BEA BEA AquaLogic Service Bus™

Configuring Message-Level Security Between .NET 2.0 and ALSB

Version 3.0
Revised: February 2008

Contents

Message-Level Security

CHAPTER 1

Message-Level Security

This chapter explains how to configure message-level security between .NET 2.0 and AquaLogic Service Bus. The chapter contains the following topics:

Message-Level Security Between .NET 2.0 and AquaLogic Service Bus

You can set up Message-level security between the Microsoft .NET 2.0 framework and AquaLogic Service Bus. Message-level security applies security checks to a SOAP message after a Web services client establishes a connection with an AquaLogic Service Bus proxy service or business service and before the proxy service or business service processes the message.

What is .NET?

The .NET framework is a software component that you can add to the Microsoft Windows operating system. It provides pre-coded solutions to common program requirements, and manages the execution of programs written specifically for the framework.

Message-Level Security Configuration in .NET For AquaLogic Service Bus Interoperability

This section provides the steps that you need to perform for .NET 2.0 and for AquaLogic Service Bus to configure message-level security.

WARNING: Before you perform these steps, you must follow the steps in Configuring Message-Level Security for Web Services in the *Security Guide* to configure inbound and outbound messaging for AquaLogic Service Bus.

To configure message-level security between .NET and AquaLogic Service Bus:

1. Verify that you completed the steps to configure inbound and outbound messaging for AquaLogic Service Bus. See the Warning above for instructions.
2. Download Web Service Enhancements (WSE) 3.0 from Microsoft and install it. WSE 3.0 is a SOAP extension managed API (`Microsoft.Web.Services3.dll`) that is compatible with the .NET 2.0 framework.
3. After you install WSE 3.0, you must enable the WSE features for your web application and enable WSE Soap Protocol Factory support. You can enable both these features using wizards in Visual Studio. See http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wse3.0/html/5a8f03b1-16ac-4c5c-9d9e-132a9c0b628a.asp for details.

   After you enable WSE 3.0, you will notice the following restrictions:

   – WSE 3.0 no longer supports WS-Policy and therefore, WS-SecurityPolicy for configuration purposes, as it did in .NET 1.1 and WSE 2.0. WSE 3.0 supports only a proprietary policy configuration using the `wse3policyCache.config` file (or equivalent .NET code) that provides similar features to those in WSE 2.0. One consequence of this is that the WSDLs for the .NET web service no longer contain WS-Policy statements. On the other hand, AquaLogic Service Bus supports a WebLogic Server-proprietary format that is based on the assertions described in the December 18, 2002 version of the Web Services Security Policy Language (WS-SecurityPolicy) specification. In order to consume .NET WSDLs in AquaLogic Service Bus, you must incorporate the equivalent AquaLogic Service Bus proprietary version of WS-Policy in the WSDL.

     The WSDL code sample in Listing 1-1 on page 7 shows how to configure WS-Policy for message-level identity propagation, confidentiality, and integrity in AquaLogic Service Bus.

– WSE 3.0 provides policy configuration for a few Turnkey Security Assertions in the `wse3policyCache.config` file, which can be selected with a wizard in Visual Studio. The certificate that maps to providing message-level security (encryption and signing, for example) is MutualCertificate10, which is compatible with the WS-Security 1.0 specification that ALSB currently supports. For details on configuring the MutualCertificate10 Security Assertion, see http://msdn2.microsoft.com/en-us/library/aa480581.aspx.

– The WSE Soap Protocol Factory does not support security with SOAP 1.2. The generated client stubs using the Web Reference option in Visual Studio contain the security-enabled operations only if you select SOAP 1.1. Message-level security interoperability works only with SOAP 1.1.

– As with .NET 1.1 and WSE 2.0, you must disable automatic signing of WS-Addressing headers and timestamps that are configured by default. You must change some of the properties in the `wse3policyCache`.config file, as shown in the following example:

```
Default Config
<protection>
     <request signatureOptions="IncludeAddressing, IncludeTimestamp,
          IncludeSoapBody" encryptBody="true" />
     <response signatureOptions="IncludeAddressing, IncludeTimestamp,
          IncludeSoapBody" encryptBody="true" />
     <fault signatureOptions="IncludeAddressing, IncludeTimestamp,
          IncludeSoapBody" encryptBody="false" />
</protection>

Required Config
<protection>
    <request signatureOptions="IncludeSoapBody" encryptBody="true" />
    <response signatureOptions="IncludeAddressing, IncludeTimestamp,
IncludeSoapBody" encryptBody="true"
         />
    <fault signatureOptions="IncludeSoapBody" encryptBody="false" />
</protection>
```

– By default, WSE 3.0 expects the key wrapping algorithm to be OAEP. However, ALSB uses the RSA15 algorithm. If the configuration remains as OAEP, the following exception appears: `Microsoft.Web.Services3.Security.SecurityFault: An unsupported signature or encryption algorithm was used System.Exception: WSE3002: The receiver is expecting the key wrapping algorithm to be` http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p, `but the incoming message used` http://www.w3.org/2001/04/xmlenc#rsa-1_5. You can change the key wrapping algorithm by configuring the security token manager.

To avoid this error, add the following configuration to the `web.config` file (on the .NET web service) and the `app.config` file (on the .NET client side) under the `<microsoft.web.services3> <security>` elements:

```
<binarySecurityTokenManager>
     <add valueType="http://docs.oasis-open.org/wss/2004/01/
          oasis-200401-wss-x509-token-profile-1.0#X509v3";>
          <keyAlgorithm name="RSA15" />
     </add>
</binarySecurityTokenManager>
```

This configuration forces WSE to use RSA15 instead of OAEP.

– For Username Token Authentication, .NET provides a `usernameForCertificateSecurity` turnkey assertion that secures the communication channel between the client and the service at the message layer using the service's X.509 certificate. However, this certificate depends on the ability to reference `<EncryptedKey>` elements as security tokens, and enables the option for signature confirmation to correlate a response message with the request that prompted it. Both of these features are available only in WS-Security 1.1. Consequently, this turnkey assertion cannot be used with AquaLogic Service Bus because it supports only WS-Security 1.0.

  An alternative for Username Token Authentication is the .NET `usernameOverTransportSecurity` turnkey assertion, which assumes that communication between the client and service will be secured at the transport layer. This approach is WS-Security 1.0 compatible and supports message-level authentication over SSL. If you want to combine the `usernameOverTransportSecurity` turnkey assertion with other message-level security mechanisms, such as encryption and signing, you must write custom code in .NET.

AquaLogic Service Bus Configuration for Message-Level Security with .NET

Before you configure AquaLogic Service Bus, the following conditions must exist:

- A .NET client invokes an AquaLogic Service Bus proxy with a plain text message (for example, message-level security does not exist between the .NET client and the AquaLogic Service Bus proxy).
- AquaLogic Service Bus enforces outbound message-level security on the SOAP request.

Note: For cases where the .NET client has message-level security enabled, you can use AquaLogic Service Bus as a pass-through proxy.

To configure AquaLogic Service Bus for message-level security with .NET:

1. Change the encryption algorithm from `tripledes-cbc` to `aes256-cbc`:

   ```
   <wssp:EncryptionAlgorithm URI="http://www.w3.org/2001/
                           04/xmlenc#aes256-cbc"/>
   ```

2. Change the sign.xml policy on the WSDL. This attribute is on the integrity assertion element.

   ```
   <wssp:Integrity SignToken='false' .... >
   ...
   </wssp:Integrity>
   ```

   By default this value is true.

3. The .NET web service expects the WS-Addressing `<wsa:To>` element to contain its own URL. As the .NET client first invokes the AquaLogic Service Bus proxy, the `<wsa:To>` element is originally set to the AquaLogic Service Bus proxy URL. Change this URL to the URL of the .NET web service in the AquaLogic Service Bus proxy message flow, using a `Replace` action as shown in the following example:

   ```
   Original URL
   <wsa:To wsu:Id="To_1mbmRK4w0bo2Dz1z"
   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
   wssecurity-utility-1.0.xsd";>http://localhost:7001/SecurityALSBProxy<;/
   wsa:To>


   URL after Replace Action
   <wsa:To wsu:Id="To_1mbmRK4w0bo2Dz1z"
   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
   wssecurity-utility-1.0.xsd";>http://localhost/SimpleSecurity/
   SecurityService.asmx<;/wsa:To>
   ```

   If you do not change this URL, the following error appears:

   ```
   Microsoft.Web.Services3.Addressing.AddressingFault: Destination
   Unreachable System.Exception: WSE846: The header must match the actor
   URI value of the web service. The actor URI value can be explicitly
   specified using SoapActorAttribute on the ASMX class. In the absence
   of the attribute, the actor URI is assumed to be equal to the HTTP
   Request Url of the incoming message. The header received contained
   "http://localhost:7001/SecurityALSBProxy"; while the receiver is
   expecting "http://localhost/SimpleSecurity/SecurityService.asmx";
   ```

4. The .NET client includes its own Timestamp elements to the SOAP header. AquaLogic Service Bus adds an additional Timestamp header that results in the following error:

   ```
   Microsoft.Web.Services3.Security.SecurityFault: An error was discovered
   processing the header Microsoft.Web.Services3.Security.Security
   FormatException: WSE001: The input was not a valid Security element
   because it contains more than one Timestamp child element.
   ```

   To solve this issue, use a Delete action to remove the original Timestamp elements that the .NET client adds in the message flow.

5. Add the CertificateRegistry certification path provider. You add this from the WLS Administration Console from
   `realm -> Providers ->Certification Path -> New`
   and then select CertificateRegistry from the drop down.

   Activate the change and restart the server.

   After you restart the server, edit the CertificateRegistry provider you just created. From the Management tab add the following three certificates:

   – The public certificate of ALSB

   – The public certificate of .NET

   – The root agency (issuer of these certificates)

   Note: One way to add the certificates is to import them from a jks store via the Migration tab. Provide the actual path of the identity store.

6. On the Configuration (Common) tab for the CertificateRegistry provider, select Current Builder to make it the current builder.

   Save these changes. Then, activate and restart the server.

7. The WLS keystore requires these same certificates:

   – The public certificate of ALSB

   – The public certificate of .NET

   – The root agency (issuer of these certificates)

   You configure the identity and trust keystores for a WebLogic Server instance on the server Configuration: Keystores page. To do this, see *Configure Identity and Trust* in the WLS online help.

Sample WSDL File

The sample WSDL in Listing 1-1 shows how to configure `WS-Policy` for message-level identity propagation, confidentiality, and integrity in ALSB.

Listing 1-1 Configuring WS-Policy for Message-Level Security

```
<?xml version='1.0' encoding='UTF-8'?>
<definitions name="SecureHello WorldServiceDefinitions"targetNamespace=
     "http://www.bea.com"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:s0="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
     wssecurity-utility-1.0.xsd"
xmlns:s1="http://www.bea.com"
xmlns:s2="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
     <wsp:UsingPolicy xmlns:n1="http://schemas.xmlsoap.org/wsdl/"
          n1:Required="true"/>
     <wsp:Policy s0:Id="Encrypt.xml">
          <wssp:Confidentialityxmlns:wssp="http://www.bea.com/wls90/
               security/policy">
               <wssp:KeyWrappingAlgorithm URI="http://www.w3.org/2001/04/
                    xmlenc#rsa-1_5"/>
               <wssp:Target>
                    <wssp:EncryptionAlgorithm URI="http://www.w3.org/2001/
                         04/xmlenc#aes256-cbc"/>
                   <wssp:MessageParts Dialect="http://schemas.xmlsoap.org
                         /2002/12/wsse#part">wsp:Body()
                         </wssp:MessageParts>
               </wssp:Target>
               <wssp:KeyInfo>
                    <wssp:SecurityToken TokenType="http://docs.oasis-open.
                         org/wss/2004/01/oasis-200401-wss-x509-token-
                         profile-1.0#X509v3"/>
                    <wssp:SecurityTokenReference>
                         <wssp:Embedded>
                             <wsse:BinarySecurityToken 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"
                                        xmlns:wsse="http://docs.oasis-open.org/
                                        wss/2004/01/oasis-200401-wss-wssecurity-
                                        secext-1.0.xsd">MIIB7DCCAZYCEN+FHomYRZU
                                        YPLiIutc0lIIwDQYJKoZIhvcNAQEEBQAweTELMAk
                                        GA1UEBhMCVVMxEDAOBgNVBAgTB015U3RhdGUxDzA
                                        NBgNVBAcTBk15VG93bjEXMBUGA1UEChMOTXlPcmd
                                        hbml6YXRpb24xGTAXBgNVBAsTEEZPUiBURVNUSU5
                                        HIE9OTFkxEzARBgNVBAMTCkNlcnRHZW5DQUIwHhc
                                        NMDYwNjA3MDQ0MDM2WhcNMjEwNjA4MDQ0MDM2WjB
                                        6MQswCQYDVQQGEwJVUzEQMA4GA1UECBYHTXlTdGF
                                        0ZTEPMA0GA1UEBxYGTXlUb3duMRcwFQYDVQQKFg5
                                        NeU9yZ2FuaXphdGlvbjEZMBcGA1UECxYQRk9SIFR
                                        FU1RJTkcgT05MWTEUMBIGA1UEAxYLYmFuZ3BsdHc
                                        zazIwXDANBgkqhkiG9w0BAQEFAANLADBIAkEAxv2
                                        nWByAF2Xr9wrb06ydrrcqPt2VQa0xcwfdZZ6oGlj
                                        1TXq+G5/Q82v7CdxjyWUQBuAzduQx9wFCrAe/aWV
                                        pgQIDAQABMA0GCSqGSIb3DQEBBAUAA0EARbwfl8w
                                        X915jL5reY+isriNF0EfUs5ck53WRNowiapJx2ea
                                        ZE03quksJgeJ0z0HekkR/aTQnkMV1xIt1HxMKRw=
                                        =</wsse:BinarySecurityToken>
                    </wssp:Embedded>
               </wssp:SecurityTokenReference>
          </wssp:KeyInfo>
     </wssp:Confidentiality>
</wsp:Policy>
<wsp:Policy s0:Id="Auth.xml">
     <wssp:Identity xmlns:wssp="http://www.bea.com/wls90/security/
     policy">
          <wssp:SupportedTokens>
               <wssp:SecurityToken TokenType="http://docs.oasis-open.
                    org/wss/2004/01/oasis-200401-wss-username-token
                    -profile-1.0#UsernameToken">
```

```
                    <wssp:UsePassword Type="http://docs.oasis-open.
                         org/wss/2004/01/oasis-200401-wss-username
                         -token-profile-1.0#PasswordText"/>
               </wssp:SecurityToken>
          </wssp:SupportedTokens>
     </wssp:Identity>
</wsp:Policy>
<wsp:Policy s0:Id="Sign.xml">
     <wssp:Integrity SignToken='false'
xmlns:wls="http://www.bea.com/wls90/security/
          policy/wsee#part"xmlns:wssp="http://www.bea.com/wls90/
          security/policy" xmlns:wsu="http://docs.oasis-open.org/wss
          /2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
          <wssp:SignatureAlgorithm URI="http://www.w3.org/2000/09/
               xmldsig#rsa-sha1"/>
          <wssp:CanonicalizationAlgorithm URI="http://www.w3.org/
               2001/10/ xml-exc-c14n#"/>
          <wssp:Target>
               <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09
                    /xmldsig#sha1"/>
               <wssp:MessageParts Dialect="http://www.bea.com/wls90/
                    security/policy/wsee#part">
               wls:SystemHeaders()
          </wssp:MessageParts>
          </wssp:Target>
          <wssp:Target>
               <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09
                    /xmldsig#sha1"/>
               <wssp:MessageParts Dialect="http://www.bea.com/wls90/
                    security/policy/wsee#part">
               wls:SecurityHeader(wsu:Timestamp)
          </wssp:MessageParts>
          </wssp:Target>
          <wssp:Target>
               <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09/
                    xmldsig#sha1"/>
               <wssp:MessageParts Dialect="http://schemas.xmlsoap.
                    org/2002/12/wsse#part">
```

```
wsp:Body()
</wssp:MessageParts>
</wssp:Target>
<wssp:SupportedTokens>
     <wssp:SecurityToken IncludeInMessage="true" TokenType=
          "http://docs.oasis-open.org/wss/2004/01/oasis-
          200401-wss-x509-token-profile-1.0#X509v3">
          <wssp:TokenIssuer>CN=CACERT,OU=FOR TESTING ONLY,
          O=MyOrganization,L=MyTown,ST=MyState,C=US,1.2.
          840.113549.1.9.1=#160f737570706f7274406265612e636
          f6d,CN=Demo Certificate Authority Constraints,OU=
          Security,O=BEA WebLogic,L=San Francisco,ST=
          California,C=US,1.2.840.113549.1.9.1=#16107365637
          572697479406265612e636f6d,CN=Demo Certificate
          Authority Constraints,OU=Security,O=BEA WebLogic,
          L=San Francisco,ST=California,C=US,CN=CertGenCAB,
          OU=FOR TESTING ONLY,O=MyOrganization,L=MyTown,ST=
          MyState,C=US,CN=Equifax Secure eBusiness CA-1,O=
          Equifax Secure Inc.,C=US,CN=VeriSign Class 1
          Public Primary Certification Authority - G3,OU=
          (c)1999 VeriSign\, Inc. - For authorized use only,
          OU=VeriSign Trust Network,O=VeriSign\, Inc.,C=US,
          OU=VeriSign Trust Network,OU=(c) 1998 VeriSign\,
          Inc. - For authorized use only,OU=Class 2 Public
          Primary Certification Authority - G2,O=VeriSign\,
          Inc.,C=US,CN=VeriSign Class 3 Public Primary
          Certification Authority - G3,OU=(c) 1999
          VeriSign\,Inc. - For authorized use only,OU=
          VeriSign Trust Network,O=VeriSign\,Inc.,C=US,CN=
          Entrust.net Client Certification Authority,OU=(c)
          2000 Entrust.net Limited,OU=www.entrust.net/
          GCCA_CPS incorp. by ref. (limits liab.),O=Entrust
          .net,OU=Go Daddy Class 2 Certification Authority,
          O=The Go Daddy Group\, Inc.,C=US,CN=GTE Cyber
          Trust Global Root,OU=GTE CyberTrust Solutions\,
          Inc., O=GTE Corporation,C=US,CN=Entrust.net
          Secure Server Certification Authority,OU=(c) 2000
          Entrust.net Limited,OU=www.entrust.net/SSL_CPS
```

```
incorp. by ref. (limits liab.),O=Entrust.net,OU=
Class 1 Public Primary Certification Authority,
O=VeriSign\, Inc.,C=US,1.2.840.113549.1.9.1=#161
9706572736f6e616c2d6261736963407468617774652e636
f6d,CN=Thawte Personal Basic CA,OU=Certification
Services Division,O=Thawte Consulting,L=Cape
Town, ST=Western Cape,C=ZA,OU=VeriSign Trust
Network, OU=(c) 1998 VeriSign\, Inc. - For
authorized use only,OU=Class 1 Public Primary
Certification Authority - G2,O=VeriSign\, Inc.,
C=US,CN=Entrust.net Secure Server Certification
Authority,OU=(c) 1999 Entrust.net Limited,OU=
www.entrust.net/CPS incorp. by ref.(limits iab.),
O=Entrust.net,C=US, 1.2.840.113549.1.9.1=#161c706
572736f6e616c2d667265656d61696c407468617774652e63
6f6d,CN=Thawte Personal Freemail CA,OU=
Certification Services Div,O=Thawte Consulting, L
=Cape Town,ST=Western Cape,C=ZA,OU=Class 3 Public
Primary Certification Authority,O=VeriSign\, Inc.
C=US,CN=GTE CyberTrust Root,O=GTE Corporation,C=
US,CN=VeriSign Class 2 Public Primary Certificate
Authority - G3,OU=(c) 1999 VeriSign\, Inc. - For
authorized use only,OU=VeriSign Trust Network,O=
VeriSign\,Inc.,C=US,1.2.840.113549.1.9.1=#1617736
5727665722d6365727473407468617774652e636f6d,CN=
Thawte Server CA,OU=Certification Services
Division,O=Thawte Consulting cc,L=Cape Town,ST=
Western Cape,C=ZA,OU=Equifax Secure Certificate
Authority,O=Equifax,C=US,1.2.840.113549.1.9.1=#16
1b706572736f6e616c2d7072656d69756d407468617774652
e636f6d,CN=Thawte Personal Premium CA,OU=
Certification Services Division,O=Thawte
Consulting,L=Cape Town,ST=Western Cape,C=ZA,1.2.
840.113549.1.9.1=#16197072656d69756d2d73657276657
2407468617774652e636f6d,CN=Thawte Premium Server
CA,OU=Certification Services Division,O=Thawte
Consulting cc,L=Cape Town,ST=Western Cape,C=ZA,
OU=VeriSign Trust Network,OU=(c) 1998 VeriSign\,
```

```
                    Inc. - For authorized use only,OU=Class 3 Public
                    Primary Certification Authority - G2,O=VeriSign\,
                    Inc.,C=US,CN=Entrust.net Certification Authority
                    (2048),OU=(c) 1999 Entrust.net Limited,OU=www
                    .entrust.net/CPS_2048 incorp. by ref. (limits
                    liab.),O=Entrust.net,1.2.840.113549.1.9.1=#1611
                    696e666f4076616c69636572742e636f6d,CN=http://www.
                    valicert.com/,OU=ValiCert Class 2 Policy
                    Validation Authority,O=ValiCert\, Inc.,L=Vali
                    cert Validation Network,CN=Baltimore CyberTrust
                    Root, OU=CyberTrust,O=Baltimore,C=IE,OU=Secure
                    Server Certification Authority,O=RSA Data
                    Security\, Inc.,C=US,CN=Entrust.net Client
                    Cert Authority,OU=(c) 1999 Entrust.net Limited,
                    OU=www.entrust.net/Client_CA_Info/CPS incorp. by
                    ref. limits liab.,O=Entrust.net,C=US,CN=GeoTrust
                    Global CA,O=GeoTrust Inc.,C=US,CN=GTE CyberTrust
                    Root 5,OU=GTE CyberTrust Solutions\, Inc.,O=GTE
                    Corporation,C=US,OU=Starfield Class 2
                    Certification Authority,O=Starfield
                    Technologies\, Inc.,C=US,CN=Equifax Secure
                    Global eBusiness CA-1,O=Equifax Secure Inc.,C=US,
                    CN=Baltimore CyberTrust Code Signing Root,OU=
                    CyberTrust,O=Baltimore,C=IE,OU=Class 2 Public
                    Primary Certification Authority,O=VeriSign\,
                    Inc.,C=US,OU=Equifax Secure eBusiness CA-2,O=
                    Equifax Secure,C=US,</wssp:TokenIssuer>
                </wssp:SecurityToken>
            </wssp:SupportedTokens>
        </wssp:Integrity>
        <wssp:MessageAge Age="60" xmlns:wssp="http://www.bea.com/wls90/
            security/policy"/>
    </wsp:Policy>
    <types>
        <xs:schema attributeFormDefault="unqualified" elementFormDefault=
            "qualified" targetNamespace="http://www.bea.com" xmlns:s0="
            http://www.bea.com" xmlns:s1="http://schemas.xmlsoap.org
            /wsdl/soap/" xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/
```

```
          09/policy" xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="sayHello">
               <xs:complexType>
                    <xs:sequence>
                        <xs:element name="s" type="xs:string"/>
                    </xs:sequence>
               </xs:complexType>
          </xs:element>
          <xs:element name="sayHelloResponse">
               <xs:complexType>
                    <xs:sequence>
                         <xs:element name="return" type="xs:string"/>
                    </xs:sequence>
               </xs:complexType>
          </xs:element>
     </xs:schema>
</types>
<message name="sayHello">
     <part element="s1:sayHello" name="parameters"/>
</message>
<message name="sayHelloResponse">
     <part element="s1:sayHelloResponse" name="parameters"/>
</message>
<portType name="SecureHelloWorldPortType" wsp:PolicyURIs="#Sign.xml
     #Auth.xml #Encrypt.xml">
     <operation name="sayHello" parameterOrder="parameters">
          <input message="s1:sayHello"/>
          <output message="s1:sayHelloResponse"/>
     </operation>
</portType>
<binding name="SecureHelloWorldServiceSoapBinding" type="s1:
     SecureHelloWorldPortType">
     <s2:binding style="document" transport="http://schemas.
          xmlsoap.org/ soap/http"/>
          <operation name="sayHello">
               <s2:operation soapAction="" style="document"/>
               <input>
                    <s2:body parts="parameters" use="literal"/>
```

```
                </input>
                <output>
                      <s2:body parts="parameters" use="literal"/>
                </output>
           </operation>
      </binding>
      <service name="SecureHelloWorldService">
           <port binding="s1:SecureHelloWorldServiceSoapBinding"
                name="SecureHelloWorldServicePort">
                <s2:address location="http://localhost:9111/
                     SecureHelloWorldService/SecureHelloWorld
                     Service"/>
           </port>
      </service>
 </definitions>
```