INTRODUCTION
This white paper describes how to secure access to JSR 168 standards-based WSRP v1 portlets from OracleAS Portal 10.1.4.1.

OVERVIEW
The Web Services for Remote Portlets (WSRP) specification indicates that Web Services Security (WS-Security) may be leveraged for providing secure identity propagation between the Portal consumer and the portlet producer. Without using such a mechanism, WSRP in and of itself, does not provide secure identity propagation of the end user’s identity to the portlet producer. The WSRP specification explicitly defers to other security standards for secure identity propagation. However, it does not go into the specific WS-Security profiles or options that should be employed.

Identity Propagation Without WS-Security
When using WSRP without WS-Security, the userContext element within the SOAP message contains user profile information and user category information. This information is not considered secure and should be used only for personalization and customization functionality. It should not be used for authorization to access sensitive resources. This information is also exposed in the JSR 168 APIs — isUserInRole(role) and getUserPrincipal():

```java
private void doViewHtml(RenderRequest request, 
RenderResponse response) 
throws PortletException, 
IOException {
    // To do: markup the required content.
    PrintWriter out = response.getWriter();
    out.println("<p class='portlet-font'>Welcome</p>");
    out.println("<p class='portlet-font'>MODERATOR</p>");
    if (request.isUserInRole("moderator")) {
    out.println("<p class='portlet-font'>MODERATOR</p>" );
    } else {
    out.println("<p class='portlet-font'>not moderator</p>" );
    }
    if (request.isUserInRole("participant")) {
    out.println("<p class='portlet-font'>PARTICIPANT</p>" );
    } else {
    out.println("<p class='portlet-font'>not participant</p>" );
    }
    if (request.isUserInRole("viewer")) {
    out.println("<p class='portlet-font'>VIEWER</p>" );
    } else {
    out.println("<p class='portlet-font'>not viewer</p>" );
    }
```
The code snippet above shows how a sample portlet’s markup rendering code may use the `isUserInRole` API to decide what content to display.

**Identity Propagation with WS-Security**

When WS-Security is leveraged with WSRP, the user’s identity is propagated outside of the SOAP message body, in the WS-Security header. This is a user assertion, using the UsernameToken or SAMLToken format, and is digitally signed to authenticate the consumer and to ensure the integrity of the assertion.

Oracle Portal 10.1.4.1 supports the use of the WS-Security UsernameToken profile token, without a password, as well as the SAMLToken with the sender-vouches semantic. It digitally signs the security token and message body to ensure authenticity and integrity.

Digitally signing the security token and the SOAP message body accomplishes a couple of objectives:

- **Consumer Authentication**
- **Identity Assertion and Message Integrity**

**Consumer Authentication**

When a portlet producer is generating sensitive information, such as paystub information, for example, it is imperative that it responds only to requests to show the information from a legitimate consumer.

By using WS-Security and having the consumer digitally sign the security token and the message body, the producer can verify the signature using the public key of the legitimate consumer. If the signature cannot be verified, it means that the request may have come from a fraudulent impostor. By requiring the verification of the digital signature, the sensitive information will only be sent to the legitimate consumer.

**Identity Assertion and Message Integrity**

In addition to verifying the identity of the consumer making the Web service requests, digitally signing the security token and the message body also ensures that the token and the message have not been tampered with. This prevents such problems as man-in-the-middle attacks where a legitimate request might be intercepted and the username in the security token replaced with another username in order to see the paystub information coming back for the other user. If the token is digitally signed, it cannot be tampered with. Any modification to the token would result in the inability to verify the signature on the producer end, and would result in returning a SOAP fault instead of the requested paystub information.

**Supported Producers**

Portal 10.1.4.1 with the patches described in this white paper supports several WS-Security token formats that allow for interoperability with several portlet producer containers that provide WS-Security support.
At a minimum, the following producers are interoperable with Portal 10.1.4.1 WS-Security:

- PeopleSoft 8.48
- WebCenter Suite 10.1.3.2 WSRP container
- WebCenter Suite 10.1.3.3 WSRP container

Other WSRP vendors may also be able to support interoperability with Portal 10.1.4.1.

**Security Domain Implication**

Note that when secure identity propagation is used as described here, the username of the user authenticated to the consumer (Portal) is simply propagated to the producer without any remapping or providing any credentials. There is an inherent assumption that the producer understands this username and can locate this user in its associated security domain. Consequently, it is highly desirable to ensure that the consumer and producer share the same security provider (identity store) to simplify the management of this configuration.

In an Oracle environment, this is easy enough to do by ensuring that the portlet producer uses the Oracle Identity Management Security Provider, and references the same Oracle Internet Directory (OID) server that is being used by the Portal 10.1.4 installation. For information on how to do this, see Configuring a 10.1.3.x Producer to Use OID on page 19.

Alternatively, out-of-band mechanisms may also be employed to simply ensure that usernames defined in the consumer’s identity store also exist in the producer’s identity store. For example, the consumer Portal would be using OID, whereas the producer may use an XML file-based identity store, with the same user population as in OID. In this case, ensure that the usernames in the XML store are in uppercase to match the Portal usernames.

**STARTING WITH AN ORACLEAS PORTAL 10.1.4.1 INSTALL**

The latest available patchset available for the OracleAS Portal Standards Release is the 10.1.4.1 Metadata Repository patchset, which goes with the 10.1.2.2 middle tier software update. You should start with this version of software to use WS-Security with WSRP producers.

To get to this version of software, do the following:

1. Start with a 10.1.2.0.2 OracleAS 10g Infrastructure and Metadata Repository, which can be installed from the 4 CD OracleAS 10.1.2.0.2 Installation set.
2. From the same set of 10.1.2.0.2 CDs, install an OracleAS Portal and Wireless middle tier, and associate it with the previously installed Infrastructure 10g. Note that the Wireless component is optional, and not required for WS-Security with WSRP.
3. Install the 10.1.2.2 patchset Software Update for the OracleAS 10g middle tier. This patchset is available on Metalink, by searching for patches with the following parameters:
   - Product or Family: Portal
Release: iAS 10.1.2.2
Patch Type: Patchset/Minipack
Select your platform, and click Go.
Just install the Software Update for the middle tier.

4. Upgrade the Portal Metadata Repository from 10.1.2.0.2 to 10.1.4.0.0 using the Portal Metadata Repository Upgrade Assistant to 10.1.4.0.0

5. Patch the Oracle RDBMS installed in the 10.1.2.0.2 Infrastructure with the Oracle RDBMS 10.1.0.5 patchset.

6. Patch the 10.1.4.0.0 Portal Metadata Repository to 10.1.4.1 by running the 10.1.2.2 Oracle Application Server 10g Portal Metadata Repository Update 10.1.2.2.0.

### PATCHING THE ORACLEAS 10.1.4.1 INSTANCE

After setting up an OracleAS Portal 10.1.4.1 Metadata Repository with a 10.1.2.2 Oracle Application Server middle tier, some additional patches need to be applied to fix some WSRP issues.

The following patches should be applied in the order shown. These patches may be obtained from [http://metalink.oracle.com](http://metalink.oracle.com).

<table>
<thead>
<tr>
<th>Metalink Patch Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6524204</td>
<td>PSE FOR BASE BUG 6524198 ON TOP OF 10.1.4.1 - VERSION2(REPOSITORY)</td>
</tr>
<tr>
<td></td>
<td>Portal WSRP Repository Bundle patch</td>
</tr>
<tr>
<td>6003718</td>
<td>MERGE LABEL REQUEST ON TOP OF 10.1.2.2 FOR BUGS 5527265 5690987 5750136 4723364</td>
</tr>
<tr>
<td></td>
<td>Java Object Cache middle tier patch.</td>
</tr>
<tr>
<td>7168573</td>
<td>PATCH REQUEST FOR MERGE BUG 7168572 ON TOP OF 10.1.2.2 - VERSION 4(MIDTIER)</td>
</tr>
<tr>
<td></td>
<td>Portal Middle Tier Patch</td>
</tr>
</tbody>
</table>

In addition, it would be wise to review *Metalink Note 460362.1 - Recommended Patches for Portal 10.1.4.x* to see if any additional patches may be required for 10.1.4.x in general. Another Metalink Note that provides information on patch status for OracleAS 10g is *Metalink Note 329361.1 - Oracle Application Server 10g Release 2 (10.1.2) Support Status and Alerts*.

The functionality documented in this white paper has been tested with the patches listed above.

### REGISTER A WSRP PRODUCER

The WSRP producers to be consumed on OracleAS Portal need to be deployed to a WSRP container. You can download Oracle WebCenter Suite 10g (10.1.3.x) for this purpose.
After installing this J2EE container, its $ORACLE_HOME/install/readme.txt file will have some URLs listed. This will include a URL for the Rich Text Portlet WSRP producer, appearing something like this:

Use the following URL to consume the Rich Text Portlet in your application:
http://host:port/richtextportlet/portlets/wsrp2?WSDL

To register this producer in OracleAS Portal 10.1.4.x, replace the wrsp2 with wrsp1, as follows:

http://host:port/richtextportlet/portlets/wsrp1?WSDL

You can register the producer in OracleAS Portal and then consume it on some page to start off testing the WSRP functionality.

**CONFIGURING FOR WS-SECURITY**

**Setting up the Keystores**

For more information on setting up keystores, Oracle wallets and certificates, see the Oracle Application Server Web Services Security Guide, 10g Release 3 (10.1.3).

**Create a Wallet for the Consumer**

The consumer’s wallet will contain the private key used to sign the WS-Security assertions.

The consumer is the Portal application. This application consumes portlets generated by the remote portlet producer over WSRP.

1. Generate a private key and a V3 certificate using `openssl`.

   ```
   $ openssl req -newkey rsa:1024 -keyout portal.key -x509 -nodes -out portal.cer
   Generating a 1024 bit RSA private key
   ...............+
   ...........................+++++
   writing new private key to 'portal.key'
   -----<snip>
   You are about to be asked to enter information that will be incorporated into your certificate request.
   What you are about to enter is what is called a Distinguished Name or a DN.
   There are quite a few fields but you can leave some blank
   For some fields there will be a default value,
   If you enter '.', the field will be left blank.
   -----<snip>
   Country Name (2 letter code) [GB]:US
   State or Province Name (full name) [Berkshire]:.
   Locality Name (eg, city) [Newbury]:
   Organization Name (eg, company) [My Company Ltd]:oracle
   Organizational Unit Name (eg, section) []:
   Common Name (eg, your name or your server's hostname) []:myportal
   Email Address []:
   ```

   This will result in a portal.cer and portal.key file.
2. Convert this into a pkcs12 file. You will need to provide a password with which to secure the generated pkcs12 file.

```bash
$ openssl pkcs12 -in portal.cer -inkey portal.key -export -out portal.p12 -nodes
Enter Export Password:
Verifying - Enter Export Password:

$ ls
portal.cer  portal.key  portal.p12
```

3. Generate a keystore using keytool (the private key generated here will be ignored).

```bash
$ keytool -genkey -dname "cn=myportal,dc=us,dc=oracle,dc=com" -keystore portal.jks
Enter keystore password:  welcome1
Enter key password for <mykey> (RETURN if same as keystore password):

4. Use pkcs12import to import the V3 certificate into the keystore (note now the keystore has two key entries: mykey is the default key and will be ignored; portalcert is the V3 certificate).

   The pkcs12import utility can be obtained from the Sun Java Web Services Developer Pack. It is available here:

   http://java.sun.com/webservices/downloads/previous/webservicespack.jsp

```bash
$ pkcs12import.sh -file portal.p12 -keystore portal.jks -alias portalcert
pkcs12-password: welcome1
keystore-password: welcome1
key-password: welcome1

$ ls
portal.cer  portal.jks  portal.key  portal.p12
```

At this point you can double-check the contents of portal.jks:

```bash
$ keytool -list -keystore portal.jks
Enter keystore password:  welcome1
Keystore type: jks
Keystore provider: SUN
Your keystore contains 2 entries
portalcert, Feb 15, 2008, keyEntry,
Certificate fingerprint (MD5):
mykey, Feb 15, 2008, keyEntry,
Certificate fingerprint (MD5):
```

The producer’s wallet will contain the public key of the consumer, with which the signed WS-Security assertions can be verified.

Create a Wallet for the Producer

The final step of the keystore setup process is generating the keystore for the producer to use. The producer will need the public key of the consumer to verify
the authenticity of the security tokens received from the consumer in the WS-Security headers of the requests it receives over its getMarkup interface.

To do so, it needs a keystore that contains the certificate of the consumer and the root certificate used to sign it. These are added to the keystore as trusted certificates.

1. Generate a keystore for the producer (again, the private key generated here will not be used).

```
$ keytool -genkey -dname "cn=producer,dc=us,dc=oracle,dc=com" -keystore myproducer.jks
Enter keystore password: welcome1
Enter key password for <mykey>
    (RETURN if same as keystore password):
```

2. Import the V3 certificate into the producer's keystore.

```
$ keytool -import -trustcacerts -file portal.cer -alias portalcert -keystore myproducer.jks
Enter keystore password: welcome1
Owner: CN=myportal, O=oracle, C=US
Issuer: CN=myportal, O=oracle, C=US
Serial number: 0
Certificate fingerprints:
Trust this certificate? [no]: yes
Certificate was added to keystore
```

At this point, you can double check the contents of myproducer.jks:

```
$ keytool -list -keystore myproducer.jks
Enter keystore password: welcome1
Keystore type: jks
Keystore provider: SUN
Your keystore contains 2 entries
portalcert, Feb 15, 2008, trustedCertEntry,
Certificate fingerprint (MD5):
mykey, Feb 15, 2008, keyEntry,
Certificate fingerprint (MD5):
```

Note that portalcert is added as a trustedCertEntry. The reason we have to do this is because this certificate is self-signed, therefore its chain of certificate authority does not terminate with a widely recognized CA. By adding it to the keystore as a trustedCertEntry and specifying this keystore for the producer, we can make sure the producer will accept this certificate.

**Configure the Producer**

The required configuration on the producer end will vary depending on the WSRP container that you are consuming through your portal. With Portal 10.1.4.1, you
can consume portlets from PeopleSoft 8.48 or from the Oracle WSRP Container running on the Oracle WebCenter Suite 10.1.3.x.

For the PeopleSoft configuration steps, refer to the PeopleSoft documentation for PeopleSoft 8.48 WS-Security support. Note that for the PeopleSoft 8.48 integration with Oracle Portal 10.1.4.1, the token type that needs to be configured is **UsernameToken** with no password, with digital signature, with the security token referenced with the X500 Serial Number. This is represented as the **UsernameTokenX500** option.

In this example, we’ll use the Oracle WebCenter Suite 10.1.3.2 WSRP Container. This is the Oracle WSRP container deployed onto OracleAS 10g Release 3 (10.1.3.2). This leverages the Web Services Security stack integration with Oracle Container for J2EE (OC4J). With this container, we can use the UsernameToken or SAMLToken, using the BST or SKI token reference mechanisms.

You can deploy your own WSRP producer and configure it for WS-Security, or to simply test the security feature, just enable WS-Security on an already deployed WSRP producer such as the RichText portlet, which is deployed on the 10.1.3.2 WSRP container on OC4J/WebCenter. If you want to use the RichText portlet or any other WSRP producer that is already deployed, you can skip ahead to “Enable Security for the Web Service” on page 13.

**Deploy the Producer**

Follow the steps in the *Oracle Portal Developer’s Guide* to deploy your standards-compliant portlet producer to the Oracle WSRP Container on OC4J. For WS-Security support you must deploy it to a WebCenter 10.1.3.2 WSRP Container.

After deploying the producer, you should use the Application Server Control application to configure the producer for WS-Security.

The `$ORACLE_HOME/install/readme.txt` file in the Oracle home of your 10.1.3.2 installation will include the URL of the AS Control for your installation.

It may look like this:

```
Use the following URL to access Application Server Control:
http://server.domain.com:7777/em
User Name: oc4jadmin
```

The password to use is the one you specified during installation.

Once in AS Control, navigate to the OC4J container and to the specific application for the producer that you deployed. If you click the Web Services tab and do not see any Web services listed, it is simply because the Web service has not been requested yet and as a result, it is not yet visible in AS Control.
To get the Web service interfaces to show up in AS Control:

1. Click the Home tab, and then click the Web module within the producer application.

2. Click the Test Web Module link to the right of the screen. You will see the base URL of the producer, as a standard Web module. To this base URL, you need to append “/info” to access the WSRPProducer Test Page.

This page has links to the WSDL URLs that you can use to register the producer with the Portal. Click the link for the WSRP v1 WSDL to obtain the Web Services Description Language (WSDL) of the producer.
The browser will show the WSDL as depicted here.

In this screen, find a line that shows the location of the WSRP Base Service. In the example above, the location indicates the following:

http://stadi65.us.oracle.com:7777/secondSampleV1/portlets/WSRPBaseService

Copy this location into a browser window to request that service. You can ignore the screen that displays, which will look something like this:

The reason for requesting that page is to get the WSRPBaseService to show up in the list of Web Services. This service, which includes the getMarkup interface is the only one that is to be configured for WS-Security.
Enable Security for the Web Service

To enable security for the Web service, log in to AS Control on the 10.1.3.2 instance and navigate to the Web Service: WSRPBaseService page for the producer of interest. For example, follow these steps for the RichText producer:

1. Log in to AS Control (Oracle Enterprise Manager 10g)
   http://host:port/em
2. Click the link for OC4J_WebCenter.
3. Click the link for Applications.
4. Click richtextportlet.
5. Click the Web Services tab.
6. Click the WSRPBaseService link.
7. Click the Administration tab.

At this point you should see a screen similar to this one:

8. Click the Enable/Disable Features button.
9. Using the shuttle control, select the Security feature and move it into the Enabled Features list. Click OK.

You should see a checkmark that indicates the Security feature is now enabled.

Configure the Keystore

1. Click the Keystore and Identity Certificates button under Port Level Configuration.

You can use an instance-level keystore or an application-specific keystore. The following instructions describe the use of an application-specific keystore.

2. Specify an arbitrary Keystore name, just to refer to this keystore.

3. Specify the path to the keystore that you created for the producer, relative to the location of the deployed Web application. For example, in our example, the Web application is deployed to $ORACLE_HOME/j2ee/OC4J_WebCenter/applications/richtextportlet. So, the entry in Keystore Path should be simply myproducer.jks, if you copy the keystore to that directory.

4. Click OK.

Configure the Inbound Policy

Back on the Edit Security Configuration screen, click the Inbound Policies button.

Authentication

1. Select the User Username /Password Authentication checkbox.
2. Click OK.

**Integrity**

1. Click the Integrity tab to set the integrity protection settings.

2. Select the “Require Message Body to Be Signed” checkbox. This will ensure that the message is tamper-proofed. Under Acceptable Signature Methods, select all signature methods as acceptable.

3. Click OK when done.

**Final Fixups for Producer Side**

In this version of AS Control, the option to configure the Web service for UsernameToken without a password is not shown on the user interface. However, it is still possible to configure this, in the `wsmgmt.xml` file.
1. Locate this file under $OC4J/config/wsmgmt.xml – where $OC4J is the location of the OC4J container under the Oracle home of the 10.1.3.2 install; that is, in our example, $ORACLE_HOME/j2ee/OC4J_WebCenter/config/wsmgmt.xml.

2. Within this file, you need to update the following verify-username-token element within the port element containing the WSRPBaseService of the producer we are configuring:

   ```xml
   <verify-username-token password-type="PLAINTEXT" require-nonce="false" require-created="false"/>
   ```

   Update it to:

   ```xml
   <verify-username-token>
     <property name="username.token.allow.nopassword" value="true"/>
   </verify-username-token>
   ```

3. Save the wsmgmt.xml file.

4. Restart OC4J_WebCenter, either through EM or on the command line.

   Note that if using the SAMLToken, no such workaround steps are required.

### Test the Producer’s Security

The portlet should fail to display because now it requires security, and the consumer hasn’t been configured to send the necessary WS-Security headers yet.

![Error Message](image)

Until the consumer side is set up to issue the appropriate WS-Security headers, it is not be able to consume this newly secured producer.

### Configure the Consumer

The consumer of the WSRP portlets is the Oracle Portal 10.1.4.1, which has been patched with the required patches for supporting WS-Security.

#### The wsrpwss.sql Script

In this release of Portal, the WS-Security support has been introduced in a patchset, so there is no UI support for updating a producer connection to use WS-Security. Rather, you need to execute a SQL script to apply the configuration updates.

The script is located in the following location:

$MIDTIER_ORACLE_HOME/portal/admin/plsql/wwc/wsrpwss.sql

There is some documentation for this script in comments at the beginning of the file.

The script accomplishes the following:

1. Updates the preference store with the Provider WS Config, Certificate alias and password to sign the WSRP SOAP request. This script is used to
store the WS-Security Config, Certificate alias and password to retrieve the private key from the keystore.

2. Updates the preference store with the Keystore type, location (absolute path), file, and password details. These details are required to retrieve the certificate and private key from the keystore. This script is used to store the keystore details such as type, file (with its absolute path), and password.

This script should be run in the Portal schema only after storing the certificate in the keystore.

Usage 1: Updating a Producer to Use WS-Security

This script should be executed from the Portal schema.

@param 1 mode (add | update | delete)
@param 2 Provider Name
@param 3 WS Config (e.g., 'UsernameTokenBST')
@param 4 Certificate alias
@param 5 Certificate Password

Example:

SQL> @wspwss.sql add RICHTEXT UsernameTokenBST portalcert welcomel

Note: The valid values for WS Config are:

- UsernameToken — same as UsernameTokenX500, see below.
- UsernameTokenX500 — UsernameToken without password, security token and message body digitally signed; signing certificate referenced via X509 Issuer Serial Number.
- UsernameTokenBST — UsernameToken without password, security token and message body digitally signed; signing certificate referenced via Binary Security Token.
- UsernameTokenSKI — UsernameToken without password, security token and message body digitally signed; signing certificate referenced via Subject Key Identifier.
- SAMLToken — same as SAMLTokenX500, see below.
- SAMLTokenX500 — SAMLToken with sender vouches, security token and message body digitally signed for integrity and authentication; signing certificate referenced via X509 Issuer Serial Number.
- SAMLTokenBST — SAMLToken with sender vouches, security token and message body digitally signed for integrity and authentication; signing certificate referenced via Binary Security Token.
- SAMLTokenSKI — SAMLToken with sender vouches, security token and message body digitally signed for integrity and authentication; signing certificate referenced via Subject Key Identifier.

The configuration you select should be compatible with the producer capabilities:
• For communicating with PeopleSoft 8.48 portlets, the UsernameTokenX500 should be used.

• For communicating with an Oracle Application Server 10.1.3.2 OC4J container-based WSRP producer, use any of the BST or SKI (direct reference) configurations.

Usage 2: De-registering a Producer from Use of WS-Security

This script should be executed from the Portal schema.

@param 1 mode delete
@param 2 Provider Name

Example:

SQL> @wsrpwss.sql delete provider_name

Usage 3: Specifying Keystore Location

This script should be executed from the Portal schema.

@param 1 mode (keystore)
@param 2 Keystore Type (jks)
@param 3 Keystore Location
(/home3/as1012p2mid/jdk/jre/lib/security)
@param 4 Keystore File (cacerts)
@param 5 Keystore Password (****)

Example:

SQL> @wsrpwss.sql keystore jks /home/as10122/keystores portal.jks welcome1

Register the Keystore Information with Portal

Log in to the Portal schema and invoke the script to register the keystore details:

$ sqlplus portal@orcl

SQL*Plus: Release 10.1.0.4.2 - Production on Mon Oct 16 14:56:59 2006
Copyright (c) 1982, 2005, Oracle. All rights reserved.
Enter password:

Connected to:
Oracle Database 10g Enterprise Edition Release 10.1.0.4.2 - Production
With the Partitioning, OLAP and Data Mining options

SQL>
SQL> @wsrpwss.sql keystore jks /scratch/pencarna/keystores portal.jks welcome1
Create path name and related entries for the Keystore
INFO: The Keystore type has been stored successfully.
INFO: The Keystore path has been stored successfully.
INFO: The Keystore password has been stored successfully.
No errors.
SQL>

Update the Producer Registrations to Use WS-Security

First, determine the name of the producer you are going to update to use WS-Security. The name is displayed in the Edit Provider Registration screen and in the list of values (LOV) used to select which provider to edit, from the Administration | Portlets -> Remote Providers portlet.
For our example, the name of the provider is RICHTEXT.

Log in to the Portal schema and invoke the script to set the provider to use WS-Security:

```bash
$ sqlplus portal@orcl
SQL*Plus: Release 10.1.0.5 - Production on Mon Oct 16 14:56:59 2006
Copyright (c) 1982, 2005, Oracle. All rights reserved.
Enter password:
Connected to:
Oracle Database 10g Enterprise Edition Release 10.1.0.5 - Production
With the Partitioning, OLAP and Data Mining options
SQL>
SQL> @wsrpwss.sql add RICHTEXT UsernameTokenBST portalcert welcome1
```

After this configuration, subsequent markup requests will be invoked with WS-Security headers included to propagate the user’s identity to the provider. After you change this configuration, or initially set it, you need to clear web cache for the pages on which to producer is consumed. This is because the Page Metadata (PMD) is cached in webcache, and the WS-Security configuration information is included in the PMD. You can clear webcache by editing the page properties, going to the Access Settings, and clearing webcache for that page; or go to the Administration | Global Settings | Caching tab and clear all of Web Cache.

**CONFIGURING A 10.1.3.X PRODUCER TO USE OID**

In order to simplify identity management for user propagation, the WSRP container should be associated with the same identity store that is being used by the portal, if possible.

The OC4J container can be associated with the Oracle Identity Management provider that Portal is using. To set this up, follow the instructions under “Associate Oracle Internet Directory with OC4J” in Chapter 8 of the Oracle® Containers for J2EE Security Guide 10g.
In order to get identity propagation to work correctly for an OC4J configured for OID, the container’s `jazn.xml` file should contain the `mapping.attribute` property:

```xml
<property name="mapping.attribute" value="uid"/>
```

To configure this, after configuring the Oracle Identity Management security provider, the `jazn.xml` in the `$ORACLE_HOME/j2ee/home/config` directory will have the proper configuration in `jazn.xml`. Copy this file to the `$OC4J/config` directory of the container you are hosting the portlets on, and add the following property within the `jazn` element:

```xml
<property name="mapping.attribute" value="uid"/>
```

The value should be the attribute name used for logging in. For a default install of the IDM and Portal, this is the `uid` attribute.

**DEBUGGING**

When working with WS-Security, it is important to be able to see what is going on under the covers to understand how to resolve issues when things aren’t going quite right.

There are several tools to help in this regard.

- **TCP Monitor (e.g., tcpmon)** for inspection of traffic over the “wire”
  
  http://www.apache.org/dyn/closer.cgi/ws/axis/1_4/

- **Producer Diagnostic Logging:** J2EE logging for producer side diagnostic logging with the oracle.portlet logger.

- **Consumer Diagnostic Logging:** J2EE logging for consumer side diagnostic logging, and Portal Metadata Repository logs
TCP Monitor

A utility such as the Apache tcpmon tunnel monitor, available in Apache’s axis.jar, provides an easy way to monitor the SOAP messages and headers that travel between the consumer and the producer.

Set up the tunnel

1. Download the axis-bin-1_4.zip (or other appropriate download format) from http://www.apache.org/dyn/closer.cgi/ws/axis/1_4/.
2. Extract the axis.jar file from this download file.
3. Execute the tcp mon by issuing the following command (assuming the Java Runtime Environment is properly setup on your machine):

   java -classpath axis.jar org.apache.axis.utils.tcpmon

This will bring up the following dialog:

![TCP Monitor dialog](image)

This tool can be run on any machine accessible from your Portal middle tier.

4. In the Listen Port # field, specify an available port on the machine on which you are running this. In this example, we selected 8081. Also specify the hostname and the same port under the Listener parameters.
5. Now, select the Proxy radio button, as we actually want to run this monitor in proxy mode so that it is easier to remove it from the configuration once we are certain that WS-Security is working properly.

6. Once the parameters are set up properly for the proxy, click the Add button, which starts the process and adds the Port 8081 tab to the dialog. Select the tab to bring it to the foreground.
7. Finally, to make it easier to read the output, click the XML Format checkbox and click the Switch Layout button to show the outbound and inbound messages side-by-side.
Define the proxy in the portal

1. On the portal, log in as an administrator and navigate to Administer | Portal | Global Settings | Proxy:

   ![Global Settings screenshot]

2. Specify the hostname and port on which you’ve set up the TCP Monitor, and click the Add button to add this as a proxy. Then click OK once you see that the proxy has been added to the list of available proxies. That’s all that’s needed on this screen.

Configure the Producer to Use the Proxy

1. Finally, navigate to the producer registration information for the producer you are configuring for WS-Security: Administer | Portlets |
2. Select the proxy you've created from the Portal Middle Tier drop-down list box. This will cause all the middle tier messages from the PPE to be run through the TCP Monitor proxy server so we can see the traffic.

The screen above shows a WSRP SOAP message being transmitted from the Portal, including a `wsse:Security` header, and the corresponding response.

When setting this up, you may see errors in the response, which may help guide you to the source of the problem. Usually, the producer-side setup may not be correct, and the message in the response may help you pinpoint the problem.
**Producer Logging**

If the SOAP message responses are not sufficient to help you isolate the problems, you may need to enable diagnostic logging on the producer side.

This can be done by editing the `%OC4J/config/j2ee-logging.xml` file and adding the loggers for the WSRP runtime container, or the Web Services Security stack. For these, the following loggers are helpful:

- `oracle.j2ee.security`
- `oracle.portal`

Adding the following snippet in `j2ee-logging.xml` and restarting the container will enable the logging of these components.

```xml
<logger name='oracle.j2ee.security' level='FINEST'>
    <handler name='oc4j-handler'/>
</logger>

<logger name='oracle.portal' level='FINEST'>
    <handler name='oc4j-handler'/>
</logger>
```

The resultant logs will be found in `$OC4J/log/$OC4J_default_group_1/oc4j/log.xml`

You will need to restart the container for any `j2ee-logging.xml` changes to take effect.

**Consumer Logging**

On the consumer side, the following log may contain relevant errors:

```
$ORACLE_HOME/j2ee/OC4J_Portal/application-deployments/portal/OC4J_Portal_default_island_1/application.log
```

Logging of additional debug detail may be enabled by configuring the logmode in `web.xml`:

```
cd $ORACLE_HOME/j2ee/OC4J_Portal/applications/portal/portal/WEB-INF
```

Edit `web.xml` and replace `perf` with `debug` for the following elements:

```
<servlet-name>PortalServlet</servlet-name>
<servlet-class>oracle.webdb.portal.servlet.PortalServlet</servlet-class>
<init-param>
    <param-name>logmode</param-name>
    <param-value>debug</param-value>
</init-param>
...
<servlet-name>page</servlet-name>
<servlet-class>oracle.webdb.page.ParallelServlet</servlet-class>
<init-param>
    <param-name>logmode</param-name>
    <param-value>debug</param-value>
</init-param>
```

The same can also be done for the `RepositoryServlet`. 
You can also enable Portal Metadata Repository diagnostic logging. This is described in the *Oracle Application Server Portal Configuration Guide*.

**SUMMARY**

The Web Services for Remote Portlets (WSRP) specification indicates that Web Services Security (WS-Security) may be leveraged for providing secure identity propagation between the portal consumer and the portlet producer. However, it does not discuss the specific WS-Security profiles or options that should be employed. This white paper describes in detail how to secure access to JSR 168 standards-based WSRP v1 portlets from OracleAS Portal 10.1.4.1.