Introduction

This article shows you step-by-step how to implement Web Services Security (WS-Security) in IBM® WebSphere® Message Broker V7 (hereafter called Message Broker). You create a message flow as a Web service provider by using the Message Broker SOAP nodes, then use the message flow to configure WS-Security for identity authentication, message signing and encryption.

Review of Web services and WS-Security

A Web service is defined by the World Wide Web Consortium (W3C) as a software system designed to support interoperable machine-to-machine interaction over a network. A Web service is described in XML via Web Service Description Language (WSDL), which provides all of the details needed to interact with the Web service, including message formats that detail the operations, transport protocols, and location, without details of how the service is implemented.

A Web service consumer uses SOAP messages to interact with the Web service, typically by using HTTP with an XML serialization in conjunction with other Web standards. A Message Broker application can participate in a Web services environment as a provider, a consumer, or both. The following pre-built SOAP nodes can be used in a message flow:

Nodes for Web service providers:

- SOAPInput
- SOAPReply

Nodes for Web service consumers:

- SOAPRequest
- SOAPAsyncRequest
- SOAPAsyncResponse

WS-Security describes a set of enhancements to SOAP messaging to provide quality of protection through message authentication, integrity, and confidentiality. Policy sets and policy set bindings are used to define and configure the WS-Security requirements. A policy set is a container for the WS-Security policy type. A policy set binding associated with a policy set contains information that is specific to the environment and platform, such as information about keys. Message Broker provides the following functions to support WS-Security:

Identity authentication

Sending security tokens as part of a message

Message confidentiality

Encrypting the message content to prevent unauthorized disclosure

Message integrity

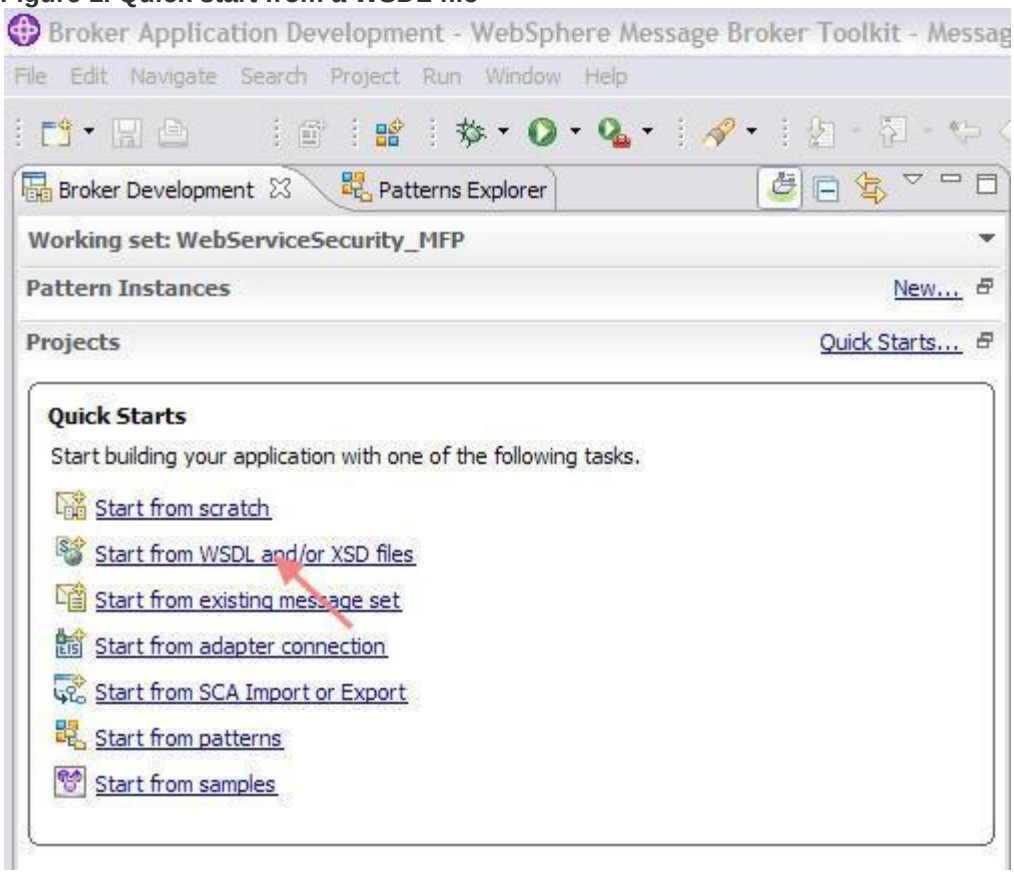
Signing the message content to prevent unauthorized and undetected modification

Creating a message flow by using the Message Broker SOAP nodes

In this article, you create a message flow as a Web service provider based on a WSDL file that describes the employee information query service. This message flow is used to configure WS-Security for authentication, signing, and encryption using the security profile, policy set, and policy set binding.

1. To create the message flow, from the Broker Application Development perspective of the Message Broker Toolkit, click **Start** from WSDL and/or XSD files:

Figure 1. Quick start from a WSDL file



2. On the Quick Start dialog, provide the project and other names as shown in Figure 2:

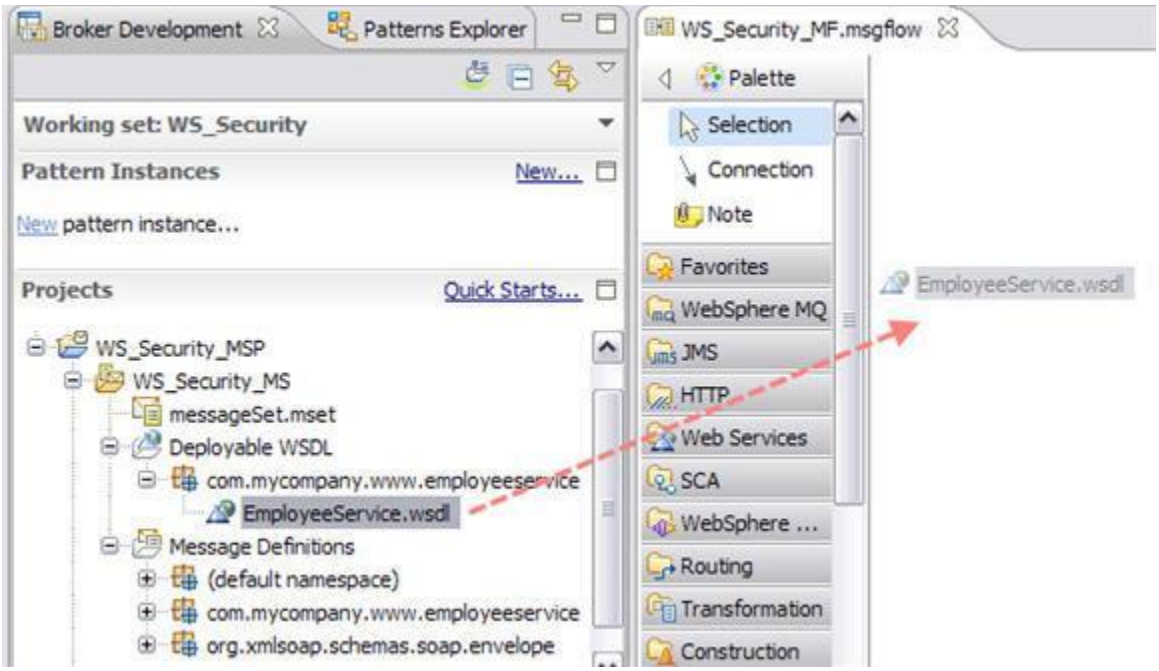
Figure 2. Setting up the basic resources

3. On the Resource Selection dialog, select **Use external resources** and provide the WSDL file EmployeeService.wsdl, as shown in Figure 3:

Figure 3. Selecting a WSDL file

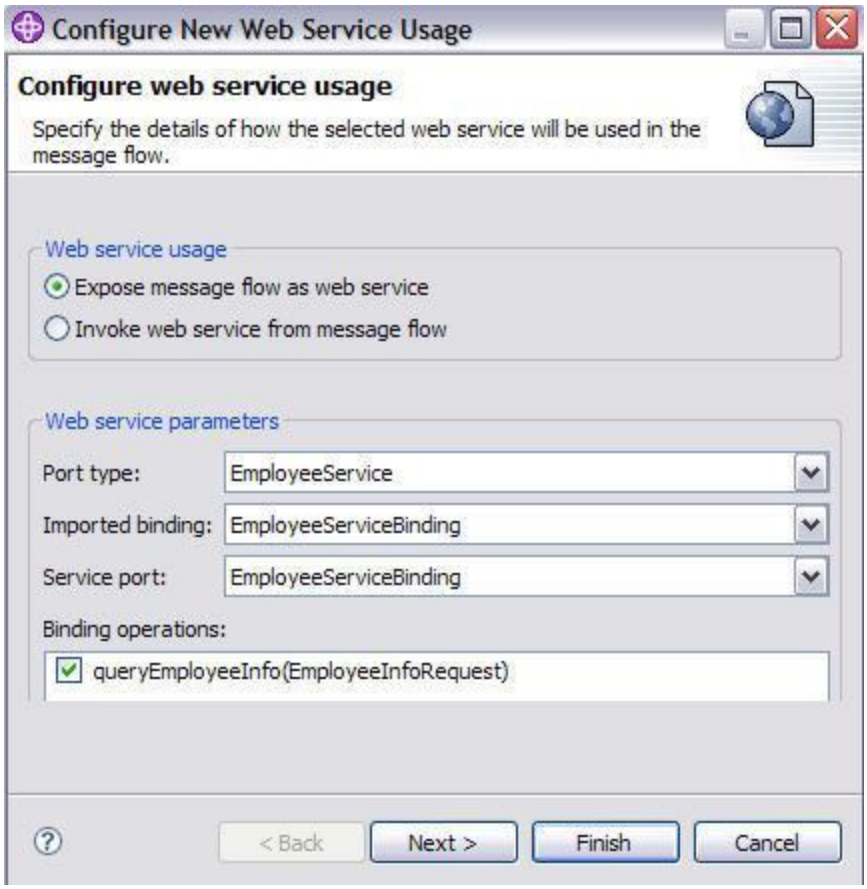
4. On the Binding Selection dialog, the **EmployeeServiceSOAP binding** should be checked by default. Click **Finish**. The message flow WS_Security_MF with no implementation is created.
5. Drag and drop the file **EmployeeService.wsdl** into the Message Flow Editor:

Figure 4. Drag and drop the WSDL file into the Flow Editor



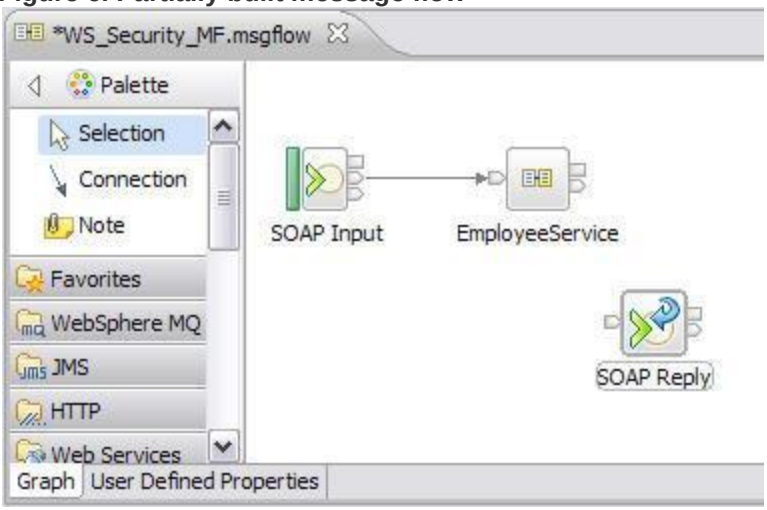
6. On the Configure Web service usage dialog, keep all the default settings, as shown in Figure 5. The message flow is built as the Web service provider:

Figure 5. Exposing the message flow as Web service provider



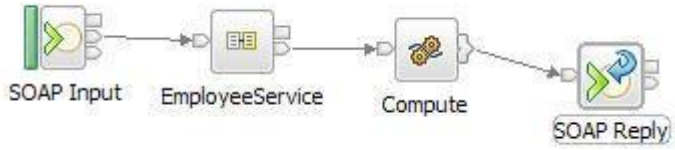
7. On the Flow Generation Details dialog, keep all the default settings. Make sure the **SOAP** node type is selected to be used in the flow. Click**Finish**. You should see the partially-built message flow, as shown in Figure 6:

Figure 6. Partially built message flow



8. Add a Compute node and wire the nodes together:

Figure 7. Adding a Compute node to the message flow



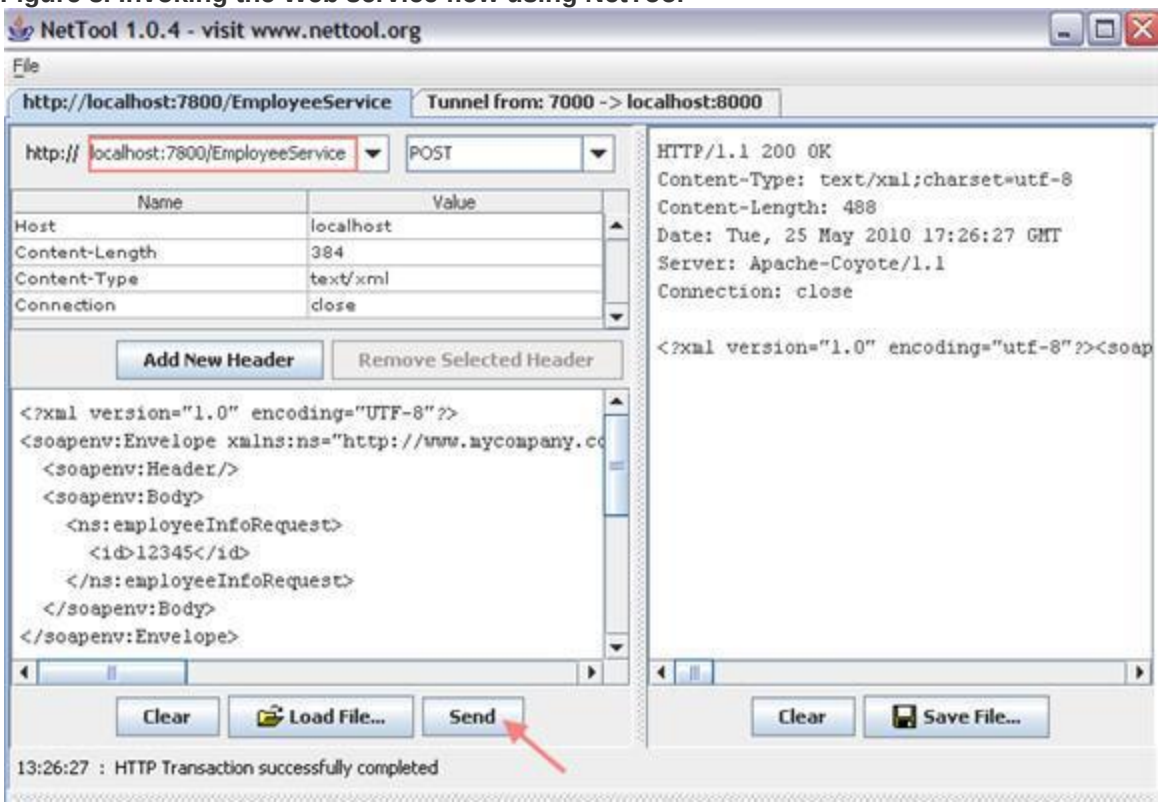
9. The following ESQL code is created in the Compute node. For simplicity, the Web service provides information only for employee id 12345 -- information for any other id is not available.

```
DECLARE ns NAMESPACE 'http://www.mycompany.com/EmployeeService';

CREATE COMPUTE MODULE WS_Security_MF_Compute
  CREATE FUNCTION Main() RETURNS BOOLEAN
  BEGIN
    IF InputRoot.XMLNSC.ns:EmployeeInfoRequest.id='12345' THEN
      SET OutputRoot.XMLNSC.ns:EmployeeInfoResponse.id =
           InputRoot.XMLNSC.ns:EmployeeInfoRequest.id;
      SET OutputRoot.XMLNSC.ns:EmployeeInfoResponse.ssn='88888888';
      SET OutputRoot.XMLNSC.ns:EmployeeInfoResponse.employeeName.firstName='John';
      SET OutputRoot.XMLNSC.ns:EmployeeInfoResponse.employeeName.lastName='Doe';
      SET OutputRoot.XMLNSC.ns:EmployeeInfoResponse.address.street='555 Creek Road';
      SET OutputRoot.XMLNSC.ns:EmployeeInfoResponse.address.city='NYC';
      SET OutputRoot.XMLNSC.ns:EmployeeInfoResponse.address.state='NY';
      SET OutputRoot.XMLNSC.ns:EmployeeInfoResponse.address.zipCode='54321';
    ELSE
      SET OutputRoot.XMLNSC.ns:EmployeeInfoResponse.status =
           'The id '||InputRoot.XMLNSC.ns:EmployeeInfoRequest.id||' does not exist';
    END IF;
    RETURN TRUE;
  END;

END MODULE;
```

31. Create a Broker Archive (BAR) file named WS_Security.bar to deploy the message flow on a broker runtime. In this example, the execution group default in the broker MB7BROKER is used.
32. Run the message flow: invoke the Web service using NetTool, a downloadable open-source graphical utility. You should see the successful

responses if the message flow is built correctly using the test messages in Figure 8:

Figure 8. Invoking the Web service flow using NetTool



The test message EmployeeInfoRequest.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:ns="http://www.mycompany.com/EmployeeService"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header/>
  <soapenv:Body>
    <ns:employeeInfoRequest>
      <id>12345</id>
    </ns:employeeInfoRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

Configuring identity authentication

An external security provider is required for the Web service identity authentication. A security profile is used to specify whether authentication, authorization, and identity mapping and propagation are performed on the identity of messages in the message flow, and if so, which external security provider is used.

Setting up the external security provider

IBM Tivoli Directory Server (LDAP) is used as the external security provider. The LDAP server has been set up and is running on the localhost (Windows XP) with port 389. The groups and users listed in Table 1 are added in the LDAP. For instructions on how to add these objects in the LDAP, see the developerWorks article Implementing message flow security in WebSphere Message Broker V7.

Table 1. Entities created in the LDAP

User	Distinguished Name (DN)	Password	Group
wmbuser1	cn=wmbuser1,ou=users,ou=wmbv7,o=ibm,c=us	user1pw	cn=authorized,ou=users,ou=wmbv7,o=ibm,c=us
wmbuser2	cn=wmbuser2,ou=users,ou=wmbv7,o=ibm,c=us	user2pw	
wmbuser3	cn=wmbuser3,ou=users,ou=wmbv7,o=ibm,c=us	user3pw	cn=authorized,ou=users,ou=wmbv7,o=ibm,c=us
wmbuser4	cn=wmbuser4,ou=users,ou=wmbv7,o=ibm,c=us	user4pw	
wmbuser5	cn=wmbuser5,ou=users,ou=wmbv7,o=ibm,c=us	user5pw	cn=authorized,ou=users,ou=wmbv7,o=ibm,c=us

Creating a security profile

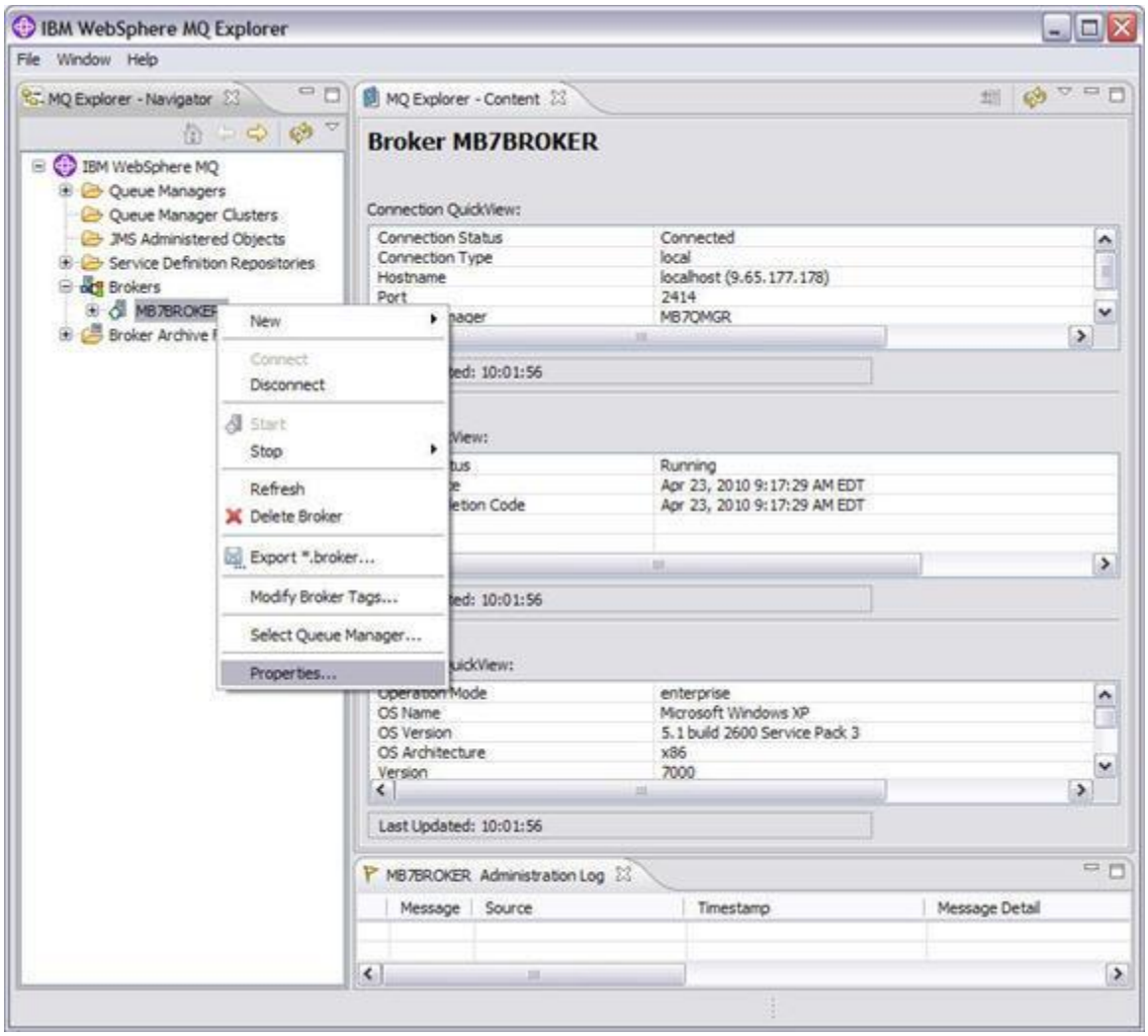
A security profile LDAP_SP1 is created with the information shown in Table 2:

Table 2. Details of the security profile LDAP_SP1

Field	Value
Authentication	LDAP
Authorization	LDAP
LDAP server	localhost:389
Identity mapping	No
Identity propagation	Yes
AuthenticationConfig	ou=users,o=wmbv7,ou=ibm,c=us (uid)
AuthorizationConfig	cn=authorized,ou=users,ou=wmbv7,o=ibm,c=us

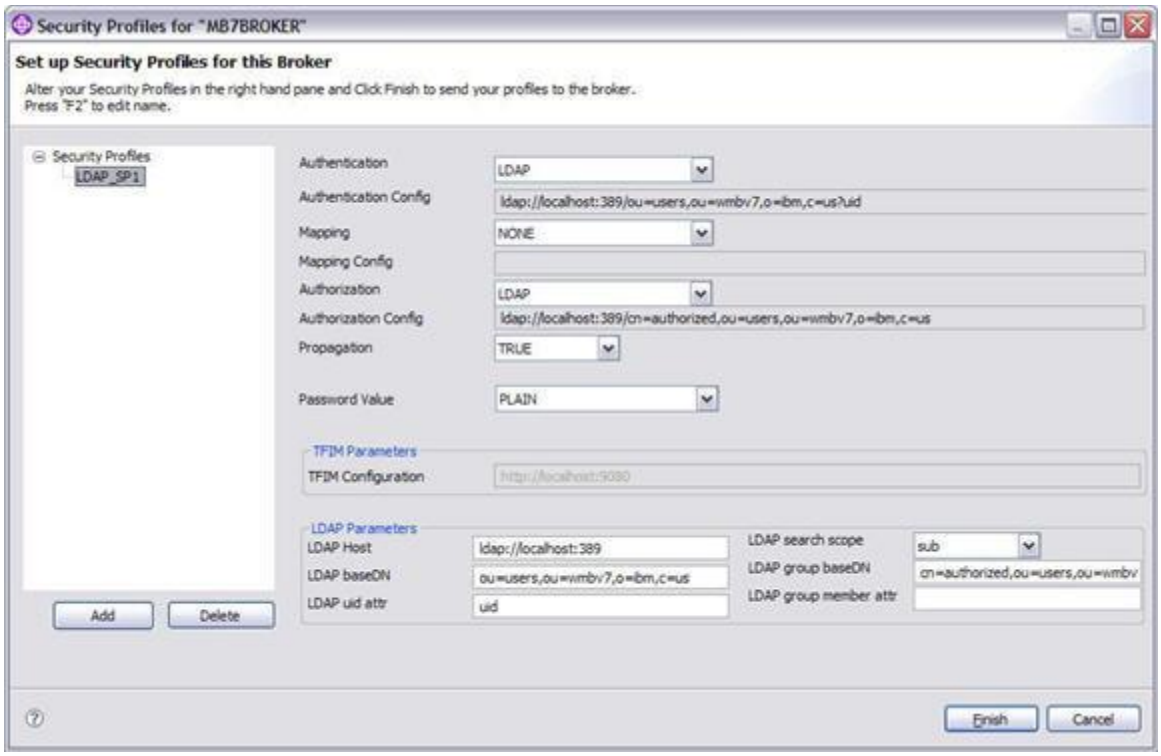
1. Open the Message Broker Explorer, right-click **MB7BROKER**, and select **Properties**:

Figure 9. Opening the broker properties



2. Click **Security and Security Profiles** on the Properties window, then click **Add** to create a new security profile. Change the name to LDAP_SP1 and provide the values based on Table 2 for the fields:

Figure 10. Creating the LDAP_SP1 security profile



Run the following command to check the details of the security profile you created:

```
mqsireportproperties MB7BROKER -c SecurityProfiles -o LDAP_SP1 -r

SecurityProfiles
  LDAP_SP1
    authentication='LDAP'
    authenticationConfig='ldap://localhost:389/ou=users,ou=wmbv7,o=ibm,c=us?uid'
    authorization='LDAP'
    authorizationConfig='ldap://localhost:389/cn=authorized,ou=users,ou=wmbv7,o=ibm,c=us'
    keyStore='keystore.jks'
    mapping='NONE'
    mappingConfig=''
    passwordValue='PLAIN'
    propagation='TRUE'
    trustStore='Reserved for future use'
```

Using the pre-built policy set and binding

Since the message flow is a Web service application implemented by using the broker SOAP nodes, the authentication tokens are defined through an appropriate policy set and binding.

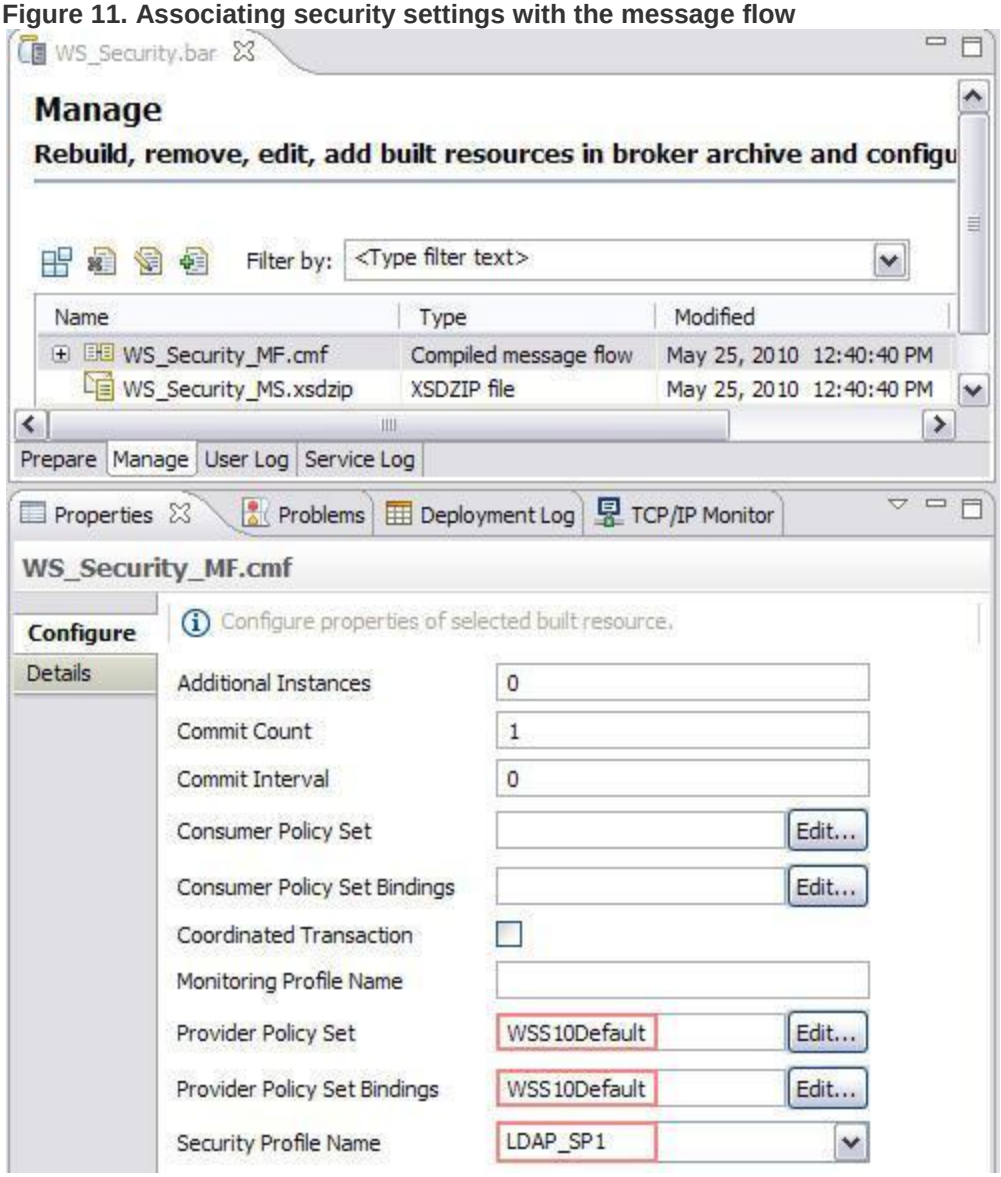
The pre-built policy set and binding WSS10Default is provided when a broker is created. A userName authentication token is defined in the policy set WSS10Default. You can use this policy set and binding pair for authentication. A new policy set and policy set binding are created later for message signing and encryption in the Section Creating a policy set and binding.

Associating the WS-Security settings with the message flow

The next step is to configure the security profile, policy set, and binding at the message flow level using the BAR File Editor. Of course, you can also set it at the SOAP node level, which overrides the security setting at the message flow level.

1. Open the BAR file **WS-Security.bar** in the Message Broker Toolkit.
2. Click **WebServiceSecurity_MF.cmf** on the Manage tab and provide the security profile LDAP_SP1 and the default policy set and binding WSS10Default:

Figure 11. Associating security settings with the message flow



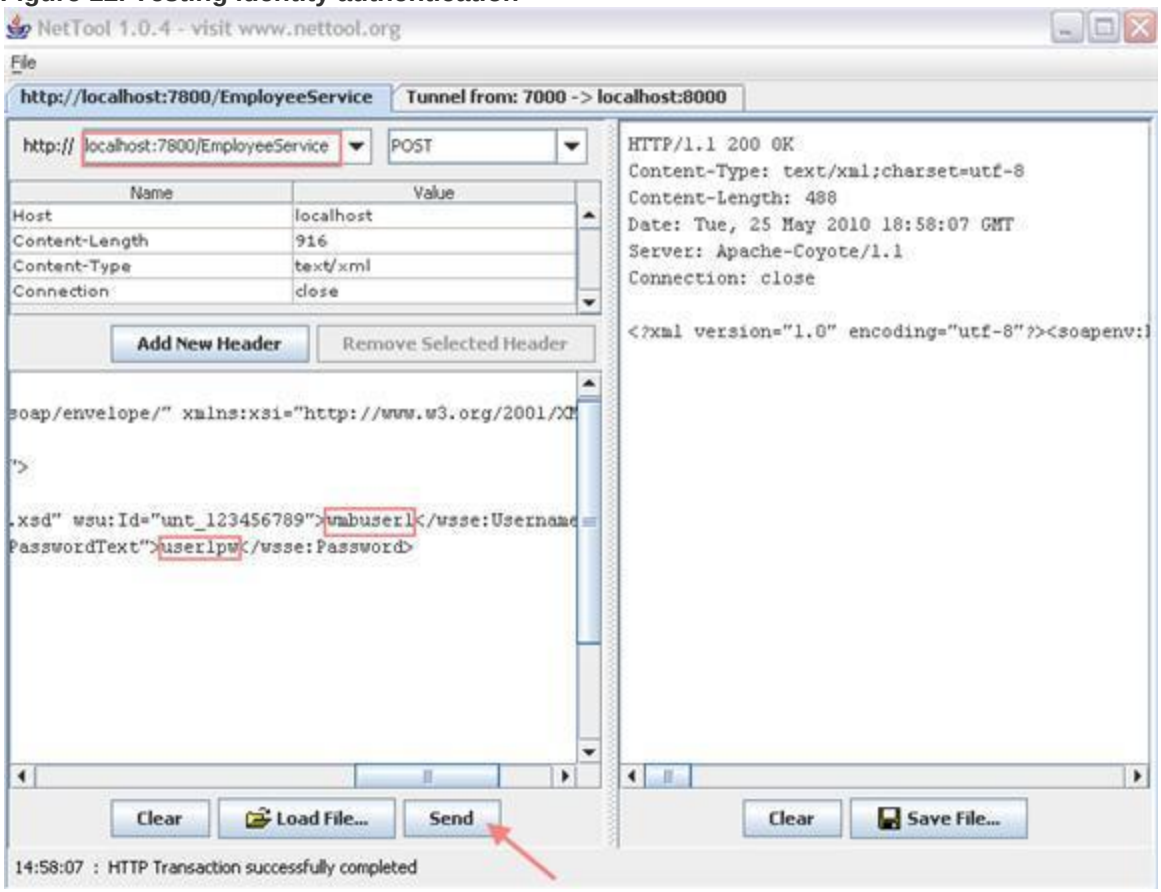
3. Save the BAR file and deploy it into the execution group default of the broker MB7BROKER. The deployment process verifies the associated security profile, policy set, and policy set binding. If any errors occur -- such

as policy set name not found or keystore setting not correct -- the deployment process fails.

Run tests to evaluate authentication

1. Before testing, make sure that the IBM Directory Server (LDAP) is running.
2. Add the WS-Security header in the input test message and ensure that the correct user ID and password are used. The WS-Security header needs to be included in the test message, because the username and password token is expected from the WS-Security header based on the WS-specification.
3. Open the test utility NetTool and provide the Web service URL: http://localhost:7800/EmployeeService. Load the file EmployeeInfoRequest_withWSSHeader.xml and click **Send**, as shown below. You should be able to access, run, and get a successful response from the message flow, as shown in Figure 12:

Figure 12. Testing identity authentication



The test message EmployeeInfoRequest_withWSSHeader.xml:

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

```
<soapenv:Envelope xmlns:ns="http://www.mycompany.com/EmployeeService"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
      wssecurity-secext-1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
          wssecurity-utility-1.0.xsd" wsu:Id="unt_907818524">wmbuser1</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
          username-token-profile-1.0#PasswordText">user1pw</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body>
    <ns:employeeInfoRequest>
      <id>12345</id>
    </ns:employeeInfoRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

The SOAP response message for the Web service EmployeeInfo request:

```
HTTP/1.1 200 OK
Content-Type: text/xml;charset=utf-8
Content-Length: 233
Date: Thu, 06 May 2010 21:19:48 GMT
Server: Apache-Coyote/1.1
Connection: close

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope">
  <soapenv:Body>
    <NS1:employeeInfoResponse xmlns:NS1="http://www.mycompany.com/EmployeeService">
      <id>12345</id>
      <ssn>88888888</ssn>
      <employeeName>
        <firstName>John</firstName>
        <lastName>Doe</lastName>
      </employeeName>
      <address>
        <street>555 Creek Road</street>
        <city>NYC</city>
        <state>NY</state>
        <zipCode>54321</zipCode>
      </address>
    </NS1:employeeInfoResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

If the invalid user token -- either wrong user ID or password -- is in the security header, then the following error occurs:

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/xml;charset=UTF-8
Content-Length: 2297
Date: Tue, 25 May 2010 19:57:20 GMT
Server: Apache-Coyote/1.1
Connection: close

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <soapenv:Fault xmlns:axis2ns2="http://schemas.xmlsoap.org/soap/envelope/">
      <faultcode>axis2ns2:Server.securityException</faultcode>
      <faultstring>CWWSS6521E: The Login failed because of an exception: javax.
      security.auth.login.LoginException: Broker security returned failure;
      system.wss.consume.unt</faultstring>
```

```
      <detail>
        <Exception>org.apache.axis2.AxisFault: CWWSS6521E: The Login failed
        because of an exception: javax.security.auth.login.LoginException: Broker
        security returned failure; system.wss.consume.unt
        at org.apache.axis2.AxisFault.makeFault(AxisFault.java:430)
        at com.ibm.ws.wssecurity.handler.WSSecurityConsumerBase.invoke
        (WSSecurityConsumerBase.java:142)
        at com.ibm.ws.wssecurity.handler.WSSecurityConsumerHandler.invoke
        (WSSecurityConsumerHandler.java:461)
        at org.apache.axis2.engine.Phase.invoke(Phase.java:295)
        at org.apache.axis2.engine.AxisEngine.invoke(AxisEngine.java:266)
        at org.apache.axis2.engine.AxisEngine.receive(AxisEngine.java:163)
        at com.ibm.broker.axis2.Axis2Invoker.processInboundRequest(Axis2Invoker.java:2481)
        at com.ibm.broker.axis2.Axis2Invoker.invokeAxis2OverHTTP(Axis2Invoker.java:2182)
        at com.ibm.broker.axis2.TomcatNodeRegistrationUtil.invokeAXIS2
        (TomcatNodeRegistrationUtil.java:377)
        Caused by: com.ibm.wsspi.wssecurity.core.SoapSecurityException: CWWSS6521E:
        The Login failed because of an exception: javax.security.auth.login.
        LoginException: Broker security returned failure; system.wss.consume.unt
        at com.ibm.wsspi.wssecurity.core.SoapSecurityException.format
        (SoapSecurityException.java:67)
        at com.ibm.ws.wssecurity.wssapi.token.impl.CommonTokenConsumer.invoke
        (CommonTokenConsumer.java:291)
        at com.ibm.ws.wssecurity.core.WSSConsumer.callTokenConsumer(WSSConsumer.java:2048)
        at com.ibm.ws.wssecurity.core.WSSConsumer.callTokenConsumer(WSSConsumer.java:1909)
        at com.ibm.ws.wssecurity.core.WSSConsumer.invoke(WSSConsumer.java:750)
        at com.ibm.ws.wssecurity.handler.WSSecurityConsumerBase.invoke
        (WSSecurityConsumerBase.java:107)
        ... 7 more</Exception>
      </detail>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

Keep in mind that if only the security profile is provided in the BAR file configuration but not the policy set and binding, the security manager of the broker can be deployed, but it rejects the access of the message flow and the following error occurs, because there is no indication where the broker security manager can get the identity token from:

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/xml;charset=UTF-8
Content-Length: 719
Date: Tue, 25 May 2010 20:04:47 GMT
Server: Apache-Coyote/1.1
Connection: close

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <soapenv:Fault>
      <faultcode>soapenv:Server</faultcode>
      <faultstring>BIP3113E: Exception detected in message flow
      WS_Security_MF.SOAP Input (broker MB7BROKER)</faultstring>
      <detail>
        <Text>BIP2708W: An input message to flow ''WS_Security_MF'' does not
        have an identity.  Input messages to the flow must have an identity for access.
        Modify the client calling the flow to provide an identity for the message.
        : :\build\S000_P\src\DataFlowEngine\MessageServices\ImbSecurityManager.cpp:
        801: ImbSecurityManager::noIdentityFound: : </Text>
      </detail>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

Of course, there is no issue if the policy set and binding are provided without the security profile on the BAR file for the message flow. In this case, the test SOAP message still needs the security header, because the policy set PS1 defines the Username token, and the broker security manager still checks the token defined in the policy set, but doesn't perform identity authentication.

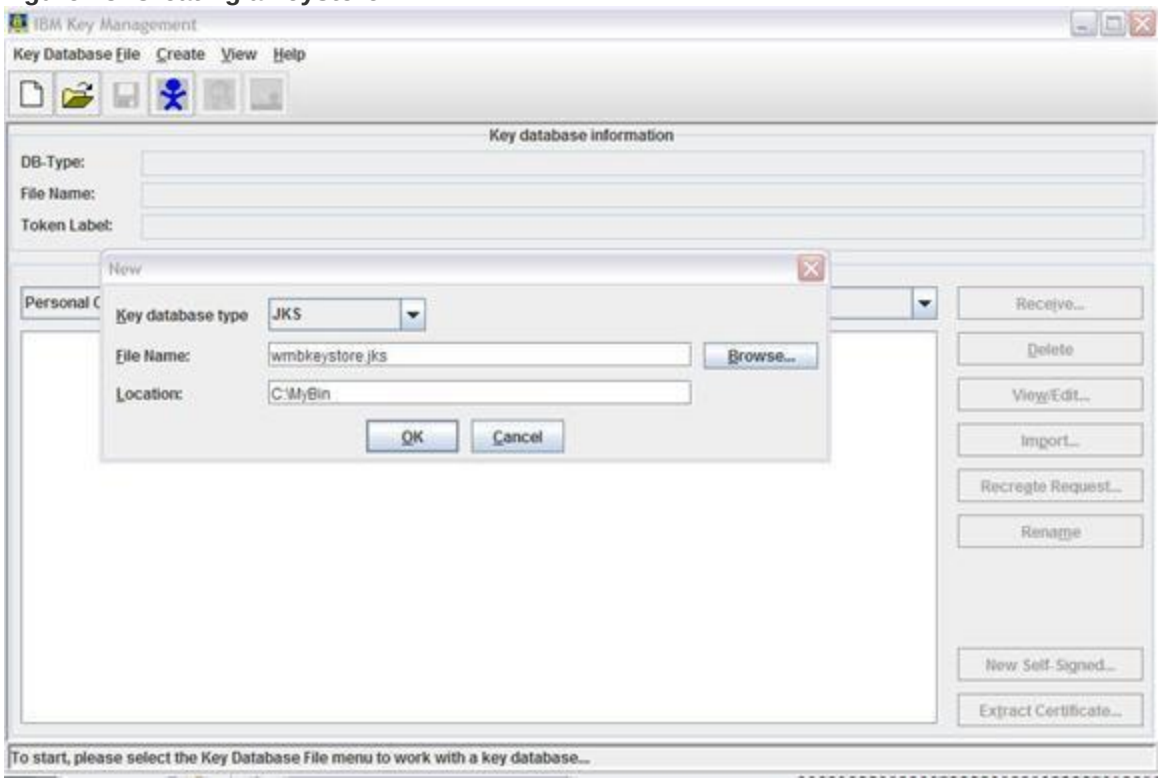
Configuring message signing and encryption

In order to sign and encrypt messages, certificates must be created and exchanged in keystores and truststores.

Creating keystores and truststores

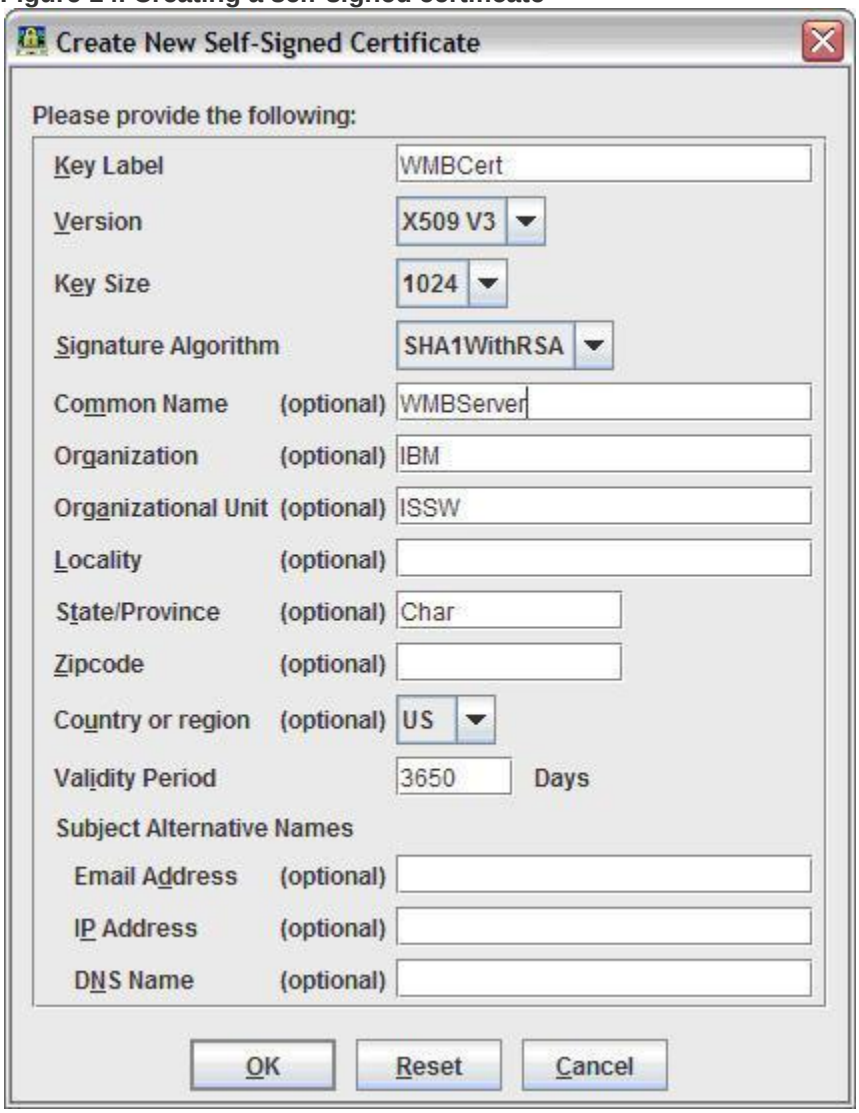
1. Run the command ikeyman.exe in the directory jre16\bin under the WebSphere Message Broker product install directory (usually C:\IBM\WMB7\jre16\bin).
2. Click **Key Database File** and select **New** to create a keystore named wmbkeystore for the Web service provider. Provide the file name and location, and make sure that the key database type is JKS, as shown in Figure 13:

Figure 13. Creating a keystore



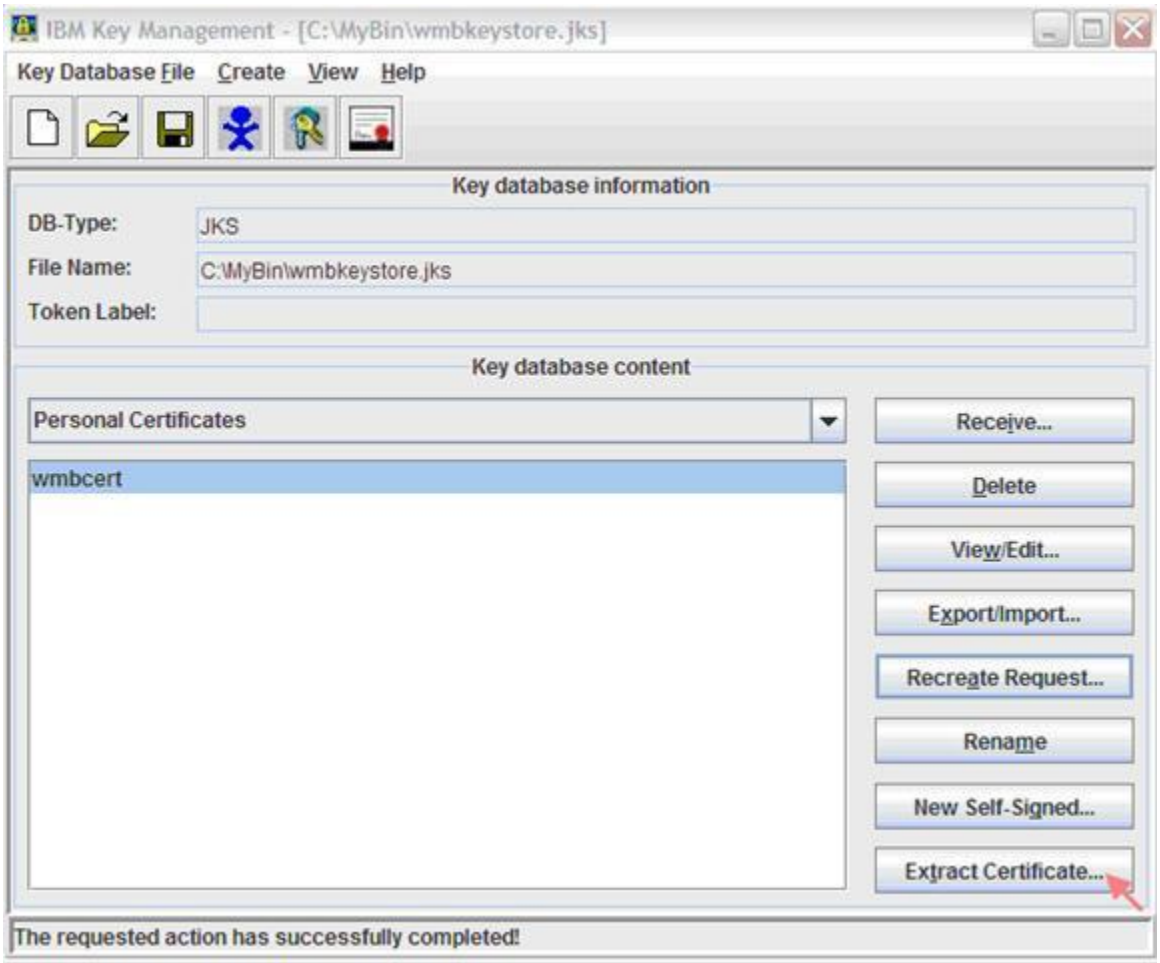
3. After providing the password wmbv7pw, the keystore is created. Click **New Self-Signed** to create a self-signed certificate:

Figure 14. Creating a self-signed certificate



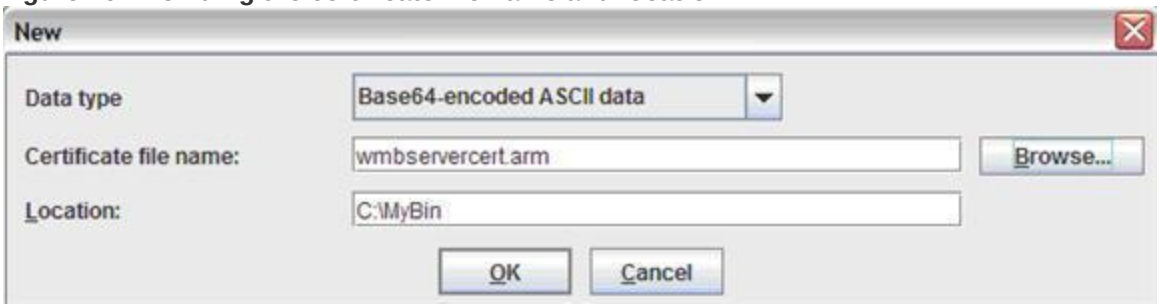
4. Click **OK**. The certificate wmbcert should be listed in Personal Certificates.
5. Click **Extract Certificate** to export the server certificate:.

Figure 15. Extracting a certificate



6. Provide the file name and location as shown in Figure 16 below and click **OK**. This server certificate will be added into the Web service consumer's keystore later.

Figure 16. Providing the certificate file name and location

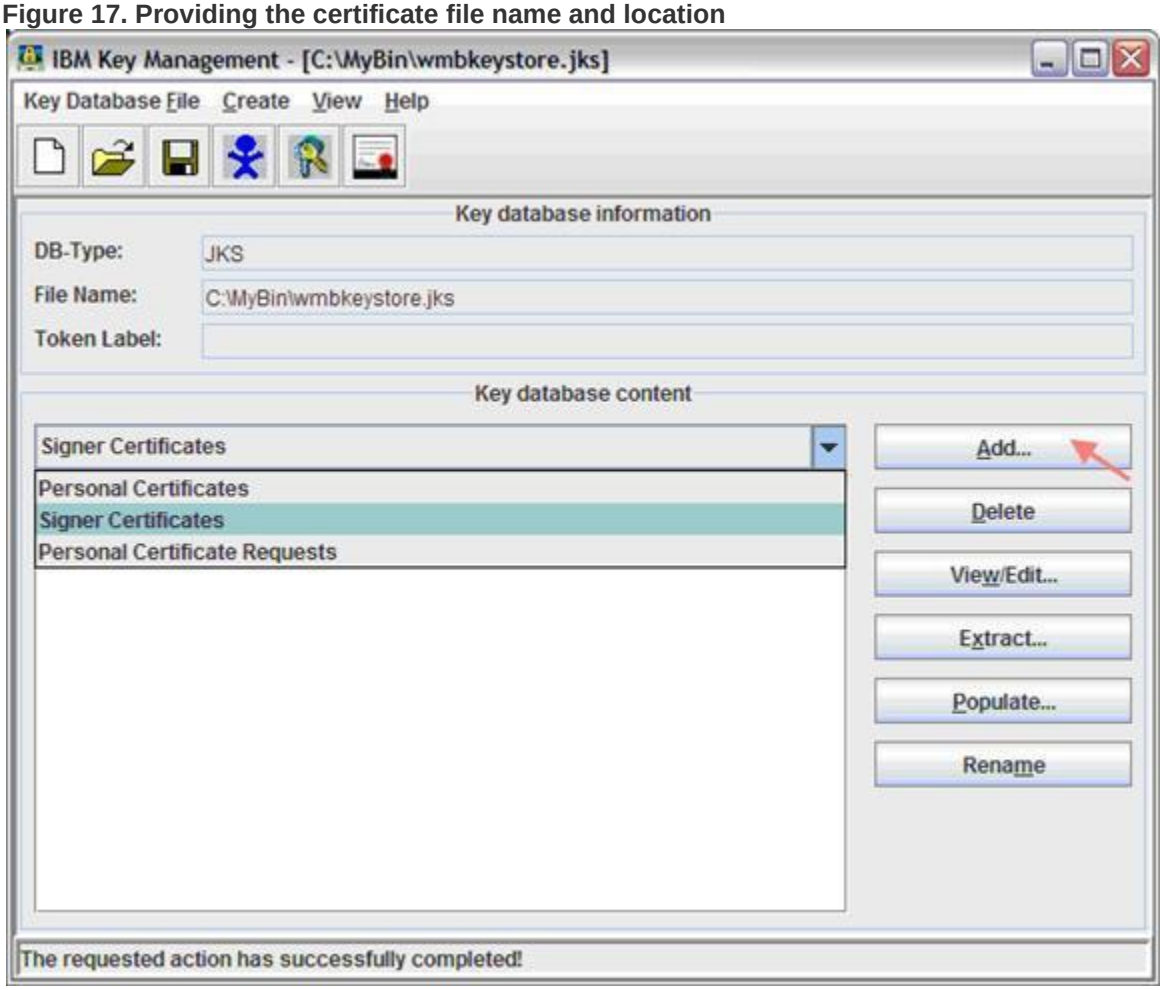


7. Repeat steps 2 through 6 above to create a keystore named wmbclientkeystore for the Web service consumer. Create a self-signed certificate and extract the public key into the file wmbclient.arm in the C:\MyBin directory. Use the same keystore password wmbv7pw.

If a keystore is used to contain trusted certificates, it is usually called a truststore. Ideally, a keystore is for private keys and a truststore is for public certificates. For the purpose of simplicity, the keystores created are also used as the truststores.

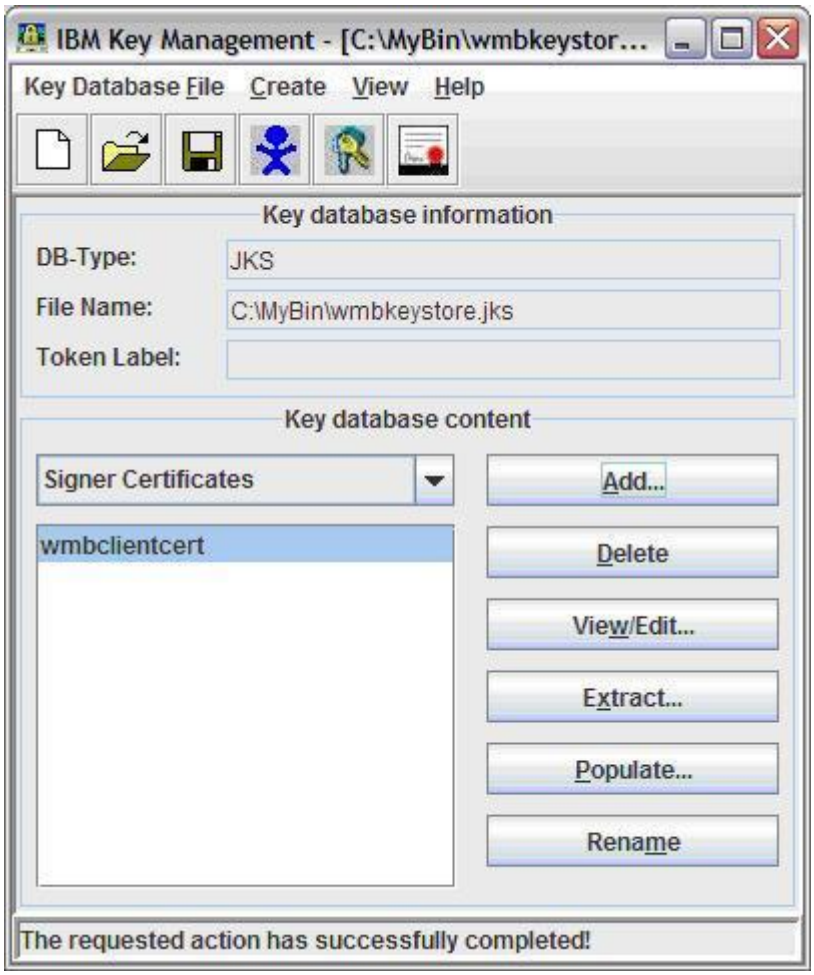
8. Import the client's certificate into the server's keystore. Assume that the keystore is still open, select **Signer Certificates** from the drop down list, and click **Add**:

Figure 17. Providing the certificate file name and location



9. Provide the file name wmbclientcert.arm and the location C:\MyBin. Click **OK** and type the password wmbv7pw for the certificate. An entry should be added in the Signer Certificates list, as shown in Figure 18:

Figure 18. The client certificate on the list of Signer Certificates



10. Repeat steps 8 and 9 above to import the server's certificate into the client's keystore.
11. Run the following command for both the server and client keystores to see the content. Each keystore should have two entries: keyEntry and trustedCertEntry.

```
keytool -list -keystore C:\MyBin\wmbkeystore.jks -v

Enter keystore password:

Keystore type: jks
Keystore provider: IBMJCE

Your keystore contains 2 entries

Alias name: wmbclientcert
```

```
Creation date: Apr 21, 2010
Entry type: trustedCertEntry

Owner: CN=WMBClient, OU=ISSW, O=IBM, ST=Char, C=US
Issuer: CN=WMBClient, OU=ISSW, O=IBM, ST=Char, C=US
Serial number: 4bcf2ac0
Valid from: 4/21/10 12:41 PM until: 4/18/20 12:41 PM
Certificate fingerprints:
         MD5:  29:E4:97:06:EC:BD:FB:3A:24:DE:C3:2F:50:DD:8D:8E
         SHA1: 3C:AC:1C:F8:AF:39:8F:90:11:5B:26:B6:E7:FF:4C:8B:08:12:C7:3F

*******************************************
*******************************************

Alias name: wmbcert
Creation date: Apr 21, 2010
Entry type: keyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=WMBServer, OU=ISSW, O=IBM, ST=Char, C=US
Issuer: CN=WMBServer, OU=ISSW, O=IBM, ST=Char, C=US
Serial number: 4bcf21d0
Valid from: 4/21/10 12:03 PM until: 4/18/20 12:03 PM
Certificate fingerprints:
         MD5:  58:8B:BE:72:F9:01:C3:5B:CD:A6:A6:77:A8:7B:32:F0
         SHA1: 32:9A:21:BB:3B:0A:B0:3F:37:F5:5E:9B:EC:E0:7F:62:52:17:57:B4

*******************************************
```

```
*******************************************
keytool -list -keystore C:\MyBin\wmbclientkeystore.jks -v


Enter keystore password:
Keystore type: jks
Keystore provider: IBMJCE
Your keystore contains 2 entries
```

```
Alias name: wmbclientcert

Creation date: Apr 21, 2010

Entry type: keyEntry

Certificate chain length: 1

Certificate[1]:

Owner: CN=WMBClient, OU=ISSW, O=IBM, ST=Char, C=US

Issuer: CN=WMBClient, OU=ISSW, O=IBM, ST=Char, C=US

Serial number: 4bcf2ac0

Valid from: 4/21/10 12:41 PM until: 4/18/20 12:41 PM

Certificate fingerprints:

         MD5:  29:E4:97:06:EC:BD:FB:3A:24:DE:C3:2F:50:DD:8D:8E

         SHA1: 3C:AC:1C:F8:AF:39:8F:90:11:5B:26:B6:E7:FF:4C:8B:08:12:C7:3F


*******************************************

*******************************************


Alias name: wmbcert

Creation date: May 7, 2010

Entry type: trustedCertEntry

Owner: CN=WMBServer, OU=ISSW, O=IBM, ST=Char, C=US

Issuer: CN=WMBServer, OU=ISSW, O=IBM, ST=Char, C=US

Serial number: 4bcf21d0

Valid from: 4/21/10 12:03 PM until: 4/18/20 12:03 PM

Certificate fingerprints:

         MD5:  58:8B:BE:72:F9:01:C3:5B:CD:A6:A6:77:A8:7B:32:F0

         SHA1: 32:9A:21:BB:3B:0A:B0:3F:37:F5:5E:9B:EC:E0:7F:62:52:17:57:B4


*******************************************

*******************************************
```

At this point, the keystores are created for the Web service provider and consumer with the certificates exchanged. The keystore information is used to create the policy set and binding, as described below.

Configuring the broker with the keystores

In order for the broker to use these keystores, you must configure the broker properties. Use the following commands to set up the provider keystore and truststore:

```
mqsichangeproperties MB7BROKER -e default -o ComIbmJVMManager -n
    keystoreFile -v C:\MyBin\wmbkeystore.jks
mqsichangeproperties MB7BROKER -e default -o ComIbmJVMManager -n
    keystoreType -v JKS
mqsichangeproperties MB7BROKER -e default -o ComIbmJVMManager -n
    keystorePass -v default::keystorepass
mqsichangeproperties MB7BROKER -e default -o ComIbmJVMManager -n
    truststoreFile -v C:\MyBin\wmbkeystore.jks
mqsichangeproperties MB7BROKER -e default -o ComIbmJVMManager -n
    truststoreType -v JKS
mqsichangeproperties MB7BROKER -e default -o ComIbmJVMManager -n
    truststorePass -v default::truststorepass
```

Run the following command to view the settings for execution group default of the broker MB7BROKER:

```
mqsireportproperties MB7BROKER -o ComIbmJVMManager -a -e default

ComIbmJVMManager
uuid='ComIbmJVMManager'
userTraceLevel='none'
traceLevel='none'
userTraceFilter='none'
traceFilter='none'
resourceStatsReportingOn='inactive'
resourceStatsMeasurements='<ResourceStatsSwitches ResourceType="JVM" version='1'>
<Measurementname="CommittedMemoryInMB" collect="on" />
<Measurement name="CumulativeGCTimeInSeconds" collect="on"/>
<Measurement name="CumulativeNumberOfGCCollections" collect="on" />
<Measurementname="InitialMemoryInMB" collect="on" />
<Measurement name="MaxMemoryInMB" collect="on" />
<Measurement name="UsedMemoryInMB" collect="on" /> </ResourceStatsSwitches>'
jvmVerboseOption='none'
jvmDisableClassGC='false'
jvmShareClasses='false'
jvmNativeStackSize='-1'
jvmJavaOSStackSize='-1'
jvmMinHeapSize='33554432'
jvmMaxHeapSize='-1'
jvmDebugPort='5000'
keystoreType='JKS'
keystoreFile='C:\MyBin\wmbkeystore.jks'
keystorePass='default::keystorepass'
truststoreType='JKS'
truststoreFile='C:\MyBin\wmbkeystore.jks'
truststorePass='default::truststorepass'
```

The value for the property keystorePass is a reference to where the password is stored. Run the following command to set up the actual password so that the broker can access the keystore. The command option -u for user is not used:

```
mqsisetdbparms MB7BROKER -n default::keystorepass -u any -p wmbv7pw
mqsisetdbparms MB7BROKER -n default::truststorepass -u any -p wmbv7pw
```

Restart the broker MB7BROKER for the setting to take effect.

Creating policy set and binding

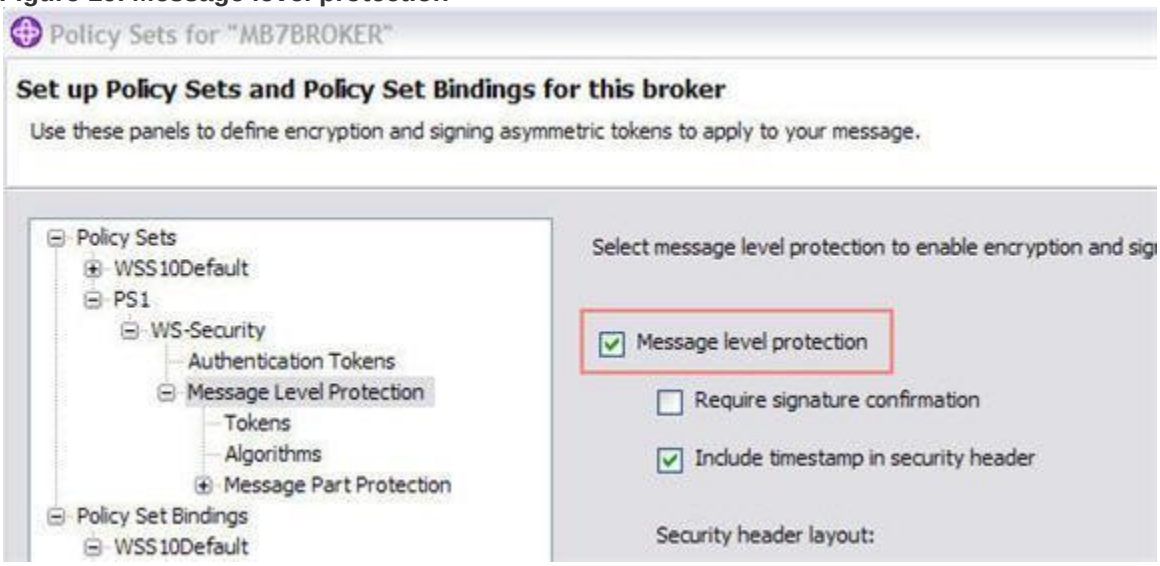
In the previous section Configuring identity authentication, the default policy set and binding WSS10Default were used, which is good for authentication purposes. In order

to accomplish WS-Security message signing and encryption, you need to create a new policy set and binding and associate them with the message flow.

Creating a policy set

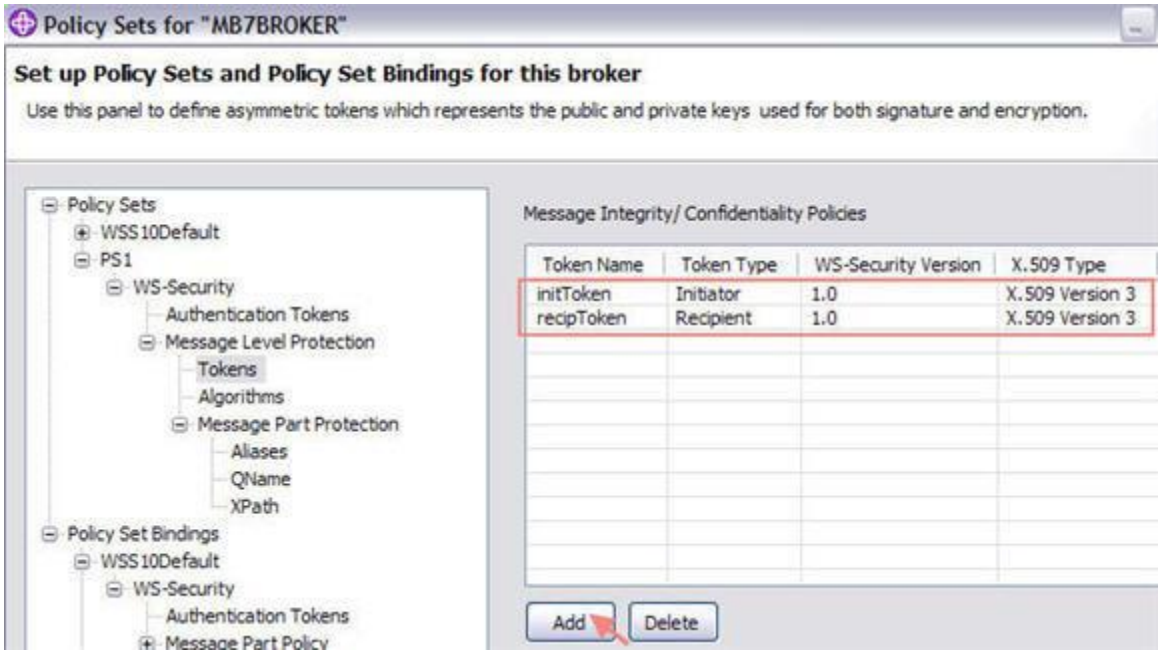
1. Open Message Broker Explorer, right-click the broker **MB7BROKER**, and select **Properties**.
2. On the Properties dialog, select **Security => Policy Sets**.
3. Select **Policy Sets => Add** to add a new policy set on the Set up Policy Sets and Policy Set Bindings for this broker dialog. Rename it as PS1.
4. Expand WS-Security and highlight **Message Level Protection** on the left. Check **Message level protection** and **Include timestamp in security header** on the right:

Figure 19. Message level protection



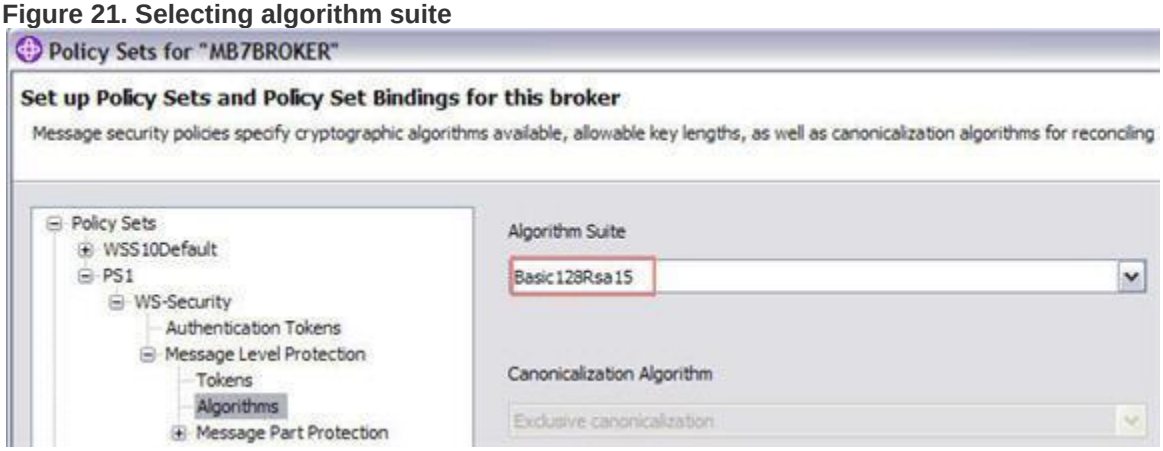
5. Click **Tokens** and add two tokens, one representing the certificate for the provider and one for the consumer:

Figure 20. Adding entries for message integrity and confidentiality policies



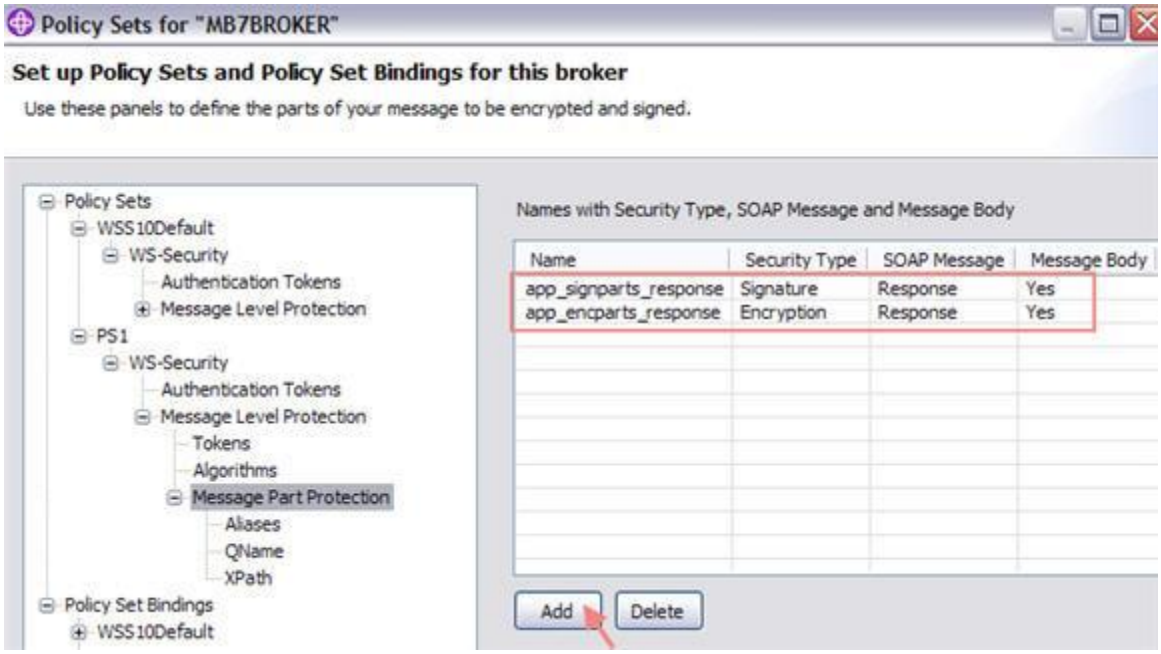
6. Click **Algorithms** and select **Basic128Rsa15**:

Figure 21. Selecting algorithm suite



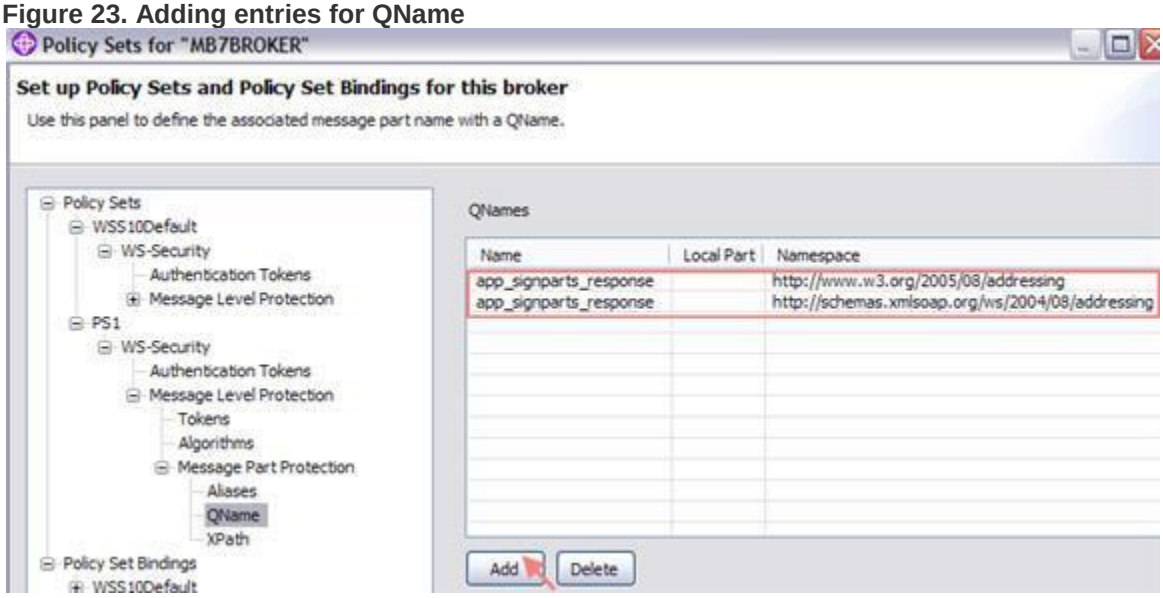
7. Highlight **Message Part Protection** and click **Add** to add two entries, one for signature and another for encryption:

Figure 22. Adding entries for message part protection



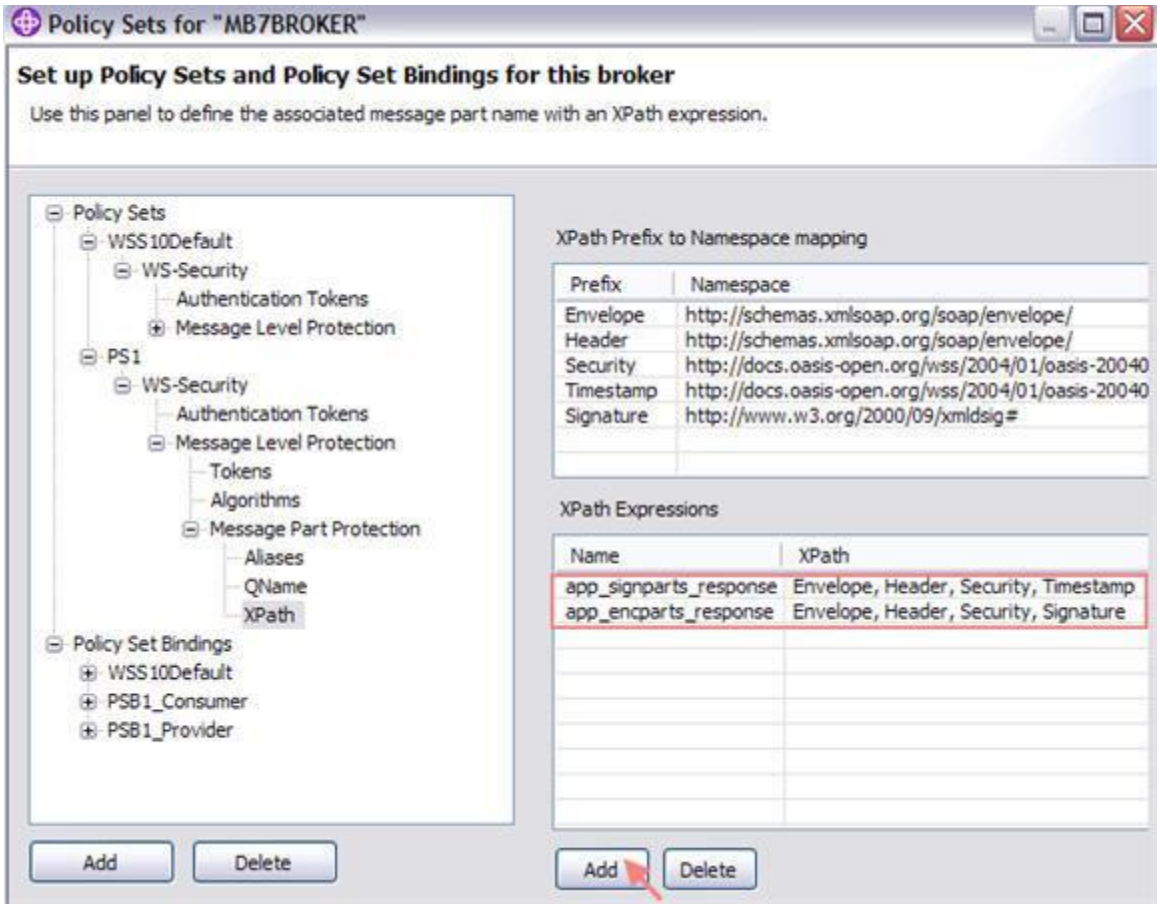
8. Highlight **QName** and click **Add** to add entries as shown in Figure 23 below. These two namespace entries indicate that the WS-Addressing headers must be signed.

Figure 23. Adding entries for QName



9. Highlight **XPath** and click **Add** to associate the pre-defined XPath to each of the entries that were created in Step 7, as shown in Figure 24:

Figure 24. Adding entries for XPath



10. Click **Finish** to save the policy set PS1.
11. Run the following command to see the details of the policy set PS1.

```
mqsireportproperties MB7BROKER -c PolicySets -o PS1 -r

PolicySets
  PS1
    config=''
    ws-security='<?xml version="1.0" encoding="UTF-8"?>
<policy:Policy xmlns:_0="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd" xmlns:_200512="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200512" xmlns:policy="http://schemas.xmlsoap.org/ws/2004/09/policy">
  <_200512:AsymmetricBinding>
    <policy:Policy>
      <_200512:InitiatorToken>
        <policy:Policy>
          <_200512:X509Token _200512:IncludeToken="http://docs.oasis-open.org/ws-sx/
          ws-securitypolicy/200512/IncludeToken/AlwaysToInitiator">
```

```
27.             <policy:Policy Name="initToken">
28.               <_200512:WssX509V3Token10/>
29.             </policy:Policy>
30.           </_200512:X509Token>
31.         </policy:Policy>
32.       </_200512:InitiatorToken>
33.       <_200512:RecipientToken>
34.         <policy:Policy>
35.           <_200512:X509Token _200512:IncludeToken="http://docs.oasis-open.org/ws-sx/
36.           ws-securitypolicy/200512/IncludeToken/AlwaysToRecipient">
37.             <policy:Policy Name="recipToken">
38.               <_200512:WssX509V3Token10/>
39.             </policy:Policy>
40.           </_200512:X509Token>
41.         </policy:Policy>
42.       </_200512:RecipientToken>
43.       <_200512:AlgorithmSuite>
44.         <policy:Policy>
45.           <_200512:Basic128Rsa15/>
46.         </policy:Policy>
47.       </_200512:AlgorithmSuite>
48.       <_200512:IncludeTimestamp/>
49.       <_200512:Layout>
50.         <policy:Policy>
51.           <_200512:Strict/>
52.         </policy:Policy>
53.       </_200512:Layout>
54.     </policy:Policy>
55.   </_200512:AsymmetricBinding>
56.   <policy:Policy _0:Id="response:app_signparts_response">
57.     <_200512:SignedElements>
58.       <_200512:XPath>/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/'
59.       and local-name()='Envelope']/*[namespace-uri()='http://schemas.xmlsoap.org/
60.       soap/envelope/' and local-name()='Header']/*[namespace-uri()='http://docs.
61.       oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd' and
```

```
62.       local-name()='Security']/*[namespace-uri()='http://docs.oasis-open.org/wss/
63.       2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd' and local-name()=
64.       'Timestamp']</_200512:XPath>
65.       <_200512:XPath>/*[namespace-uri()='http://www.w3.org/2003/05/soap-envelope'
66.       and local-name()='Envelope']/*[namespace-uri()='http://www.w3.org/2003/05/soap-
67.       envelope' and local-name()='Header']/*[namespace-uri()='http://docs.oasis-open.
68.       org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd' and local-name()=
69.       'Security']/*[namespace-uri()='http://docs.oasis-open.org/wss/2004/01/oasis-
70.       200401-wss-wssecurity-utility-1.0.xsd' and local-name()='Timestamp']
71.       </_200512:XPath>
72.     </_200512:SignedElements>
73.     <_200512:SignedParts>
74.       <_200512:Body/>
75.       <_200512:Header Namespace="http://www.w3.org/2005/08/addressing"/>
76.       <_200512:Header Namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing"/>
77.     </_200512:SignedParts>
78.   </policy:Policy>
79.   <policy:Policy _0:Id="response:app_encparts_response">
80.     <_200512:EncryptedParts>
81.       <_200512:Body/>
82.     </_200512:EncryptedParts>
83.     <_200512:EncryptedElements>
84.       <_200512:XPath>/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/'
85.       and local-name()='Envelope']/*[namespace-uri()='http://schemas.xmlsoap.org/
86.       soap/envelope/' and local-name()='Header']/*[namespace-uri()='http://docs.
87.       oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd' and
88.       local-name()='Security']/*[namespace-uri()='http://www.w3.org/2000/09/
89.       xmldsig#' and local-name()='Signature']</_200512:XPath>
90.       <_200512:XPath>/*[namespace-uri()='http://www.w3.org/2003/05/soap-envelope'
91.       and local-name()='Envelope']/*[namespace-uri()='http://www.w3.org/2003/05/soap-
92.       envelope' and local-name()='Header']/*[namespace-uri()='http://docs.oasis-open.
93.       org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd' and local-name()=
94.       'Security']/*[namespace-uri()='http://www.w3.org/2000/09/xmldsig#' and
95.       local-name()='Signature']</_200512:XPath>
96.     </_200512:EncryptedElements>
```

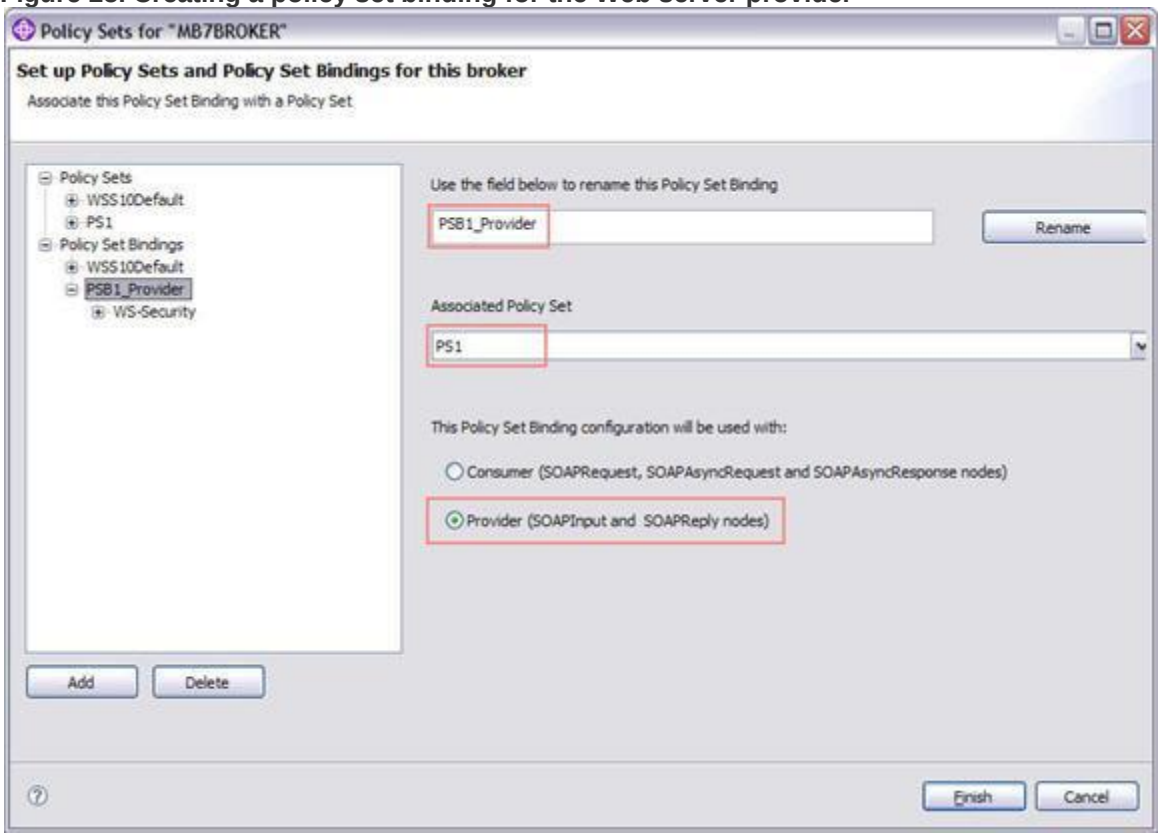
```
97.   </policy:Policy>
</policy:Policy>'
```

The policy set is created only for the Web service provider (SOAPInput and SOAPReply nodes), since the message flow to be configured for the WS-Security in this article is the Web service provider. Of course, you can add entries in the policy set for the Web service consumer if needed.

Creating a policy set binding for the Web service provider

1. Right-click the broker **MB7BROKER**, select **Properties and Security**, and then click **Policy Sets**.
2. Select **Policy Set Bindings** on the left and then click **Add** to create a new entry. Rename it to PSB1_Provider. Select **PS1** to be associated, and make sure the binding is used with Provider, as shown in Figure 25:

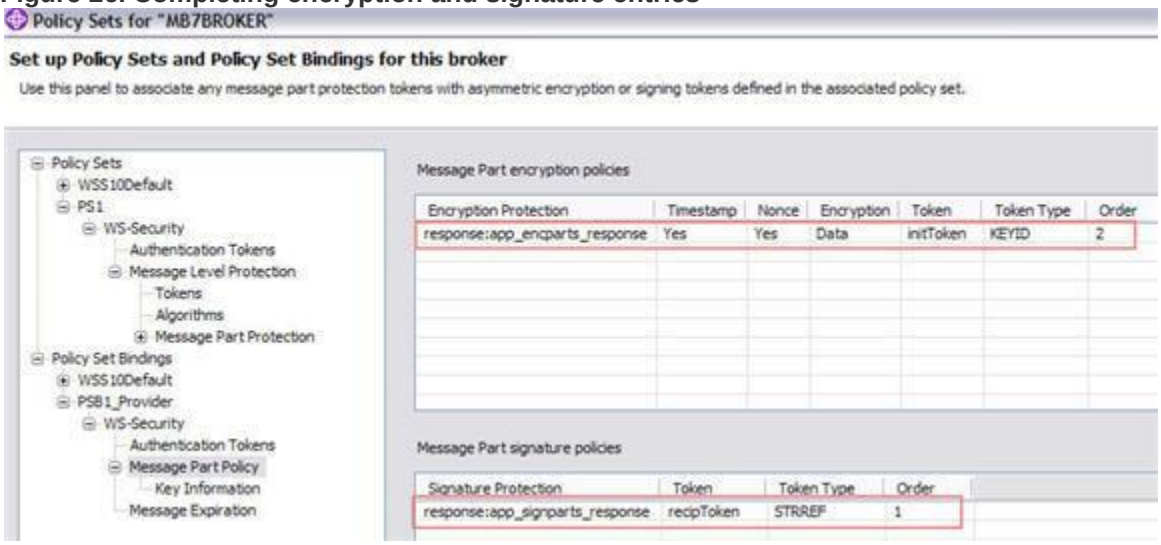
Figure 25. Creating a policy set binding for the Web server provider



3. Select **PSB1_Provider => WS-Security => Message Part Policy**. The entries are partially completed based on the security policy based on the policy set associated. Complete encryption and signature entries as shown in Figure 26. The Order indicates whether the message is encrypted or

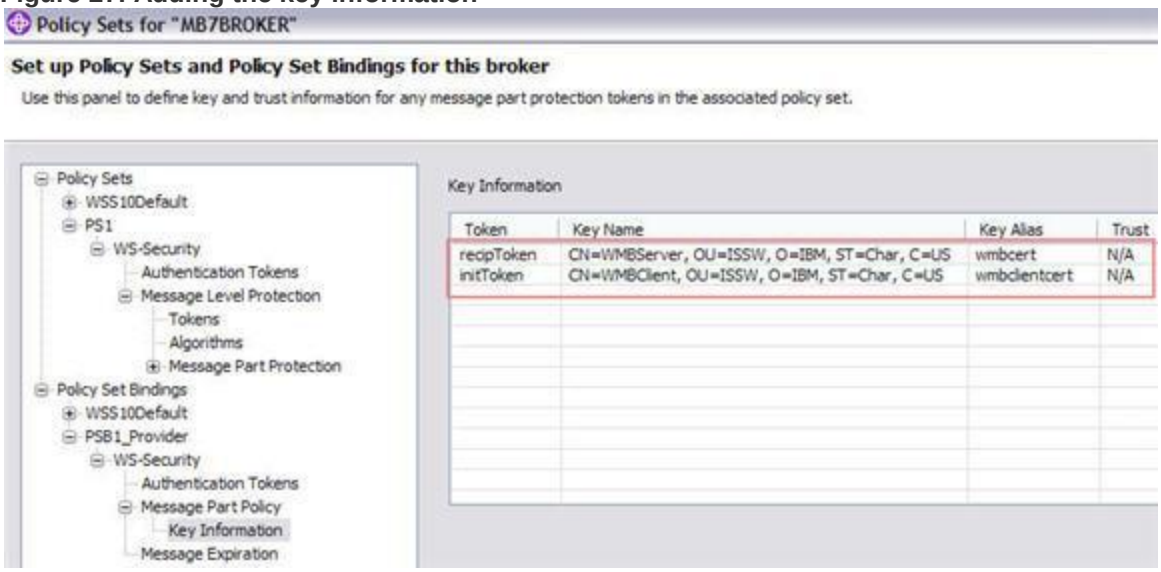
singed first. In this case, the response message is signed and then encrypted.

Figure 26. Completing encryption and signature entries



4. Expand Message Part Policy and select Key Information. Provide values as shown in Figure 27.

Figure 27. Adding the key information



5. Click Finish button to save the binding.
6. Run the following command to see details of the policy set binding.
   mqsireportproperties MB7BROKER -c PolicySetBindings -o PSB1_Provider -r

```
PolicySetBindings
```

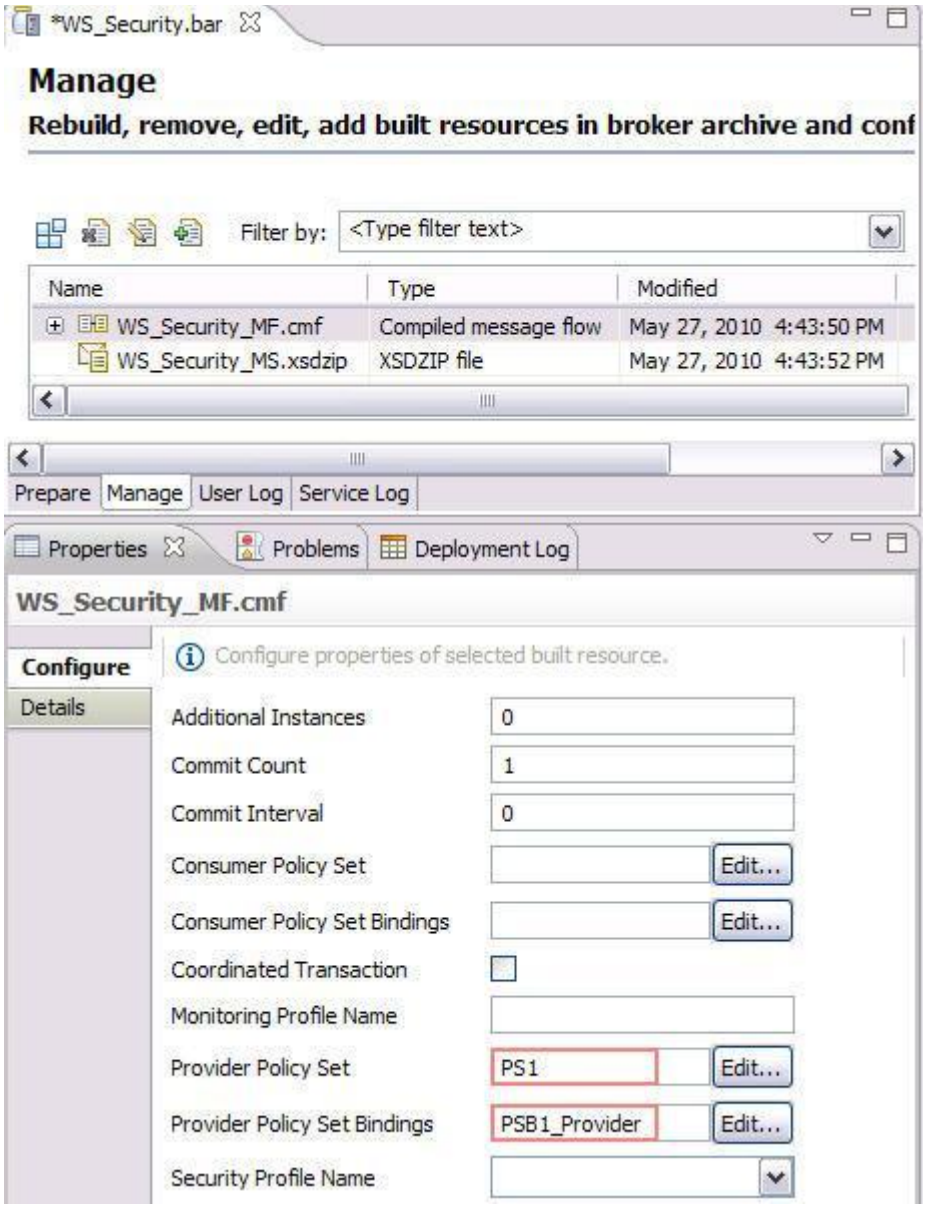
```
  PSB1_Provider
    associatedPolicySet='PS1'
    config=''
    ws-security='<?xml version="1.0" encoding="UTF-8"?>
<securitybinding:securityBindings xmlns:securitybinding="http://www.ibm.com/xmlns/
prod/websphere/200608/ws-securitybinding">
  <securitybinding:securityBinding name="application">
    <securitybinding:securityOutboundBindingConfig>
      <securitybinding:signingInfo name="gen_app_signparts_response" order="1">
        <securitybinding:signingKeyInfo reference="gen_recipToken_signapp_signparts_
        response_keyinfo"/>
        <securitybinding:signingPartReference reference="response:app_signparts_
        response">
          <securitybinding:transform algorithm="http://www.w3.org/2001/10/xml-exc-
          c14n#"/>
        </securitybinding:signingPartReference>
      </securitybinding:signingInfo>
      <securitybinding:encryptionInfo name="gen_app_encparts_response" order="2">
        <securitybinding:keyEncryptionKeyInfo reference="gen_initToken_encapp_
        encparts_response_keyinfo"/>
        <securitybinding:encryptionPartReference reference="response:app_
        encparts_response"/>
      </securitybinding:encryptionInfo>
      <securitybinding:keyInfo classname="com.ibm.ws.wssecurity.wssapi.
      CommonContentGenerator" name="gen_recipToken_signapp_signparts_response_
      keyinfo" type="STRREF">
        <securitybinding:tokenReference reference="gen_responseapp_signparts_
        response"/>
      </securitybinding:keyInfo>
      <securitybinding:keyInfo classname="com.ibm.ws.wssecurity.wssapi.
      CommonContentGenerator" name="gen_initToken_encapp_encparts_response_keyinfo"
      type="KEYID">
        <securitybinding:tokenReference reference="gen_responseapp_encparts_
        response"/>
      </securitybinding:keyInfo>
```

```
      <securitybinding:tokenGenerator classname="com.ibm.ws.wssecurity.wssapi.token.
      impl.CommonTokenGenerator" name="gen_responseapp_signparts_response">
        <securitybinding:valueType localName="http://docs.oasis-open.org/wss/2004/01/
        oasis-200401-wss-x509-token-profile-1.0#X509v3"/>
        <securitybinding:jAASConfig configName="system.wss.generate.x509"/>
        <securitybinding:callbackHandler classname="com.ibm.websphere.wssecurity.
        callbackhandler.X509GenerateCallbackHandler">
          <securitybinding:keyStore path="*MQSIBROKERSTOREPATHMQSI*" storepass=
          "*MQSIBROKERSTOREPWDMQSI*" type="JKS"/>
          <securitybinding:key alias="wmbcert" keypass="*MQSIBROKERSTOREKEYPASS
          wmbcertMQSI*" name="CN=WMBServer, OU=ISSW, O=IBM, ST=Char, C=US"/>
        </securitybinding:callbackHandler>
      </securitybinding:tokenGenerator>
      <securitybinding:tokenGenerator classname="com.ibm.ws.wssecurity.wssapi.token.
      impl.CommonTokenGenerator" name="gen_responseapp_encparts_response">
        <securitybinding:valueType localName="http://docs.oasis-open.org/wss/2004/01/
        oasis-200401-wss-x509-token-profile-1.0#X509v3"/>
        <securitybinding:jAASConfig configName="system.wss.generate.x509"/>
        <securitybinding:callbackHandler classname="com.ibm.websphere.wssecurity.
        callbackhandler.X509GenerateCallbackHandler">
          <securitybinding:keyStore path="*MQSIBROKERSTOREPATHMQSI*" storepass=
          "*MQSIBROKERSTOREPWDMQSI*" type="JKS"/>
          <securitybinding:key alias="wmbclientcert" keypass="*MQSIBROKERSTOREKEYPASS
          wmbclientcertMQSI*" name="CN=WMBClient, OU=ISSW, O=IBM, ST=Char,C=US"/>
        </securitybinding:callbackHandler>
      </securitybinding:tokenGenerator>
    </securitybinding:securityOutboundBindingConfig>
    <securitybinding:securityInboundBindingConfig/>
  </securitybinding:securityBinding>
</securitybinding:securityBindings>'
```

Associate the WS-Security settings with the message flow

Follow the same steps described in the previous section to associate the policy set PS1 and the policy set binding PSB1_Provider with the message flow, as shown in Figure 28:

Figure 28. Associating WS-Security settings with the message flow



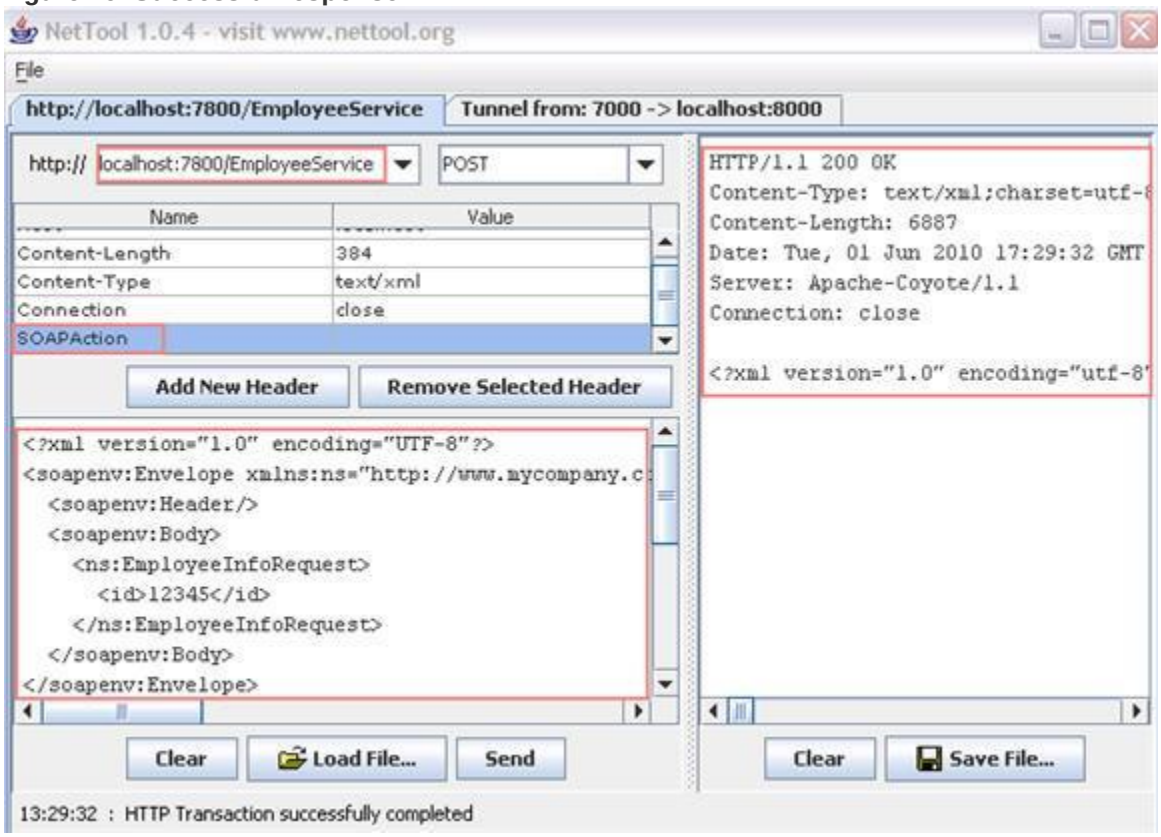
After any changes to policy sets or policy set bindings are saved directly to the associated broker, the broker or the execution group where the message flow is deployed must be restarted for the new configuration to take effect.

Run tests to evaluate message signing and encryption

1. Open the test utility NetTool. Click **Add New Header** to add the header SOAPAction.
2. Provide the Web service URL http://localhost:7800/EmployeeService. Load the file EmployeeInfoRequest.xml. There is no wsse:Security under the header in the test SOAP message, because no authentication is required.

3. Click **Send** to invoke the Web service message flow. You should see this response:

Figure 29. Successful response



The signed and encrypted response message is shown below:


```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
    wssecurity-secext-1.0.xsd" soapenv:mustUnderstand="1">
      <wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
      wssecurity-utility-1.0.xsd" wsu:Id="wssecurity_signature_id_0">
        <wsu:Created>2010-06-01T17:29:22.15Z</wsu:Created>
      </wsu:Timestamp>
      <wsse:BinarySecurityToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
      200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="x509bst_2" EncodingType="http://docs.
      oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
      ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
      profile-1.0#X509v3">
      MIICETCCAXqgAwIBAgIES88h0DANBgkqhkiG9w0BAQUFADBNMQswCQYDVQQGEwJVUzENMAsGA1UECBMEQ2h
      hcjEMMAoGA1UEChMDSUJNMQ0wCwYDVQQLEwRJU1NXMRIwEAYDVQQDEwlXTUJTZXJ2ZXIwHhcNMTAwNDIxMT
      YwMzI4WhcNMjAwNDE4MTYwMzI4WjBNMQswCQYDVQQGEwJVUzENMAsGA1UECBMEQ2hhcjEMMAoGA1UEChMDS
      UJNMQ0wCwYDVQQLEwRJU1NXMRIwEAYDVQQDEwlXTUJTZXJ2ZXIwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJ
      AoGBAKdjFeBi+zPL4Hp2LzoZy2DFcx/FXobChjZBMylmkwuVYR6JvJC71GRd4ViSyLM8zdirBm2Ef+KtV/R
      WqDEZFhEycHnFOhlZFIgoHR+3Sp4jrT72DWRCUzVqpmv0pjbHldXmvqwfXYO9AHDg0pXcgyXoGhmk/kce50
      7Yi/fqrpB1AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAnSdZ/R1YSc7KqgHOAi30kuV/ISPeCpfZwWBy7EAGn
      O7Iv8k3HOKllMmyMRJT39HNh62e0/Pl2+t0YA38XWSo2Y3437xIgukFIJIYIEHKCnNZBThg6H1nKbg269Um
      ZOu4QRFO9c3NLa+HwmmRm6iu3mGJU/QFEIOZlpR6DOPkchA=</wsse:BinarySecurityToken>
      <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
        <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5">
        </EncryptionMethod>
```

```
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier EncodingType="http://docs.oasis-open.org/wss/2004/01/
          oasis-200401-wss-soap-message-security-1.0#Base64Binary"
          ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-
          token-profile-1.0#X509SubjectKeyIdentifier">V6GdFyibFnet6w3QTJgmjG+x1rY=
          </wsse:KeyIdentifier>
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
      <CipherData>
        <CipherValue>
        FikfNEXeKPY1tUPOmldv2z3fuTvOTEAvsSdnaJhckFQSdyz9aJOIveOpDaqoZdOEaxHRTbPPlTSBMbu
        xvSKdlVrPboxIP8riupzfEVoENOyv2y73lq9lYm7zAl3guTW6u267Jm5tCVaoe2Ts12j5AtkqYoWB8y
        7TNPZJixkGEiU=</CipherValue>
      </CipherData>
      <ReferenceList>
        <DataReference URI="#wssecurity_encryption_id_3"></DataReference>
        <DataReference URI="#wssecurity_encryption_id_4"></DataReference>
      </ReferenceList>
    </EncryptedKey>
    <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#" Id="wssecurity_
    encryption_id_3" Type="http://www.w3.org/2001/04/xmlenc#Element">
      <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc">
      </EncryptionMethod>
      <CipherData>
        <CipherValue>8pcJa2f7GHjp/SDopXeXlnYz1YU9XtogGNBFeBftL7goQOolvM85ZR+ruJbt2FqiYW
        IsyextSkNDGkl/DN8r7WNlEEht6pXswMfL7NmRhTl6DoHERjX4bi27+NZBNdYswfEYfOkB9dvI8EOCY
        zQzRSvd1cRZgup+BoZVojYN3n5WHsGCitYNIGj6hJZdeaQuVr4hrsi+7kxV/amuuLtTysYPlLOfV9A4
        xI9OyUF5kRdaC9mZsCuPEePQGSmTqKoMCgNDjen4UmO/sSFYgsW5Wqq/XDWbxPkzYrApqi6wve2p3me
        F2x+2QReFxtIxjrsWjLumX5vUrL2RT5bQm+lF9q1cs4wJrGgff0R5lXrO7xMic9OV3FTJLSpnjZPDbT
        +A4Whz5HNGldLveu1bdtCzq+vvM56aE4skZDvO66GpUsMedsuV+M+Qcf8Pu1M6izDorP7iic6Dvgbpw
        tiSnnpaGKUd59G71QldRJfZrdAL5fhiN1SMZIVYJraUg9cUFyRMy7tBRrizzg/7fcI5sNxcpx/+NGUJ
        px1m9SNoadbkFlnOvKzxgzkK6uLz2oafkd3Pg72GERJi9FHOCJ6aig9hZKhLRQ4rHTvOHG/r1UGVv1w
        d7rG+3D6D3cw6aKC6chTT+iRdAVsTrqGCtJ1OS97yP6rqfdnadeU1QGbJfry+sjEXgyfIzUHWeezegd
        xgNLmoj8g8t6OZ1nRTzZvOaThPZA39E3Sr6nwH9tzbQz9lUvh9uRsUNr8NiTTNQJ4ROrIMHOPN/pzeY
        kQgkstq3VDhUomy/2hMCxARqbfytMoxAqJzc/FZSugE8q9f1G+jJC4E4oga8PtON3gtuAVIut9yfJ7J
        BNSn9BfIW7VGCEMU7n/Z6MlOdofNWbiEzfOXBF4e255bsEzkPEKemLtSke2hAfXy1BPKgxuLG8EZFRK
        VNEg6Eo1usDlN6opHfdLMtXIeNqERgpxfOjLWkW4O8+UwN/XoR7/nepK78BxR/qwhlGV35DlrW82+R1
        au5F9laYHHDqNhuBoJHhirri5jTDDgO3HswwTpyk3KYQMPhYcNlR5PZJMUZbcGbRVONYbitBcDgHooh
        sQm5bBpE+KxRxKOD5ANjZ7cZt0UbMj1Y44vwUaWLxr6VRSq+vUlaYIKBL1igOjRCcF98dVKqTgBaDOE
        wn8ar8BAXW9alhNOIEQTpvXkiLrjjksMFXvqGoh0fFo1eg+3r+foIvbO7O7ULV+PhfLK8WCYepy2BvE
        Wf1x8QMlWwRSuXQuEx6ZwcYw/GvSbWAhQgqSaaIoq7x5ujrkPlOjP2oYEzLEhhPO+1CRVBUou4VoL/u
        ZdWLwMRP3H2gtoubK/yexFiwO78cNvqs6ZmPPTpMaUOXwrxaZoEwA+M9Q71KqENC6YDP52RvOyYpHs3
        Zskqc93bpUsOvSTmN9xHYjMqkDJi9ZnTfrnV3bS76gZqisbFq/6X3k412trpwVtAekP5/SS7lJyyQ5r
        Kg9nufoglT2djx5WWoMViLklMFOM9OYL+UeCvag6wUy8GS5W3+cJCB88fq4dGnrEQCv2BeURQPGxiuJ
        EwJrTpLkJUxW5ljhK9+Tgem8qqnW1kHIHZ8U6LPEAdwurLF8t5UH+RJ7G+hsmMZm4D52UCtOOvYpdwq
        /44YElFn4VrMBO3OxXCgxJGP7hCakYguVObFP/hENoKWrMD2ZSKvKYPVpsMrlL2S8QQ67epjO9jOcKY
        n5NlAlANy7PHuc77KQCnoOGWsY8iSQT9cT58IaAph8KlxwAv9wfBTYSfjHdkcE/fmbh1uNiWUSN4C9u
        csHb+q8SjpJiEtDlG42AjAJtmINfGn+Im4K2jq3MrUSTW3KrMQuy7w4VE9zCBGHOVz+udS83o1AxQ3V
        gU2ZUBPIpcO9Qj4hlvFDsdMeCRlL+mdWDwOvFtmmA5wMa8uAuurlnH2S1kqrH497VXLqyU2rEUKUHT+
        NeZtO+oOTBBqgZN6yIzXCT7jlDXOW9u+3PV2FO8bcf4nQmMuDlChnUfKTZ+4Eml9ueEouRei1a+AySP
        ziQHC3sgmle9Zg++7zD9eDlolByiROIZVLje81psyaF8tOS+sXNSZVF7mfweOLYSJW3p3FBmHMtkukv
        OOddCbY29obVfQA91LOG/fDt6ue3Zcqmvclzb7z9JOGikNPVYCokNunFfOTsMoikiCDu1XtHyqG18TW
        cuMZyvyJk2d6/4Y66eTZf47FSByOFz5NVRVo/wl1CFX4uZD46UXQ1CHkwGCV8jTaIOie+RTM6bBPTXq
        i4i5hYZ3UHOZcY9U2AhZUyQsVO5VhaKqiVcNVZgcWbeFNSgOInYT8THuT5pJpC5MniLyYOgjggY36Q9
        peKI67KSMM6gb/OdcpQ6Jts3aUDawUab59jNMmxBW7ggISPHzWAiVsvIQOl5PQYYYPWuu6mb12ZCdMm
        Tn91Wmn3ElvsHlWbKuZNVcHieORYoXjjAQ+1QqOV6jw+mYyEFjNYGHu5lIrjlqahx5DS6AQbepdjmAB
        XuMmNjYY/ljk6dkgyO5JmROhvHgg3gUjKOYmYlKYPchgJ4z/mt1C4GOHxjfTpOPRhIUkkYr1j5Ce/wg
        O5qSIOFItEagyNdK2tMXpQVohFDxKiyV7vsX9P6vGHtFVLgJJTqHUrOOSs3+f4IYOa1NiACGxRZsscB
        BV2fEh94blAbuDRJ9mPoXxo35VSzpCPowsdtZMiwTYetw/q5w2CXc8LuFafDWjZoAdjv7veQpsRDZ/z
        trkWHO4dYgvzf1rKgRnOHt4GN53OHLj+fQmI/lxNKmC4/R/cul2P2Fyuxr456El44Ioo3PkL6h2YGi1
        LtreOnVHNiJmHrerGzLhWWpq3YeAhZOLJyQMr6I7Gs5YXlOHysm4+CXN6rueHRgyurglxEZFVVg/Tlc
        Ip4+pgHv3EKXr9KqFmV5fKaLiXbWilP8IMsuAtuteCTZWUN/KLfORYn1QHHz2yAy+MQcIXxglSHz+Nk
        V2IPpFg3tiG53sTgbcR7Y2l9eiUS4hZ4hFYnqXGlbBKc85siwLsNztetHBqUQif36bSNB/TN6+qGszO
        2K4W</CipherValue>
      </CipherData>
    </EncryptedData>
```

```
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
  wssecurity-utility-1.0.xsd" wsu:Id="wssecurity_signature_id_1">
    <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#" Id="wssecurity_encryption_
    id_4" Type="http://www.w3.org/2001/04/xmlenc#Content">
      <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc">
      </EncryptionMethod>
      <CipherData>
        <CipherValue>owaONrOnpEpw/B8NTPMPWERNOgqrDPnsTL/g4Ymr9ODQ+r91MPIhkYvUGsdDfmPQbyNS
        1WG9wSTFg6BYYyHrgdCQAJM9fHGgDMcHjFEK/nIQDIfC1A34VSnEAc5M36VvL9ZcoI2HKNACb2SOd72ZS
        IvzKX6M9KvEv6v4LXXjdCSOOPuq6WqW3XgZ16PRlAPgFj8DTnwZk1TykT77CTNR8elf1brqDCNsyN+BBA
        8m+7SiMEIjvNDfz49tW14un8i2k/PbQGU2FE2ooo+k4DGTXqhQNksgES3p+y5Y6WbhL1IddgCLoqllfhr
        Hn9QlbmhBEC6e6AdK7VgfNPOQotxMGPiLsHcPWLiLF7j6TgTGsFWpo+6W/n11Ja8MoIXBQwJf5CcKLDtm
        YuO/98cOwT+5mPmMXgHwWXRHOKpaeP4FJdIJhE8yV281QCOtUwDOtZr4vFsBpn1a8Ebtnrye0kYclw==
        </CipherValue>
      </CipherData>
    </EncryptedData>
  </soapenv:Body>
</soapenv:Envelope>
```


Conclusion

WebSphere Message Broker supports WS-Security for identity authentication, message signing and encryption. In this article, a message flow as the Web service provider is built based on a WSDL file. The WS-Security is implemented on the message flow using the security profile, and the policy set and binding.