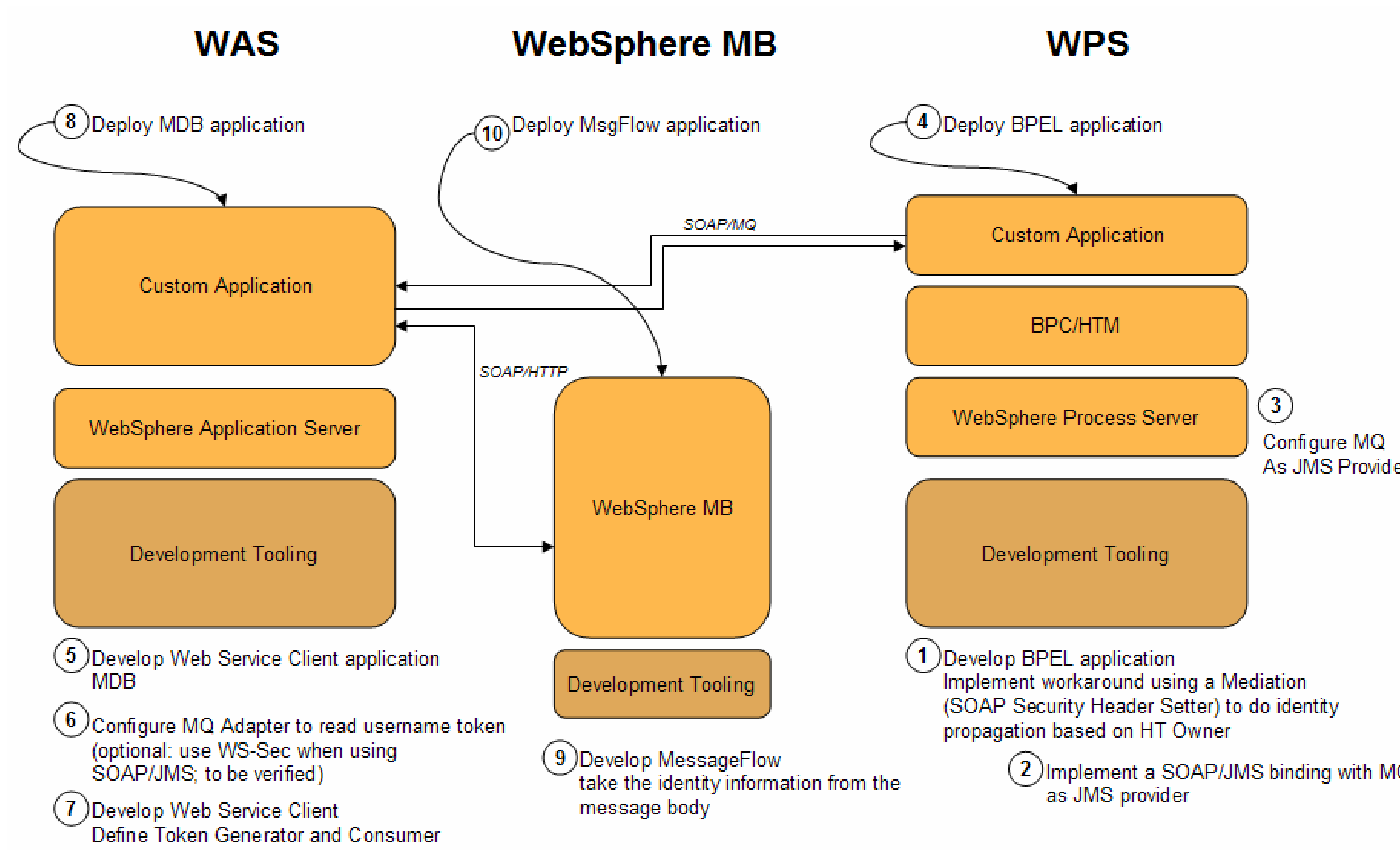
- Identity propagation not based on process starter identity but on HT owner of preceding activity
- SOAP/MQ call from WPS to WAS

	Client Application	Server Application
1	ShowcaseApp (WPS, SCA Import – SOAP/JMS Binding)	InternalSupplierOrder_16 (WAS)

The figure below shows the relevant part in the sequence diagram:

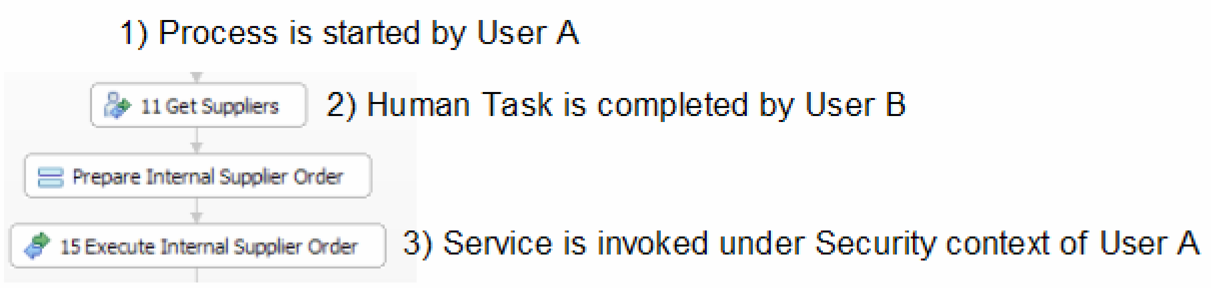


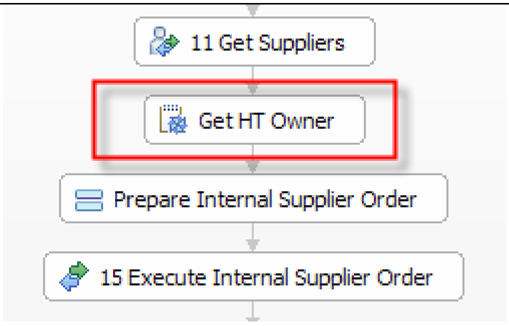
The figure below shows the high-level implementation and configuration steps that have to be performed. Find details of the steps in the next sections.


WAS
WebSphere MB
WPS
8 Deploy MDB application
10 Deploy MsgFlow application
4 Deploy BPEL application
Custom Application
SOAP/MQ
Custom Application
BPC/HTM
SOAP/HTTP
WebSphere Application Server
WebSphere MB
WebSphere Process Server
3 Configure MQ As JMS Provide
Development Tooling
Development Tooling
Development Tooling
5 Develop Web Service Client application MDB
6 Configure MQ Adapter to read username token (optional: use WS-Sec when using SOAP/JMS; to be verified)
7 Develop Web Service Client Define Token Generator and Consumer
9 Develop MessageFlow take the identity information from the message body
1 Develop BPEL application Implement workaround using a Mediation (SOAP Security Header Setter) to do identity propagation based on HT Owner
2 Implement a SOAP/JMS binding with M as JMS provider

7.1.9.1 Step 1 – WPS: Identity propagation based on HT owner of the preceding activity

This section demonstrates how to call a service with an HT owner identity of a preceding activity.

	By default, the behaviour is that a invoke is done under the security context of the process starter  1) Process is started by User A 11 Get Suppliers 2) Human Task is completed by User B Prepare Internal Supplier Order 15 Execute Internal Supplier Order 3) Service is invoked under Security context of User A
	To achieve that the service call is done by the human task owner, we have to add a Java snippet which assigns the activity owner of the human task to a BPEL variable.


11 Get Suppliers
Get HT Owner
Prepare Internal Supplier Order
15 Execute Internal Supplier Order

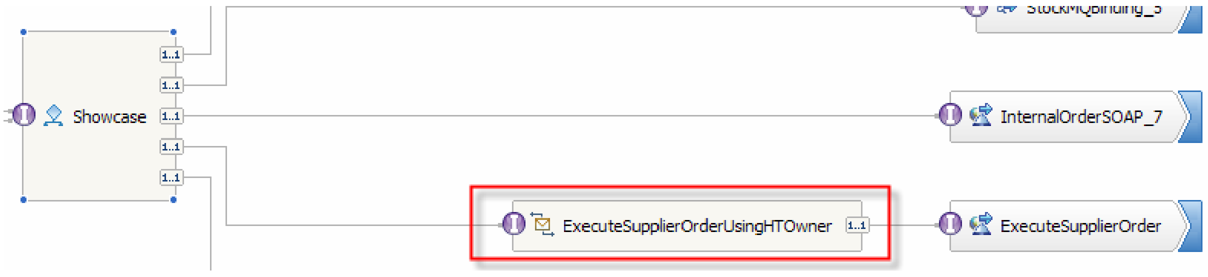
The snippet contains the following code:

```
com.ibm.bpe.api.ActivityInstanceData aid = activityInstance("GetSuppliers");
HTOwner = aid.getOwner();
```

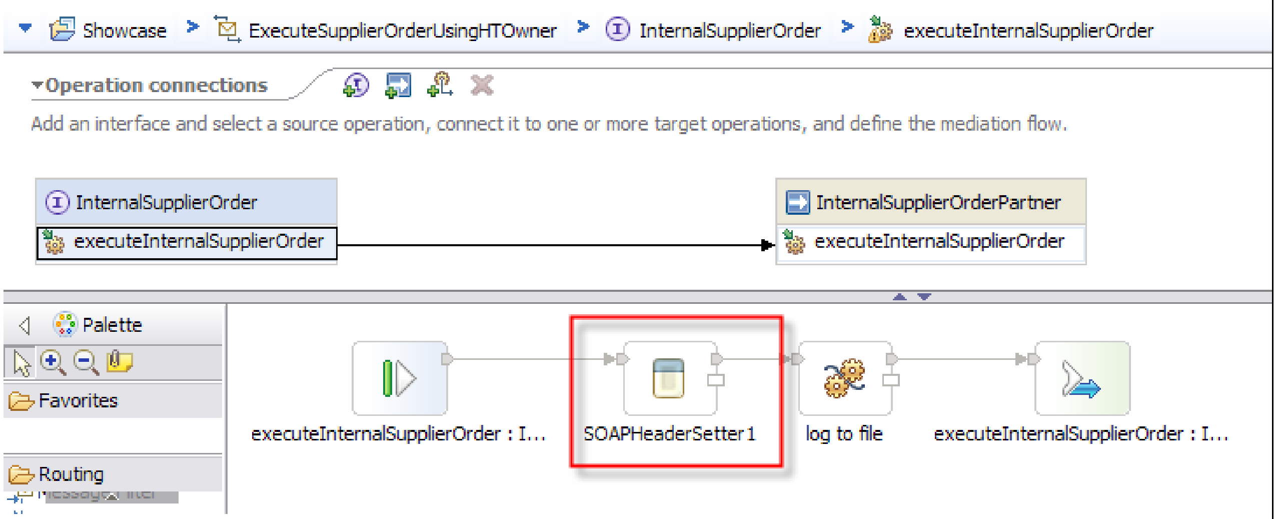
Important: This approach works only with an inline human task.

The BPEL variable with the HT owner must be added to the payload of the message we will sent to the back-end service.

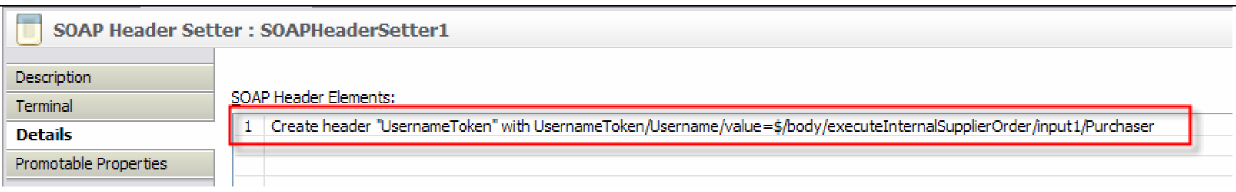
Now, the back-end service is not invoked directly by the Invoke activity. The invoke will call a mediation module. The mediation module is in between the BPEL invoke and the back-end service.



The mediation module has one main node:


Showcase
ExecuteSupplierOrderUsingHTOwner
InternalSupplierOrder
executeInternalSupplierOrder
Operation connections
Add an interface and select a source operation, connect it to one or more target operations, and define the mediation flow.
InternalSupplierOrder
executeInternalSupplierOrder
InternalSupplierOrderPartner
executeInternalSupplierOrder
Palette
Favorites
Routing
executeInternalSupplierOrder : I...
SOAPHeaderSetter1
log to file
executeInternalSupplierOrder : I...

The SOAPHeaderSetter node creates a Username Token Header element and assigns the user ID from the payload to it.

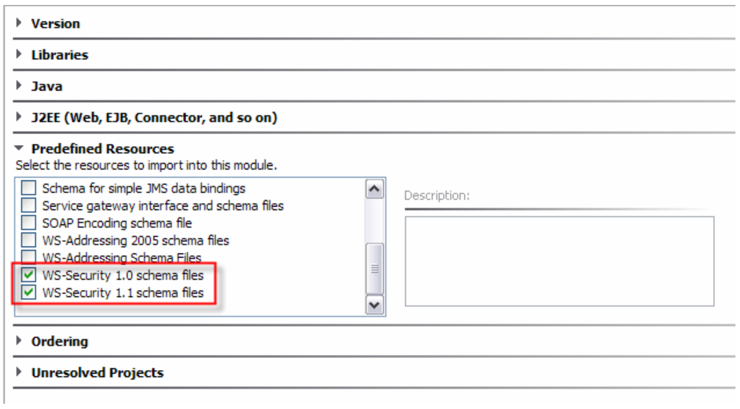


SOAP Header Setter : SOAPHeaderSetter1

SOAP Header Elements:

1 Create header "UsernameToken" with UsernameToken/Username/value=$/body/executeInternalSupplierOrder/input1/Purchaser

To create the Username Token in the security header, first of all you have to add WS-Security schema files in the dependencies editor.

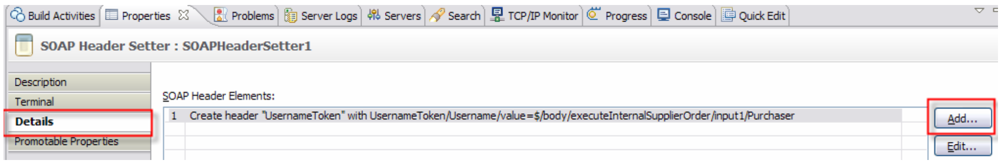


Predefined Resources

Select the resources to import into this module.

- [ ] Schema for simple JMS data bindings
- [ ] Service gateway interface and schema files
- [ ] SOAP Encoding schema file
- [ ] WS-Addressing 2005 schema files
- [ ] WS-Addressing Schema Files
- [x] WS-Security 1.0 schema files
- [x] WS-Security 1.1 schema files

Then, click the SOAPHeaderSetter node and select *Properties > Details > Add..*

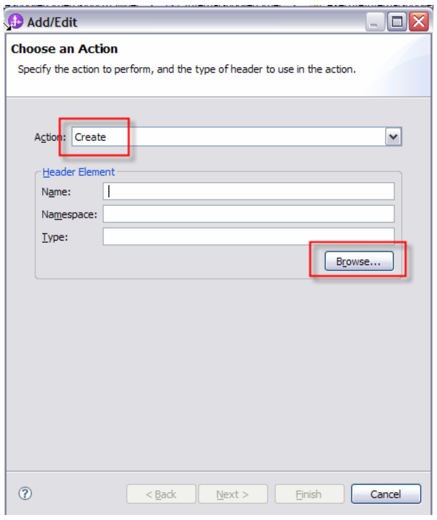


SOAP Header Setter : SOAPHeaderSetter1

SOAP Header Elements:

1 Create header "UsernameToken" with UsernameToken/Username/value=$/body/executeInternalSupplierOrder/input1/Purchaser

In the dialog box, select as Action *Create* and click *Browse…*



Add/Edit

Choose an Action

Specify the action to perform, and the type of header to use in the action.

Action: Create

Header Element

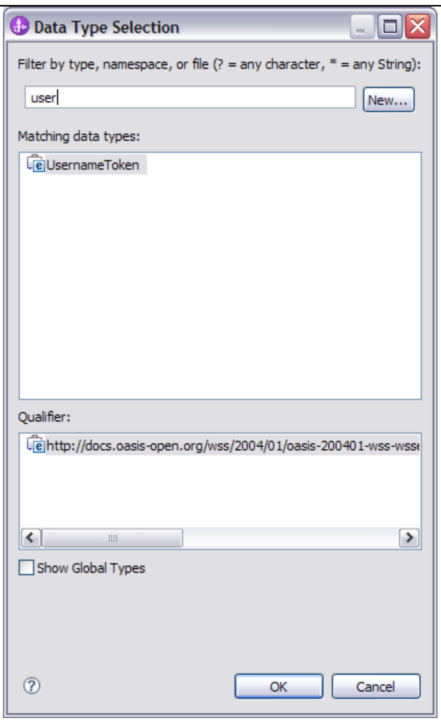
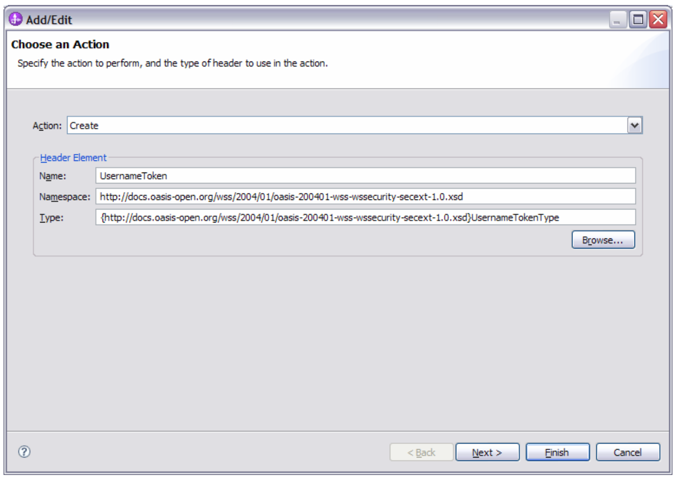
Name:

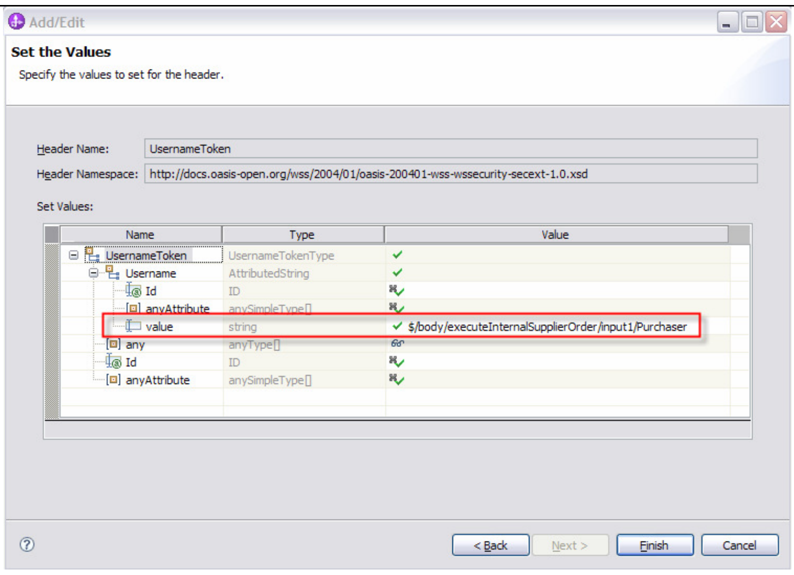
Namespace:

Type:

Select as Data Type *UsernameToken*. If you did not add the WS-Security schema file to the dependencies, you will not find the UsernameToken here.

Click *OK*

	 Data Type Selection Filter by type, namespace, or file (? = any character, * = any String): user Matching data types: UsernameToken Qualifier: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse Show Global Types
	Click *Next*  Add/Edit **Choose an Action** Specify the action to perform, and the type of header to use in the action. Action: Create Header Element Name: UsernameToken Namespace: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd Type: {http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd}UsernameTokenType
	Insert an XPATH expression where to find the user ID in the input message to the value field.

	
	Click Finish and Save

Set the Values

Specify the values to set for the header.

Header Name: UsernameToken

Header Namespace: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd

Set Values:

Name	Type	Value
UsernameToken	UsernameTokenType	✓
Username	AttributedString	✓
Id	ID	✓
anyAttribute	anySimpleType[]	✓
value	string	✓ $/body/executeInternalSupplierOrder/input1/Purchaser
any	anyType[]	
Id	ID	✓
anyAttribute	anySimpleType[]	✓

When the Service call is now done using SOAP, the token contains the user ID of the human task owner.

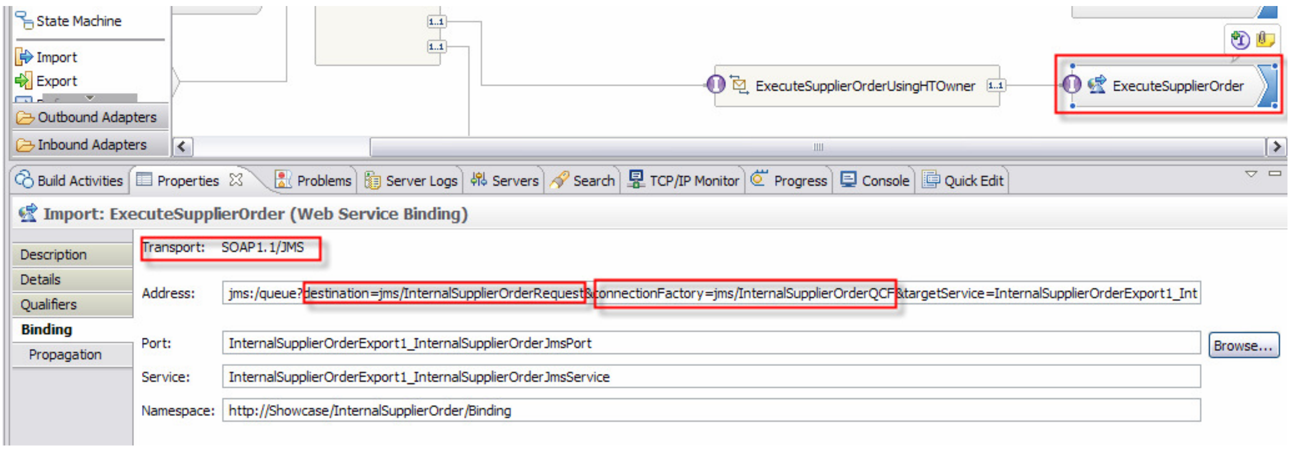
7.1.9.2 Step 2 and 3 – WPS: Implement a SOAP/JMS binding

In our showcase we use a SOAP/MQ call.

There are two options to do this:

1. Using a SOAP Datahandler with the MQ binding. However, the disadvantage here is, that just the BO itself is converted to a SOAP message and not the SOAP Headers. The SOAP Headers are ignored by the Datahandler. Therefore this is not an option for the showcase.
2. Another option, which we implemented, is to use a SOAP/JMS binding with MQ as JMS provider. Using this approach, the SOAP header we set in the mediation is inserted into the SOAP envelope.

The JMS/SOAP binding defines *Address* properties containing destination and queue connection factory.



Import: ExecuteSupplierOrder (Web Service Binding)

Transport: SOAP1.1/JMS

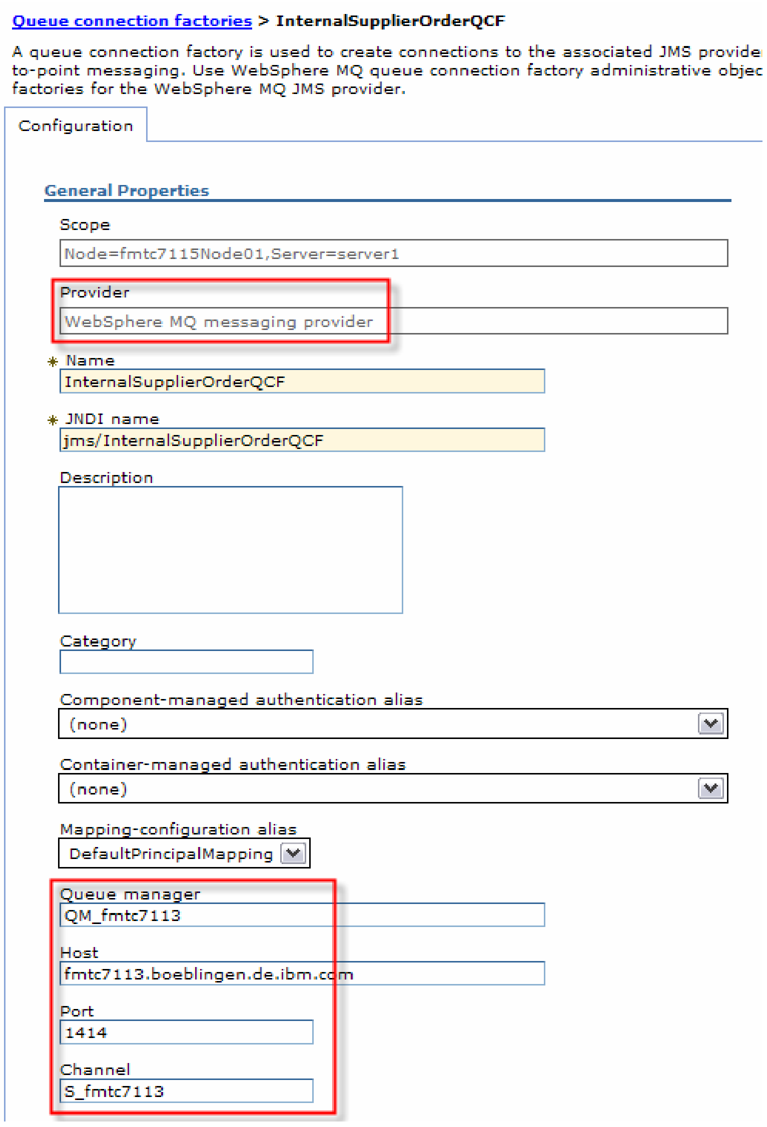
Address: jms:/queue?destination=jms/InternalSupplierOrderRequest&connectionFactory=jms/InternalSupplierOrderQCF&targetService=InternalSupplierOrderExport1_Int

Port: InternalSupplierOrderExport1_InternalSupplierOrderJmsPort

Service: InternalSupplierOrderExport1_InternalSupplierOrderJmsService

Namespace: http://Showcase/InternalSupplierOrder/Binding

The queue connection factory is based on MQ:

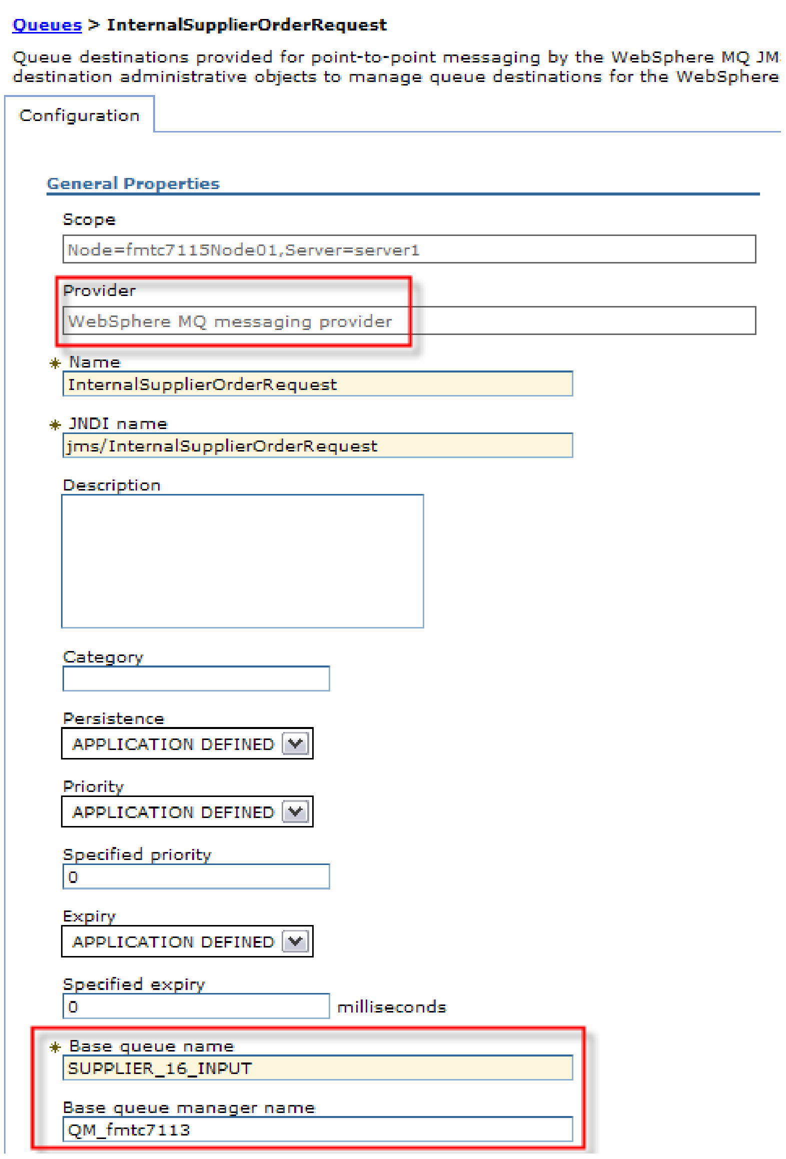

Queue connection factories > InternalSupplierOrderQCF

A queue connection factory is used to create connections to the associated JMS provide
to-point messaging. Use WebSphere MQ queue connection factory administrative objec
factories for the WebSphere MQ JMS provider.

General Properties

Scope	Node=fmtc7115Node01,Server=server1
Provider	WebSphere MQ messaging provider
* Name	InternalSupplierOrderQCF
* JNDI name	jms/InternalSupplierOrderQCF
Description	
Category	
Component-managed authentication alias	(none)
Container-managed authentication alias	(none)
Mapping-configuration alias	DefaultPrincipalMapping
Queue manager	QM_fmtc7113
Host	fmtc7113.boeblingen.de.ibm.com
Port	1414
Channel	S_fmtc7113

And the destination is also based on MQ:


Queues > InternalSupplierOrderRequest

Queue destinations provided for point-to-point messaging by the WebSphere MQ JM
destination administrative objects to manage queue destinations for the WebSphere

Configuration

General Properties

Scope
Node=fmtc7115Node01,Server=server1

Provider
WebSphere MQ messaging provider

* Name
InternalSupplierOrderRequest

* JNDI name
jms/InternalSupplierOrderRequest

Description

Category

Persistence
APPLICATION DEFINED

Priority
APPLICATION DEFINED

Specified priority
0

Expiry
APPLICATION DEFINED

Specified expiry
0 milliseconds

* Base queue name
SUPPLIER_16_INPUT

Base queue manager name
QM_fmtc7113

7.1.9.3 Step 4 – WPS: Deploy the BPEL application

Detailed deployment steps are described in the appendix.

7.1.9.4 Step 5 – WAS: Develop the WAS application

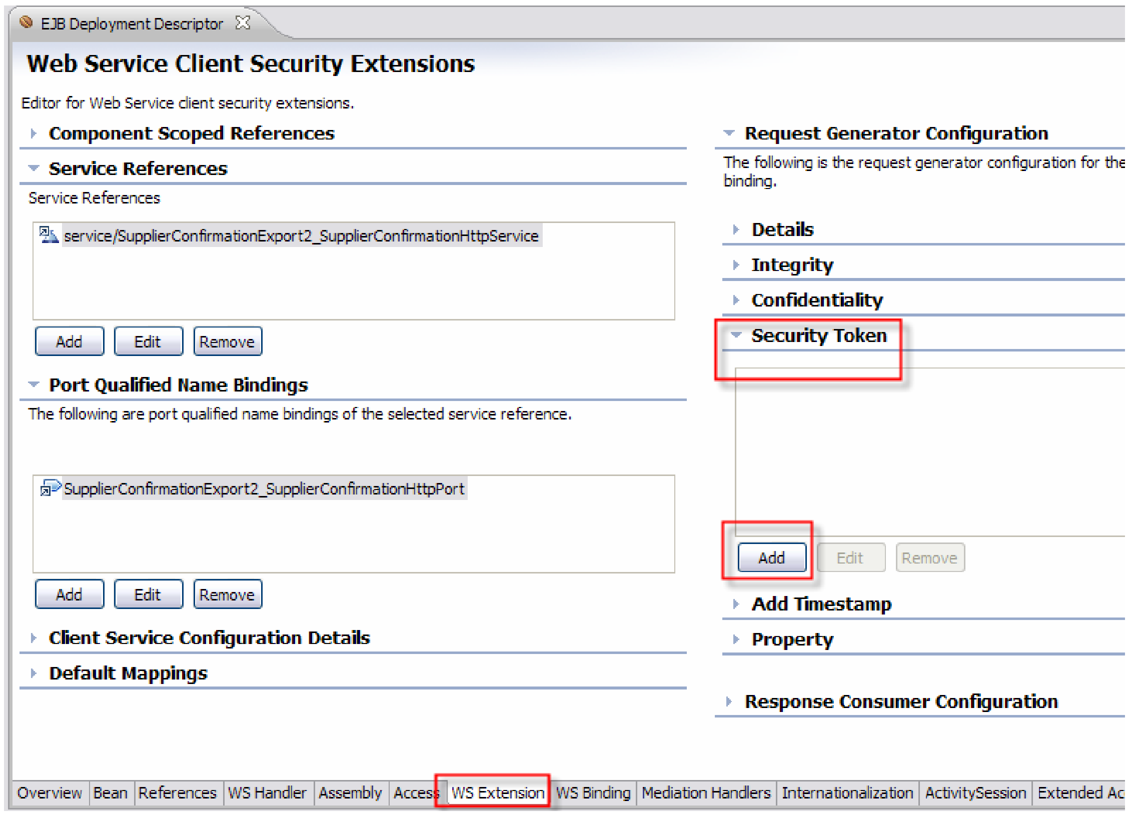
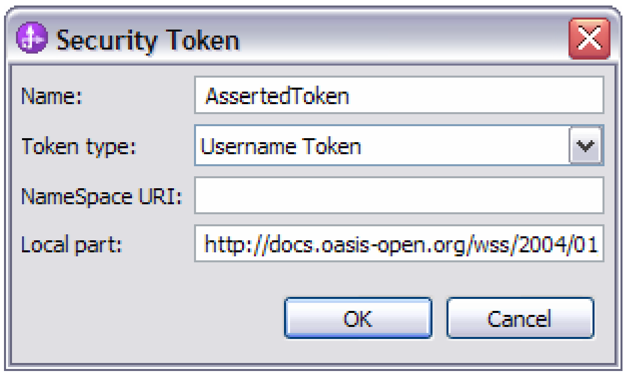
The WAS application picks up the message from the JMS MQ queue with a message-driven bean.
The SOAP message is parsed and a new SOAP message is created, which is sent to Message Broker via SOAP/HTTP.

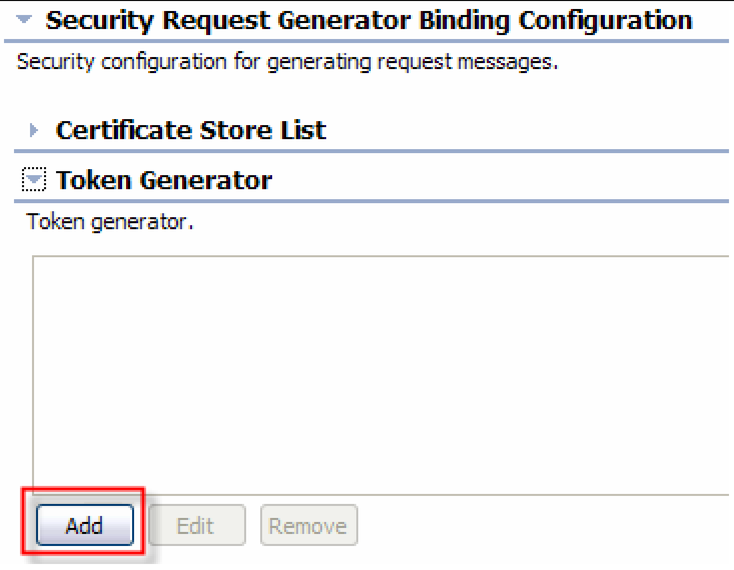
7.1.9.5 Step 6 – WAS: Configure MQ Adapter

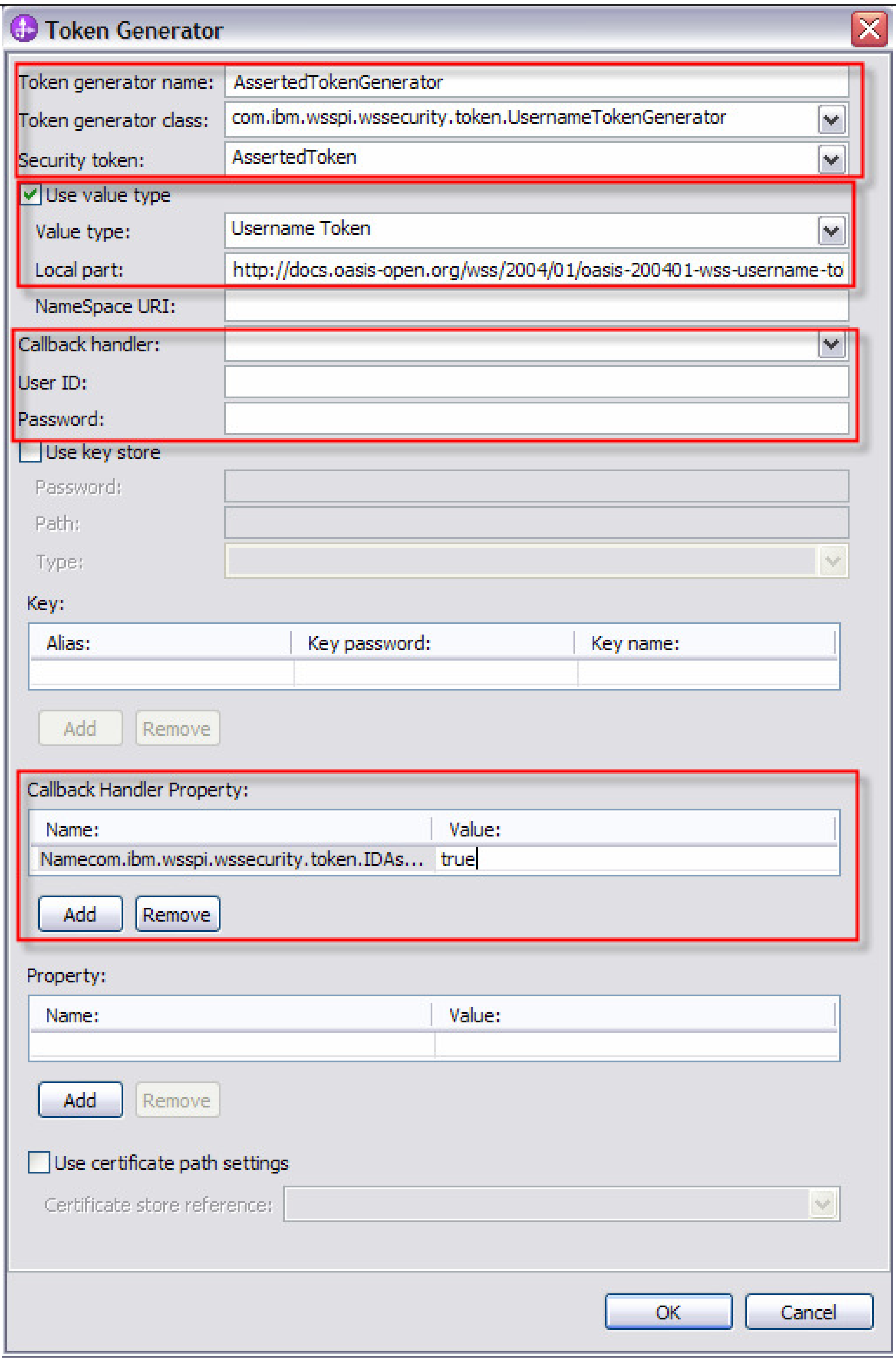
Refer to chapter 5.9.4.5 Step 5 – WAS: Configure MQ Adapter WAS application

7.1.9.6 Step 7 – WAS: Define Token Generator and Consumer

Define Token Generator:

	Open the deployment descriptor on the client application
	In the deployment descriptor open the WS Extension tab, and add a new Security Token  EJB Deployment Descriptor Web Service Client Security Extensions Editor for Web Service client security extensions. Component Scoped References Service References Service References service/SupplierConfirmationExport2_SupplierConfirmationHttpService Port Qualified Name Bindings The following are port qualified name bindings of the selected service reference. SupplierConfirmationExport2_SupplierConfirmationHttpPort Client Service Configuration Details Default Mappings Request Generator Configuration The following is the request generator configuration for the binding. Details Integrity Confidentiality Security Token Add Timestamp Property Response Consumer Configuration Overview | Bean | References | WS Handler | Assembly | Access | WS Extension | WS Binding | Mediation Handlers | Internationalization | ActivitySession | Extended Ac
	Add a new Username token  Security Token Name: AssertedToken Token type: Username Token NameSpace URI: Local part: http://docs.oasis-open.org/wss/2004/01 OK Cancel
	Go to the *WS Binding* tab and add a new Token Generator

	 **Security Request Generator Binding Configuration** Security configuration for generating request messages. **Certificate Store List** **Token Generator** Token generator.
	Token Generator Name: AssertedTokenGenerator Token generator class: com.ibm.wsspi.wssecurity.token.UsernameTokenGenerator Security Token: AssertedToken Use value type: Checked Callback handler: Blank UserID Blank Password Blank Callback handler Properties: com.ibm.wsspi.wssecurity.token.IDAssertion.isUsed=true com.ibm.wsspi.wssecurity.token.IDAssertion.useRunAsIdentity=true *You'll need to click the Add button to add a row and then select name and value fields to type over.*



Token Generator

Token generator name: AssertedTokenGenerator

Token generator class: com.ibm.wsspi.wssecurity.token.UsernameTokenGenerator

Security token: AssertedToken

☑ Use value type

Value type: Username Token

Local part: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-to

NameSpace URI:

Callback handler:

User ID:

Password:

☐ Use key store

Password:

Path:

Type:

Key:

Alias:	Key password:	Key name:

Callback Handler Property:

Name:	Value:
Namecom.ibm.wsspi.wssecurity.token.IDAs...	true

Property:

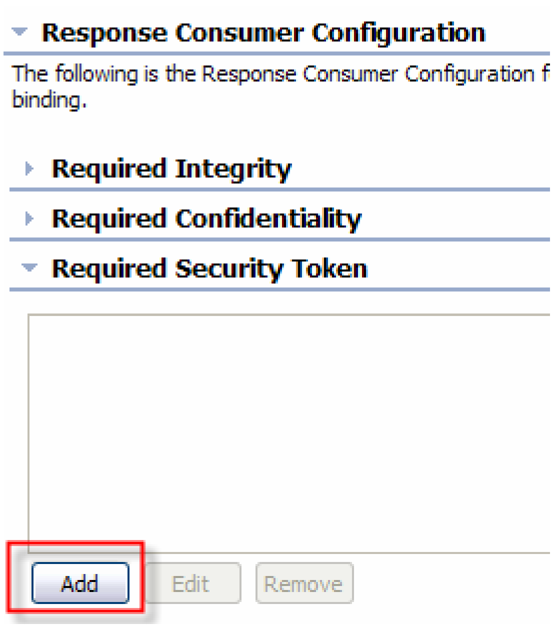
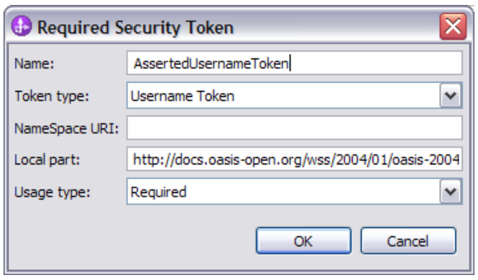
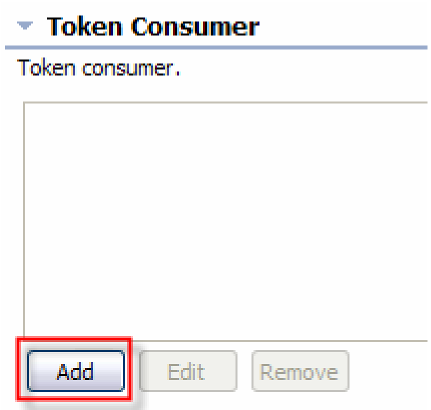
Name:	Value:

☐ Use certificate path settings

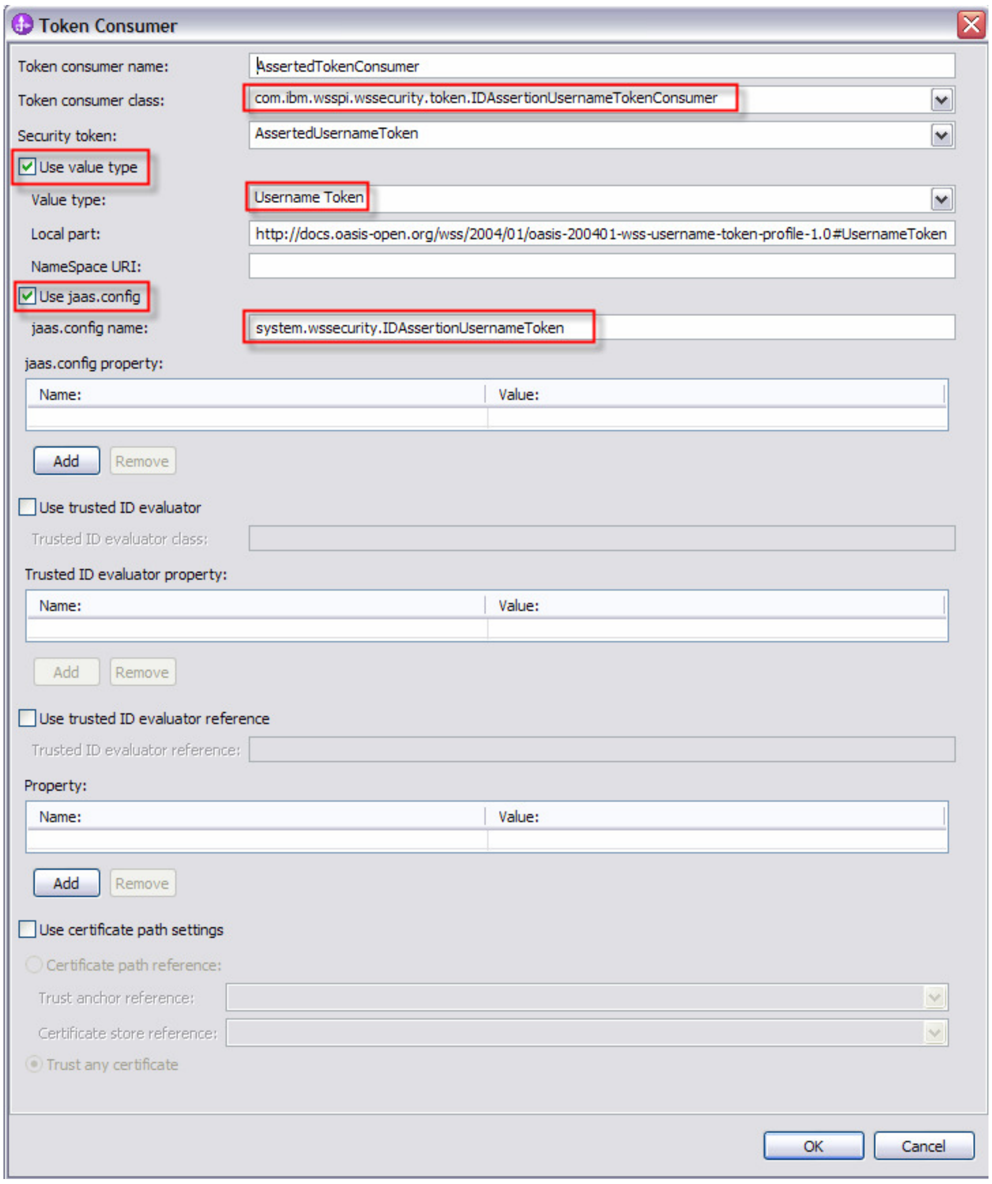
Certificate store reference:

Define Token Consumer:

1.	To create a *Response Consumer Security Token* open the deployment descriptor and goto the tab WS Extension

2.	Open *Response Consumer Service Configuration Details > Required Security Token* and Click *Add* Response Consumer Configuration The following is the Response Consumer Configuration f binding. Required Integrity Required Confidentiality Required Security Token
3.	Name the token for example *AssertedUsernameToken* Select as Token type *Username Token* Local Part is set automatically when choosing *Username Token* Usage type is *Required* Click *OK* Required Security Token Name: AssertedUsernameToken Token type: Username Token NameSpace URI: Local part: http://docs.oasis-open.org/wss/2004/01/oasis-2004 Usage type: Required
4.	Open the WS-*Binding* Tab Open *Response Consumer Binding Configuration Details > Token Consumer* Click *Add* Token Consumer Token consumer.
5.	• In the Token Consumer dialog box enter a consumer name, e.g AssertedTokenConsumer • Select as Token consumer class ***com.ibm.wsspi.wssecurity.token.IDAssertionUsernameTokenConsumer***

- As Security Token select *AssertedUsernameToken*
- Check *Use value type*
- Select as Value type: *Username Token*
- Local Part is generated automatically
- Check *Use.jaas.config*
- Enter as jaas.config name: *system.wssecurity.IDAssertionUsernameToken*
  by selecting the *IDAssertionUsernameToken* we define that we just need the user ID, and no password
- Click *OK*


Token Consumer

Token consumer name: AssertedTokenConsumer
Token consumer class: com.ibm.wsspi.wssecurity.token.IDAssertionUsernameTokenConsumer
Security token: AssertedUsernameToken
[✓] Use value type
Value type: Username Token
Local part: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken
NameSpace URI:
[✓] Use jaas.config
jaas.config name: system.wssecurity.IDAssertionUsernameToken
jaas.config property:

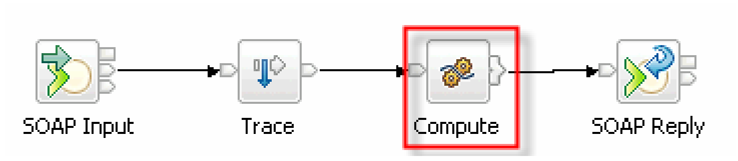
Name:
Value:
Add
Remove
Use trusted ID evaluator
Trusted ID evaluator class:
Trusted ID evaluator property:
Name:
Value:
Add
Remove
Use trusted ID evaluator reference
Trusted ID evaluator reference:
Property:
Name:
Value:
Add
Remove
Use certificate path settings
Certificate path reference:
Trust anchor reference:
Certificate store reference:
Trust any certificate
OK
Cancel

7.1.9.7 Step 8 – WAS: Deploy the WAS application

Detailed deployment steps are described in the appendix.

7.1.9.8 Step 9 – WMB: Develop the message flow

In the Compute node we copy the SOAP Header to the output message:



```
--COPY SOAP UsernameToken
SET OutputRoot.SOAP.Header = InputRoot.SOAP.Header;
```

7.1.9.9 Step 10 – WMB: Deploy th message flow

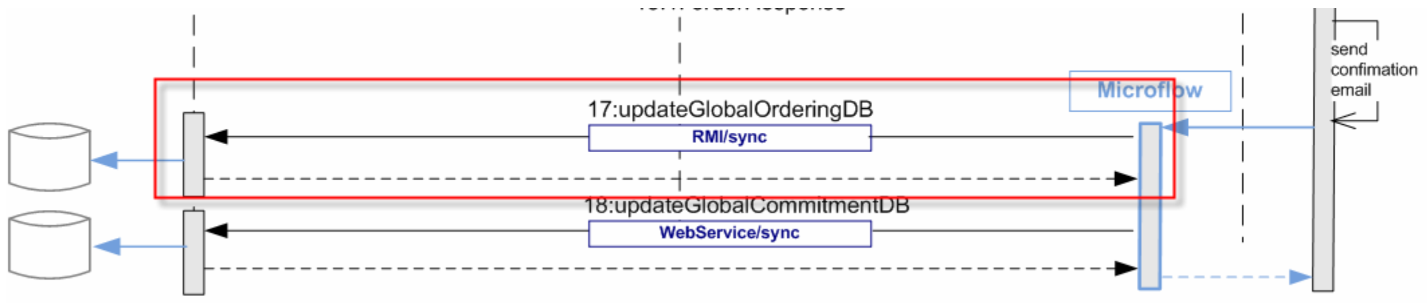
Detailed deployment steps are described in the appendix.

7.1.10 Interaction 17 – RMI between WPS and WAS

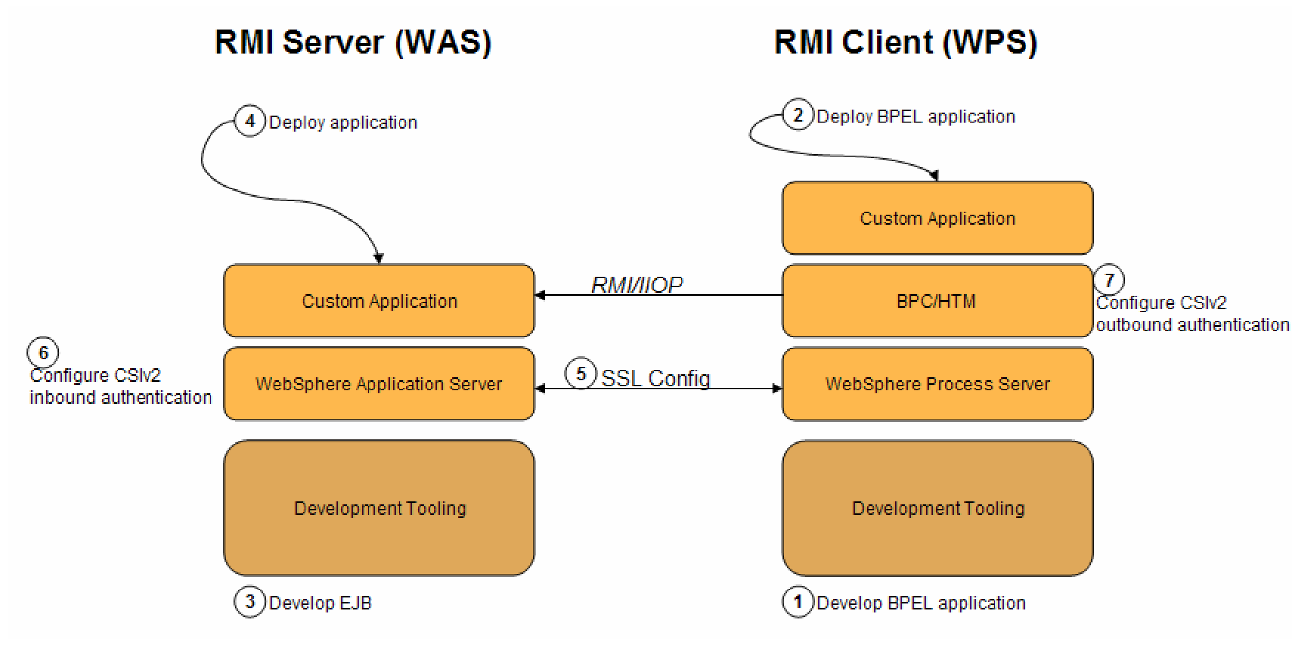
In this section we describe:

- SSL with RMI/IIOP
- Identity propagation

The figure below shows the relevant part in the sequence diagram:

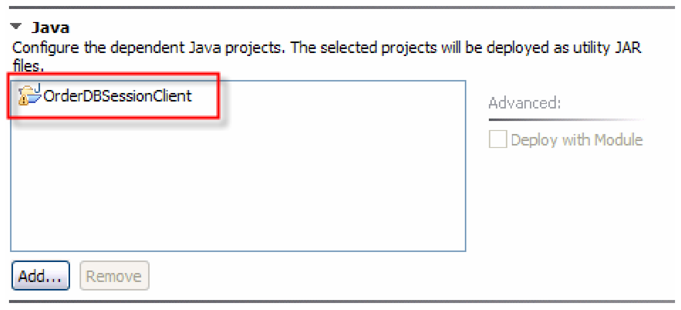


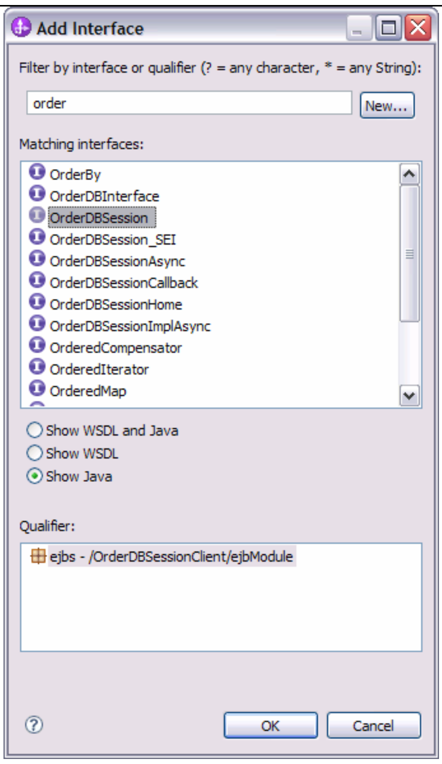
The figure below shows the high-level implementation and configuration steps that have to be performed. Find details of the steps in the next sections.



7.1.10.1 Step 1 – WPS: Develop BPEL application

To make an EJB call in a SCA application, the following basic steps have to be performed:

1.	Add the EJB client as dependency to the SCA module  Java Configure the dependent Java projects. The selected projects will be deployed as utility JAR files. OrderDBSessionClient Advanced: Deploy with Module Add... Remove
2.	Create a new import component in the Assembly editor
3.	Add the Java remote interface of the bean you want to access


Add Interface

Filter by interface or qualifier (? = any character, * = any String):

order

Matching interfaces:

- OrderBy
- OrderDBInterface
- OrderDBSession
- OrderDBSession_SEI
- OrderDBSessionAsync
- OrderDBSessionCallback
- OrderDBSessionHome
- OrderDBSessionImplAsync
- OrderedCompensator
- OrderedIterator
- OrderedMap

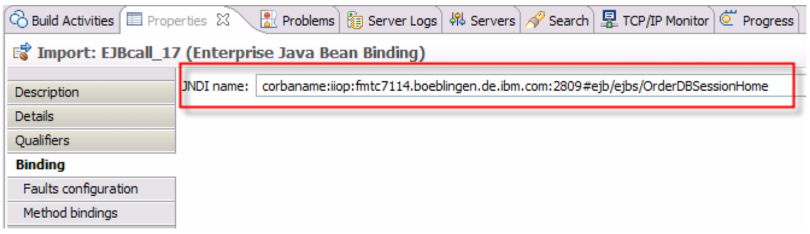
( ) Show WSDL and Java
( ) Show WSDL
[✓ Show Java]

Qualifier:

ejbs - /OrderDBSessionClient/ejbModule

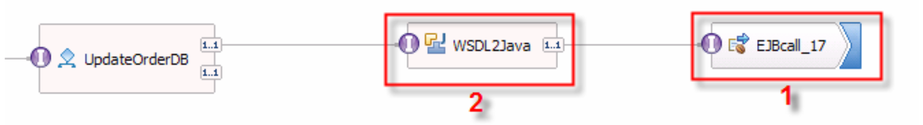
4. Generate a Enterprise Java Bean binding and name it for example EJBcall_17

5. Add following JNDI name to the EJB binding


Import: EJBcall_17 (Enterprise Java Bean Binding)

JNDI name: corbaname:iiop:fmtc7114.boeblingen.de.ibm.com:2809#ejb/ejbs/OrderDBSessionHome

6. EJB bindings can only have Java interfaces, not WSDL interfaces (from WPS 6.2 on they can also have WSDL interfaces). BPEL components can only have WSDL interfaces. Therefore we need a Java component “WSDL to Java bridge”, which transforms the WSDL interface to a Java interface (screenshot #2)



7. To create the “WSDL2Java” component, add a Java component to the Assembly Diagram and add a WSDL interface and a Java reference to it.
   - `WSDl interface: GlobalOrderingDB`
   - `Java interface: public interface OrderDBSession extends javax.ejb.EJBObject, OrderDBInterface`

8. Generate the implementation of the Java component

9. Add following code to access the EJB

```
    public String store(DataObject input1) {
            OrderDBRequestData orderDBRequestData = new
```

```
                OrderDBRequestData(input1.getString("clientOrderId"),
                input1.getInt("partCount"), input1.getString("partNumber"));

          //0=OK, 1=NOK
                OrderDBResponseData orderDBResponseData = new OrderDBResponseData(1);

          try {
                        orderDBResponseData =
                        locateService_OrderDBSessionPartner().createOrderEntry(orderDBRequestDa
                        ta);

          } catch (RemoteException e) {
               // TODO Auto-generated catch block
               e.printStackTrace();
          }

          String status;
          if (orderDBResponseData.getStatus()==0)
               status = "OK";
          else
               status = "NOK";

          return status;
     }
```

7.1.10.2 Step 2 – WPS: Deploy the BPEL application

Detailed deployment steps are described in the appendix.

7.1.10.3 Step 3 – WAS: Develop the application

Detailed implementation steps, which are not security relevant, are not described. Refer to the WID artefacts to see how the BPEL application is developed.

7.1.10.4 Step 4 – WAS: Deploy the application

Detailed deployment steps are described in the appendix.

7.1.10.5 Step 5 – SSL configuration

Refer to chapter "SSL between WPS and WAS"

7.1.10.6 Step 6 and 7 – Configure CSIv2 authentication

7.1.10.6.1 Option 1: Basic Authentication and identity assertion

Option 1 describes how to establish trust between the servers using Basic Authentication and identity assertion (technical user ID). Basic Authentication with identity assertion is identity propagation without the need for a common authentication infrastructure. For example the sending server and target server do not share LTPA keys.

Servers require some form of trust. In this identity assertion scenario the target server authenticates the sending server to establish trust. If the server is trustworthy, the target server accepts the asserted identity token. Two mechanisms to authenticate trusted user exist:

1. **Basic Authentication (implemented as Option 1 in this showcase)**
   - Outbound server's security ID (technical user ID) and password is sent
   - With WAS V6.1 can specify id to use

        - Inbound server validates user ID and password in registry
    2. Client Certificate Authentication (not implemented in the showcase)
        - Outbound server's client certificate (KeyRing used by IIOP) is verified by the inbound server, that is, the signing certificate used to sign the client's certificate (whether CA issued or self-signed) must be in the server's key ring
        - Certificate identity is then mapped to an identity in the receiving server's registry
        - Then by using the trusted server list, WAS determines if calling server is authorized to assert identity

Identity assertion behavior:

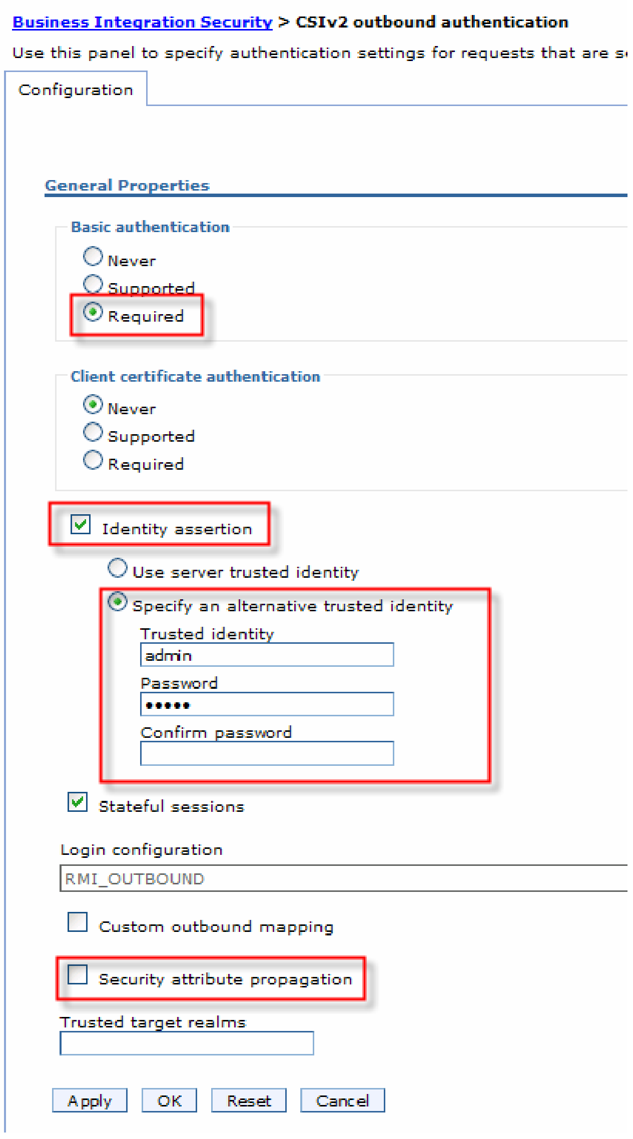
- Outbound server sends the asserted user's identity as a user ID
- Inbound server accepts the user's id and creates credentials by querying its registry
- No validation is performed on asserted identity (no password, token, etc)
- Both outbound and inbound servers can insert login modules to customize this process
    - RMI_INBOUND - inbound server's JAAS login configuration
    - RMI_OUTBOUND – outbound server's JAAS login configuration
- Either module can perform identity mapping

To configure Basic Authentication with identity assertion you have to configure

- CSIv2 outbound authentication on caller side (WPS)
- CSIv2 inbound authentication on provider side (WAS)

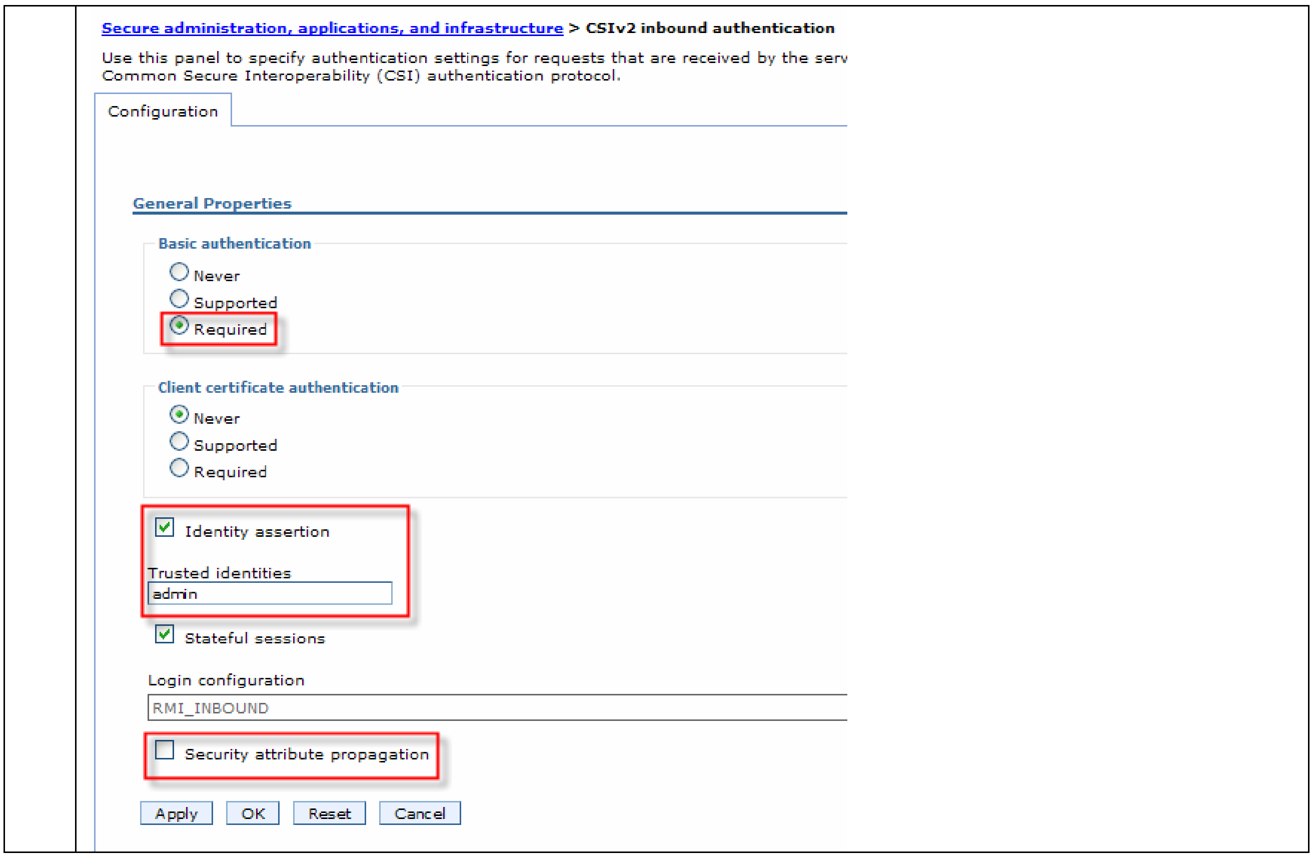
Configure CSIv2 outbound authentication on caller side (WPS)

1.	In the administrative console navigate *to Secure administration, applications, and infrastructure > RMI/IIOP security > CSIv2 outbound authentication*
2.	Set Basic Authentication to *required* This option indicates that clients communicating with this server must specify a user ID and password for any method request
3.	Enable *identity assertion* Specify as alternate trusted identity a technical user that is known on the provider side
4.	Disable *security attribute propagation* LTPA is the only authentication mechanism supported when you enable the security attribute propagation feature

	 Business Integration Security > CSIv2 outbound authentication Use this panel to specify authentication settings for requests that are s Configuration General Properties Basic authentication Never [✓ Required] Supported Client certificate authentication [✓ Never] Supported Required ✓ Identity assertion Use server trusted identity [✓ Specify an alternative trusted identity] Trusted identity: admin Password: ••••• Confirm password: ✓ Stateful sessions Login configuration: RMI_OUTBOUND ☐ Custom outbound mapping ☐ Security attribute propagation Trusted target realms: Apply OK Reset Cancel
5.	Save

Configure CSIv2 inbound authentication on provider side (WAS)

1.	In the administrative console navigate to *Secure administration, applications, and infrastructure > RMI/IIOP security > CSIv2 inbound authentication*
2.	Set Basic Authentication to *required*
3.	Enable identity assertion and enter the trusted identity
4.	Disable *security attribute propagation*



Secure administration, applications, and infrastructure > CSIv2 inbound authentication

Use this panel to specify authentication settings for requests that are received by the serv
Common Secure Interoperability (CSI) authentication protocol.

Configuration

General Properties

Basic authentication

- ○ Never
- ○ Supported
- ● Required

Client certificate authentication

- ● Never
- ○ Supported
- ○ Required

☑ Identity assertion

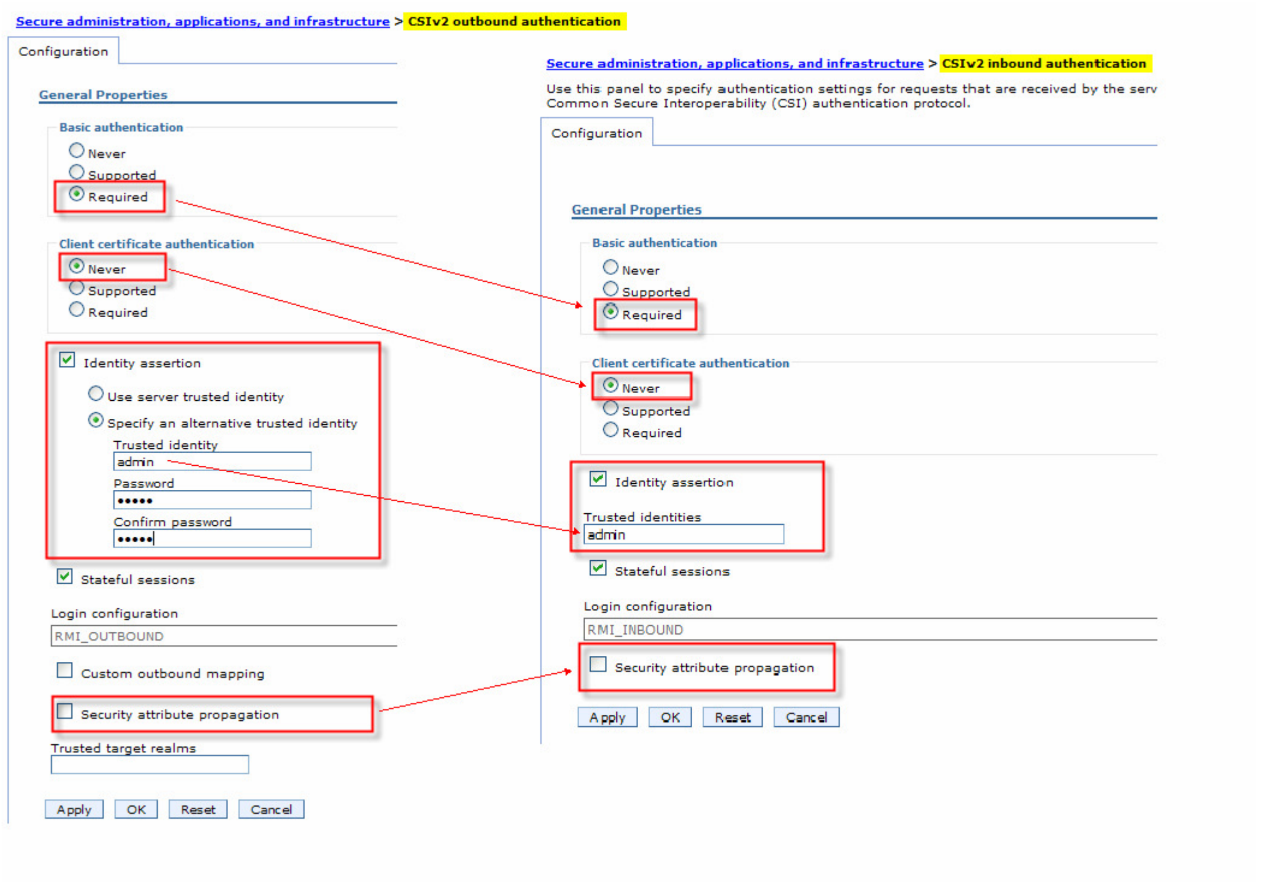
Trusted identities: admin

☑ Stateful sessions

Login configuration: RMI_INBOUND

☐ Security attribute propagation

The following picture shows both inbound and outbound authentication properties:


Secure administration, applications, and infrastructure > CSIv2 outbound authentication

Configuration

General Properties

Basic authentication
Never / Supported / [✓ Required]

Client certificate authentication
[✓ Never] / Supported / Required

[✓] Identity assertion
Use server trusted identity / [✓ Specify an alternative trusted identity]
Trusted identity: admin
Password: •••••
Confirm password: •••••

[✓] Stateful sessions

Login configuration: RMI_OUTBOUND

[ ] Custom outbound mapping

[ ] Security attribute propagation

Trusted target realms:

Apply | OK | Reset | Cancel

Secure administration, applications, and infrastructure > CSIv2 inbound authentication

Use this panel to specify authentication settings for requests that are received by the serv
Common Secure Interoperability (CSI) authentication protocol.

Configuration

General Properties

Basic authentication
Never / Supported / [✓ Required]

Client certificate authentication
[✓ Never] / Supported / Required

[✓] Identity assertion
Trusted identities: admin

[✓] Stateful sessions

Login configuration: RMI_INBOUND

[ ] Security attribute propagation

Apply | OK | Reset | Cancel

7.1.10.6.2 Option 2: Basic Authentication without identity assertion

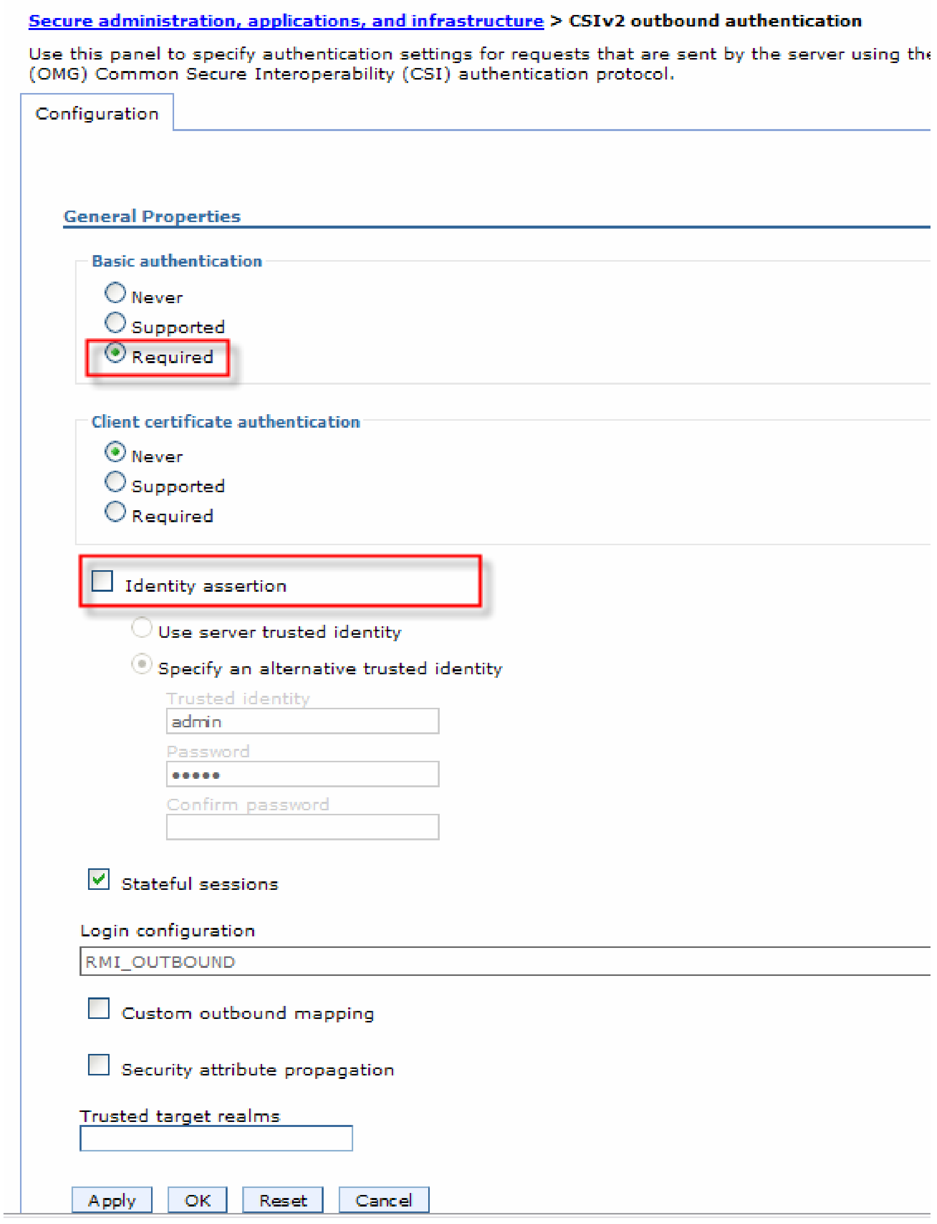
Option 2 describes how to use Basic authentication to authenticate with the current user using LTPA tokens. Prerequisite is that the sending server and target server share LTPA keys.

- The client sends an LTPA token to the target server via the IIOP channel.
- Option 2 is only applicable if both servers share the realm or part of a trusted realm
- Option 2 is only applicable if both servers share the LTPA key

To configure Basic Authentication with identity assertion you have to configure
- CSIv2 outbound authentication on caller side (WPS)
- CSIv2 inbound authentication on provider side (WAS)

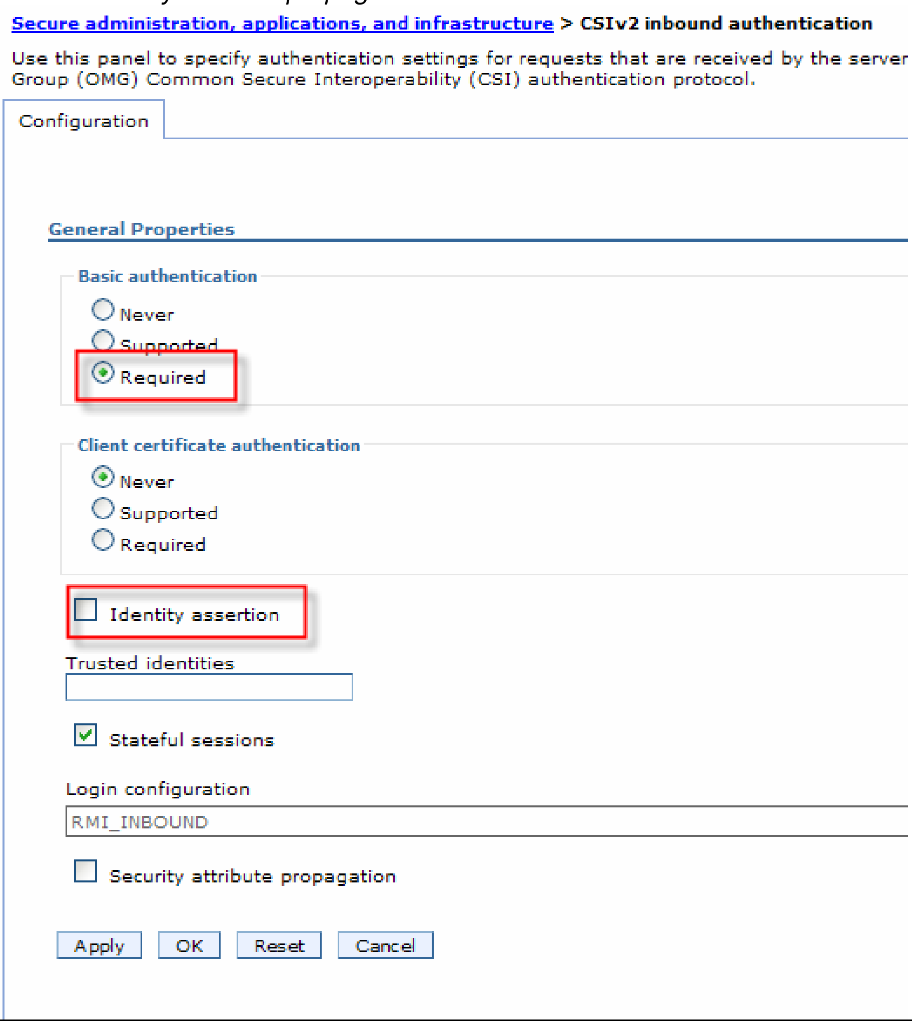
Configure CSIv2 outbound authentication on caller side (WPS)

1.	In the administrative console navigate to *Secure administration, applications, and infrastructure > RMI/IIOP security > CSIv2 outbound authentication*
2.	Set Basic Authentication to *required* a. Never i. This option indicates that this server cannot accept user ID and password authentication. b. Supported i. This option indicates that a client communicating with this server can specify a user ID and password. However, a method might be invoked without this type of authentication. For example, an anonymous or client certificate might be used instead.

	c. Required i. This option indicates that clients communicating with this server must specify a user ID and password for any method request.
3.	Disable *identity assertion*
4.	Disable *security attribute propagation*  Secure administration, applications, and infrastructure > **CSIv2 outbound authentication** Use this panel to specify authentication settings for requests that are sent by the server using th (OMG) Common Secure Interoperability (CSI) authentication protocol. Configuration General Properties Basic authentication Never Supported [✓ Required] Client certificate authentication [✓ Never] Supported Required ☐ Identity assertion Use server trusted identity [✓ Specify an alternative trusted identity] Trusted identity: admin Password: ••••• Confirm password: ☑ Stateful sessions Login configuration: RMI_OUTBOUND ☐ Custom outbound mapping ☐ Security attribute propagation Trusted target realms: Apply OK Reset Cancel
5.	Save

Configure CSIv2 inbound authentication on provider side (WAS)

1.	In the administrative console navigate to *Secure administration, applications, and infrastructure > RMI/IIOP security > CSIv2 inbound authentication*

2.	Set Basic Authentication to *required*
3.	disable identity assertion
4.	disable *security attribute propagation* 

Secure administration, applications, and infrastructure > **CSIv2 inbound authentication**

Use this panel to specify authentication settings for requests that are received by the server
Group (OMG) Common Secure Interoperability (CSI) authentication protocol.

Configuration

General Properties

Basic authentication
Never Supported [✓ Required]

Client certificate authentication
[✓ Never] Supported Required

☐ Identity assertion

Trusted identities

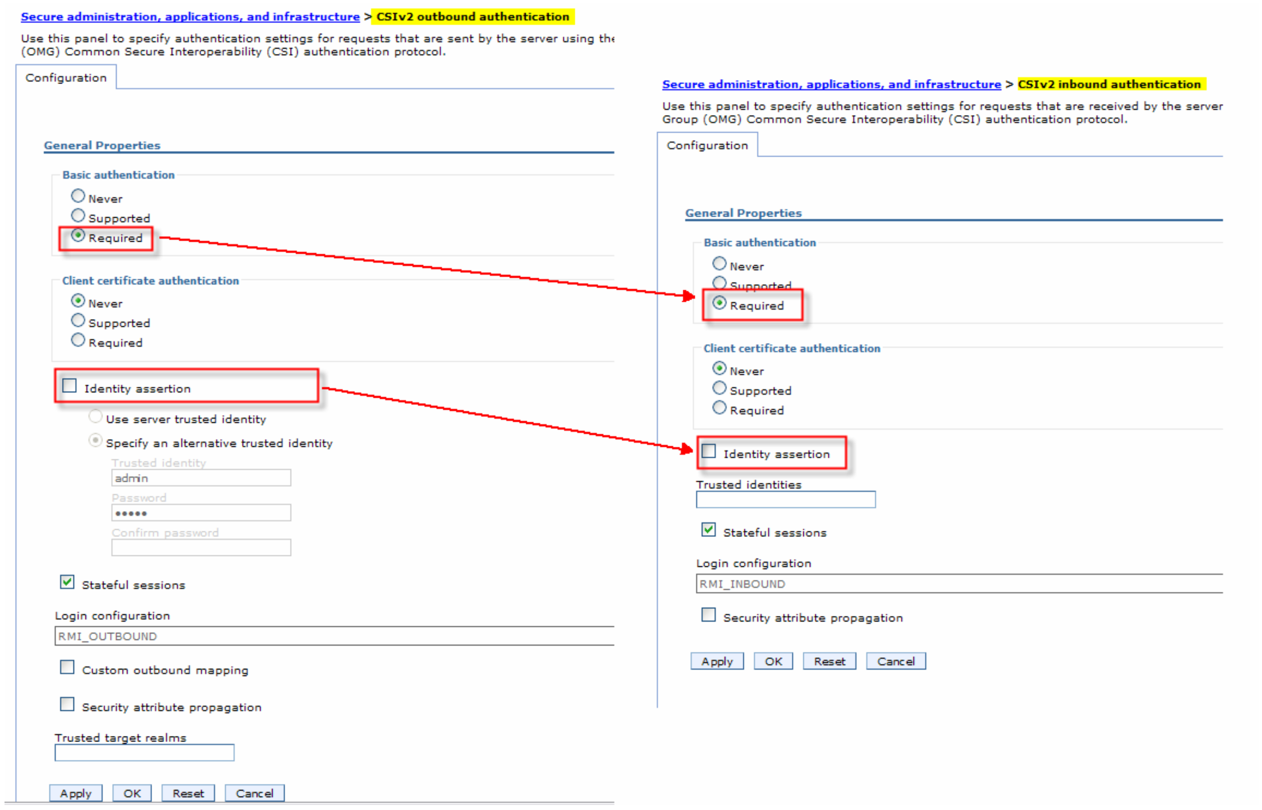
✓ Stateful sessions

Login configuration
RMI_INBOUND

☐ Security attribute propagation

Apply OK Reset Cancel

The following picture shows both inbound and outbound authentication properties:



Secure administration, applications, and infrastructure > CSIv2 outbound authentication

Use this panel to specify authentication settings for requests that are sent by the server using th
(OMG) Common Secure Interoperability (CSI) authentication protocol.

Configuration

General Properties

Basic authentication
Never Supported [✓ Required]

Client certificate authentication
[✓ Never] Supported Required

☐ Identity assertion
Use server trusted identity
[✓ Specify an alternative trusted identity]
Trusted identity: admin
Password: •••••
Confirm password:

☑ Stateful sessions

Login configuration: RMI_OUTBOUND

☐ Custom outbound mapping

☐ Security attribute propagation

Trusted target realms:

Secure administration, applications, and infrastructure > CSIv2 inbound authentication

Use this panel to specify authentication settings for requests that are received by the server
Group (OMG) Common Secure Interoperability (CSI) authentication protocol.

Configuration

General Properties

Basic authentication
Never Supported [✓ Required]

Client certificate authentication
[✓ Never] Supported Required

☐ Identity assertion

Trusted identities:

☑ Stateful sessions

Login configuration: RMI_INBOUND

☐ Security attribute propagation

7.1.10.6.3 When to use LTPA, identity assertion, Basic Authentication and Certificates

CSIv2 panel shows:

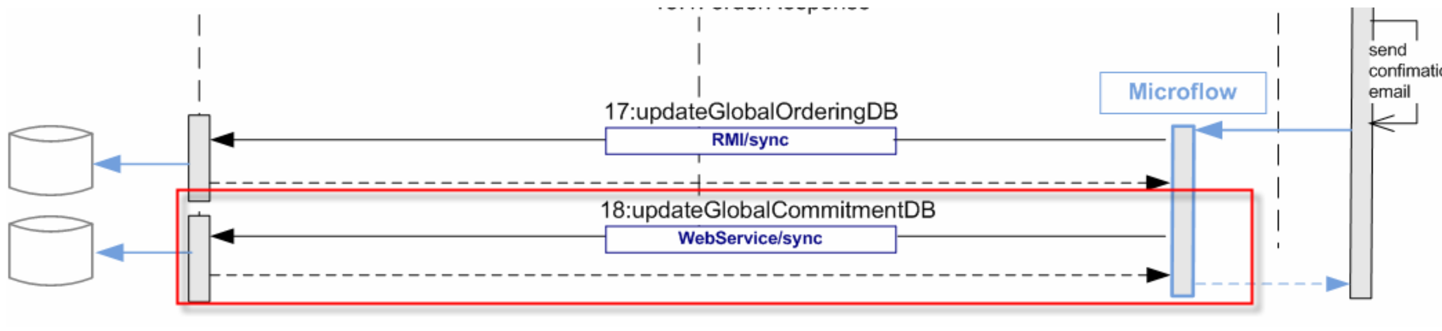
- Identity assertion
- Basic auth
- Certificate
- LTPA is implicitly there as an option but not shown.

The recommendations are:

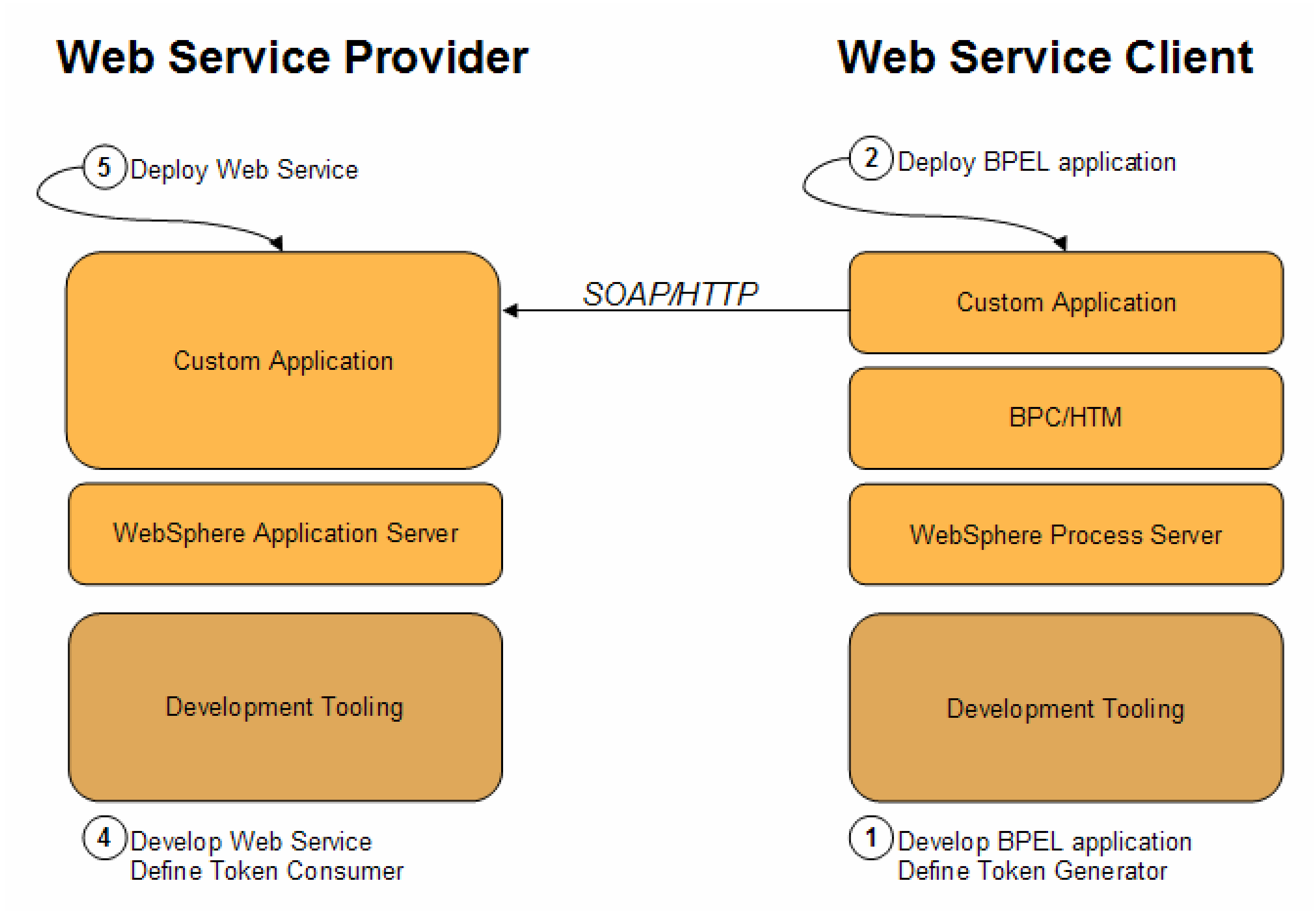
- If identity assertion is available
  - LTPA token not used or sent for user identity
  - Basic auth or certificate used for server to server authentication
- If identity assertion is not available
  - LTPA token used if available
  - Basic auth or certificate used if no LTPA is available.

7.1.11 Interaction 18 – Update Global Order DB - SOAP HTTPs between WPS and WAS

The figure below shows the relevant part in the sequence diagram:



The figure below shows the high-level implementation and configuration steps that have to be performed. Find details of the steps in the next sections.



This interaction step is not documented in detail, because it is not security relevant

8 Basis setup and application install reference

8.1 Configuration of JMS resources

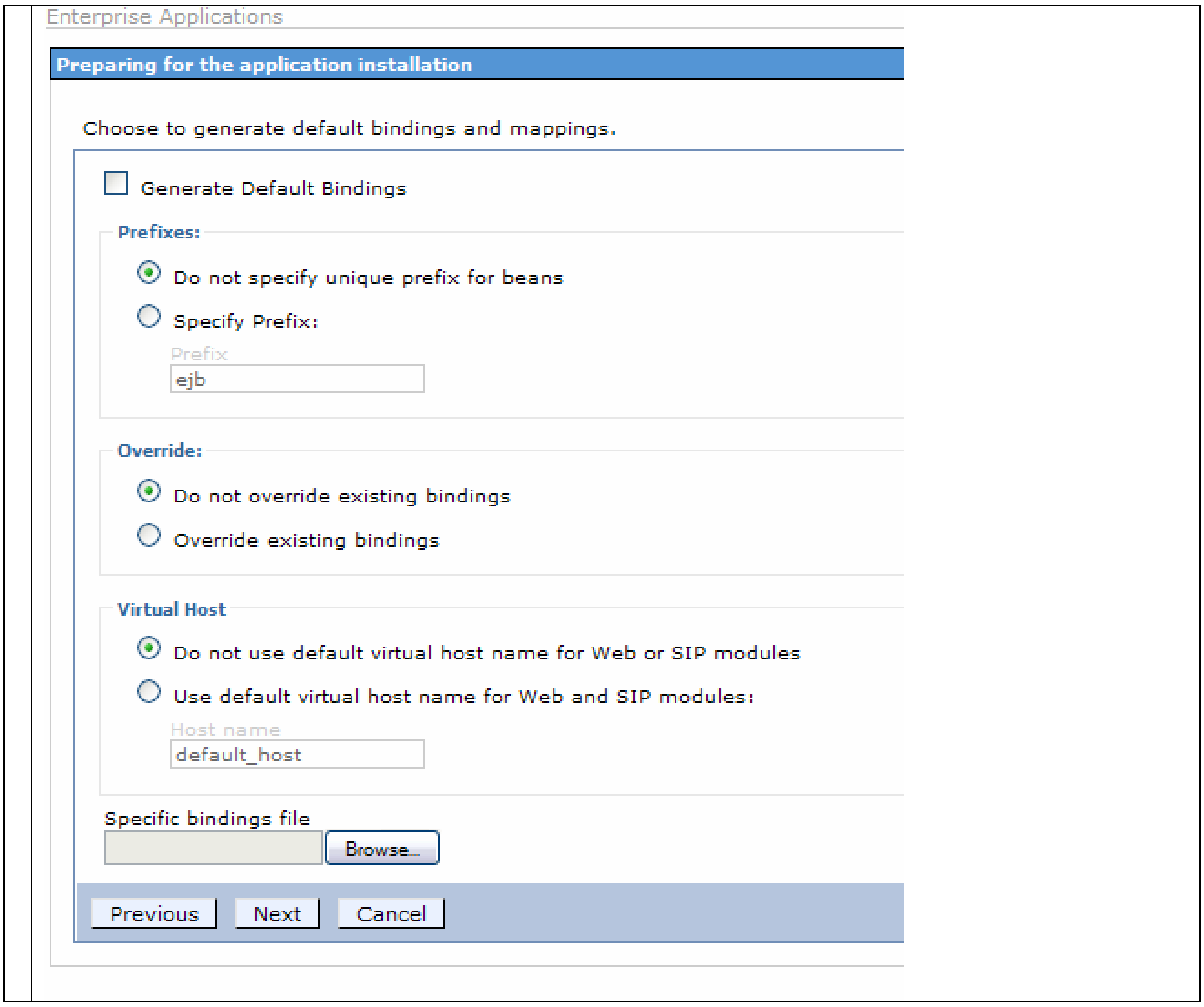
All JMS resources on WPS are generated automatically during deployment of the applications.

8.2 Configuration of JDBC resources

All JDBC resources on WPS are generated automatically during deployment of the applications.

8.3 Deployment of the showcase SCA module

For the deployment to WPS we use the default deployment settings:



Enterprise Applications

Preparing for the application installation

Choose to generate default bindings and mappings.

☐ Generate Default Bindings

Prefixes:

[✓ Do not specify unique prefix for beans]

Specify Prefix:

Prefix: ejb

Override:

[✓ Do not override existing bindings]

Override existing bindings

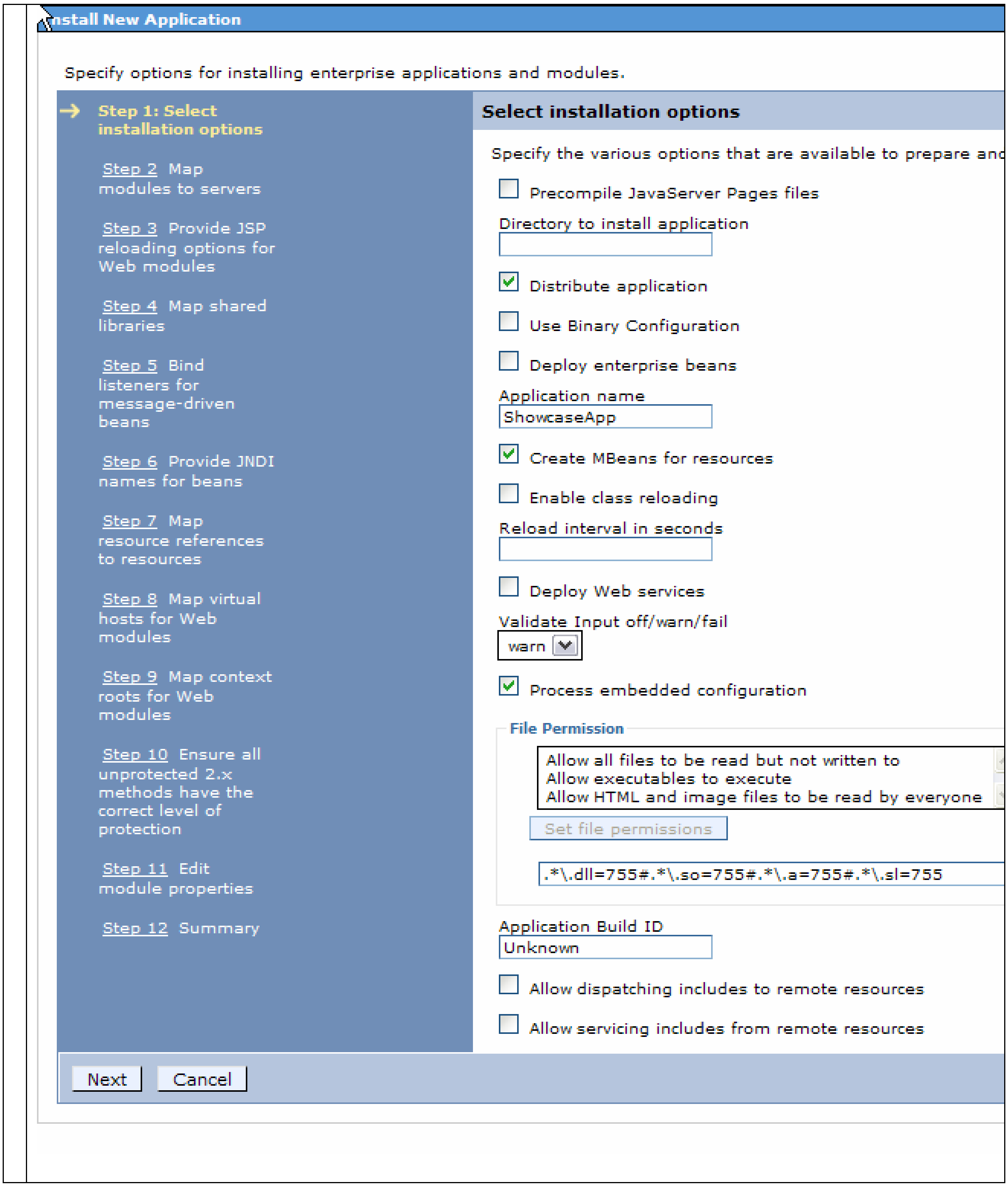
Virtual Host

[✓ Do not use default virtual host name for Web or SIP modules]

Use default virtual host name for Web and SIP modules:

Host name: default_host

Specific bindings file


Install New Application

Specify options for installing enterprise applications and modules.

Step 1: Select installation options

Step 2 Map modules to servers

Step 3 Provide JSP reloading options for Web modules

Step 4 Map shared libraries

Step 5 Bind listeners for message-driven beans

Step 6 Provide JNDI names for beans

Step 7 Map resource references to resources

Step 8 Map virtual hosts for Web modules

Step 9 Map context roots for Web modules

Step 10 Ensure all unprotected 2.x methods have the correct level of protection

Step 11 Edit module properties

Step 12 Summary

Select installation options

Specify the various options that are available to prepare and

☐ Precompile JavaServer Pages files

Directory to install application

✓ Distribute application

☐ Use Binary Configuration

☐ Deploy enterprise beans

Application name: ShowcaseApp

✓ Create MBeans for resources

☐ Enable class reloading

Reload interval in seconds

☐ Deploy Web services

Validate Input off/warn/fail: warn

✓ Process embedded configuration

File Permission

Allow all files to be read but not written to
Allow executables to execute
Allow HTML and image files to be read by everyone

Set file permissions

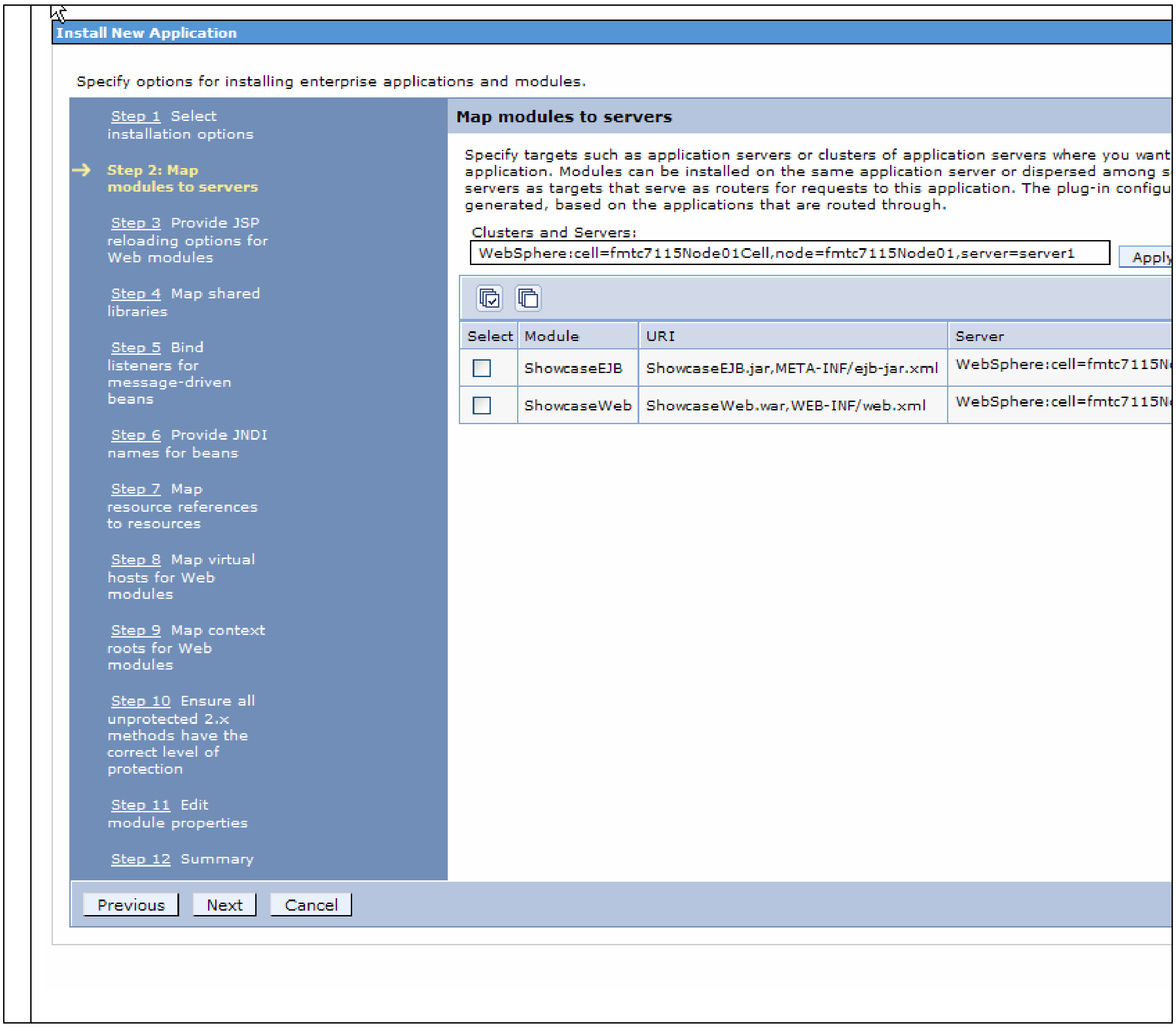
.*\.dll=755#.*\.so=755#.*\.a=755#.*\.sl=755

Application Build ID: Unknown

☐ Allow dispatching includes to remote resources

☐ Allow servicing includes from remote resources

Next | Cancel


Install New Application

Specify options for installing enterprise applications and modules.

Step 1 Select installation options

→ **Step 2: Map modules to servers**

Step 3 Provide JSP reloading options for Web modules

Step 4 Map shared libraries

Step 5 Bind listeners for message-driven beans

Step 6 Provide JNDI names for beans

Step 7 Map resource references to resources

Step 8 Map virtual hosts for Web modules

Step 9 Map context roots for Web modules

Step 10 Ensure all unprotected 2.x methods have the correct level of protection

Step 11 Edit module properties

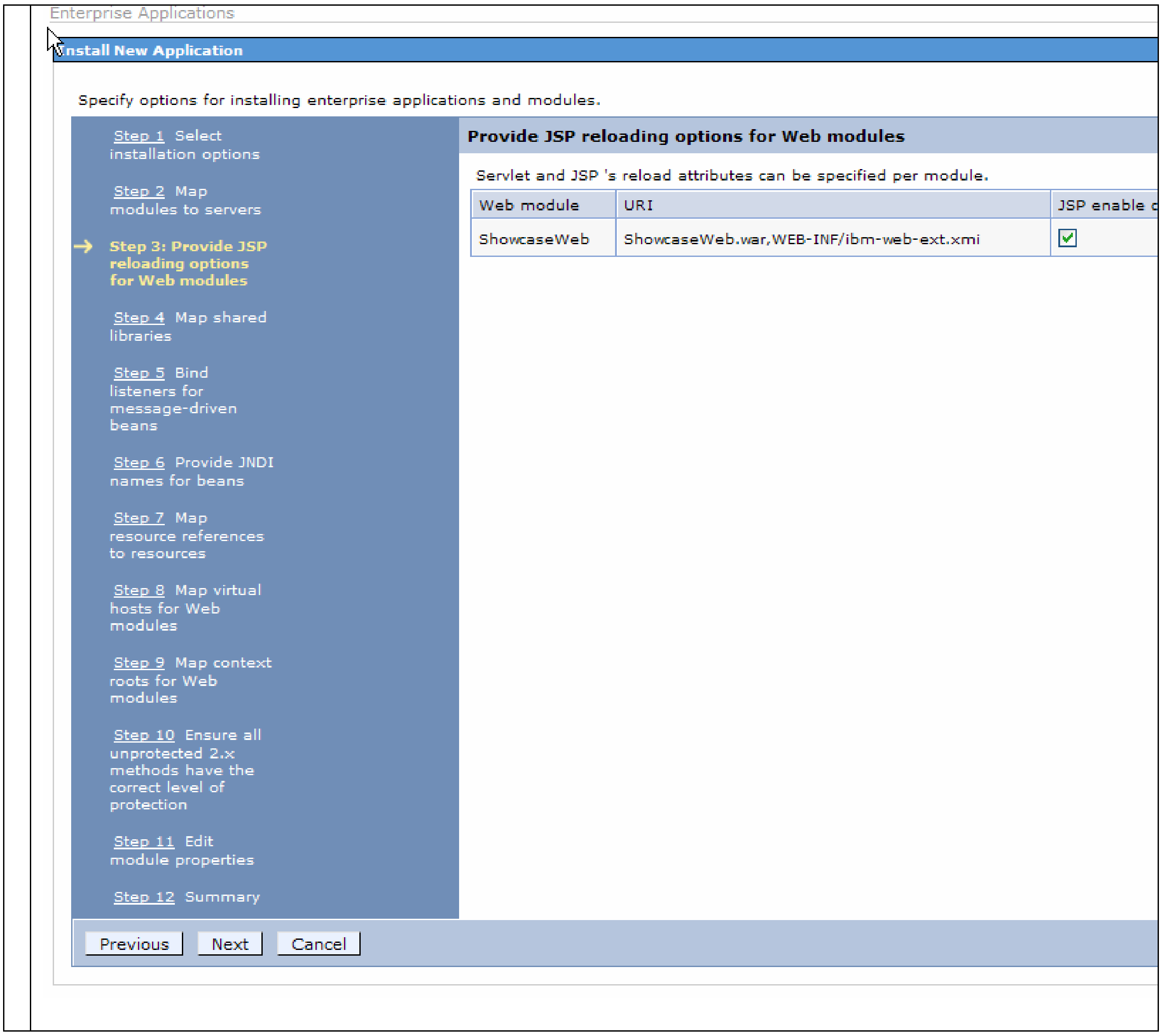
Step 12 Summary

Map modules to servers

Specify targets such as application servers or clusters of application servers where you want application. Modules can be installed on the same application server or dispersed among s servers as targets that serve as routers for requests to this application. The plug-in configu generated, based on the applications that are routed through.

Clusters and Servers:
WebSphere:cell=fmtc7115Node01Cell,node=fmtc7115Node01,server=server1

Select	Module	URI	Server
☐	ShowcaseEJB	ShowcaseEJB.jar,META-INF/ejb-jar.xml	WebSphere:cell=fmtc7115N
☐	ShowcaseWeb	ShowcaseWeb.war,WEB-INF/web.xml	WebSphere:cell=fmtc7115N



Enterprise Applications

Install New Application

Specify options for installing enterprise applications and modules.

Step 1 Select installation options

Step 2 Map modules to servers

→ **Step 3: Provide JSP reloading options for Web modules**

Step 4 Map shared libraries

Step 5 Bind listeners for message-driven beans

Step 6 Provide JNDI names for beans

Step 7 Map resource references to resources

Step 8 Map virtual hosts for Web modules

Step 9 Map context roots for Web modules

Step 10 Ensure all unprotected 2.x methods have the correct level of protection

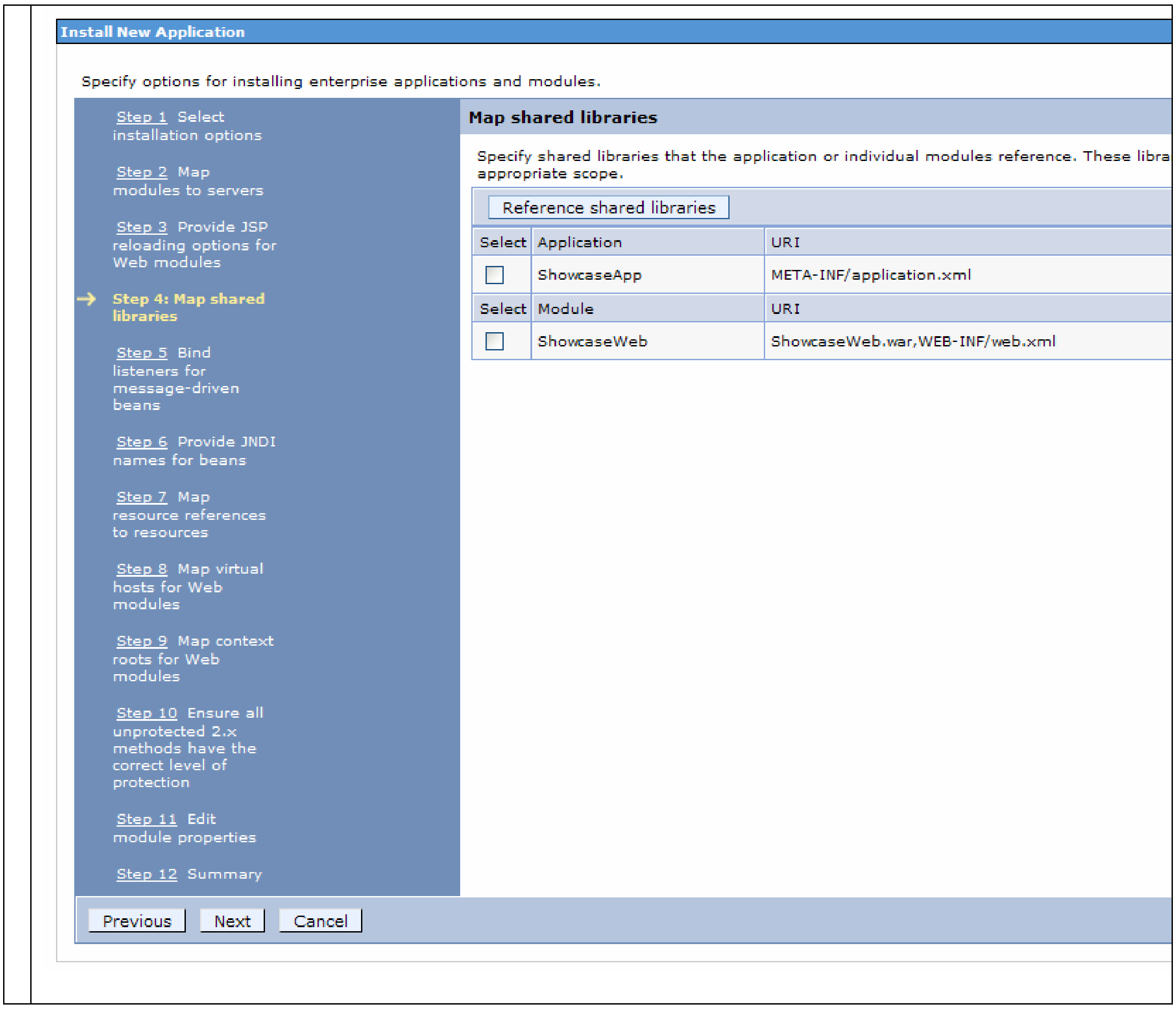
Step 11 Edit module properties

Step 12 Summary

Provide JSP reloading options for Web modules

Servlet and JSP 's reload attributes can be specified per module.

Web module	URI	JSP enable c
ShowcaseWeb	ShowcaseWeb.war,WEB-INF/ibm-web-ext.xmi	✓


Install New Application

Specify options for installing enterprise applications and modules.

Step 1 Select installation options

Step 2 Map modules to servers

Step 3 Provide JSP reloading options for Web modules

→ **Step 4: Map shared libraries**

Step 5 Bind listeners for message-driven beans

Step 6 Provide JNDI names for beans

Step 7 Map resource references to resources

Step 8 Map virtual hosts for Web modules

Step 9 Map context roots for Web modules

Step 10 Ensure all unprotected 2.x methods have the correct level of protection

Step 11 Edit module properties

Step 12 Summary

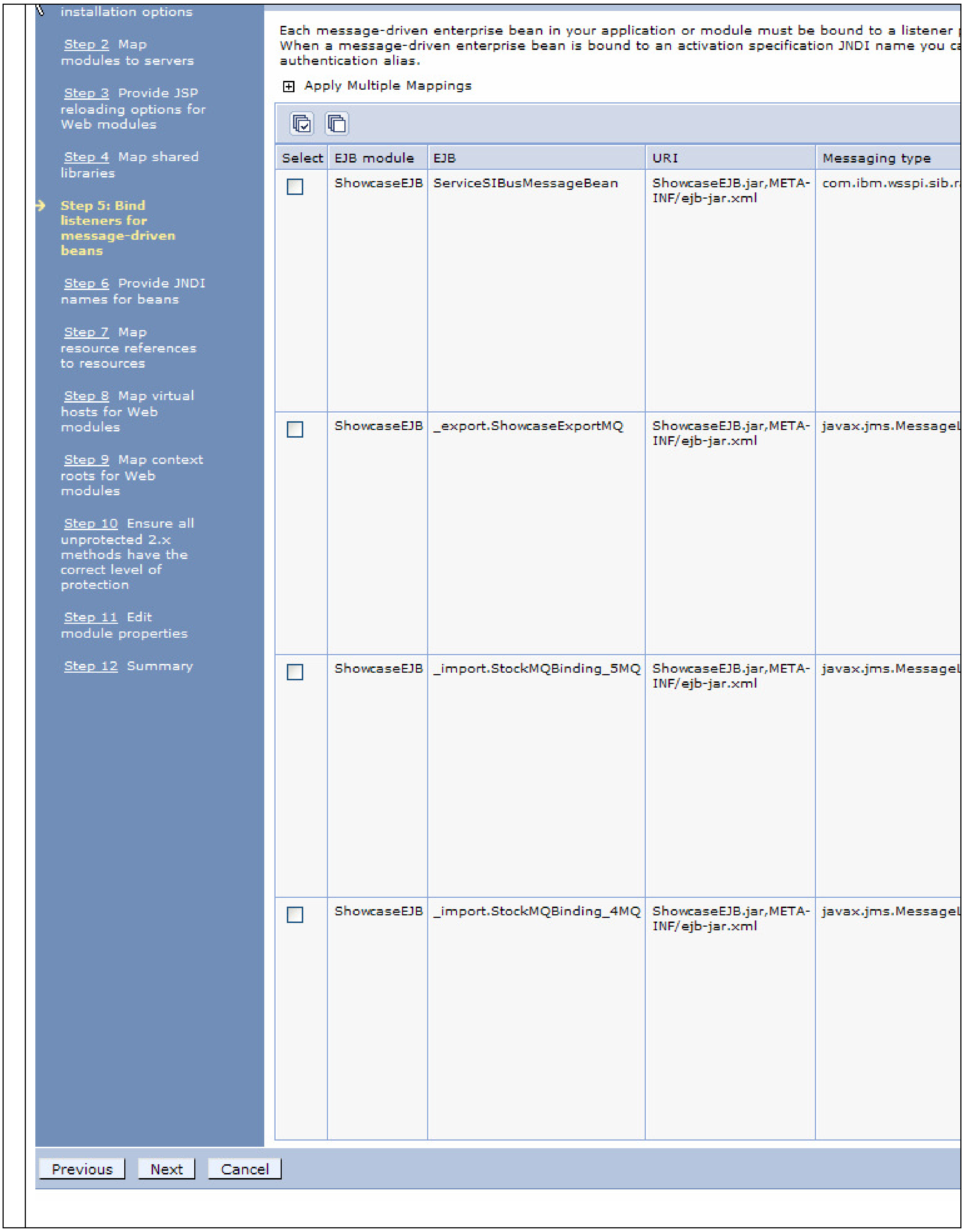
Map shared libraries

Specify shared libraries that the application or individual modules reference. These libra
appropriate scope.

Reference shared libraries

Select	Application	URI
☐	ShowcaseApp	META-INF/application.xml

Select	Module	URI
☐	ShowcaseWeb	ShowcaseWeb.war,WEB-INF/web.xml



installation options

Step 2 Map modules to servers

Step 3 Provide JSP reloading options for Web modules

Step 4 Map shared libraries

Step 5: Bind listeners for message-driven beans

Step 6 Provide JNDI names for beans

Step 7 Map resource references to resources

Step 8 Map virtual hosts for Web modules

Step 9 Map context roots for Web modules

Step 10 Ensure all unprotected 2.x methods have the correct level of protection

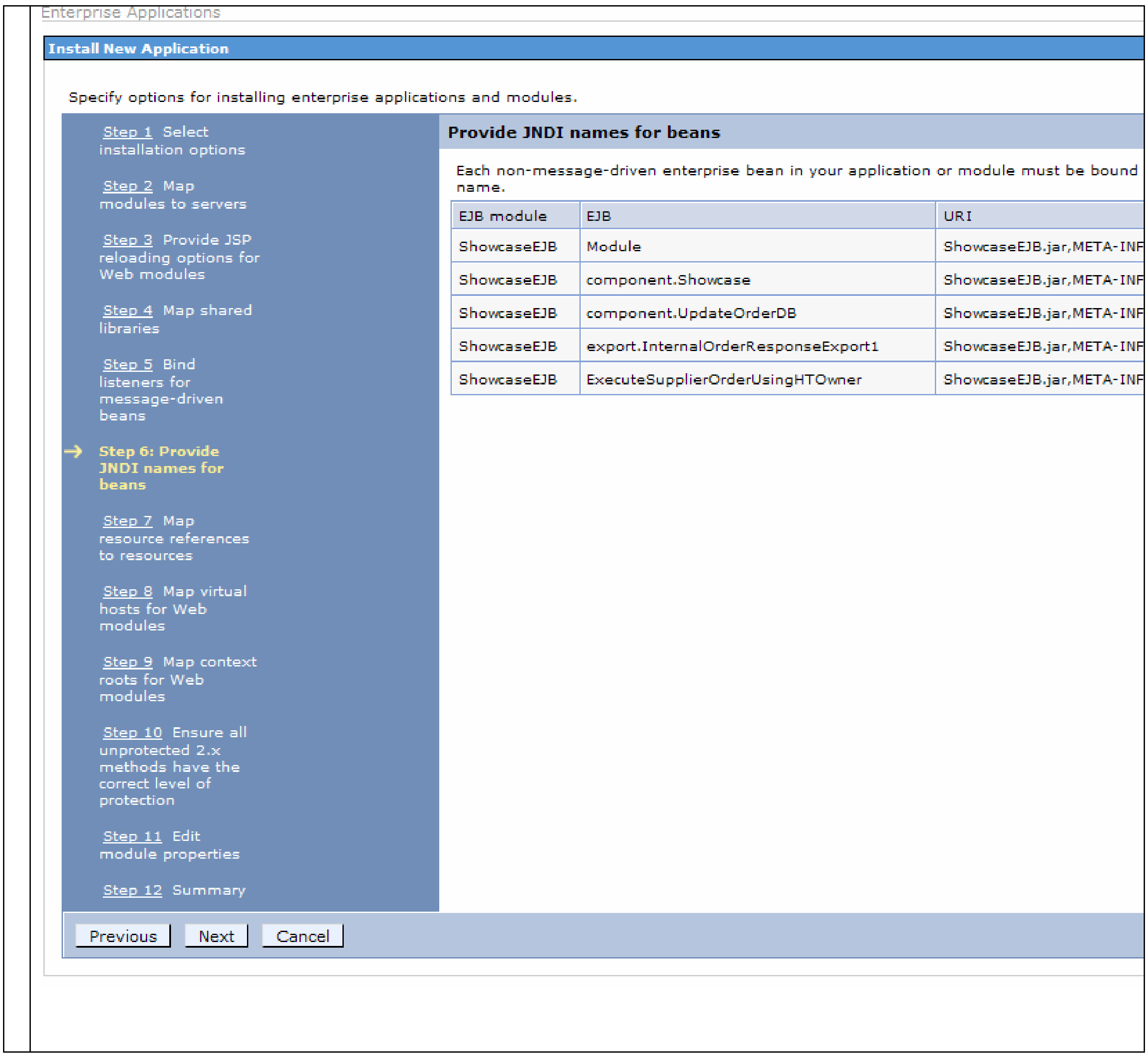
Step 11 Edit module properties

Step 12 Summary

Each message-driven enterprise bean in your application or module must be bound to a listener
When a message-driven enterprise bean is bound to an activation specification JNDI name you c
authentication alias.

⊞ Apply Multiple Mappings

Select	EJB module	EJB	URI	Messaging type
☐	ShowcaseEJB	ServiceSIBusMessageBean	ShowcaseEJB.jar,META-INF/ejb-jar.xml	com.ibm.wsspi.sib.r
☐	ShowcaseEJB	_export.ShowcaseExportMQ	ShowcaseEJB.jar,META-INF/ejb-jar.xml	javax.jms.Messagel
☐	ShowcaseEJB	_import.StockMQBinding_5MQ	ShowcaseEJB.jar,META-INF/ejb-jar.xml	javax.jms.Messagel
☐	ShowcaseEJB	_import.StockMQBinding_4MQ	ShowcaseEJB.jar,META-INF/ejb-jar.xml	javax.jms.Messagel


Enterprise Applications

Install New Application

Specify options for installing enterprise applications and modules.

Step 1 Select installation options

Step 2 Map modules to servers

Step 3 Provide JSP reloading options for Web modules

Step 4 Map shared libraries

Step 5 Bind listeners for message-driven beans

→ **Step 6: Provide JNDI names for beans**

Step 7 Map resource references to resources

Step 8 Map virtual hosts for Web modules

Step 9 Map context roots for Web modules

Step 10 Ensure all unprotected 2.x methods have the correct level of protection

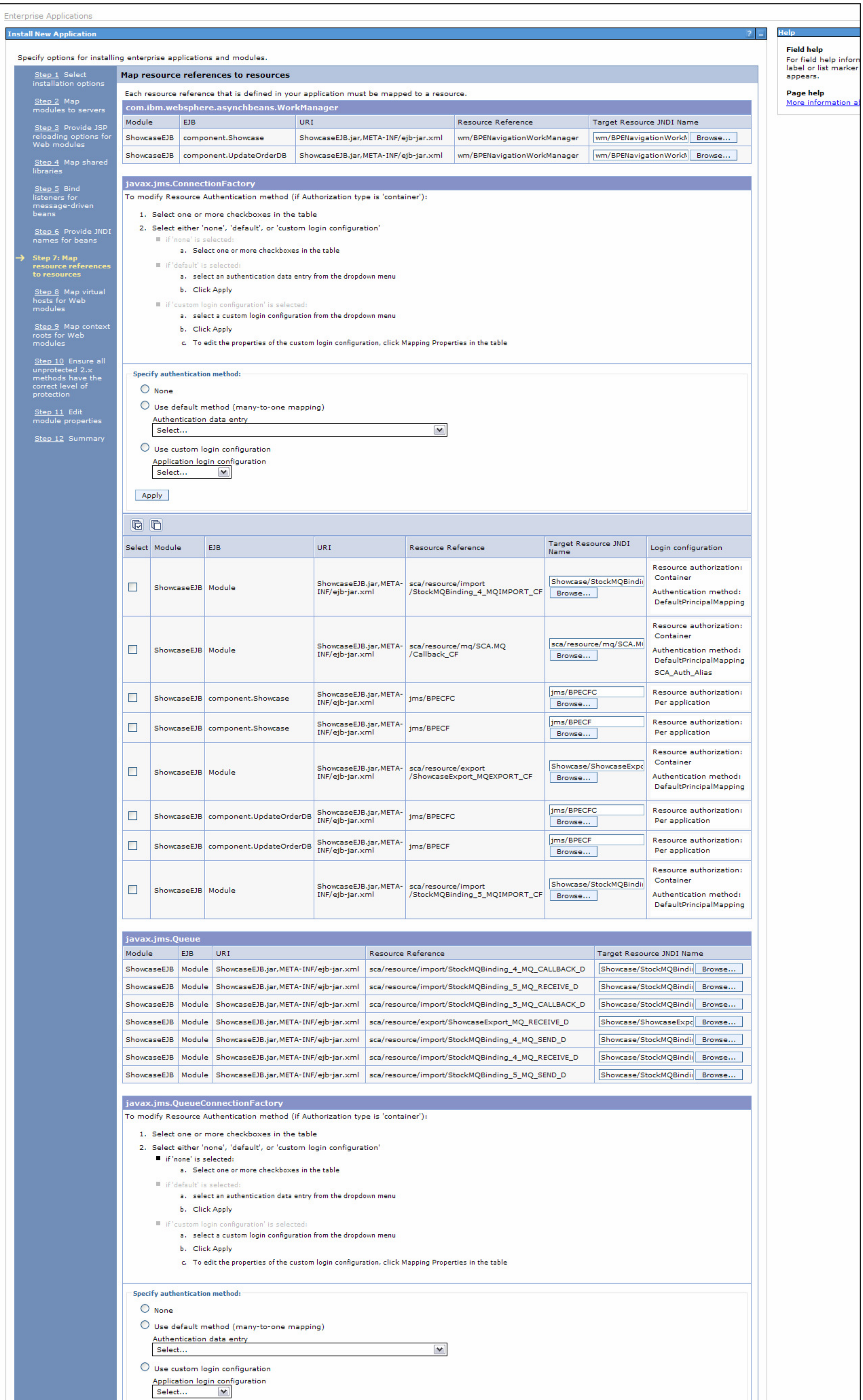
Step 11 Edit module properties

Step 12 Summary

Provide JNDI names for beans

Each non-message-driven enterprise bean in your application or module must be bound name.

EJB module	EJB	URI
ShowcaseEJB	Module	ShowcaseEJB.jar,META-INF
ShowcaseEJB	component.Showcase	ShowcaseEJB.jar,META-INF
ShowcaseEJB	component.UpdateOrderDB	ShowcaseEJB.jar,META-INF
ShowcaseEJB	export.InternalOrderResponseExport1	ShowcaseEJB.jar,META-INF
ShowcaseEJB	ExecuteSupplierOrderUsingHTOwner	ShowcaseEJB.jar,META-INF



Enterprise Applications

Install New Application

Specify options for installing enterprise applications and modules.

Step 1 Select installation options
Step 2 Map modules to servers
Step 3 Provide JSP reloading options for Web modules
Step 4 Map shared libraries
Step 5 Bind listeners for message-driven beans
Step 6 Provide JNDI names for beans
→ **Step 7: Map resource references to resources**
Step 8 Map virtual hosts for Web modules
Step 9 Map context roots for Web modules
Step 10 Ensure all unprotected 2.x methods have the correct level of protection
Step 11 Edit module properties
Step 12 Summary

Map resource references to resources

Each resource reference that is defined in your application must be mapped to a resource.

com.ibm.websphere.asynchbeans.WorkManager

Module	EJB	URI	Resource Reference	Target Resource JNDI Name
ShowcaseEJB	component.Showcase	ShowcaseEJB.jar,META-INF/ejb-jar.xml	wm/BPENavigationWorkManager	wm/BPENavigationWorkN Browse...
ShowcaseEJB	component.UpdateOrderDB	ShowcaseEJB.jar,META-INF/ejb-jar.xml	wm/BPENavigationWorkManager	wm/BPENavigationWorkN Browse...

javax.jms.ConnectionFactory

To modify Resource Authentication method (if Authorization type is 'container'):

1. Select one or more checkboxes in the table
2. Select either 'none', 'default', or 'custom login configuration'
   - if 'none' is selected:
     a. Select one or more checkboxes in the table
   - if 'default' is selected:
     a. select an authentication data entry from the dropdown menu
     b. Click Apply
   - if 'custom login configuration' is selected:
     a. select a custom login configuration from the dropdown menu
     b. Click Apply
     c. To edit the properties of the custom login configuration, click Mapping Properties in the table

Specify authentication method:

○ None
○ Use default method (many-to-one mapping)
Authentication data entry
Select...
○ Use custom login configuration
Application login configuration
Select...
Apply

Select	Module	EJB	URI	Resource Reference	Target Resource JNDI Name	Login configuration
☐	ShowcaseEJB	Module	ShowcaseEJB.jar,META-INF/ejb-jar.xml	sca/resource/import /StockMQBinding_4_MQIMPORT_CF	Showcase/StockMQBindi Browse...	Resource authorization: Container Authentication method: DefaultPrincipalMapping
☐	ShowcaseEJB	Module	ShowcaseEJB.jar,META-INF/ejb-jar.xml	sca/resource/mq/SCA.MQ /Callback_CF	sca/resource/mq/SCA.M Browse...	Resource authorization: Container Authentication method: DefaultPrincipalMapping SCA_Auth_Alias
☐	ShowcaseEJB	component.Showcase	ShowcaseEJB.jar,META-INF/ejb-jar.xml	jms/BPECFC	jms/BPECFC Browse...	Resource authorization: Per application
☐	ShowcaseEJB	component.Showcase	ShowcaseEJB.jar,META-INF/ejb-jar.xml	jms/BPECF	jms/BPECF Browse...	Resource authorization: Per application
☐	ShowcaseEJB	Module	ShowcaseEJB.jar,META-INF/ejb-jar.xml	sca/resource/export /ShowcaseExport_MQEXPORT_CF	Showcase/ShowcaseExpo Browse...	Resource authorization: Container Authentication method: DefaultPrincipalMapping
☐	ShowcaseEJB	component.UpdateOrderDB	ShowcaseEJB.jar,META-INF/ejb-jar.xml	jms/BPECFC	jms/BPECFC Browse...	Resource authorization: Per application
☐	ShowcaseEJB	component.UpdateOrderDB	ShowcaseEJB.jar,META-INF/ejb-jar.xml	jms/BPECF	jms/BPECF Browse...	Resource authorization: Per application
☐	ShowcaseEJB	Module	ShowcaseEJB.jar,META-INF/ejb-jar.xml	sca/resource/import /StockMQBinding_5_MQIMPORT_CF	Showcase/StockMQBindi Browse...	Resource authorization: Container Authentication method: DefaultPrincipalMapping

javax.jms.Queue

Module	EJB	URI	Resource Reference	Target Resource JNDI Name
ShowcaseEJB	Module	ShowcaseEJB.jar,META-INF/ejb-jar.xml	sca/resource/import/StockMQBinding_4_MQ_CALLBACK_D	Showcase/StockMQBindi Browse...
ShowcaseEJB	Module	ShowcaseEJB.jar,META-INF/ejb-jar.xml	sca/resource/import/StockMQBinding_5_MQ_RECEIVE_D	Showcase/StockMQBindi Browse...
ShowcaseEJB	Module	ShowcaseEJB.jar,META-INF/ejb-jar.xml	sca/resource/import/StockMQBinding_5_MQ_CALLBACK_D	Showcase/StockMQBindi Browse...
ShowcaseEJB	Module	ShowcaseEJB.jar,META-INF/ejb-jar.xml	sca/resource/export/ShowcaseExport_MQ_RECEIVE_D	Showcase/ShowcaseExpo Browse...
ShowcaseEJB	Module	ShowcaseEJB.jar,META-INF/ejb-jar.xml	sca/resource/import/StockMQBinding_4_MQ_SEND_D	Showcase/StockMQBindi Browse...
ShowcaseEJB	Module	ShowcaseEJB.jar,META-INF/ejb-jar.xml	sca/resource/import/StockMQBinding_4_MQ_RECEIVE_D	Showcase/StockMQBindi Browse...
ShowcaseEJB	Module	ShowcaseEJB.jar,META-INF/ejb-jar.xml	sca/resource/import/StockMQBinding_5_MQ_SEND_D	Showcase/StockMQBindi Browse...

javax.jms.QueueConnectionFactory

To modify Resource Authentication method (if Authorization type is 'container'):

1. Select one or more checkboxes in the table
2. Select either 'none', 'default', or 'custom login configuration'
   - if 'none' is selected:
     a. Select one or more checkboxes in the table
   - if 'default' is selected:
     a. select an authentication data entry from the dropdown menu
     b. Click Apply
   - if 'custom login configuration' is selected:
     a. select a custom login configuration from the dropdown menu
     b. Click Apply
     c. To edit the properties of the custom login configuration, click Mapping Properties in the table

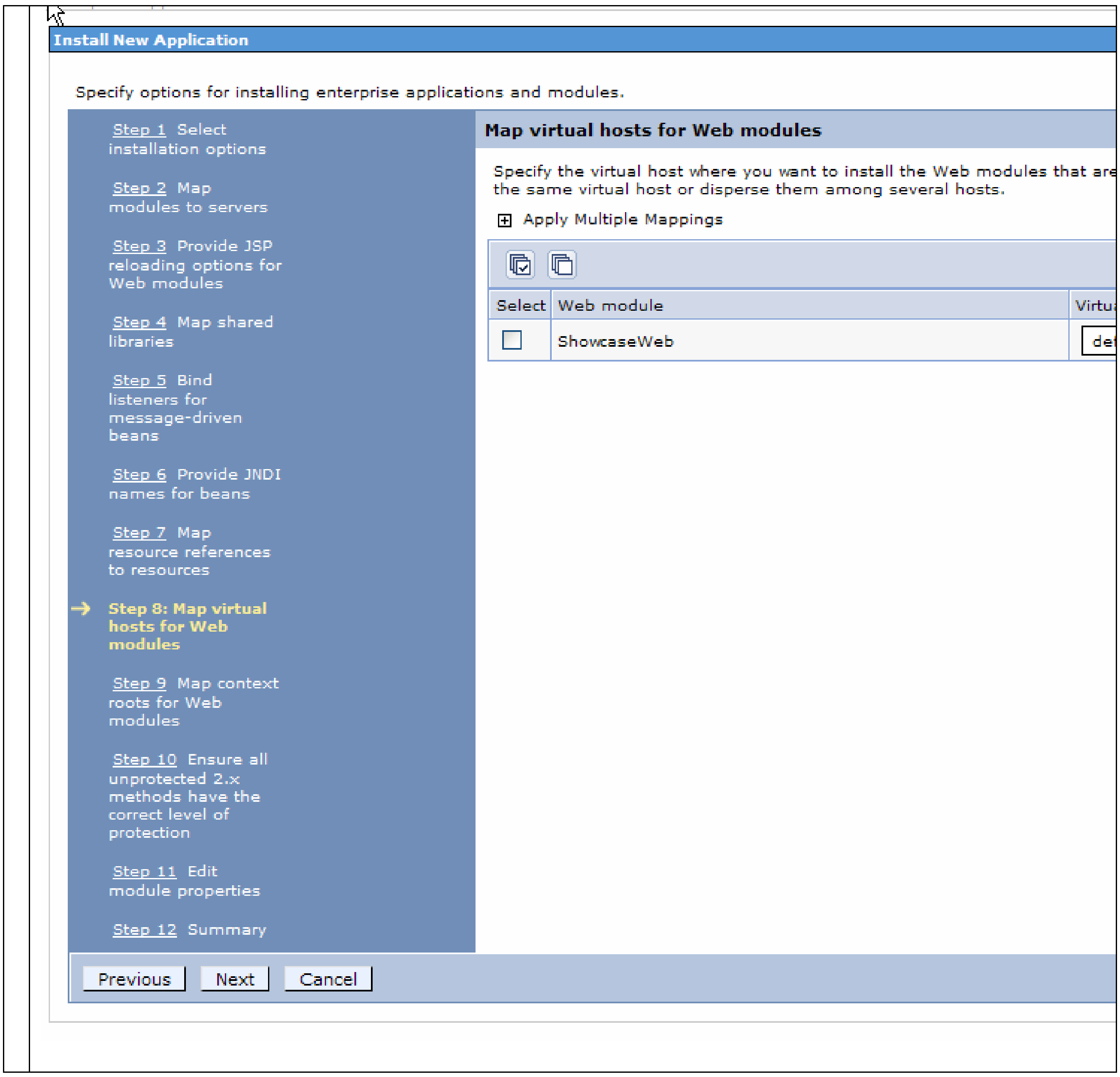
Specify authentication method:

○ None
○ Use default method (many-to-one mapping)
Authentication data entry
Select...
○ Use custom login configuration
Application login configuration
Select...

Help

Field help
For field help inforn
label or list marker
appears.

Page help
More information a


Install New Application

Specify options for installing enterprise applications and modules.

Step 1 Select installation options

Step 2 Map modules to servers

Step 3 Provide JSP reloading options for Web modules

Step 4 Map shared libraries

Step 5 Bind listeners for message-driven beans

Step 6 Provide JNDI names for beans

Step 7 Map resource references to resources

→ **Step 8: Map virtual hosts for Web modules**

Step 9 Map context roots for Web modules

Step 10 Ensure all unprotected 2.x methods have the correct level of protection

Step 11 Edit module properties

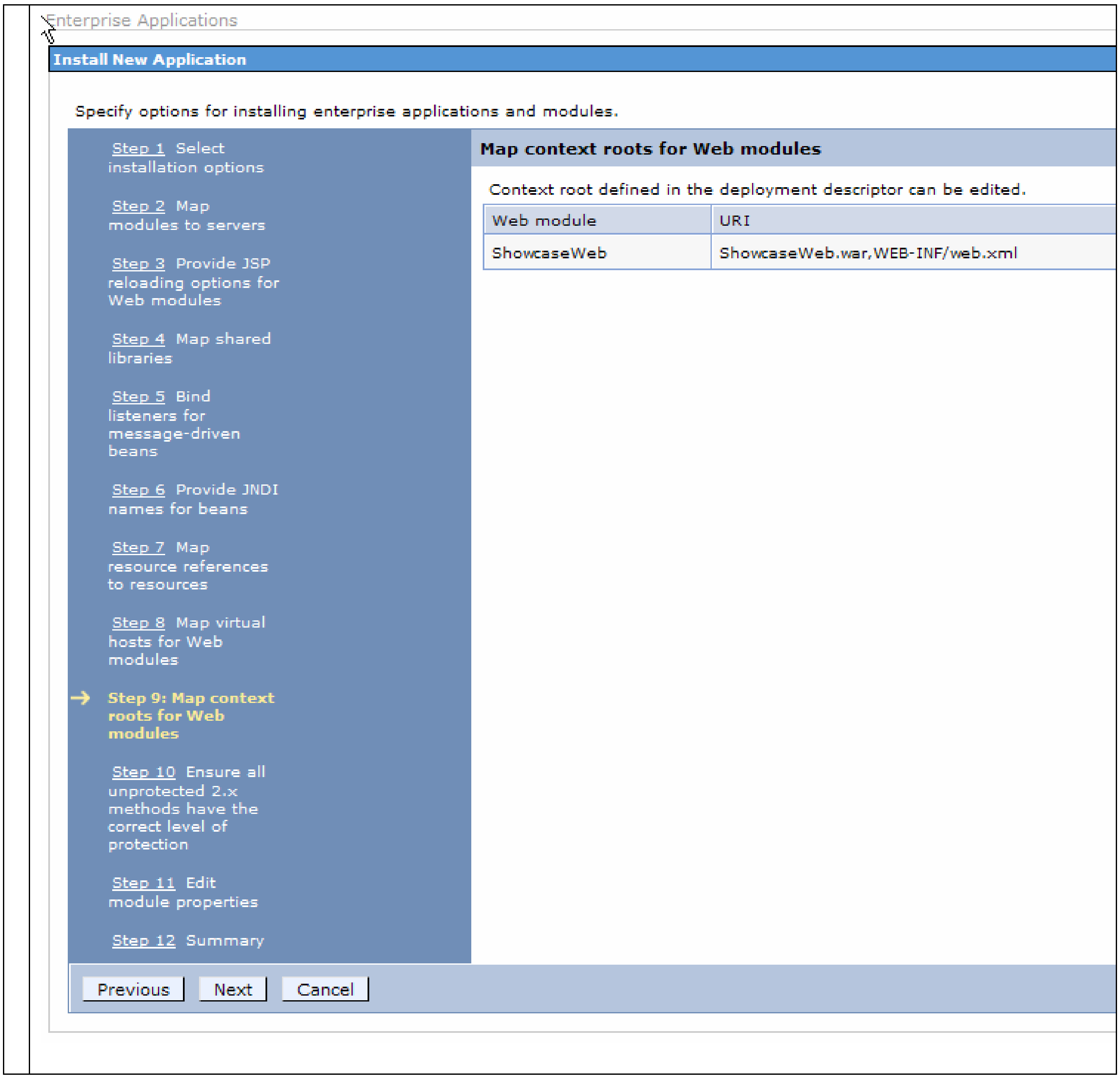
Step 12 Summary

Map virtual hosts for Web modules

Specify the virtual host where you want to install the Web modules that are
the same virtual host or disperse them among several hosts.

⊞ Apply Multiple Mappings

Select	Web module	Virtu
☐	ShowcaseWeb	de


Enterprise Applications

Install New Application

Specify options for installing enterprise applications and modules.

Step 1 Select installation options

Step 2 Map modules to servers

Step 3 Provide JSP reloading options for Web modules

Step 4 Map shared libraries

Step 5 Bind listeners for message-driven beans

Step 6 Provide JNDI names for beans

Step 7 Map resource references to resources

Step 8 Map virtual hosts for Web modules

→ Step 9: Map context roots for Web modules

Step 10 Ensure all unprotected 2.x methods have the correct level of protection

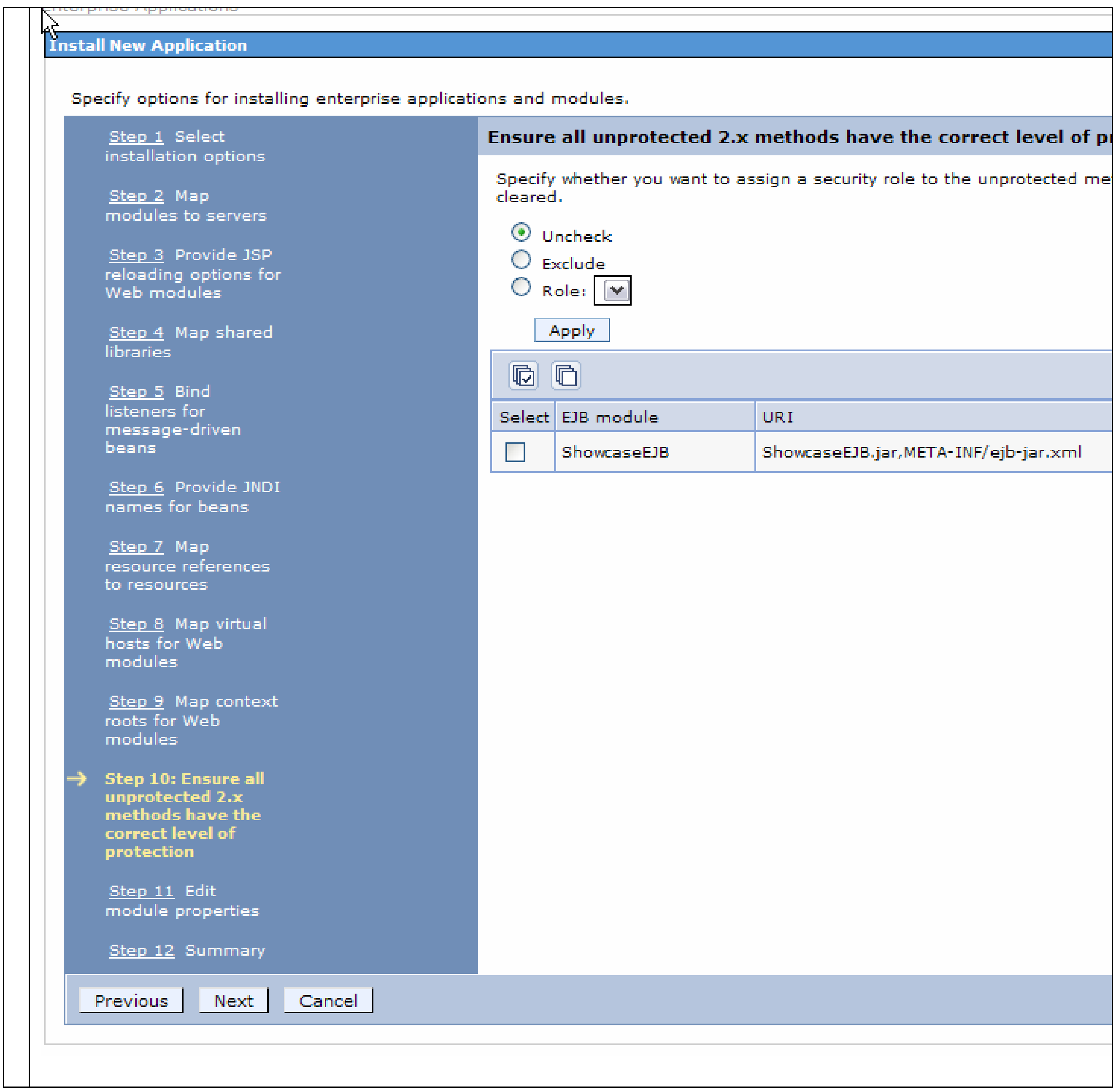
Step 11 Edit module properties

Step 12 Summary

Map context roots for Web modules

Context root defined in the deployment descriptor can be edited.

Web module	URI
ShowcaseWeb	ShowcaseWeb.war,WEB-INF/web.xml



Install New Application

Specify options for installing enterprise applications and modules.

Step 1 Select installation options

Step 2 Map modules to servers

Step 3 Provide JSP reloading options for Web modules

Step 4 Map shared libraries

Step 5 Bind listeners for message-driven beans

Step 6 Provide JNDI names for beans

Step 7 Map resource references to resources

Step 8 Map virtual hosts for Web modules

Step 9 Map context roots for Web modules

→ **Step 10: Ensure all unprotected 2.x methods have the correct level of protection**

Step 11 Edit module properties

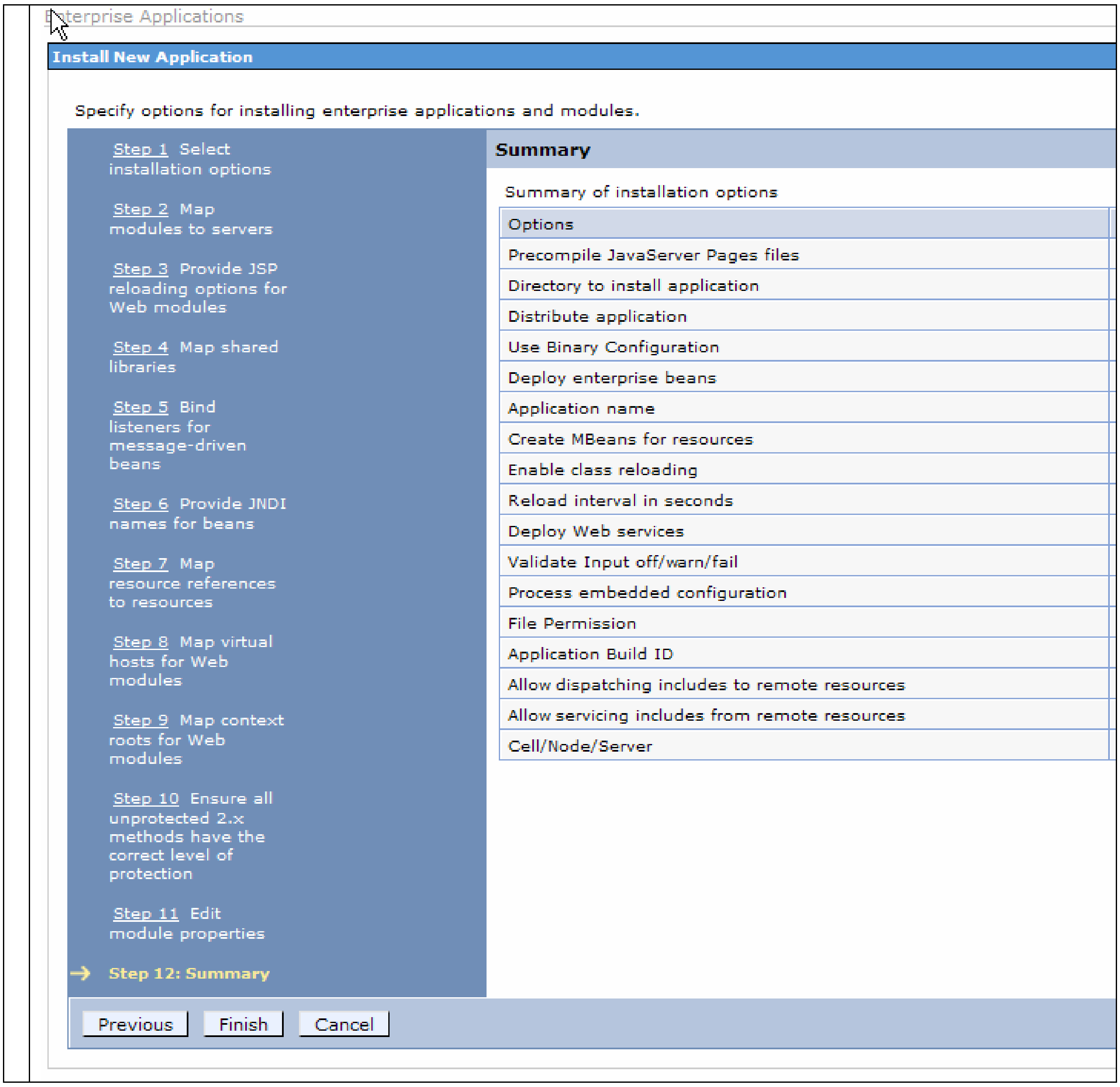
Step 12 Summary

Ensure all unprotected 2.x methods have the correct level of pr

Specify whether you want to assign a security role to the unprotected me cleared.

[✓ Uncheck] Exclude Role:

Select	EJB module	URI
☐	ShowcaseEJB	ShowcaseEJB.jar,META-INF/ejb-jar.xml


Enterprise Applications

Install New Application

Specify options for installing enterprise applications and modules.

Step 1 Select installation options

Step 2 Map modules to servers

Step 3 Provide JSP reloading options for Web modules

Step 4 Map shared libraries

Step 5 Bind listeners for message-driven beans

Step 6 Provide JNDI names for beans

Step 7 Map resource references to resources

Step 8 Map virtual hosts for Web modules

Step 9 Map context roots for Web modules

Step 10 Ensure all unprotected 2.x methods have the correct level of protection

Step 11 Edit module properties

→ **Step 12: Summary**

Summary

Summary of installation options

Options
Precompile JavaServer Pages files
Directory to install application
Distribute application
Use Binary Configuration
Deploy enterprise beans
Application name
Create MBeans for resources
Enable class reloading
Reload interval in seconds
Deploy Web services
Validate Input off/warn/fail
Process embedded configuration
File Permission
Application Build ID
Allow dispatching includes to remote resources
Allow servicing includes from remote resources
Cell/Node/Server

8.4 WebSphere Message Broker resources

8.4.1 Message Flow definition

Projectname	Content	Interaction Step
CheckStock_4	CheckStockMQ_5.msgflow CheckStockSOAP_6.msgflow	4a 5
InternalOrder_7	InternalOrder_7Flow.msgflow	7
Supplier_13	Supplier.msgflow	13

SupplierOrder_16	SupplierOrder_16.msgflow	15/16

8.4.2 Queue definition

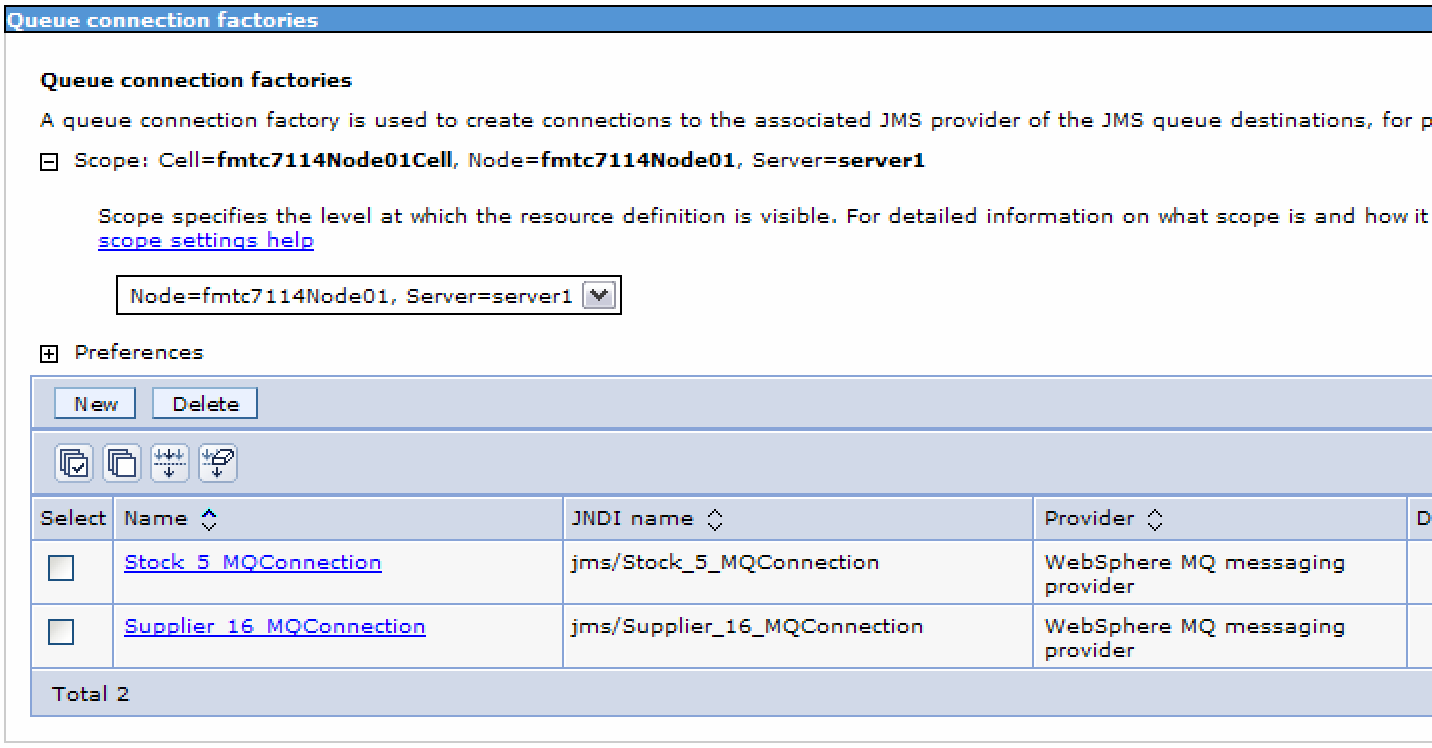
Name	Type	Content	Interaction Step
STOCK_5_INPUT_J2EE	queue		4d
STOCK_5_INPUT_WPS	queue		4c
STOCK_5_OUTPUT_J2EE	queue		4b
STOCK_5_OUTPUT_WPS	queue		4a
STOCK_6_INPUT_WPS	queue		5d
STOCK_6_OUTPUT_WPS	queue		5a
SUPPLIER_16_INPUT	queue		16.1
SUPPLIER_16_OUTPUT	queue		16.1

8.5 WebSphere Application Server resources

Projectname	Type	Content	Interaction Step
StartProcessEAR_3 StartProcessWeb_3	JSF	Start Process	3
CheckStock1_EAR_5 CheckStock1MdbEjb_5	MDB	Reads MQ request message ; modify payload; sends MQ message	5
CheckStock2EAR_6 CheckStock2WAR_6	Web Service	Returns stock amount, via SOAP/HTTP	6
InternalOrderEAR_8 InternalOrderWAR_8	Web Service	Executes internal order, via SOAP/HTTP	8
HumanTaskInterfaceEAR_9 HumanTaskInterfaceWAR_9	JSF	HumanTask Web Service API client	9
InternalSupplierOrderEAR_16 InternalSupplierOrderEJB_16	Web Service MQ		16
OrderDBEAR_17 OrderDBEntity OrderDBSession	EntityBean	Insert into DB	17

8.5.1 JMS Connection Factory resources

Create following two JMS Queue Connection Factories:


Queue connection factories

Queue connection factories

A queue connection factory is used to create connections to the associated JMS provider of the JMS queue destinations, for p

⊟ Scope: Cell=**fmtc7114Node01Cell**, Node=**fmtc7114Node01**, Server=**server1**

Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it
scope settings help

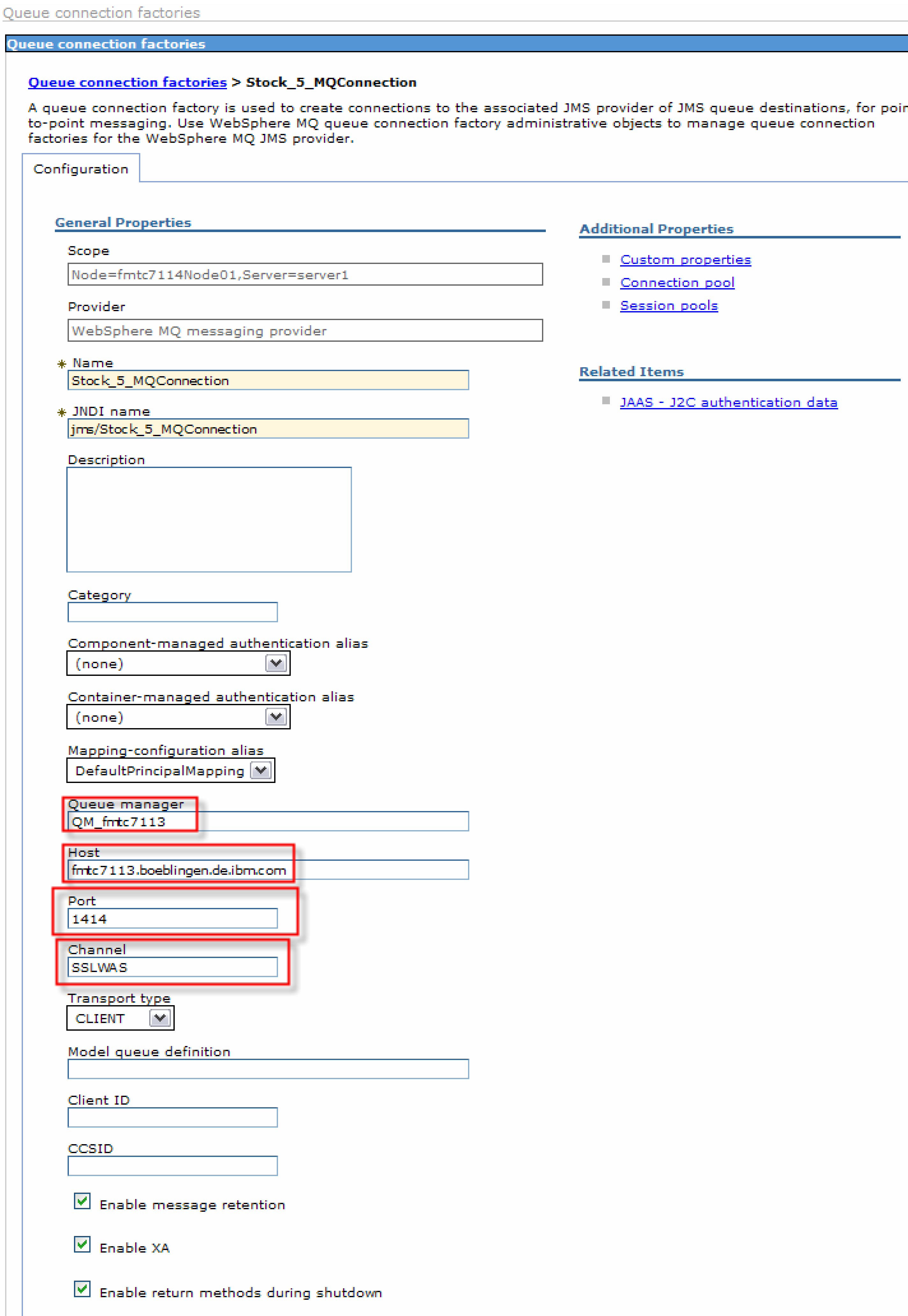
Node=fmtc7114Node01, Server=server1

⊞ Preferences

New | Delete

Select	Name	JNDI name	Provider	D
☐	Stock_5_MQConnection	jms/Stock_5_MQConnection	WebSphere MQ messaging provider	
☐	Supplier_16_MQConnection	jms/Supplier_16_MQConnection	WebSphere MQ messaging provider	

Total 2


Queue connection factories

Queue connection factories

Queue connection factories > Stock_5_MQConnection

A queue connection factory is used to create connections to the associated JMS provider of JMS queue destinations, for poir
to-point messaging. Use WebSphere MQ queue connection factory administrative objects to manage queue connection
factories for the WebSphere MQ JMS provider.

Configuration

General Properties

Scope
Node=fmtc7114Node01,Server=server1

Provider
WebSphere MQ messaging provider

* Name
Stock_5_MQConnection

* JNDI name
jms/Stock_5_MQConnection

Description

Category

Component-managed authentication alias
(none)

Container-managed authentication alias
(none)

Mapping-configuration alias
DefaultPrincipalMapping

Queue manager
QM_fmtc7113

Host
fmtc7113.boeblingen.de.ibm.com

Port
1414

Channel
SSLWAS

Transport type
CLIENT

Model queue definition

Client ID

CCSID

✓ Enable message retention

✓ Enable XA

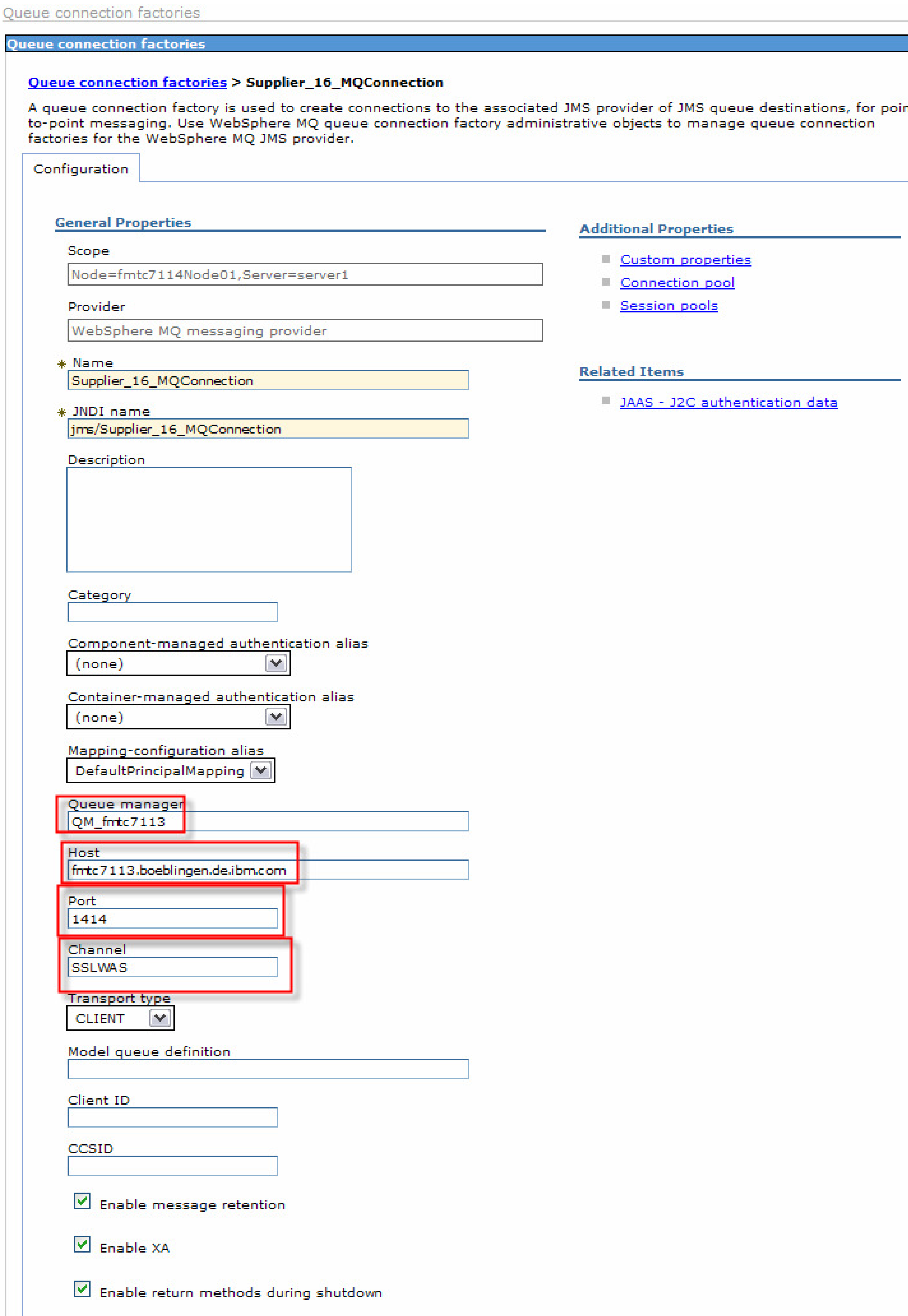
✓ Enable return methods during shutdown

Additional Properties

- Custom properties
- Connection pool
- Session pools

Related Items

- JAAS - J2C authentication data


Queue connection factories

Queue connection factories

Queue connection factories > **Supplier_16_MQConnection**

A queue connection factory is used to create connections to the associated JMS provider of JMS queue destinations, for poir to-point messaging. Use WebSphere MQ queue connection factory administrative objects to manage queue connection factories for the WebSphere MQ JMS provider.

Configuration

General Properties

Scope: Node=fmtc7114Node01,Server=server1

Provider: WebSphere MQ messaging provider

* Name: Supplier_16_MQConnection

* JNDI name: jms/Supplier_16_MQConnection

Description:

Category:

Component-managed authentication alias: (none)

Container-managed authentication alias: (none)

Mapping-configuration alias: DefaultPrincipalMapping

Queue manager: QM_fmtc7113

Host: fmtc7113.boeblingen.de.ibm.com

Port: 1414

Channel: SSLWAS

Transport type: CLIENT

Model queue definition:

Client ID:

CCSID:

✓ Enable message retention

✓ Enable XA

✓ Enable return methods during shutdown

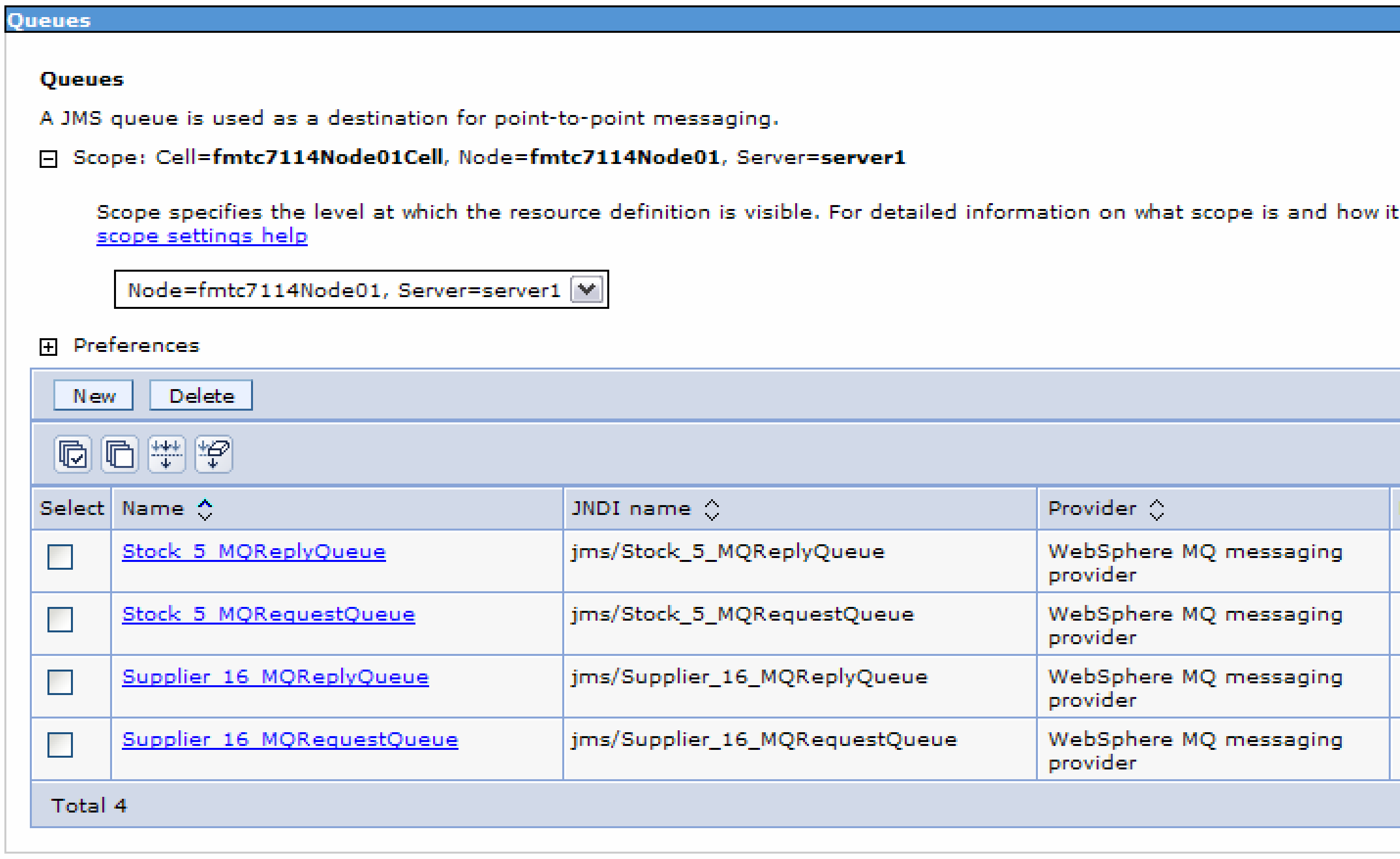
Additional Properties

- Custom properties
- Connection pool
- Session pools

Related Items

- JAAS - J2C authentication data

JMS Queues:


Queues

Queues

A JMS queue is used as a destination for point-to-point messaging.

⊟ Scope: Cell=**fmtc7114Node01Cell**, Node=**fmtc7114Node01**, Server=**server1**

Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it
scope settings help

Node=fmtc7114Node01, Server=server1

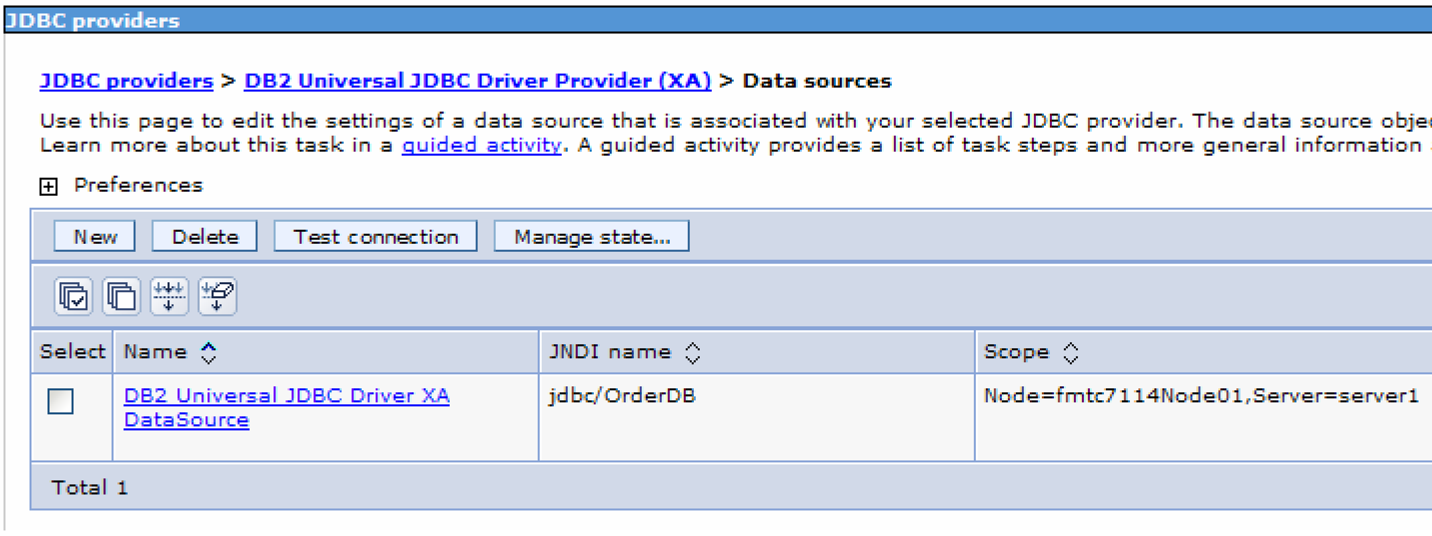
⊞ Preferences

New | Delete

Select	Name	JNDI name	Provider
☐	Stock_5_MQReplyQueue	jms/Stock_5_MQReplyQueue	WebSphere MQ messaging provider
☐	Stock_5_MQRequestQueue	jms/Stock_5_MQRequestQueue	WebSphere MQ messaging provider
☐	Supplier_16_MQReplyQueue	jms/Supplier_16_MQReplyQueue	WebSphere MQ messaging provider
☐	Supplier_16_MQRequestQueue	jms/Supplier_16_MQRequestQueue	WebSphere MQ messaging provider

Total 4

8.5.2 Configuration of JDBC resources


JDBC providers

JDBC providers > DB2 Universal JDBC Driver Provider (XA) > Data sources

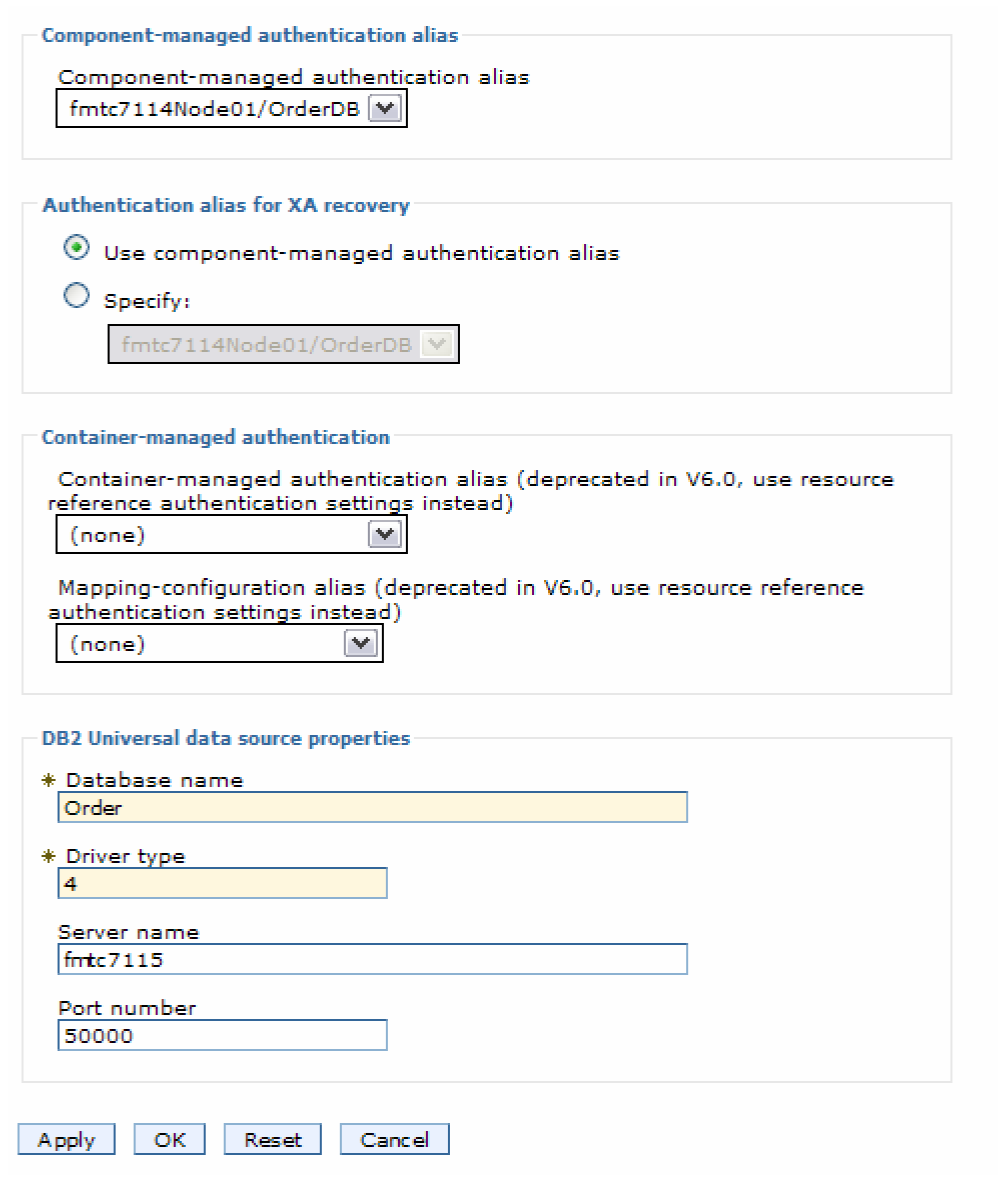
Use this page to edit the settings of a data source that is associated with your selected JDBC provider. The data source obje
Learn more about this task in a guided activity. A guided activity provides a list of task steps and more general information

⊞ Preferences

New | Delete | Test connection | Manage state...

Select	Name	JNDI name	Scope
☐	DB2 Universal JDBC Driver XA DataSource	jdbc/OrderDB	Node=fmtc7114Node01,Server=server1

Total 1



Component-managed authentication alias

Component-managed authentication alias: fmtc7114Node01/OrderDB

Authentication alias for XA recovery

[✓ Use component-managed authentication alias]

Specify: fmtc7114Node01/OrderDB

Container-managed authentication

Container-managed authentication alias (deprecated in V6.0, use resource reference authentication settings instead): (none)

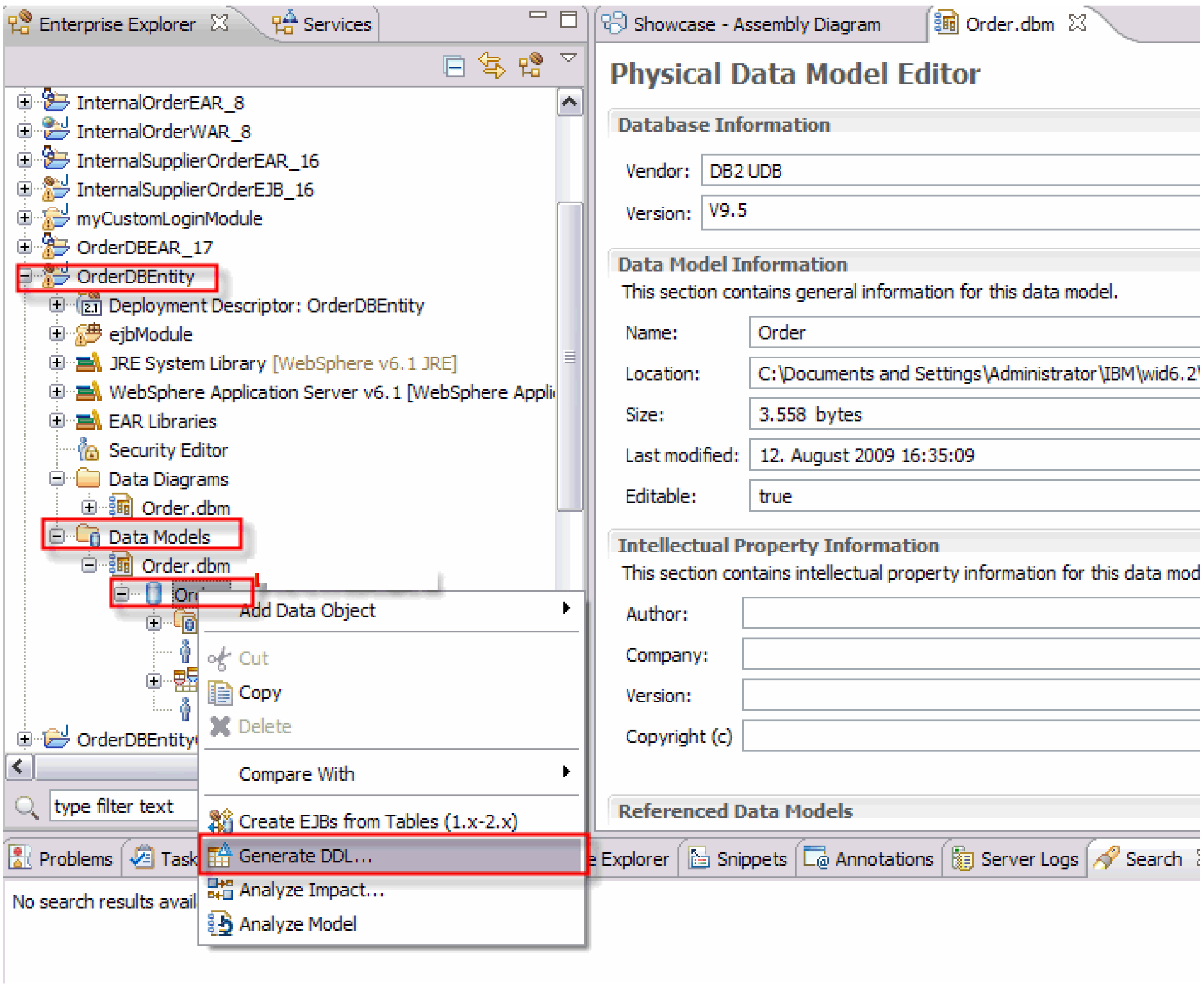
Mapping-configuration alias (deprecated in V6.0, use resource reference authentication settings instead): (none)

DB2 Universal data source properties

* Database name	Order
* Driver type	4
Server name	fmtc7115
Port number	50000

8.5.3 Configuration of DB2 for WAS applications

To create DB2 tables for the showcase interaction step 17 create DDLs using the Data Model in OrderDBEntity:



Physical Data Model Editor

Database Information

Vendor: DB2 UDB
Version: V9.5

Data Model Information

This section contains general information for this data model.

Name: Order
Location: C:\Documents and Settings\Administrator\IBM\wid6.2
Size: 3.558 bytes
Last modified: 12. August 2009 16:35:09
Editable: true

Intellectual Property Information

This section contains intellectual property information for this data mod

Author:
Company:
Version:
Copyright (c)

Referenced Data Models

In the showcase we used the db2 admin user ID to access the database during runtime. The user needs at least the rights to do sql insert, delete and recover.

8.5.4 Deployment of the WAS applications

For the deployment to WAS we use the default settings of the deployment steps.

9 Terms

Token	A security token represents a set of claims made by a client that may include a name, password, identity, key, certificate, group, or privilege. Web services security provides a general-purpose mechanism to associate security tokens with messages for single-message authentication. A specific type of security token is not required by Web services security
Username Token	A Username Token consists of a user name and, optionally, password information
Asserted (Username) Token	A asserted Username Token consists of a user name without password information
LTPA Token	Lightweight Third-Party Authentication Token. Encrypted Token, carries User identiy. Prereq for use is, that servers exchange their LTPA keys.
Identity assertion	When using the identity assertion (IDAssertion) authentication method, the security token generated is a <wsse:UsernameToken> element that contains a <wsse:Username> element. On the request sender side, a callback handler is invoked to generate the security token. On the request receiver side, the security token is validated.
Identity propagation	An identity is carried within a request call from one system to another system

10 Abbreviations

BPC	Business Process Choreographer
HTM	Human Task Container
WAS	WebSphere Application Server
WID	WebSphere Integration Developer
WMB	WebSphere Message Broker
MQ	WebSphere MQ
WPS	WebSphere Process Server

11 Referenced Documents

WPS

[WPS01] WID info center
http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.wbit.620.help.nav.doc/topics/welcome.html

[WPS02] WPS info center
http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.websphere.wps.620.doc/welcome_wps.html

[WPS03] Pamela Fong: Asynchronous Processing in WebSphere Process Server, Asynchronous processing in WebSphere Process Server
http://www.ibm.com/developerworks/websphere/library/techarticles/0904_fong/0904_fong.html

[WPS04] IBM Redbook, Using IBM WebSphere Message Broker as an ESB with WebSphere Process Server, http://www.redbooks.ibm.com/redbooks/pdfs/sg247527.pdf

Message Broker

[WMB01] WMB info center

[WMB02] IBM Redpaper, Using the New Features in WebSphere Message Broker V6.1, http://www.redbooks.ibm.com/abstracts/redp4458.html?Open

[WMB03] Mike Johnson , Signing Flows for WebServices Security, http://www.ibm.com/developerworks/library/ws-security/index.html
Summary: Set up Web Services Security (WS-Security) for signing data that your applications send to and receive from IBM® WebSphere® Message Broker. This article describes basic concepts, how to set up the environment, and how to configure WebSphere Message Broker to sign the data. The information provided here is platform-independent and operating system-independent, but you can see examples of specific operating systems where appropriate. A section on terminology at the end of this article helps clarify the concepts described.

[WMB04] Rob Henley, Matthew Golby-Kirk,
http://www.ibm.com/developerworks/websphere/library/techarticles/0902_henley/0902_henley.html
Summary: SOAP nodes in WebSphere Message Broker V6.1 send and receive SOAP-based Web services messages, enabling a message flow to interact with Web service endpoints. The messages may be plain SOAP, SOAP with Attachments (SwA), or Message Transmission Optimization Mechanism (MTOM). You can configure the nodes using WSDL, and they support the WS-Security and WS-Addressing standards. This four-part series describes the SOAP nodes, the logical tree for the new SOAP domain, configuration, and runtime behavior. Part 4 describes runtime validation, performance, scalability, message flow design, and use of WS-Addressing.

[WMB05] Rob Henley, Matthew Golby-Kirk, SOAP nodes in IBM WebSphere Message Broker V6.1, Part 1:
http://www.ibm.com/developerworks/library/ws-soapnode/index.html
SOAP nodes send and receive SOAP-based Web services messages, allowing a message flow to interact with Web service endpoints. The messages might be plain SOAP, SOAP with Attachments (SwA), or Message Transmission Optimization Mechanism (MTOM). The nodes are configured using Web Services Description Language (WSDL) and support WS-Security and WS-Addressing. This four-part series describes the SOAP nodes, the logical tree for the new SOAP domain, and details of configuration and runtime behavior. In this first article, you learn about the basic use of the nodes. You should have a general familiarity with SOAP-based Web services and WSDL to follow along with this article series.

[WMB06] Rob Henley, (rhenley@uk.ibm.com), Matthew Golby-Kirk (mgk@uk.ibm.com), SOAP nodes in IBM WebSphere Message Broker V6.1, Part 2:
http://www.ibm.com/developerworks/library/ws-soapnode2/index.html
This article, Part 2, describes the new logical tree format used by the SOAP domain. You should have a general familiarity with SOAP-based Web services and WSDL to follow along with this article series. Note: This article relates to IBM WebSphere Message Broker V6.1 Fix Pack 6.1.0.2. Some details could differ slightly from the 6.1 GA version.

[WMB07] Rob Henley, (rhenley@uk.ibm.com), Matthew Golby-Kirk (mgk@uk.ibm.com), SOAP Nodes in WebSphere Message Broker V6.1, Part 4:
http://www.ibm.com/developerworks/websphere/library/techarticles/0902_henley/0902_henley.html
SOAP nodes in WebSphere Message Broker V6.1 send and receive SOAP-based Web services messages, enabling a message flow to interact with Web service endpoints. The messages may be plain SOAP, SOAP with Attachments (SwA), or Message Transmission Optimization Mechanism (MTOM). You can configure the nodes using WSDL, and they support the WS-Security and WS-Addressing standards. This four-part series describes the SOAP nodes, the logical tree for the new SOAP domain, configuration, and runtime behavior. Part 4 describes runtime validation, performance, scalability, message flow design, and use of WS-Addressing.

WAS

[WAS01] WAS Info Center Web services security token propagation,
http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/cwbs_securitytokenPropagationwbs.html

[WAS02] WAS Security in General (Technical Library
http://www.ibm.com/developerworks/views/websphere/libraryview.jsp?end_no=100&lcl_sort_order=desc&type_by=All+Types&sort_order=desc&show_all=false&start_no=1&product_by=WebSphere+Application+Servers&search_by=&sort_by=Date&count=100&topic_by=Security&search_flag=&show_abstract=true

[WAS04] Keys Botzum, Keys Botzum's Home Page, http://www.keysbotzum.com/

[WAS05] Web Services Handbook for WebSphere Application Server Version 6.1, Chapter 19 "WS-Addressing and WS-Resource", SG247257

[WAS06] DeveloperWorks Article “Driving WS-Addressing in WebSphere Application Server Version 6.1” at http://www.ibm.com/developerworks/webservices/library/ws-soa-wsawsa/