Oracle Web Service Manager 11g
Interop with WLS WS-Security (in WLS)
March, 2012

Step-by-Step Instruction Guide

Author: Prakash Yamuna
Senior Development Manager
Oracle Corporation
# Table of Contents

Use Case ........................................................................................................................................... 5  
Description ........................................................................................................................................ 5  
Objective ............................................................................................................................................ 5  
Scenario ............................................................................................................................................. 5  
Policies Used ...................................................................................................................................... 6  
Software Requirements ........................................................................................................................ 7  
Prerequisites ...................................................................................................................................... 7  
Verified Product Version .................................................................................................................... 7  
Potentially Applies to Product Version(s) .......................................................................................... 7  
Download Main Page .......................................................................................................................... 7  
Product URLs ..................................................................................................................................... 7  
Install Locations ............................................................................................................................... 7  
Topology ........................................................................................................................................... 8  
Install & Topology Verification ........................................................................................................... 8  
   Verify all Product Consoles are reachable ......................................................................................... 8  
Step by Step Instructions ................................................................................................................... 10  
Create HelloWorld WLS POJO JAX-WS ......................................................................................... 10  
Create Application “HelloWorldApp” ............................................................................................... 10  
Create Project “HelloWorld” .............................................................................................................. 11  
Create a POJO Java Class “HelloWorld” ......................................................................................... 12  
Add method “hello” to the Java Class “HelloWorld” ....................................................................... 15  
Create Web Service from HelloWorld POJO Class ........................................................................ 15  
Attach Weblogic WS-Security Policies to the HelloWorld Web Service ............................................. 21  
Deploy HelloWorld Web Service ...................................................................................................... 22
Create WAR .................................................................................................................. 22
Deploy HelloWorldApp-HelloWorld-context-root.war ............................................. 25
Validate HelloWorldApp-HelloWorld-context-root.war successfully deployed .......... 28
Validate Web Service .................................................................................................. 29
Create the HelloWorldClient Web App ..................................................................... 30
Create a HTTP Servlet “HelloWorldClientServlet” .................................................. 32
Create “HelloWorldService” Web Service Proxy ...................................................... 36
Invoke the HelloWorld Web Service from the HelloWorldClientServlet ............... 41
Attach OWSM Client Policy to the HelloWorldClientServlet .................................. 43

Setup KeyStore & WLS Security Configuration ......................................................... 45
Create Keystore using keytool .................................................................................... 45
  Validate content of keystore ..................................................................................... 46
Create truststore using keytool .................................................................................. 47
  Export the certificates from the default-keystore.jks .............................................. 47
  Import the certificates into default-truststore.jks ..................................................... 47
  Validate the contents of the default-truststore.jks .................................................. 48
Copy Keystore & Truststore ......................................................................................... 48
Change WLS Keystore Configuration to use the Custom Keystore and Truststore created above ................................................................. 49
  Change SSL to use the Custom Identity and Custom Trust ..................................... 54
Create Web Service Security Configuration .............................................................. 55
  Edit “wss-username-config” .................................................................................... 58
Use the “wss-username-config” in the HelloWorld Service ..................................... 78
Setup OWSM KeyStore & Credential Store ............................................................... 78
  Configure jps-config.xml to use default-keystore.jks using EM ......................... 78
  Validate the necessary credentials for Keystore access has been created using EM 82
  Create the credentials “mycreds” using EM .......................................................... 83
Deploy the HelloWorldClient Web App ................................................................. 84
Create Custom Deployment Settings ................................................................. 84
Create helloworldclient WAR ........................................................................... 88
Deploy the helloworldclient.war ................................................................. 89
Testing ............................................................................................................. 90
Positive testing ............................................................................................. 90
Negative testing .......................................................................................... 90
Delete Credential “mycreds” using EM .......................................................... 90
Use Case

Description

Objective
Provide a detailed step by step instruction for enabling interoperability between OWSM and WLS WS-Security for username token with message protection scenario.

Scenario
This How-To illustrates the following:

1. How to Build a HelloWorld WLS POJO JAX-WS.
2. How to secure the HelloWorld WLS POJO JAX-WS with WLS WS-Security using username token with message protection
3. Deploy the HelloWorld WLS POJO JAX-WS.
4. How to Build a HelloWorld Client Web App that will invoke the HelloWorld WLS POJO JAX-WS
5. How to secure the HelloWorld Client Web App with OWSM security
7. Setup the Keystore and Credential Store required for OWSM.
**Policies Used**

<table>
<thead>
<tr>
<th>Service/Client</th>
<th>Policy</th>
<th>Policy Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>JEE Client</td>
<td>oracle/wss11_username_token_with_message_protection_client_policy</td>
<td>Client</td>
</tr>
</tbody>
</table>

- Wssp1.2-2007-Wss1.1-UsernameToken-Plain-EncryptedKey-Basic128.xml
- Wssp1.2-2007-SignBody.xml
- Wssp1.2-2007-EncryptBody.xml
Software Requirements

Prerequisites

<table>
<thead>
<tr>
<th>#</th>
<th>Product</th>
<th>Download URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Install SOA Suite 11.1.1.5 with JDeveloper</td>
<td></td>
</tr>
</tbody>
</table>

Verified Product Version

<table>
<thead>
<tr>
<th>#</th>
<th>Product</th>
<th>Release Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WebLogic</td>
<td>10.3.5</td>
</tr>
<tr>
<td>2</td>
<td>SOA</td>
<td>11.1.1.5</td>
</tr>
<tr>
<td>3</td>
<td>JDeveloper</td>
<td>11.1.1.5</td>
</tr>
</tbody>
</table>

Potentially Applies to Product Version(s)

<table>
<thead>
<tr>
<th>#</th>
<th>Product</th>
<th>Release Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WebLogic</td>
<td>10.3.3, 10.3.4, 10.3.5, 10.3.6</td>
</tr>
<tr>
<td>2</td>
<td>SOA</td>
<td>11.1.1.4, 11.1.1.5, 11.1.1.6</td>
</tr>
</tbody>
</table>

Download Main Page


Product URLs

<table>
<thead>
<tr>
<th>Product</th>
<th>URL</th>
<th>Login/Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>EM Fusion Middle Control</td>
<td>http://admin_host:admin_port/em</td>
<td>User: weblogic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Password: welcome1</td>
</tr>
<tr>
<td>Weblogic Console</td>
<td>http://admin_host:admin_port/console</td>
<td>User: weblogic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Password: welcome1</td>
</tr>
</tbody>
</table>

Install Locations

This How-To does not provide installation instructions for the pre-requisite components. Please consult the appropriate Install guides.

The components in this How-To are installed at the following locations:

<table>
<thead>
<tr>
<th>Component</th>
<th>Install location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Weblogic 11g PS4</td>
<td>D:\Oracle11gR1\Middleware\wlserver_10.3</td>
</tr>
<tr>
<td>Oracle Web Services Manager (OWSM) 11g PS4</td>
<td>D:\Oracle11gR1\Middleware\oracle_common</td>
</tr>
<tr>
<td>Oracle SOA Suite (SOA) 11g PS4</td>
<td>D:\Oracle11gR1\Middleware\Oracle_SOA1</td>
</tr>
<tr>
<td>Oracle Enterprise Manager (EM) 11g PS4</td>
<td>D:\Oracle11gR1\Middleware\oracle_common</td>
</tr>
<tr>
<td>JDeveloper</td>
<td>D:\Oracle11gR1\Middleware\jdeveloper</td>
</tr>
<tr>
<td>JDK</td>
<td>D:\Oracle11gR1\Middleware\jdk160_24</td>
</tr>
</tbody>
</table>
**Topology**

This How-To uses a single domain. The domain includes a one weblogic server. The steps provided in this How-To can vary based on Topology.

- Domain Name: soa_domain
- Weblogic Server: AdminServer

**Install & Topology Verification**

Start the Admin Server

Navigate to: `D:\Oracle11gR1\Middleware\user_projects\domains\soa_domain\bin`

```
C:\WINDOWS\system32\cmd.exe - startWebLogic.cmd
```

Verify all Product Consoles are reachable

Go to the product console URL and provide username as weblogic and password as welcome1.

<table>
<thead>
<tr>
<th>Product</th>
<th>URL</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle WebLogic</td>
<td><a href="http://localhost:7001/console">http://localhost:7001/console</a></td>
<td>WebLogic Administration Console</td>
</tr>
<tr>
<td>Oracle Web Services Manager (OWSM)</td>
<td><a href="http://localhost:7001/wsm-pm">http://localhost:7001/wsm-pm</a></td>
<td>Indicates status of OWSM Policy Manager. Presence of this page indicates that the Policy Manager has started</td>
</tr>
<tr>
<td></td>
<td><a href="http://localhost:7001/wsm-pm/validator">http://localhost:7001/wsm-pm/validator</a></td>
<td>Show you all the out-of-the-box</td>
</tr>
<tr>
<td>Interop with WLS WS-Security in WLS using Oracle Web Services Manager 11g</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

policy. If you see that page, OWSM policy store is properly deployed and running

<table>
<thead>
<tr>
<th>Oracle SOA Suite (SOA)</th>
<th><a href="http://localhost:7001/soa-infra">http://localhost:7001/soa-infra</a></th>
<th>List all deployed SOA Composites. Also has a test console to test the SOA Composite</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Enterprise Manager (EM)</td>
<td><a href="http://localhost:7001/em">http://localhost:7001/em</a></td>
<td>Oracle Enterprise Manager</td>
</tr>
</tbody>
</table>

Oracle Enterprise Manager (EM)
Step by Step Instructions

Create HelloWorld WLS POJO JAX-WS

Create Application “HelloWorldApp”
Click on File->New to start creation of the HelloWorld App in JDeveloper.
Create Project “HelloWorld”
Create a POJO Java Class “HelloWorld”

Right Click on HelloWorld project and select “New”.
Select “Java Class” and click “OK” button as shown below.
Enter the name of the Java class as "HelloWorld" and click on "OK" button.
Add method “hello” to the Java Class “HelloWorld”

```java
public String hello(String str) {
    return "Hello " + str;
}
```

Create Web Service from HelloWorld POJO Class

Right Click on the HelloWorld class and click on “Create Web Service”
Select the deployment platform that matches your requirements.

**Platform**

- **J2EE 1.4 JAX-RPC, with support for 10.3 WLS**
  Support for deploying JAX-RPC Web Services to release 10.3 of WLS.

- **Java EE 1.5, with support for JAX-WS Annotations**
  Support for deploying to 10.3 WLS with Java annotations using JAX-WS annotation specification.

- **Java EE 1.5, with support for JAX-WS RI**
  Support for deploying to any container that supports the Sun JAX-WS Reference Implementation.

**Generation Options**

Select the Java class or Stateless Session EJB you want to publish, and the J2EE and annotations options for web service.

- **Component To Publish:**
  `Hello.HelloWorld`

- **Web Service Name:**
  `HelloWorldService`

- **Port Name:**
  `HelloWorldPort`

- **Add SFI**
Interop with WLS WS-Security in WLS using Oracle Web Services Manager 11g

**Message Format**

- Use the controls on this page to choose the binding option and the settings that control the structure of the SOAP messages transmitted to and from the web service.

  - SOAP 1.1 Binding
  - SOAP 1.2 Binding

  **SOAP Message Format:** Document/Wrapped

- Binary encoding style for this service is MTOM. Enable MTOM to write MTOM annotation for this service.
  - Enable MTOM

**Methods**

- Select at least one method for the web service to expose.

  **Available Methods:**

  - `hello(java.lang.String str) : java.lang.String`
Interop with WLS WS-Security in WLS using Oracle Web Services Manager 11g

**Create Java Web Service - Step 6 of 9**

**Additional Classes**

Use this step to specify any extra classes for which you wish to create XML schema definitions. The Find Subclasses button will search the project for subclasses of classes used as parameters or return types used by the service’s operations.

**Additional Classes:**

- Add...
- Remove
- Find Subclasses

**Create Java Web Service - Step 7 of 9**

**Configure Policies**

In this page you decide the security policy option to use for the web service. Based on the selection the appropriate policy configuration page will be shown.

- [ ] OWSM Policies
- [ ] WLS Policies
- [x] No Policies

Help

< Back

Next >
Create Java Web Service - Step 8 of 9

Provide Handler Details

Specify any handler classes you have which will deal with the web service message. The defined handlers may have associated initialization parameters, SOAP roles or SOAP headers.

Ports: HelloWorldService

Defined Handlers:

Finish

You have finished describing your Java web service.

When you click Finish, the wizard will create the web service.
Attach Weblogic WS-Security Policies to the HelloWorld Web Service

You can attach Weblogic WS-Security Policies using the @Policy and @Policies annotation.
The final class looks as follows:

```java
package hello;
import javax.jws.WebMethod;
import javax.jws.WebService;
import weblogic.jws.Policies;
import weblogic.jws.Policy;

@WebService
@Policy(uri="policy:Wssp1.2-2007-Wss1.1-UsernameToken-Plain-EncryptedKey-Basic128.xml")
public class HelloWorld {
  public HelloWorld() {
    super();
  }

  @WebMethod
  @Policies({
    @Policy(uri="policy:Wssp1.2-2007-SignBody.xml"),
    @Policy(uri="policy:Wssp1.2-2007-EncryptBody.xml"))
  public String hello(String str) {
    return "Hello " + str;
  }
}
```

**Deploy HelloWorld Web Service**

**Create WAR**

Right Click on the HelloWorld project. Select “Deploy” item from the menu.
Interop with WLS WS-Security in WLS using Oracle Web Services Manager 11g

Deployment Action

Select a deployment action from the list below.

- Deploy to Application Server
- Deploy to WAR

Creates a Web Archive (WAR) File for the platform defined in the Profile.

Summary

Deployment Summary:

- Output file: D:\Developer\my\work\HelloWorldApp\HelloWorld\deploy\HelloWorldApp\HelloWorld-context-root.war
- Include manifest: No
- Compressed: No
Deploy HelloWorldApp-Helloworld-context-root.war
Click on “Finish” to complete the deployment.
Validate HelloWorldApp-HelloWorld-context-root.war successfully deployed
Validate Web Service
Create the HelloWorldClient Web App

Click on File->New to start creating a new Application.

Select JSP and Servlets as the “Project Technologies” for the project and click on “Next”.

Oracle Corporation | Interop with WLS WS-Security | Version 1.0
Create a HTTP Servlet “HelloWorldClientServlet”

Right Click on the HelloWorldClient Project and click on “New” menu item.

Select “Servlets” and Select “HTTP Servlet” and click on “OK”.

![Screenshot of creating a HTTP Servlet](image-url)
Interop with WLS WS-Security in WLS using Oracle Web Services Manager 11g

Welcome

Welcome to the Create HTTP Servlet Wizard

This wizard will help you to create a new HTTP Servlet.

[Options for creating a new HTTP Servlet]

Skip this Page Next Time

Help

< Back Next > Finish Cancel
A web application does not yet exist in this project. Select the version to create.

**Web Application Version:**
- Servlet 2.3/JSP 1.2 (J2EE 1.3)
- Servlet 2.4/JSP 2.0 (J2EE 1.4)
- Servlet 2.5/JSP 2.1 (Java EE 1.5)

**Create HTTP Servlet - Step 1 of 3: Servlet Information**

**Enter servlet details**

- **Class:** HelloWorldClientServlet
- **Package:** helloworldclient
- **Generate Content Type:** HTML
- **Generate Header Comments**

**Implement Methods**
- doGet()
- doPost()
- service() (default)
- doPut()
- doDelete()
Interop with WLS WS-Security using Oracle Web Services Manager 11g

Create HTTP Servlet - Step 2 of 3: Mapping Information

Enter servlet mapping.
While this is not required to create a servlet, it is required to run a servlet.

- Specify a name and mapping for the servlet.

**Mapping Details**

<table>
<thead>
<tr>
<th>Name</th>
<th>HelloWorldClientServlet</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL Pattern</td>
<td>/helloworldclientservlet</td>
</tr>
</tbody>
</table>

Create HTTP Servlet - Step 3 of 3: Servlet Parameters

Enter servlet parameters:

<table>
<thead>
<tr>
<th>Name*</th>
<th>Type*</th>
<th>Variable*</th>
<th>Desc</th>
<th>Default</th>
</tr>
</thead>
</table>

New | Remove

Help | Next > | Finish | Cancel
Create "HelloWorldService" Web Service Proxy
Interop with WLS WS-Security in WLS using Oracle Web Services Manager 11g

Oracle Corporation | Interop with WLS WS-Security | Version 1.0
Welcome to the Create Web Service Proxy Wizard.

This wizard helps you to create Java classes that can be used to access web services.

Click Next to continue.
Interop with WLS WS-Security in WLS using Oracle Web Services Manager 11g

Create Web Service Proxy - Step 4 of 7

Port Endpoints

If you wish, you can change the endpoint URLs that the proxy will use to access the ports of the service. This is useful if you want your proxy to run against a service on Integrated WLS, or a different external server.

<table>
<thead>
<tr>
<th>Port Name</th>
<th>Endpoint URL</th>
</tr>
</thead>
</table>

Create Web Service Proxy - Step 5 of 7

Asynchronous Methods

You can choose to add methods to your proxy that allow clients to access the service in an asynchronous manner. This step allows you to choose the asynchronous methods to be added.

- Don't generate any asynchronous methods
- Generate asynchronous methods where specified by the JAX-WS binding
- Generate asynchronous methods for all operations
Invoke the HelloWorld Web Service from the HelloWorldClientServlet
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.jsp;
import javax.servlet.http.*;

import oracle.xml.soap.v100.SecurityPolicyFeature;

public class HelloWorldClientServlet extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=windows-1252";

    public void init(ServletConfig config) throws ServletException {
        super.init(config);
    }

    public void doGet(HttpServletRequest request,
                      HttpServletResponse response) throws ServletException,
                      IOException {
        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head><title>HelloWorldClientServlet</title></head>");
        out.println("<body>");
        out.println("<p>The servlet has received a GET. This is the reply.</p>");

        HelloWorldService helloWorldService = new HelloWorldService();
        HelloWorld helloWorld = null;
        try {
            helloWorld = helloWorldService.getHelloWorldPort();
            out.println("<p>Calling Web Service:</p>");
            out.println("HelloWorld: " + helloWorld.toString());
            out.println("<p>Calling Web Service done:</p>");
        } finally {
            // Close the connection.
        }
    }
}
Make the following modifications to the generated code in the HelloWorldClientServlet:

```java
HelloWorldService helloWorldService = new HelloWorldService();
HelloWorld helloWorld = null;
try {
    helloWorld = helloWorldService.getHelloWorldPort();
    out.println("<p>Calling Web Service:" +
              helloWorld.hello("HelloWorldClient app calling") +"</p>");
    out.println("</body></html>");
    out.close();
} finally {
    out.close();
    ((java.io.Closeable)helloWorld).close();
}
```

Attach OWSM Client Policy to the HelloWorldClientServlet
Make the following code changes to attach the OWSM Client Policy:

```
SecurityPolicyFeature[] securityFeature = new SecurityPolicyFeature[] {
    new SecurityPolicyFeature("policy:oracle/wss11_username_token_with_message_protection_client_policy")
};

try {
    helloWorld = helloWorldService.getHelloWorldPort(securityFeature);((BindingProvider)helloWorld).getRequestContext().put(SecurityConstants.ClientConstants.WSS_CSF_KEY, "mycreds");
}
```
Setup KeyStore & WLS Security Configuration

Create Keystore using keytool

We will create a keystore using keytool. The keystore will be called default-keystore.jks. We will use the alias orakey and oratest.

Note: All passwords will be welcome1 (keystore, truststore passwords and alias passwords).

```
D:\Oracle11gR1\Middleware\jdk160_24\bin>keytool -genkey -keyalg RSA -alias orakey -keystore default-keystore.jks -storepass welcome1 -validity 3600
What is your first and last name? [Unknown]: weblogic
What is the name of your organizational unit? [Unknown]: oracle
What is the name of your organization? [Unknown]: oracle
What is the name of your City or Locality? [Unknown]: us
What is the name of your State or Province? [Unknown]: us
What is the two-letter country code for this unit? [Unknown]: us
Is CN=weblogic, OU=oracle, O=oracle, L=us, ST=us, C=us correct? [no]: yes
Enter key password for <orakey>
<RETURN if same as keystore password>:
Re-enter new password:
```
Validate content of keystore

```
D:\Oracle11gR1\Middleware\jdk160_24\bin>keystore -list -keystore default-keystore.jks -storepass welcome1

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 2 entries:

oracle, Dec 23, 2011, PrivateKeyEntry,
oratess, Dec 23, 2011, PrivateKeyEntry,
```
Create truststore using keytool

Export the certificates from the default-keystore.jks

```
C:\Oracle\Middleware\jdk160_24\bin>keytool -export -alias orakey -file orakey.cer -keystore default-keystore.jks -storepass welcome
Certificate stored in file <orakey.cer>

C:\Oracle\Middleware\jdk160_24\bin>keytool -export -alias oratest -file oratest.cer -keystore default-keystore.jks -storepass welcome
Certificate stored in file <oratest.cer>
```

Import the certificates into default-truststore.jks
Validate the contents of the default-truststore.jks

Copy Keystore & Truststore

Copy the keystore & truststore to the $MW_HOME/user_projects/domains(soa_domain)/config/fmwconfig
Change WLS Keystore Configuration to use the Custom Keystore and Truststore created above

- Login into WLS Console.
- Click on “Servers” under Environment on the Left Hand Navigation
- Click on “AdminServer”
Click on the “Keystore” tab. Click on “Change” button to change the keystore and truststore.
Select “Custom Identity and Custom Trust” option and click on “Save”.

---
Enter the following values in the Custom Identity and Custom Trust page and click on “Save” button:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom Identity</td>
<td>D:\Oracle11gR1\Middleware\user_projects\domains\soa_domain\config\fmwconfig\default-keystore.jks</td>
</tr>
<tr>
<td>Custom Identity</td>
<td>Jks</td>
</tr>
<tr>
<td>Custom Identity</td>
<td>welcome1</td>
</tr>
<tr>
<td>Custom Identity</td>
<td>welcome1</td>
</tr>
<tr>
<td>Custom Trust Keystore:</td>
<td>D:\Oracle11gR1\Middleware\user_projects\domains\soa_domain\config\fmwconfig\default-truststore.jks</td>
</tr>
<tr>
<td>Custom Trust Keystore:</td>
<td>Jks</td>
</tr>
<tr>
<td>Custom Trust Keystore:</td>
<td>welcome1</td>
</tr>
<tr>
<td>Confirm Custom Trust</td>
<td>welcome1</td>
</tr>
</tbody>
</table>

Oracle Corporation | Interop with WLS WS-Security | Version 1.0
Interop with WLS WS-Security in WLS using Oracle Web Services Manager 11g

**Key Stores:**
- Custom Identity and Custom Trust: Change
  - Which configuration rules should be used for finding the server's identity and trust keystores? More Info...

**Identity**
- Custom Identity Keystore: mglsdefaultkeystore.jks
  - The path and file name of the identity keystore. More Info...
- Custom Identity Keystore Type: jks
  - The type of the keystore. Generally, this is JKS. More Info...
- Custom Identity Keystore Passphrase: **************
  - The encrypted custom identity keystore's passphrase. If empty or null, then the keystore will be opened without a passphrase. More Info...
- Confirm Custom Identity Keystore Passphrase: **************

**Trust**
- Custom Trust Keystore: D:\Oracle11gR1\Middleware
  - The path and file name of the custom trust keystore. More Info...
- Custom Trust Keystore Type: jks
  - The type of the keystore. Generally, this is JKS. More Info...
- Custom Trust Keystore Passphrase: **************
  - The custom trust keystore's passphrase. If empty or null, then the keystore will be opened without a passphrase. More Info...
- Confirm Custom Trust Keystore Passphrase: **************

**Security**
- Save
Change SSL to use the Custom Identity and Custom Trust

Enter the following values in the SSL configuration page.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private Key Alias:</td>
<td>orakey</td>
</tr>
<tr>
<td>Private Key Passphrase:</td>
<td>welcome1</td>
</tr>
<tr>
<td>Confirm Private Key Passphrase:</td>
<td>welcome1</td>
</tr>
</tbody>
</table>
Create Web Service Security Configuration

Click on “soa_domain” under the Domain Structure on the Left Hand Tree.
Click on “Web Service Security” tab on the Right hand side.
Click on the “New” button to create new web service security configuration.

Provide the following value “wss-username-config” and then click on “OK” button.
Edit "wss-username-config"
Click on the “Credential Provider” tab to create Credential providers.

**Create Credential Provider**

Enter the following values:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name:</td>
<td>Username</td>
</tr>
<tr>
<td>Class Name:</td>
<td>weblogic.wsee.security.bst.ServerBSTCredentialProvider</td>
</tr>
<tr>
<td>Token Type:</td>
<td>X509</td>
</tr>
</tbody>
</table>

Click on “Finish” button.
Interop with WLS WS-Security in WLS using Oracle Web Services Manager 11g

Create a Web Service Security Credential Provider - soa_domain - WLS Console - Mozilla Firefox

Oracle WebLogic Server Administration Console

Domain Structure
soa_domain
- Environment
  - Servers
  - Clusters
  - Virtual Hosts
  - Migratable Targets
  - Coherence Servers
  - Coherence Clusters
  - Machines
  - Work Managers
  - Startup and Shutdown Classes
- Deployments
- Services
- Security Realm

How do I...
- Create a Web service security configuration
- Specify the key pair used to sign SOAP messages
- Specify the key pair used to encrypt SOAP messages
- Use a password digest in SOAP messages
- Use X.509 certificates to establish identity

Create a Web Service Security Credential Provider

Create Credential Provider

This assistant helps you create a new credential provider for an existing Web service security configuration.

* Indicates required fields

What would you like to name your credential provider?
* Name: username

Specify the class name that implements this credential provider.
* Class Name: BSTCredentialProvider

Specify the token type of the credential provider, such as x509.
* Token Type: x509
Create Credential Provider Properties

Create ConfidentialityKeyStore Credential Provider Property
Create ConfidentialityKeyStorePassword Credential Provider Property
Create ConfidentialityKeyAlias Credential Provider Property
Create ConfidentialityKeyPassword Credential Provider Property
Create IntegrityKeyStore Credential Provider Property
Create a Credential Provider property - soa_domain - WLS Console - Mozilla Firefox

Welcome, weblogic Connected to: soa_domain

Create a Credential Provider property

Select Encryption

Use this page to select the encryption for the property that is going to be created.

- Is Encrypted

Back | Next | Finish | Cancel
Create IntegrityKeyStorePassword Credential Provider Property
Interop with WLS WS-Security in WLS using Oracle Web Services Manager 11g

Create a Credential Provider property - sea_domain - WLS Console - Mozilla Firefox

Change Center

View changes and restarts
Configuration editing is enabled. Future changes will automatically be activated as you modify, add or delete items in this domain.

Domain Structure
sea_domain

<table>
<thead>
<tr>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users</td>
</tr>
<tr>
<td>Machines</td>
</tr>
<tr>
<td>Virtual Hosts</td>
</tr>
<tr>
<td>Work Managers</td>
</tr>
<tr>
<td>Start up and Shutdown Classes</td>
</tr>
</tbody>
</table>

Find: eWsc

Create a Credential Provider property

Select Encryption

Use this page to select the encryption for the property that is going to be created.

- Is Encrypted

Next

Create a Credential Provider property

Credential Provider Properties

Use this page to create a property associated with an existing credential provider of a Web service security configuration.

Name: grtKeyStorePassword

Value: *********

Confirm Value: *********

Oracle Corporation | Interop with WLS WS-Security | Version 1.0
Create IntegrityKeyAlias Credential Provider Property
Create IntegrityKeyPassword Credential Provider Property
Interop with WLS WS-Security in WLS using Oracle Web Services Manager 11g

Configuration editing is enabled. Future changes will automatically be activated as you modify, add or delete items in this domain.

Create a Credential Provider property

Name: IntegrityKeyPassword
Value: ********
Confirm Value: ********
Interop with WLS WS-Security in WLS using Oracle Web Services Manager 11g

Settings for username - soa_domain - WLS Console - Mozilla Firefox

Configuration

Name: username
Class: Name: weblogic.wsse.security
Token Type: x509

Customize this table

Credential/Provider Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Is Encrypted</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConfidentialKeyAlias</td>
<td>orakey</td>
<td>false</td>
</tr>
<tr>
<td>ConfidentialKeyPassword</td>
<td>*****************</td>
<td>true</td>
</tr>
<tr>
<td>ConfidentialKeyStore</td>
<td>D:\Oracle11g\Middleware\user_projects\domain\soa_domain\config\conf\default\keystore.ks</td>
<td>true</td>
</tr>
<tr>
<td>ConfidentialKeyStorePassword</td>
<td>*****************</td>
<td>true</td>
</tr>
<tr>
<td>IntegrityKeyAlias</td>
<td>orakey</td>
<td>false</td>
</tr>
<tr>
<td>IntegrityKeyPassword</td>
<td>*****************</td>
<td>true</td>
</tr>
<tr>
<td>IntegrityKeyStore</td>
<td>D:\Oracle11g\Middleware\user_projects\domain\soa_domain\config\conf\default\keystore.ks</td>
<td>true</td>
</tr>
<tr>
<td>IntegrityKeyStorePassword</td>
<td>*****************</td>
<td>true</td>
</tr>
</tbody>
</table>
Use the “wss-username-config” in the HelloWorld Service

```java
package hello;

import java.io.InputStream;
import java.io.OutputStream;
import java.security.KeyStore;
import java.security.cert.Certificate;

import weblogic.jws.Policies;
import weblogic.jws.Policy;
import weblogic.jws.security.UseConfiguration;

@WebService
@Policy(url="policy:wsapi.2-2007-Web1.1-UsernameToken-Fixed-EncryptedKey-Symmetric.xml")
@UseConfiguration(value="wss-username-config")

public class HelloWorld {
    public HelloWorld() {
        super();
    }

    @WebMethod
    @Policy(url="policy:wsapi.2-2007-SignBody.xml")
    @Policy(url="policy:wsapi.2-2007-EncryptBody.xml")
    public String hello(String str) {
        return "Hello " + str;
    }
}
```

Redeploy the app as described in section 3.

Setup OWSM KeyStore & Credential Store

For the purposes of this How-To we will use the same default-keystore.jks created in section 0

Configure jps-config.xml to use default-keystore.jks using EM
Interop with WLS WS-Security in WLS using Oracle Web Services Manager 11g

Security Provider Configuration: /farm_osd_domain/osd_domain (Oracle WebLogic Domain) - Oracle Enterprise Manager (weblogic) - Mozilla Firefox

Oracle Enterprise Manager 11g Fusion Middleware Control

Farms

soa_domain

WebLogic Domain

Web Services Manager Authentication Providers

You can configure the login modules and keystores for Web Services Manager authentication.

Login Modules

The following table lists all configured login modules for Web Services Manager. Use this list to create, configure or delete a login module.

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>ControlFlag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jwt.module</td>
<td>oracle.security.jwt.internal.java.module.jwt.module</td>
<td>Required</td>
<td>JWT Login Module</td>
</tr>
<tr>
<td>saml2.module</td>
<td>oracle.security.jsp.internal.java.module.saml2.SAMLLoginModule</td>
<td>Required</td>
<td>SAML Login Module</td>
</tr>
<tr>
<td>x509.module</td>
<td>oracle.security.jsp.internal.java.module.x509.x509Module</td>
<td>Required</td>
<td>X.509 Certificate Module</td>
</tr>
<tr>
<td>digest.module</td>
<td>oracle.security.jsp.internal.java.module.digest.DigestLoginModule</td>
<td>Required</td>
<td>Digest Login Module</td>
</tr>
<tr>
<td>sso.module</td>
<td>oracle.security.jsp.internal.java.module.sso.SSOLoginModule</td>
<td>Required</td>
<td>SSO Login Module</td>
</tr>
<tr>
<td>user.module</td>
<td>oracle.security.jsp.internal.java.module.user.UserLoginModule</td>
<td>Required</td>
<td>User Authentication Module</td>
</tr>
<tr>
<td>user.asser module</td>
<td>oracle.security.jsp.internal.java.module.userAssertion.UserAssertionLoginModule</td>
<td>Required</td>
<td>User Assertion Login Module</td>
</tr>
</tbody>
</table>

Keystores

Use this section to specify the keystores to store public and private keys for all secure connections within the WebLogic Domain.

Type: JKS
Path: Default-keystore.jks

Single Sign-On Provider

Advanced Properties
Interop with WLS WS-Security in WLS using Oracle Web Services Manager 11g

Keystore Configuration: /Farm soo_domain/soa_domain (Oracle WebLogic Domain) - Oracle Enterprise Manager (weblogic) - Mozilla Firefox

Keystore Configuration

Security Provider Configuration > Configure Key Store

Information

All changes made in this page require a server restart to take effect.

Keystore Configuration

A keystore is a key database that contains both public and private keys. Keystore needs to be configured only at the WebLogic Domain level. You will need to provide the keystore name, path, password and information about default identity certificates.

Keystore Type: Java Key Store (JKS)

Access Attributes

- Key Alias: oracle
- Password: **********
- Confirm Password: **********

Identity Certificates

Specify the default identity certificates (signature and encryption keys) for this keystore. Web Services that are configured to use this keystore will use these identity certificates.

Signature Key

- Key Alias: oracle
- Signature Password: **********
- Confirm Password: **********

Encryption Key

- Key Alias: oracle
- Crypt Password: **********
- Confirm Password: **********
Validate the necessary credentials for Keystore access has been created using EM
Create the credentials “mycreds” using EM
Deploy the HelloWorldClient Web App

*Create Custom Deployment Settings*
Select “Use Custom Settings” and Click on “Customize Settings” button.
Click on “New” button.
Interop with WLS WS-Security in WLS using Oracle Web Services Manager 11g

Click OK to create your new deployment profile and immediately open it to see its configuration.

Archive Type:
- WAR File

Name:
helloworldclient

Description:
Creates a profile for deploying the Java EE web module (WAR) to an application server. The WAR consists of the web components (JSPs and servlets) and the corresponding deployment descriptors.

Edit WAR Deployment Profile Properties

General

WAR File:
/mywork/HelloWorldClient/HelloWorldClient/deploy/helloworldclient.war

Deployment Plan:

Web Application's Context Root:
- Use Project's Java EE Web Context Root
- Specify Java EE Web Context Root:

Deployment Client Maximum Heap Size (in Megabytes): Auto

Oracle Corporation | Interop with WLS WS-Security | Version 1.0
Create helloworldclient WAR
Deploy the helloworldclient.war

Deploy the helloworldclient.war via Weblogic Console. The steps are similar to those described in section 0.
Testing

Positive testing

Negative testing
Just to make sure things are working as we would expect. Delete the “mycreds” from the Credential Store using EM.

Delete Credential “mycreds” using EM
Interop with WLS WS-Security in WLS using Oracle Web Services Manager 11g

Credentials: jFarm.soa_domain/soa_domain (Oracle WebLogic Domain) - Oracle Enterprise Manager (weblogic) - Mozilla Firefox

Oracle Enterprise Manager 11g Fusion Middleware Control

Logged in as weblogic

Page Refreshed Dec 23, 2011 8:36:29 PM PST

Information
The credential key, mcreds, has been created.

Credentials
A credential store is the repository of security data that certify the authority of entities used by Java 2, J2EE, and ADF applications. Applications can use the Credential Store, a single, consolidated service provider to store and manage their credentials securely.

Credential Store Provider

<table>
<thead>
<tr>
<th>Credential</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BPM-CRYPTO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>default</td>
<td></td>
<td></td>
</tr>
<tr>
<td>oracle.wm.security</td>
<td></td>
<td></td>
</tr>
<tr>
<td>step-0f-key</td>
<td>Password</td>
<td></td>
</tr>
<tr>
<td>basic-credentials</td>
<td>Password</td>
<td></td>
</tr>
<tr>
<td>user-step-key</td>
<td>Password</td>
<td></td>
</tr>
<tr>
<td>mcreds</td>
<td>Password</td>
<td></td>
</tr>
<tr>
<td>keystore-step-key</td>
<td>Password</td>
<td></td>
</tr>
</tbody>
</table>
Restart the server.
Error 500--Internal Server Error

javax.xml.ws.WebServiceException: javax.xml.rpc.JAXRPCException: oracle.xml.soap.MTOMException: WSS-00015 : The user name is missing.
  at com.sun.xml.ws.epi.pipe.Filter.__doRun(Filter.java:166)
  at com.sun.xml.ws.epi.pipe.Filter._doRun(Filter.java:81)
  at com.sun.xml.ws.epi.pipe.Filter.doRun(Filter.java:178)
  at com.sun.xml.ws.epi.pipe.Filter.doRun(Filter.java:192)
  at com.sun.xml.ws.client.Sub.process(Sub.java:272)
  at com.sun.xml.ws.client.sel.SELMethodHandler.invoke(SELMethodHandler.java:115)
  at com.sun.xml.ws.client.sel.SELMethodHandler.invoke(SELMethodHandler.java:96)
  at com.sun.xml.ws.client.sel.ETFSub.invoke(ETFSub.java:116)
  at FProxy.$H ello(unknown source)
  at sun.reflect.NativeMethodAccessorImpl.invoke(Native Method)
  at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:19)
  at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:26)
  at java.lang.reflect.Method.invoke(Method.java:505)
  at weblogic.xml.ws.epi.ClientInvocationInterceptor.invoke(ClientInvocationInterceptor.java:64)
  at FProxy.$H ello(unknown source)
  at helloWorldClient.HelloWorldClientServlet.doGet(HelloWorldClientServlet.java:42)
  at javax.servlet.http.HttpServlet.service(HttpServlet.java:797)
  at javax.servlet.http.HttpServlet.service(HttpServlet.java:820)
  at weblogic.servlet.internal.ServletSecurityHelper$ServletServiceAction.run(ServletSecurityHelper.java:227)
  at weblogic.servlet.internal.ServletSecurityHelper.invokeServlet(ServletSecurityHelper.java:125)
  at weblogic.servlet.internal.ServletActionImpl.execute(ServletActionImpl.java:100)
  at weblogic.servlet.internal.TailFilter.doFilter(TailFilter.java:26)
  at weblogic.servlet.internal.FilterChainImpl.doFilter(FilterChainImpl.java:106)
  at oracle.security.jsps.security.JpsFilter.run(JpsFilter.java:115)
  at java.security.AccessController.doPrivileged(AccessController.java:308)
  at oracle.security.jsps.util.HttpSessionSubject.doPrivileged(HttpSessionSubject.java:125)
  at oracle.security.jsps.security.JpsFilter.run(JpsFilter.java:415)
  at oracle.security.jsps.security.JpsFilter.run(JpsFilter.java:141)
  at oracle.security.jsps.security.JpsFilter.doFilter(JpsFilter.java:72)

Oracle Corporation | Interop with WLS WS-Security | Version 1.0