Oracle® Communications Billing and Revenue Management

Web Services Manager

Release 7.5

E16724-15

November 2018

ORACLE®

Oracle Communications Billing and Revenue Management Web Services Manager, Release 7.5

E16724-15

Contents

Preface

This document contains guidelines for installing and setting up Oracle Communications Billing and Revenue Management (BRM) Web Services Manager. The sample procedures use Oracle WebLogic Server, but you can apply the concepts to supported application server.

Before reading this document, you should be familiar with implementing Web services using WebLogic Server. See your WebLogic Server documentation for more information.

Audience

This document is intended for systems integrators, system administrators, database administrators, and other individuals who are responsible for installing, configuring, and customizing Web services for BRM.

Downloading Oracle Communications Documentation

Product documentation is located on Oracle Help Center:

`http://docs.oracle.com`

Additional Oracle Communications documentation is available from the Oracle software delivery Web site:

`https://edelivery.oracle.com`

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Document Revision History

The following table lists the revision history for this book.

Version	Date	Description
E16724-01	November 2011	Initial release.
E16724-02	May 2012	Documentation updates for BRM 7.5 Patch Set 1. ■ Minor formatting and text changes.
E16724-03	December 2012	Documentation updates for BRM 7.5 Patch Set 3. ■ Added documentation about enabling Web Services Manager to support custom opcodes: Setting Up Web Services Manager to Support Custom Opcodes
E16724-04	March 2013	Documentation updates for BRM 7.5 Patch Set 4. ■ Added documentation about configuring Java logging in Oracle WebLogic Server: Configuring Java Logging for the Application Server
E16724-05	August 2013	On HP-UX IA64, BRM 7.5 is certified as of BRM 7.5 Patch Set 5. Documentation added for HP-UX IA64.
E16724-06	February 2014	Documentation updates for BRM 7.5 Patch Set 7. ■ Minor formatting and text changes.
E16724-07	May 2014	Documentation updates for BRM 7.5 Patch Set 8. ■ Added documentation about configuring security, security policy, and role-based authorization for Web Services Manager: Configuring Security for Web Services Manager Configuring Role-Based Authentication for Apache Axis in WebLogic Server ■ Added documentation about Web services that take payload as an XML element data type: About WSDL Files and BRM Opcodes ■ Added documentation about customizing Web services: Creating a Custom Web Service ■ Added documentation about generating schema files for opcodes: Generating Schema Files for Your System Generating Schema for Your Opcodes ■ Added a sample SOAP request and response message for a Web service: Example of Creating an Account in BRM Using Web Services

Version	Date	Description
E16724-08	August 2014	Documentation updates for BRM 7.5 Patch Set 9. ■ Added documentation about installing and configuring GlassFish Server and the Apache Tomcat server: Installing and Configuring Oracle GlassFish Server Installing and Configuring Apache Tomcat Server ■ Added documentation about deploying Web Services Manager on GlassFish Server and the Apache Tomcat server: Deploying and Launching Web Services Manager on GlassFish Server Deploying and Launching Web Services Manager on Tomcat Server ■ Added documentation about configuring security for Web Services Manager in GlassFish Server and the Apache Tomcat server: Configuring Security for Web Services Manager in GlassFish Server Configuring Security for Web Services Manager in Tomcat Server ■ Added documentation about data masking in Web services responses: About Masked Fields in Web Services Responses
E16724-09	October 2014	Documentation updates for BRM 7.5 Patch Set 10. ■ Updated the following sections: About WSDL Files and BRM Opcodes Deploying and Launching Web Services Manager on WebLogic Server Table 6–2, " Web Services Included in Web Services Manager that Support XML Element Payload"
E16724-10	August 2015	Documentation updates for BRM 7.5 Patch Set 12. ■ Minor formatting and text changes.
E16724-11	December 2015	Documentation updates for BRM 7.5 Patch Set 14. ■ Updated the entire document.
E16724-12	August 2016	Documentation updates for BRM 7.5 Patch Set 16. ■ Added the following sections: Finding WSDL V2 Files Creating a Custom Web Service that Supports XML String Payload ■ Updated the following sections: Deploying and Launching Web Services Manager on Tomcat Server
E16724-13	December 2016	Documentation updates for BRM 7.5 Patch Set 17. ■ Added the following chapter about securing Web Services Manager with OAuth2: Chapter 4, "Securing Web Services Manager with OAuth2"

Version	Date	Description
E16724-14	August 2017	Documentation updates for BRM 7.5 Patch Set 19. ■ Updated the following section: Creating a Custom Web Service that Supports XML Element Payload
E16724-15	November 2018	Documentation updates for BRM 7.5 Patch Set 22. ■ Added the following section: Starting and Stopping GlassFish Server ■ Updated the following section: Deploying and Launching Web Services Manager on GlassFish Server

1

Installing Web Services Manager

This chapter describes how to install Oracle Communications Billing and Revenue Management (BRM) Web Services Manager.

BRM Web Services Manager allows BRM opcodes to be exposed as Web Services.

Software Requirements

Before you install and configure Web Services Manager, you must install the following:

- A supported, standards-compliant server. See "Supported Servers" for a list of the servers supported by Web Services Manager. See server documentation for more information.
- BRM. See *BRM Installation Guide* for more information.
- The Third-Party software package, which includes the Perl libraries and JRE required for installing BRM components. See “Installing the Third-Party Software” in *BRM Installation Guide*.

Supported Operating Systems

Web Services Manager is available on for the following operating systems:

- HP-UX IA64
- Oracle Solaris
- Oracle Linux
- AIX

Supported Servers

Web Services Manager is supported on the following servers:

- Oracle WebLogic Server
- Oracle GlassFish Server
- Apache Tomcat server

Installing Web Services Manager

Note: If you have already installed Web Services Manager, features that are already installed cannot be reinstalled without uninstalling them first. To reinstall a feature, uninstall it and then install it again.

The Web Services Manager package includes Web Services Manager, Synchronization Queue Data Manager, and JCA Resource Adapter. You must download the Web Services Manager package and extract these components before you can install Web Services Manager. For information about Synchronization Queue Data Manager and JCA Resource Adapter, see:

- Understanding the Synchronization Queue Data Manager in *BRM Synchronization Queue Manager*.
- Connecting J2EE-Compliant Applications to BRM in *BRM JCA Resource Adapter*.

You must also increase the heap size used by the Java Virtual Machine (JVM) before running the installation program to avoid "Out of Memory" error messages in the log file. For information, see "Increasing Heap Size to Avoid "Out of Memory" Error Messages" in *BRM Installation Guide*

Note: Oracle recommends that you install Web Services Manager on the system on which BRM is installed.

To download Web Services Manager:

1. Go to the Oracle software delivery Web site:

   `https://edelivery.oracle.com`

2. Download the **7.5_WebServicesMgr_***platform***_opt.tar.Z** software pack to a temporary directory (*temp_dir*), where *platform* is the operating system name.
3. In *temp_dir*, extract the **7.5_WebServicesMgr_***platform***_opt.tar.Z** software pack:

   The following files are extracted:

   - **7.5.0_BRM_JCA_Adapter_***platform***_opt.bin**: JCA Resource Adapter
   - **7.5.0_DM_AQ_***platform***_opt.bin**: Synchronization Queue Data Manager
   - **7.5.0_WebServicesMgr_***platform***_opt.bin**: Web Services Manager

4. Go to the directory in which you installed the Third-Party package and source the **source.me** file.

   Caution: You must source the **source.me** file to proceed with installation; otherwise, "suitable JVM not found" and other error messages appear.

   Bash shell:

   ```
   source source.me.sh
   ```

   C shell:

   ```
   source source.me.csh
   ```

5. Go to the *temp_dir* and enter the following command:

```
7.5.0_WebServicesMgr_platform_32_opt.bin
```

Note: You can use the **-console** parameter to run the installation in command-line mode. To enable a graphical user interface (GUI) installation, install a GUI application such as X Windows and set the DISPLAY environment variable before you install the software.

6. Follow the instructions displayed during installation. The default installation directory for Web Services Manager is *BRM_Home***/deploy/web_services**. If BRM or Web Services Manager is not already installed on the machine, the installation program will prompt you for an installation directory: specify a directory in which to install the package.

Your Web Services Manager installation is now complete.

Uninstalling Web Services Manager

To uninstall Web Services Manager:

1. Go to the *BRM_Home***/uninstaller/WebServicesMgr** directory.
2. Enter the following command:

```
uninstaller.bin
```

2 Deploying Web Services Manager

This chapter describes how to deploy and launch the Oracle Communications Billing and Revenue Management (BRM) Web Services Manager.

Before reading this chapter, you should have read "Installing Web Services Manager" and you should be familiar with the administration console of your chosen application server.

About Deploying Web Services Manager

Web Services Manager includes a servlet-based implementation that hosts the deployed web service. Web Services Manager uses the JAX-WS API to support SOAP, WSDL, and other low-level web service protocols.

Web Services Manager can be deployed on the following servers:

- Oracle WebLogic Server. See "Deploying and Launching Web Services Manager on WebLogic Server."
- Oracle GlassFish Server. See "Deploying and Launching Web Services Manager on GlassFish Server."
- Apache Tomcat Server. See "Deploying and Launching Web Services Manager on Tomcat Server."

Deploying and Launching Web Services Manager on WebLogic Server

You can deploy Web Services Manager on WebLogic server through the WebLogic Server Administration Console. Depending on the type of payload supported by web services, deploy one of the following files:

- **infranetwebsvc.war**: Includes web services that support the payload as an XML string data type.
- **BrmWebServices.war**: Includes web services that support the payload as an XML element data type.

If you customize web services, regenerate **infranetwebsvc.war** or **BrmWebServices.war** and use the generated version. Otherwise, you should use the default **infranetwebsvc.war** or **BrmWebServices.war** file. For more information about customizing web services, see "Customizing Web Services."

To deploy Web Services Manager on WebLogic Server:

1. Create the WebLogic Server domain. See the discussion about creating a WebLogic domain in *Fusion Middleware Creating Domains Using the Configuration Wizard* for detailed instructions.

2. If you deploy the **BrmWebServices.war** file, set the heap size required to start WebLogic Server:

   a. Open the *WebLogic_Home*/**user_projects/domains/***Domain_Name*/**setDomainEnv.sh** file in a text editor.

      Where:

      WebLogic_Home is the directory in which WebLogic Server is installed.

      Domain_Name is the name of the domain you created in step 1.

   b. Add the following entry:

      ```
      USER_MEM_ARGS ="-Xms2048m -Xmx2048m"
      ```

   c. Save and close the file.

   d. Restart WebLogic Server.

3. Do one of the following:

   - If you customized web services:

     – Extract the *BRM_Home*/**deploy/web_services/infranetwebsvc.war** or the *BRM_Home*/**deploy/web_services/BrmWebServices.war** file to *local_dir*.

       Where:

       BRM_Home is the directory in which BRM is installed.

       local_dir is a directory on the machine on which you installed WebLogic Server.

     – Copy the **CustomFields.jar** files to the *local_dir*/**WEB-INF/lib** directory. See "Setting Up Web Services Manager to Support Custom Opcodes" for more information.

     Note: The JRE version that was used to generate **CustomFields.jar** must be the same or lower than the version of the WebLogic Server JRE.

     – Open the *BRM_Home*/**deploy/web_services/Infranet.properties** file in a text editor.

     – Modify the following entry:

       ```
       infranet.custom.field.package = package
       ```

       where *package* is the name of the package that contains the **CustomOp.java** file; for example, **com.portal.classFiles**.

     – Add all the custom fields to the **Infranet.properties** file.

     – Save and close the file.

     – Copy the *BRM_Home*/**deploy/web_services/Infranet.properties** file to the *local_dir*/**WEB-INF/classes** or in the home directory on the machine on which WebLogic Server is installed.

     – Regenerate the WAR file by running one of the following commands:

       To regenerate the **infranetwebsvc.war** file:

       ```
       jar -cvf infranetwebsvc.war *
       ```

    To regenerate the **BrmWebServices.war** file:

    ```
    jar -cvf BrmWebServices.war *
    ```

- If you did not customize web services:
  - Extract the *BRM_Home*/**deploy/web_services/infranetwebsvc.war** or the *BRM_Home*/**deploy/web_services/BrmWebServices.war** file to *local_dir*.
  - Copy the *BRM_Home*/**deploy/web_services/Infranet.properties** file to the *local_dir*/**WEB-INF/classes** directory or in the home directory on the machine on which WebLogic Server is installed.
  - Regenerate the WAR file by running one of the following commands:

    To regenerate the **infranetwebsvc.war** file:

    ```
    jar -cvf infranetwebsvc.war *
    ```

    To regenerate the **BrmWebServices.war** file:

    ```
    jar -cvf BrmWebServices.war *
    ```

4. Log in to WebLogic Server Administration Console.
5. In the Domain Structure pane, click **Deployments**.

   The Summary of Deployments page appears.
6. Click **Install**.

   The Install Application Assistant page appears.
7. Enter the local directory path for **infranetwebsvc.war** or **BrmWebServices.war** in the Path field and press **Enter**.

   Note: If you use WebLogic Server 12.2, make sure you put serverlet-to-URL within a comment.

8. Select **Install this deployment as an application** and click **Next**.
9. Select the server or servers on which you want to deploy Web Services Manager and click **Next**.
10. Click **Next**.

    Note: When you deploy Web Services Manager on WebLogic Server, select the **Custom Roles and Policies** option in the Security section of WebLogic Server Administration Console.

11. Verify your deployment options and click **Finish**.

    WebLogic Server displays the deployed application on the Deployments page.

To launch Web Services Manager for web services, do one of the following:

- If you have deployed **infranetwebsvc.war**, launch web services from the WebLogic Server Administration Console:
  1. On the Home page, select **Deployments** in the Domain Structure pane.

     The Summary of Deployments page appears.

2. Click the **infranetwebsvc** link.

   The Settings page for the deployment appears.

3. Click the **Testing** tab.
4. Click the default url link.

   The Apache Axis Web page appears.

5. Click the link for the list of the supported BRM web services for this Web Services Manager deployment.

- If you have deployed **BrmWebServcies.war**, launch web services from the WebLogic Server Administration Console:

  1. On the Home page, select **Deployments** in the Domain Structure pane.

     The Summary of Deployments page appears.

  2. Click the **BrmWebServices** link, a list of all the web services is displayed.
  3. Click on any of the web service.
  4. Click the **Testing** tab.
  5. The WebLogic Server Administration console displays an URL for the BRM web services.

Deploying and Launching Web Services Manager on GlassFish Server

You can deploy Web Services Manager on GlassFish Server through the GlassFish Server Administration Console. Depending on the type of payload supported by web services, deploy one of the following files:

- **infranetwebsvc.war**: Includes the web services that support the payload as an XML string data type.
- **BrmWebServices.war**: Includes the web services that support the payload as an XML element data type.

If you customize web services, regenerate **infranetwebsvc.war** or **BrmWebServices.war** and use the generated version. Otherwise, you should use the default **infranetwebsvc.war** or **BrmWebServices.war** file. For more information about customizing web services, see "Customizing Web Services."

To deploy Web Services Manager on GlassFish Server:

1. Create the GlassFish Server domain. See the GlassFish documentation for detailed instructions.

   If you plan to deploy the **BrmWebServices.war** file, do the following:

   a. Open the **BrmWebServices.war/WEB-INF/web.xml** file in a text editor.

   b. Uncomment the serverlet-to-URL mapping.

   c. Save and close the file.

2. Copy the *BRM_Home***/deploy/web_services/infranetwebsvc.war** file or the *BRM_Home***/deploy/web_services/BrmWebServices.war** to a *local_dir*.

   Where:

   BRM_Home is the directory in which BRM is installed.

   local_dir is a directory on the machine on which you installed GlassFish Server.

3. Log in to the GlassFish Server Administration Console.
4. In the Common Tasks pane, click **Applications**.

   The Applications page appears.
5. On the Applications page, click **Deploy...**

   The Deploy Applications or Modules page appears.
6. In the **Location: Packaged File to Be Uploaded to the Server** field, click **Choose File** and do one of the following:

   - To support the payload as an XML string data type, select **infranetwebsvc.war** from the list of available applications and click **OK** to deploy the **infranetwebsvc.war** file.
   - To support the payload as an XML payload data type, select **BrmWebServices.war** from the list of available applications and click **OK** to deploy the **BrmWebServices.war** file.

   GlassFish Server displays the deployed application on the Applications page.
7. To launch Web Services Manager for web services, do one of the following:

   If you have deployed **infranetwebsvc.war**, launch web services from the GlassFish Server Administration Console:

   a. On the Applications page, select the **infranetwebsvc** checkbox and click **Launch**.

      The GlassFish Server Administration console displays an HTTP and an HTTPS URL for the web services.

   b. Click on a URL for any available web service.

   If you have deployed **BrmWebServices.war**, launch web services from the GlassFish Server Administration Console:

   a. On the Applications page, select the **BrmWebServices** checkbox and click **Launch**.

      The GlassFish Server Administration console displays an HTTP and an HTTPS URL for the BRM web services.

      A sample URL for the BRMCUSTServices_v2Web service is as follows:

      ```
      http://ipaddress:port/BrmWebServices/BRMCUSTServices_v2?wsdl
      ```

      where:

      – *ipaddress* is the domain IP address of the application server on which Web Services Manger is deployed.

      – *port* is the domain port number of the application server on which Web Services Manger is deployed.
8. Restart GlassFish Server. See "Starting and Stopping GlassFish Server".

For more information on the BRM web services included in Web Services Manager that take the payload as an XML element data type, see the Table 6–2, " Web Services Included in Web Services Manager that Support XML Element Payload".

Starting and Stopping GlassFish Server

Note: You must restart GlassFish Server after you deploy or undeploy Web services on GlassFish Server.

To start GlassFish Server:

1. Go to the directory in which GlassFish Server is installed.
2. Run the following command, which starts GlassFish Server:

   ```
   start-domain GlassFish_domain
   ```

   where *GlassFish_domain* is the GlassFish Server domain; for example, domain1.

To stop GlassFish Server:

1. Go to the directory in which GlassFish Server is installed.
2. Run the following command, which stops GlassFish Server:

   ```
   stop-domain GlassFish_domain
   ```

Deploying and Launching Web Services Manager on Tomcat Server

You can deploy Web Services Manager on Tomcat Server through the Tomcat Web Application Manager. Depending on the type of payload supported by web services, deploy one of the following files:

- **infranetwebsvc.war**: Includes the web services that support the payload as an XML string data type. See "Deploying and Launching infranetwebsvc.war."
- **BrmWebServices.war**: Includes the web services that support the payload as an XML element data type. See "Deploying and Launching BrmWebServices.war."

If you customize web services, regenerate **infranetwebsvc.war** or **BrmWebServices.war** and use the generated version. Otherwise, you should use the default **infranetwebsvc.war** or **BrmWebServices.war** file. For more information about customizing web services, see "Customizing Web Services."

Deploying and Launching infranetwebsvc.war

To deploy Web Services Manager for web services that support the payload as an XML string data type, on Tomcat server:

1. Create the Tomcat server domain. See Tomcat documentation for detailed instructions.
2. Download JAX-WS Reference Implementation (RI) library from the JAX-WS RI page (`http://jax-ws.java.net/`).
3. Extract the **jaxws-ri-2.2.10.zip** file and copy the following files to *Tomcat_home*/**lib**, where *Tomcat_home* is the directory in which the Tomcat server is installed:
   - gmbal-api-only.jar
   - jaxb-api.jar
   - jaxb-impl.jar
   - jaxws-rt.jar
   - management-api.jar

- policy.jar
- stax-ex.jar
- jaxb-core.jar
- ha-api.jar
- jaxws-tools.jar
- jaxb-xjc.jar
- jaxb-impl.jar
- jaxb-jxc.jar
- streambuffer.jar

4. In the **War file to deploy** section, click **Browse**.
5. Download **jaxrpc.jar** from the Download jaxrpc.jar web page (`http://www.java2s.com/Code/Jar/j/Downloadjaxrpcjar.htm`) and copy the file to *Tomcat_home*/**lib**.
6. Click **Deploy**.

   Tomcat Web Application Manager displays the deployed application in the **Applications** list.

   Note: (UNIX only) The GUI user must have execute permissions for **infranetwebsvc.war**. If the GUI user does not have the appropriate permissions, Tomcat will deploy the application, but will be unable to start it. Verify that the Running column of the Applications list has a value of true for **/intranetwebsvc** to ensure that Web Services Manager has been deployed successfully.

To launch Web Services Manager for web services that support the payload as an XML string data type, from the Tomcat Web Application Manager:

1. In the Applications list, click the **/infranetwebsvc** link.
2. Click the **Services** link to view a list of the supported BRM web services for this Web Services Manager deployment.

Deploying and Launching BrmWebServices.war

To deploy Web Services Manager for web services that support the payload as an XML element data type, on Tomcat server:

1. Create the Tomcat server domain. See Tomcat documentation for detailed instructions.
2. Download JAX-WS RI library from the JAX-WS Reference Implementation page (`http://jax-ws.java.net/`).
3. Extract the **jaxws-ri-2.2.10.zip** file and copy the following files to *Tomcat_home*/**lib**, where *Tomcat_home* is the directory in which Tomcat server is installed:
   - gmbal-api-only.jar
   - jaxb-api.jar
   - jaxb-impl.jar
   - jaxws-rt.jar

- management-api.jar
- policy.jar
- stax-ex.jar
- jaxb-core.jar
- ha-api.jar
- jaxws-tools.jar
- jaxb-xjc.jar
- jaxb-impl.jar
- jaxb-jxc.jar
- streambuffer.jar

4. Download **jaxrpc.jar** from the Download jaxrpc.jar web page (`http://www.java2s.com/Code/Jar/j/Downloadjaxrpcjar.htm`) and copy the file to *Tomcat_home*/**lib**.

   If you plan to deploy the BrmWebServices.war file, do the following:

   a. Open the **BrmWebServices.war/WEB-INF/web.xml** file in a text editor.
   b. Uncomment the serverlet-to-URL mapping.
   c. Save and close the file.

5. Copy the *BRM_home*/**deploy/web_services/BrmWebServices.war** file to a local directory on the machine on which you installed Tomcat server, where *BRM_home* is the directory in which BRM is installed.
6. Log in to the Tomcat Web Application Manager.
7. In the **War file to deploy** section, click **Browse...**.
8. Select the **BrmWebServices.war** file.
9. Click **Deploy**.

   Tomcat Web Application Manager displays the deployed application in the **Applications** list.

To launch Web Services Manager for web services that support the payload as an XML element data type, from the Tomcat Web Application Manager:

1. In the Applications list, click the **/BrmWebServices** link.
2. Web Services Manager displays the WSDL URLs for each available service.

   A sample URL for the BRMCUSTServices_v2Web service is as follows:

   ```
   http://ipaddress:port/BrmWebServices/BRMCUSTServices_v2?wsdl
   ```

   where:

   – *ipaddress* is the domain IP address of the application server on which Web Services Manger is deployed.

   – *port* is the domain port number of the application server on which Web Services Manger is deployed.

For more information on the BRM web services included in Web Services Manager that take the payload as an XML element data type, see Table 6–2, " Web Services Included in Web Services Manager that Support XML Element Payload".

3 Configuring Web Services Manager

This chapter describes how to configure the deployed Oracle Communications Billing and Revenue Management (BRM) Web Services Manager application. Configuring Web Services Manager requires connecting the deployed application to the BRM system and configuring security, authorization, and Java logging for the deployed application.

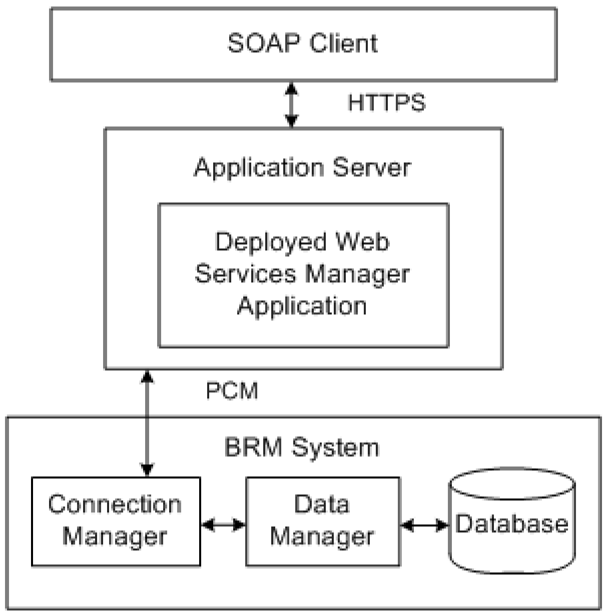
This chapter is intended for system administrators and system integrators.

Before reading this chapter, you should have installed BRM and Web Services Manager and deployed Web Services Manager on a supported application server. You should also be familiar with the administration console of your chosen applications server and with using BRM. For installation procedures, see "Installing Web Services Manager." For deployment procedures, see "Deploying Web Services Manager." For information about BRM, see *BRM Concepts*.

About Connecting Web Services Manager to the BRM System

Web Services Manager connects to the BRM system through a BRM Connection Manager (CM). Figure 3–1 shows how BRM and the SOAP client communicate with the deployed application. Web Services Manager translates Portal Communication Module (PCM) communications sent from a CM in the BRM system into SOAP requests sent to the SOAP client over HTTP. Web Services Manager translates SOAP responses sent from the SOAP client over HTTP into PCM communications that are returned to the CM.

Figure 3–1 Architecture of Web Services Manager in the BRM System



Connecting Web Services Manager to the BRM System

Before you connect Web Services Manager to the BRM system, verify that your instance of Web Services Manager is deployed to an application server.

If you customized web services, use the custom **infranetwebsvc.war** or **BrmWebServices.war** file. Otherwise, you should use the default **infranetwebsvc.war** or **BrmWebServices.war** file. For more information about customizing web services, see "Customizing Web Services."

To connect Web Services Manager to the BRM system, do the following:

1. On your application server, copy the *BRM_Home*/**deploy/web_services/Infranet.properties** file to one of the following:

   - *local_dir*/**WEB-INF/classes** directory, where *local_dir* is a directory on the machine on which you installed your application server.

     Note: If you copy the **Infranet.properties** file to the *local_dir*/**WEB-INF/classes** directory, extract the **infranetwebsvc.war** file or **BrmWebServices.war** file to a local directory (*local_dir*) on the system on which your application server is installed.

   - home directory on the machine on which you installed your application server.

2. Open the **Infranet.properties** file in a text editor.
3. Locate the following lines:

```
infranet.connection=pcp://root.0.0.0.1:password@ipAddress:port/0.0.0.1/service
/admin_client 1
infranet.login.type=1
```

4. Do the following:

   a. Replace *password* with the password for the BRM server.
   b. Replace *ipAddress* with the IP address of the system on which BRM is installed.
   c. Replace *port* with the port number used by the application server on which BRM is installed.

5. If SSL is enabled in connection manager (CM), locate the following lines and update the parameters if necessary:

   ```
   infranet.pcp.ssl.enabled=true
   infranet.pcp.ssl.wallet.location=wallet_directory
   ```

   where *wallet_directory* is the path to your client Oracle wallet. The client Oracle wallet contains the optional client SSL certificate and the private key, and it contains the Trusted CA certificate.

6. If you added custom opcodes or custom fields for Web services, add the enum values of the custom fields.

   For example, if you created the **custom_fld_usage_id** custom field and the enum value for the **custom_fld_usage_id** field is 10001, add the following entry:

   ```
   infranet.custom.field.10001=custom_fld_usage_id
   ```

   For information about mapping enum values, see "Creating Custom Fields" in *BRM Developer's Guide*.

7. (Optional) To configure the connection pool parameters, modify the following entries:

   ```
   infranet.connectionpool.minsize=min_connections
   infranet.connectionpool.maxsize=max_connections
   infranet.connectionpool.timeout=connection_timeout
   ```

   where:

   min_connections is the minimum number of connections allowed in the pool. The default number is **1**.

   max_connections is the maximum number of connections allowed in the pool.

   connection_timeout is the connection pool timeout in milliseconds.

8. (Optional) To configure logging for Web Services Manager, modify the following entry:

   ```
   webservices.log.enabled=log_value
   ```

   where *log_value* is one of the following:

   - **true** enables logging. This option saves and displays the log files as standard output in the application server console.
   - **false** disables logging. This option saves the log files in the **/domain/logs/BRMWebSvcMgr.log** file. Configure the *BRM_Home*/**deploy/web_services/lib/weblogic_ws_startup.jar** file to use this option.

9. Save and close the file.

10. If you are working in the *local_dir*/**WEB-INF/classes** directory, regenerate the WAR file by running one of the following commands:

    To regenerate the **infranetwebsvc.war** file:

```
jar -cvf infranetwebsvc.war *
```

To regenerate the **BrmWebServices.war** file:

```
jar -cvf BrmWebServices.war *
```

11. Deploy the regenerated **infranetwebsvc.war** or **BrmWebServices.war**file on the server. See "Deploying Web Services Manager."

Changing the Instance of BRM to which Web Services Manager Connects

If you customized web services, use the custom **infranetwebsvc.war** or **BrmWebServices.war** file. Otherwise, you should use the default **infranetwebsvc.war**or **BrmWebServices.war** file. For more information about customizing web services, see "Customizing Web Services."

To change the instance of BRM to which Web Services Manager connects, do the following:

1. On your application server, copy the *BRM_home*/**deploy/web_services/Infranet.properties** file to one of the following:
   - *local_dir*/**WEB-INF/classes** directory, where *local_dir* is a directory on the machine on which you installed your application server.

     Note: If you copy the **Infranet.properties** file to the *local_dir*/**WEB-INF/classes** directory, extract the **infranetwebsvc.war** or **BrmWebServices.war** file to a local directory (*local_dir*) on the system on which your application server is installed.

   - home directory on the machine on which you installed your application server.
2. Open the copied **Infranet.properties** file.
3. Locate the following lines:

```
infranet.connection=pcp://root.0.0.0.1:password@ipAddress:port/0.0.0.1/service
/admin_client 1
infranet.login.type=1
```

4. Do the following:
   a. Replace *password* with the password for the BRM server.
   b. Replace *ipAddress* with the IP address of the system on which BRM is installed.
   c. Replace *port* with the port number used by the application server on which BRM is installed.
5. If SSL is enabled in the Connection Manager (CM), locate the following lines and update the parameters if necessary:

```
infranet.pcp.ssl.enabled=true
infranet.pcp.ssl.wallet.location=wallet_directory
```

   where *wallet_directory* is the path to your client Oracle wallet. The client Oracle wallet contains the optional client SSL certificate and the private key, and it contains the Trusted CA certificate.

6. If you added custom opcodes or custom fields for Web services, add the enum values of the custom fields.

   For example, if you created the **custom_fld_usage_id** custom field and the enum value for the **custom_fld_usage_id** field is 10001, add the following entry:

   ```
   infranet.custom.field.10001=custom_fld_usage_id
   ```

   For information about mapping enum values, see "Creating Custom Fields" in *BRM Developer's Guide*.

7. (Optional) To configure the connection pool parameters, modify the following entries:

   ```
   infranet.connectionpool.minsize=min_connections
   infranet.connectionpool.maxsize=max_connections
   infranet.connectionpool.timeout=connection_timeout
   ```

   where:

   min_connections is the minimum number of connections allowed in the pool. The default number is **1**.

   max_connections is the maximum number of connections allowed in the pool.

   connection_timeout is the connection pool timeout in milliseconds.

8. (Optional) To configure logging for Web Services Manager, modify the following entry:

   ```
   webservices.log.enabled=log_value
   ```

   where *log_value* is one of the following:

   - **true** enables logging. This option saves and displays the log files as standard output in the application server console.
   - **false** disables logging. This option saves the log files in the **/domain/logs/BRMWebSvcMgr.log** file. Configure the *BRM_Home***/deploy/web_services/lib/weblogic_ws_startup.jar** file to use this option.

9. (Optional) To configure searching in BRM using the PCM_OP_SEARCH opcode, restrict the PCM_OP_SEARCH opcode to pre-defined search templates by modifying the following entry:

   ```
   allowed.search.template.ids=template_id
   ```

   where *template_id* is the template ID of the search template that you want the PCM_OP_SEARCH opcode to use for searching. Use a comma (,) to separate multiple template IDs. If you do not want to restrict the PCM_OP_SEARCH opcode to any pre-defined search templates, set *template_id* to **None**.

   For a list of template IDs, connect to the BRM database and check the list of POIDS and the respective templates in the SEARCH_T table in the BRM database. For more information, see "Searching for Objects in the BRM Database" in *BRM Developer's Guide*.

10. If you added custom opcodes or custom fields for web services, add the enum values of the custom fields. For information about mapping enum values, see "Creating Custom Fields" in *BRM Developer's Guide*.

    For example, if you created the **custom_fld_usage_id** custom field and the enum value for the **custom_fld_usage_id** field is 10001, add the following entry:

```
infranet.custom.field.10001=custom_fld_usage_id
```

11. Save and close the file.
12. If you are working in the *local_dir*/**WEB-INF/classes** directory, regenerate the WAR file by running one of the following commands:

    To regenerate the **infranetwebsvc.war** file:

    ```
    jar -cvf infranetwebsvc.war *
    ```

    To regenerate the **BrmWebServices.war** file:

    ```
    jar -cvf BrmWebServices.war *
    ```

13. Deploy the regenerated **infranetwebsvc.war** or **BrmWebServices.war** file on the server. See "Deploying Web Services Manager."

Configuring Security for Web Services Manager

By default, secure sockets layer (SSL) security for Web Services Manager is disabled. You can enable SSL security for Web Services Manager by configuring security parameters and enabling the SSL security feature in the application server on which Web Services Manager is deployed.

Configuring Security for Web Services Manager in WebLogic Server

Before you configure security for Web Services Manager, ensure that WebLogic Server and Web Services Manager are installed and that Web Services Manager has been deployed on a WebLogic Server domain. See "Installing Web Services Manager" and "Deploying Web Services Manager" for more information.

To configure security for Web Services Manager in WebLogic Server, do the following:

- Configure authentication for Web Services Manager. See "Configuring Authentication for WebLogic Server."
- Configure authorization for Web Services Manager by doing one of the following:
  - If you have deployed **infranetwebsvc.war**, configure role-based authentication for Apache Axis. See "Configuring Role-Based Authentication for Apache Axis in WebLogic Server."
  - If you have deployed **BrmWebServices.war**, configure WebLogic security policy for JAX-WS. See "Configuring WebLogic Security Policy on BRM Web Services for JAX-WS in WebLogic Server."

Configuring Authentication for WebLogic Server

Before you configure authentication for Web Services Manager, create a user, group, and security realm for Web Services Manager in WebLogic Server. For more information about creating users and groups, see the discussion about users, groups, and security roles in *Fusion Middleware Securing Resources Using Roles and Policies for Oracle WebLogic Server*. For more information about security realms, see the discussion about security realms in WebLogic Server in *Fusion Middleware Securing Oracle WebLogic Server*.

To configure authentication for Web Services Manager in WebLogic Server:

1. Open the *local_dir*/**infranetwebsvc.war/WEB-INF/weblogic.xml** file in a text editor, where *local_dir* is a directory on the WebLogic host where you copied the **infranetwebsvc.war** file.
2. Uncomment the following lines:

```
# <security-role-assignment>
   # <role-name>brmws</role-name>
   # <externally-defined/>
# </security-role-assignment>
```

3. Save and close the file.
4. Open the *local_dir*/**infranetwebsvc.war/WEB-INF/web.xml** file in a text editor.
5. Uncomment the following lines:

```
# <security-constraint>
  # <web-resource-collection>
   #  <web-resource-name>restricted web services</web-resource-name>
    # <url-pattern>/*</url-pattern>
    # <http-method>GET</http-method>
    # <http-method>POST</http-method>
  # </web-resource-collection>
 # <auth-constraint>
    # <role-name>brmws</role-name>
  # </auth-constraint>
  # <user-data-constraint>
   # <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  # </user-data-constraint>
# </security-constraint>

# <login-config>
 # <auth-method>BASIC</auth-method>
 # <realm-name>default</realm-name>
# </login-config>
# <security-role>
    # <role-name>brmws</role-name>
# </security-role>
```

6. Save and close the file.
7. Log in to WebLogic Server Administration Console.
8. Expand **Environment** and select **Servers**.

   The Summary of Servers page appears.
9. Select the server for which you want to enable the SSL port.
10. Click the **Configuration** tab.
11. Click the **General** subtab.
12. Select the **SSL Listen Port Enabled** check box.
13. In the **SSL Listen Port** field, enter a free port number. The default is **7002**.
14. Click **Save**, which configures Web Services Manager with the following default port numbers:
    - HTTP connection: 7001

      Web services that supports XML element payload have the default connection set to HTTP.

- HTTPS connection: 7002

If you use a SOAP development application to generate a web service client and use port numbers other than the default port numbers, the URLs for the web services that supports XML element payload will show port numbers that do not match the port numbers you configured in WebLogic Server Administration Console. Populate the correct port numbers in the URLS for the WSDL files that are generated dynamically by your SOAP development application by doing either of the following:

- Change the port numbers manually in your SOAP development application request.
- Change the port numbers in the **infranetwebsvc.war/WEB-INF/conf/axis2.xml** file as follows:

  Open the *local_dir*/**infranetwebsvc.war/WEB-INF/conf/axis2.xml** file in a text editor.

  Locate the following line and change the HTTP port number:

  ```
  <parameter name="port">7001</parameter>
  ```

  Locate the following line and change the HTTPS port number:

  ```
  <parameter name="port">7002</parameter>
  ```

  Save and close the file.

  Regenerate the **infranetwebsvc.war** file by running the following command:

  ```
  jar -cvf infranetwebsvc.war *
  ```

  Deploy the regenerated **infranetwebsvc.war** file on WebLogic Server. See "Deploying and Launching Web Services Manager on WebLogic Server" for instructions on deploying Web Services Manager.

Configuring Role-Based Authentication for Apache Axis in WebLogic Server

You configure access restrictions for Web Services Manager by creating security roles and by adding conditions to the security roles.

To configure role-based authorization for Apache Axis in WebLogic Server:

1. Log in to WebLogic Server Administration Console.
2. In the Domain Structure pane, click **Deployments** and click the **infranetwebsvc** link.
3. Click the **Security** tab.
4. Click the **Application Scope** subtab.
5. Click the **Roles** subtab.
6. Click **New**.

   The Create Stand-Alone Web Application Scoped Roles page appears.
7. In the **Name** field, enter a name for the role. For example **brmws**.
8. In the **Provider Name** list, select **XACMLRoleMapper**.
9. Click **OK**.
10. Click the link with the role name you created, for example the **brmws** link.

11. Click **Add Conditions**.
12. In the **Predicate List** list, select **Group**.
13. Click **Next**.
14. In the **Group Argument Name** field, enter the name of the desired group.
15. Click **Add**.
16. Click **Finish**.
17. Click **Save**.

Configuring WebLogic Security Policy on BRM Web Services for JAX-WS in WebLogic Server

You define access restrictions for web services in security policies in WebLogic Server.

To configure WebLogic Security Policy on BRM web Services for JAX-WS in WebLogic Server:

1. Log in to WebLogic Server Administration Console.
2. In the Domain Structure pane, click **Deployments** and click the **BrmWebServices** link. A list of all the web services is displayed.
3. Click any of the web service.
4. Click the **Configuration** tab.
5. Click the **WS-Policy** subtab.
6. Click the WS-Policy files associated with this web service.
7. Select **WebLogic** in the **Configure the Policy Type for a Web Service** section.
8. Add the policies from the **Available Endpoint Policies** for the selected service.

   If you want to use the policy for HTTPS with basic authentication, add **policy:***policy_name***-Https-BasicAuth.xml**, where *policy_name* is name of the policy for the selected service; for example, policy:Wssp1.2-2007-Https-BasicAuth.xml.

   If you want to use the policy for HTTPS without authentication, add **policy:***policy_name***-Https.xml**, where *policy_name* is name of the policy for the selected service; for example, policy:Wssp1.2-2007-Https.xml.
9. Click **Finish**.
10. Click **OK** in the **Save Deployment Plan Assistant** section.

    If you have multiple deployments, then the **plan.xml**, which is created when you assign a policy to the service, should be saved in its respective deployment directory.
11. Click the **Security** tab.
12. Click the **Policies** subtab.
13. In the **Web Service Methods** list, select the web method that you want to secure.
14. Click **Add Conditions**.
15. In the **Predicate List** list, select one of the following: Roles, Users, or Groups.
16. Click **Next**.
17. In the **User Argument Name** field, add the user/group.
18. Click **Add**.

19. Click **Finish**.

20. Click **Save**.

If you have enabled SSL, add the following entry in the **BRMWebServices.war/WEB-INF/web.xml** file to enable cookie security:

```
<cookie-config>
            <secure>true</secure>
      </cookie-config>
```

Note: This entry should be added in the session-config element of the **BRMWebServices.war/WEB-INF/web.xml** file.

Configuring Security for Web Services Manager in GlassFish Server

Before you configure security for Web Services Manager, ensure that GlassFish Server and Web Services Manager are installed and that Web Services Manager has been deployed on a GlassFish Server domain. See "Installing Web Services Manager" and "Deploying Web Services Manager" for more information.

To configure security for Web Services Manager in GlassFish Server, do the following:

- Configure authentication for Web Services Manager for Apache Axis in GlassFish Server. See "Configuring Authentication for Web Services Manager for Apache Axis in GlassFish Server."
- Configure authentication for Web Services Manager for JAX-WS in GlassFish Server. See "Configuring Authentication for Web Services Manager for JAX-WS in GlassFish Server."

Configuring Authentication for Web Services Manager for Apache Axis in GlassFish Server

To configure authentication for Web Services Manager for Apache Axis in GlassFish Server:

1. Open the *local_dir*/**infranetwebsvc.war/WEB-INF/web.xml** file in a text editor.

2. Uncomment the following lines:

```
# <security-constraint>
  # <web-resource-collection>
   #  <web-resource-name>restricted web services</web-resource-name>
    # <url-pattern>/*</url-pattern>
    # <http-method>GET</http-method>
    # <http-method>POST</http-method>
  # </web-resource-collection>
 # <auth-constraint>
    # <role-name>brmws</role-name>
  # </auth-constraint>
  # <user-data-constraint>
   # <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  # </user-data-constraint>
# </security-constraint>

# <login-config>
 # <auth-method>BASIC</auth-method>
 # <realm-name>default</realm-name>
# </login-config>
# <security-role>
    # <role-name>brmws</role-name>
# </security-role>
```

3. Locate the following line and change the realm name to **file**:

   ```
   <realm-name>file</realm-name>
   ```

4. Save and close the file.
5. Open the *local_dir*/**infranetwebsvc.war/WEB-INF/sun-web.xml** file in a text editor.
6. Locate the following line and specify a name for your Web Services Manager user group:

   ```
   <group-name>groupname</group-name>
   ```

   where *groupname* is the name of your Web Services Manager user group for GlassFish Server.
7. Save and close the file.
8. Regenerate the **infranetwebsvc.war** file by running the following command:

   ```
   jar -cvf infranetwebsvc.war *
   ```

9. Deploy the regenerated **infranetwebsvc.war** or **BrmWebServices.war** file on the server. See "Deploying and Launching Web Services Manager on GlassFish Server" for more information.
10. Log in to GlassFish Server Administration Console.
11. In the Common Tasks pane, under Configurations, select **server-config**.
12. Click **Security** and select **Realms**.
13. Select **file**.

    The Edit Realm page appears.
14. Click **Manage Users** and add a user. See the GlassFish Server documentation for instructions about creating a user.
15. Associate the user to the group that you created in step 6. See the GlassFish Server documentation for instructions about associating a user to a group.
16. Click **Save**.
17. In the Common Tasks pane, under **Configurations**, select **server-config**.
18. Select **Network Config**.
19. Select **Network Listeners**.
20. Select the port that is enabled for an HTTPS connection.

    Note: By default, **http-listener -1** is for an HTTP connection and **http-listener-2** is for an HTTPS connection.

21. Click the SSL tabbed page.
22. Select the **SSL3** check box.
23. Deselect the **TLS** check box.

24. Enter the details about the SSL certificate and the keystore in their relevant fields.

    For details about the SSL certificate and the keystore, open the *local_dir*/**glassfish/config** directory on the server on which you installed GlassFish Server.

25. Click **Save**.

Configuring Authentication for Web Services Manager for JAX-WS in GlassFish Server

To configure authentication for Web Services Manager for JAX-WS in GlassFish Server:

1. Open the *local_dir*/**BrmWebServices.war/WEB-INF/web.xml** file in a text editor.
2. Add the following lines:

```
# <security-constraint>
  # <web-resource-collection>
   #  <web-resource-name>restricted web services</web-resource-name>
    # <url-pattern>/*</url-pattern>
    # <http-method>GET</http-method>
    # <http-method>POST</http-method>
  # </web-resource-collection>
 # <auth-constraint>
    # <role-name>*</role-name>
  # </auth-constraint>
  # <user-data-constraint>
   # <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  # </user-data-constraint>
# </security-constraint>

# <login-config>
 # <auth-method>BASIC</auth-method>
 # <realm-name>default</realm-name>
# </login-config>
```

3. Locate the following line and change the realm name to **file**:

```
<realm-name>file</realm-name>
```

4. Save and close the file.
5. Go to the *local_dir*/**BrmWebServices.war**/**WEB-INF** directory and create the **sun-web.xml** file.
6. Open the *local_dir*/**BrmWebServices.war**/**WEB-INF**/**sun-web.xml** file in a text editor.

   Add the following lines:

```
<?xml version="1.0" encoding="UTF-8"?>
     <!DOCTYPE sun-web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Application
 Server 9.0 Servlet 2.5//EN"
 "http://www.sun.com/software/appserver/dtds/sun-web-app_2_5-0.dtd">
     <sun-web-app error-url="">
       <security-role-mapping>
         <role-name>rolename</role-name>
         <!--Add group name as configured in the security realm 'file' -->
         <group-name>groupname</group-name>
       </security-role-mapping>
     </sun-web-app>
```

where:

rolename is the name of your Web Services Manager role for GlassFish Server.

groupname is the name of your Web Services Manager user group for GlassFish Server.

7. Save and close the file.
8. Regenerate the **BrmWebServices.war** file by running the following command:

   To regenerate the **BrmWebServices.war** file:

   ```
   jar -cvf BrmWebServices.war *
   ```

9. Deploy the regenerated **infranetwebsvc.war** or **BrmWebServices.war** file on the server. See "Deploying and Launching Web Services Manager on GlassFish Server" for more information.
10. Log in to GlassFish Server Administration Console.
11. In the Common Tasks pane, under Configurations, select **server-config**.
12. Click **Security** and select **Realms**.
13. Select **file**.

    The Edit Realm page appears.
14. Click **Manage Users** and add a user. See the GlassFish Server documentation for instructions about creating a user.
15. Associate the user to the group that you created in step 6. See the GlassFish Server documentation for instructions about associating a user to a group.
16. Click **Save**.
17. In the Common Tasks pane, under **Configurations**, select **server-config**.
18. Select **Network Config**.
19. Select **Network Listeners**.
20. Select the port that is enabled for an HTTPS connection.

    Note: By default, **http-listener -1** is for an HTTP connection and **http-listener-2** is for an HTTPS connection.

21. Click the SSL tabbed page.
22. Select the **SSL3** check box.
23. Deselect the **TLS** check box.
24. Enter the details about the SSL certificate and the keystore in their relevant fields.

    For details about the SSL certificate and the keystore, open the *local_dir***/glassfish/config** directory on the server on which you installed GlassFish Server.
25. Click **Save**.

Configuring Security for Web Services Manager in Tomcat Server

Before you configure security for Web Services Manager, ensure that Tomcat server and Web Services Manager are installed and that Web Services Manager has been deployed on a Tomcat server domain. See "Installing Web Services Manager" and "Deploying Web Services Manager"for more information.

To configure security for Web Services Manager in Tomcat server, do the following:

- Configure authentication for Web Services Manager for Apache Axis in Tomcat server. See "Configuring Authentication for Web Services Manager for Apache Axis in Tomcat Server."
- Configure authentication for Web Services Manager for JAX-WS in Tomcat server. See "Configuring Authentication for Web Services Manager for JAX-WS in Tomcat Server."
- Enable SSL in Tomcat server. See "Enabling SSL in Tomcat Server."

Configuring Authentication for Web Services Manager for Apache Axis in Tomcat Server

To configure authentication for Web Services Manager for Apache Axis in Tomcat server:

1. Open the *local_dir*/**infranetwebsvc.war/WEB-INF/web.xml** file in a text editor.
2. Uncomment the following lines:

```
# <security-constraint>
  # <web-resource-collection>
   #  <web-resource-name>restricted web services</web-resource-name>
    # <url-pattern>/*</url-pattern>
  # </web-resource-collection>
 # <auth-constraint>
    # <role-name>brmws</role-name>
  # </auth-constraint>
# <user-data-constraint>
    # <transport-guarantee>CONFIDENTIAL</transport-guarantee>
# </user-data-constraint>
# </security-constraint>

# <login-config>
 # <auth-method>BASIC</auth-method>
 # <realm-name>default</realm-name>
# </login-config>
# <security-role>
    # <role-name>brmws</role-name>
# </security-role>
```

3. Locate the following line and specify the Web resource name as follows:

```
<web-resource-name>Protected Resource</web-resource-name>
```

4. Locate the following lines and specify the realm name as follows:

```
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>Apache version</realm-name>
</login-config>
```

   where *version* is the Tomcat server version on which you deployed Web Services Manager.

5. Save and close the file.
6. Open the *local_dir*/**apache-tomcat-7.0.54/conf/tomcat-users.xml** file in a text editor.

7. Locate the following lines and specify the login details of the user:

```
<role rolename="brmws"/>
<user username="username" password="password" roles="brmws"/>
```

where:

- *username* is the user name for accessing Web services.
- *password* is the password for accessing Web services.

8. Save and close the file.
9. Open the **config/server.xml** file in a text editor.
10. In the **<Engine>** section, add the following class path:

```
 <Realm className="org.apache.catalina.realm.MemoryRealm" />
```

11. Save and close the file.
12. Restart the Tomcat server.

Configuring Authentication for Web Services Manager for JAX-WS in Tomcat Server

To configure authentication for Web Services Manager for JAX-WS in Tomcat server:

1. Open the *local_dir*/**BrmWebServices.war/WEB-INF/web.xml** file in a text editor.
2. Add the following lines:

```
# <security-constraint>
  # <web-resource-collection>
   #  <web-resource-name>restricted web services</web-resource-name>
    # <url-pattern>/*</url-pattern>
    # <http-method>GET</http-method>
    # <http-method>/POST</http-method>
  # </web-resource-collection>
 # <auth-constraint>
    # <role-name>brmws</role-name>
  # </auth-constraint>
# <user-data-constraint>
    # <transport-guarantee>CONFIDENTIAL</transport-guarantee>
# </user-data-constraint>
# </security-constraint>

# <login-config>
 # <auth-method>BASIC</auth-method>
# </login-config>
# <security-role>
    # <role-name>brmws</role-name>
# </security-role>
```

3. Save and close the file.
4. Open the *local_dir*/**apache-tomcat-7.0.62/conf/tomcat-users.xml** file in a text editor.
5. Locate the following lines and specify the login details of the user:

```
<role rolename="brmws"/>
<user username="username" password="password" roles="brmws"/>
```

where:

- *username* is the user name for accessing Web services.

- *password* is the password for accessing Web services.

6. Save and close the file.
7. Open the **config/server.xml** file in a text editor.
8. In the **<Engine>** section, add the following class path:

   ```
   <Realm className="org.apache.catalina.realm.MemoryRealm" />
   ```

9. Save and close the file.
10. Restart the Tomcat server.

Enabling SSL in Tomcat Server

To enable secure communication for Web Services Manager, enable secure sockets layer (SSL) in the Tomcat server domain on which you deploy Web Services Manager.

To enable SSL for Tomcat server:

1. Generate the keystore by running the following command:

   ```
   keytool -genkey -alias mykes -keyalg RSA -keystore mykeystore
   ```

   where:

   - *mykes* is the alias.
   - *mykeystore* is the name of the keystore.

2. Open the **conf/server.xml** file in a text editor.
3. Uncomment the following lines and specify the path for the keystore file:

   ```
   # <Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
               # address="IPAddress"
               # maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
               # clientAuth="false" sslProtocol="TLS"
               # keystoreFile="/glassfish/glassfish3/bin/filepath"
               # keystorePass="password" />
   ```

   where:

   - *IPAddress* is the IP address of the machine on which you installed the Apache Tomcat server.
   - *filepath* is the keystore file path.
   - *password* is the password for the keystore file.

4. Save and close the file.

Configuring Java Logging for the Application Server

Depending on your configuration, you may wish to change the level of Java logging on the application server. To configure the Java logging level, do the following:

- For WebLogic Server, see "Configuring Java Logging for WebLogic Server" for Web Services Manager-specific configuration. For more information, see the discussion about application logging and WebLogic logging services in *Fusion Middleware Using Logging Services for Application Logging for Oracle WebLogic Server*.
- For GlassFish Server, see the discussion about configuring logging services in *GlassFish Server Administration Guide*.

- For Tomcat server, see the discussion about logging in Tomcat in *Tomcat User Guide*.

Configuring Java Logging for WebLogic Server

To configure Java logging in WebLogic Server:

1. Specify the Java Unified Logging (JUL) mechanism. See "Specifying the Java Unified Logging (JUL) Mechanism."
2. Create a startup class. See "Creating a Startup Class."

Specifying the Java Unified Logging (JUL) Mechanism

Specifying the JUL mechanism allows Web Services Manager to use JUL in addition to the WebLogic Server Administration Console logging.

To specify the JUL mechanism:

1. Open the *BRM_Home*/**deploy/web_services/Infranet.properties** file in a text editor.
2. Uncomment the following entry:

   ```
   # webservices.log.enabled = true
   ```

3. Change the value to **false**:

   ```
   webservices.log.enabled = false
   ```

4. Save and close the file.

Creating a Startup Class

You define a startup class to enable JUL and create log files for the following Web service classes:

- **com.portal.webservices.BRMFlistToXML**
- **com.portal.webservices.BRMXMLToFlist**
- **com.portal.webservices.OpcodeCaller**
- **com.portal.webservices.WebServicesUtilities**

To create a startup class:

1. Copy the *BRM_Home*/**deploy/web_services/weblogic_ws_startup.jar** file to the *server_domain*/**lib** directory, where *server_domain* is the WebLogic Server domain in which Web Services Manager is deployed.
2. Log in to WebLogic Server Administration Console.
3. Click **Lock and Edit**.
4. In the **Domain Structure** pane, expand **Environment** and then click **Startup and Shutdown classes**.

   The Startup and Shutdown Classes pane appears.
5. Click **New**.

   The Configure a New Startup or Shutdown Class: Class Type pane appears.
6. Select **Startup Class** and click **Next**.

The Configure a New Startup or Shutdown Class: Startup Class Properties pane appears.

7. In the **Name** field, enter **BRMWSLoggerStartUpClass**.
8. In the **Class Name** field, enter **com.portal.webservices.BRMWSLoggerStartUp**.
9. In the **Argument** field, set the log level. This field sets the log level for all the classes in Web Services Manager:
   - To log problems that require attention from the system administrator, enter **SEVERE**. This is the default.
   - To log the most detailed trace and debug messages, enter **FINEST**.
   - To log highly detailed trace and debug messages, enter **FINER**.
   - To log trace and debug messages for performance monitoring, enter **FINE**.
10. Click **Next**.

    The Configure a New Startup or Shutdown Class: Select Targets pane appears.
11. From the **Servers** list, select the server on which to deploy the class.

    The Startup and Shutdown Classes pane appears.
12. Click **Finish**.
13. Click **BRMWSLoggerStartUpClass**.

    The Settings for BRMWSLoggerStartUpClass pane appears.
14. Select **Run Before Application Deployments** and **Run Before Application Activations** and click **Save**.
15. Click **Activate Changes**.
16. Restart the WebLogic server, which applies changes.
17. Redeploy any existing Web Services Manager deployments. See "Deploying Web Services Manager."

By default, log files are created in the *WebLogic_Home*/**user_projects/domains/***Domain_Name*/**logs/BRMWebServicesMgrLogs/BRMWebServicesMgr.log** file

where:

- *WebLogic_Home* is the directory in which WebLogic Server is installed.
- *Domain_Name* is the name of the domain you are configuring.

4 Securing Web Services Manager with OAuth2

This chapter provides information about securing Web Services Manager with the OAuth2 authorization framework.

The OAuth2 authorization framework enables you to secure clients' access to Oracle Communications Billing and Revenue Management (BRM) web services. When you set up Web Services Manager with OAuth2 authorization, client applications use a token to obtain authorization. Web Services Manager validates the token to authorize users' access to BRM web services. Web Services Manager supports the Authorization Code Grant flow to secure access to web services. Authorization Code Grant flow enables servers to protect the client application's secret (password). Web Services Manager requires and uses Oracle Identity and Access Management to manage the identity and access of clients that access BRM web services.

To set up Web Services Manager with OAuth2 authorization:

1. Install Oracle Identity and Access Management. For installation instructions, see *Oracle Fusion Middleware Installation Guide for Oracle Identity and Access Management*.
2. Configure Oracle Access Manager. See "Configuring Oracle Access Manager" for instructions.
3. Configure Web Services Manager. See "Configuring Web Services Manager" for instructions.
4. Enable OAuth validation for Web Services Manager. See "Enabling OAuth Validation for Web Services Manager" for instructions.

Configuring Oracle Access Manager

Oracle Access Manager is a component of Oracle Identity and Access Management.

To configure Oracle Access Manager:

1. Create a resource. See "Creating a Resource" for instructions.
2. Create a validation client. See "Creating an OAuth Validation Client" for instructions.
3. Create a web client. See "Creating Web Service Clients" for instructions.

Creating a Resource

To create a resource:

1. Log in to Oracle Access Manager Console.

2. In your default identity domain, create a resource with any name. See Oracle Access Manager Console Help for instructions on creating resources.
3. Add all the BRM web services as scopes.
4. Specify the token settings to override the default settings.

Creating an OAuth Validation Client

An OAuth validation client acts as an internal client and runs within Web Services Manager. The validation client validates clients' access tokens. When a BRM web service is accessed from a web service client, the OAuth validation client validates it.

To create an OAuth validation client:

1. In Oracle Access Manager Console, in your default identity domain, create a client with the following specifications:
   - **Name**: **ServiceTokenValidator**
   - **Client ID**: **ServiceTokenValidator**
2. Specify the settings to allow token attributes retrieval.
3. Generate a client secret, which you will enter in the *BRM_home***/deploy/web_services/Infranet.properties** file when you configure Web Services Manager.

Creating Web Service Clients

You can create a single client for all available services or one client for each service.

To create a web service client:

1. In Oracle Access Manager Console, in your default identity domain, create a web client with the following specifications:
   - **Name**: **BRMOAuthClient**
   - **Client ID**: **BRMOAuthClient**
2. Specify the settings to allow token attributes retrieval.
3. Specify the URL of your client. For example, if you use a servlet test client to call BRMBalWebService, specify the URL of your servlet.
4. Add all the BRM web services as scopes.
5. Specify the Grant Type to use Authorization Code.

Configuring Web Services Manager

To configure Web Services Manager:

1. Copy the *BRM_home***/deploy/web_services/Infranet.properties** file to the directory of the server where the web server is installed.
2. Open the copied **Infranet.properties** file in a text editor.
3. Specify the following parameters:

```
infranet.OAuthAccessTokenUrl=URL
infranet.OAuthClientId=ServiceTokenValidator
infranet.OAuthClientSecret=clientSecret
infranet.OAuthGrantType=grantType
infranet.OAuthTokenAction=validate
infranet.OAuthTokenAttributes=tokenAttribute
infranet.ResourceScopePrefix=scopePrefix
```

where:

- *URL* is the IP address and port number of the Oracle Access Manager server.
- *ServiceTokenValidator* is the ID of the client that you added as the service token validator.
- *clientSecret* is the client secret that you generated when you created the OAuth validation client.

> **Note:** The client secret can be added as plain text. However, you can encrypt this secret using the **pin_crypt_app** utility and add the encrypted secret to the file.

- *grantType* is the type of grant. The default is: **oracle-idm%3A%2Foauth%2Fgrant-type%2Fresource-access-token%2Fjwt**.
- *tokenAttribute* is the token attribute. The default is: **oracle_token_attrs_retrieval=iss%20aud%20exp%20prn%20jti%20exp%20iat%20oracle.oauth.scope%20oracle.oauth.client_origin_id%20oracle.oauth.user_origin_id%20oracle.oauth.user_origin_id_type%20oracle.oauth.tk_context%20oracle.oauth.id_d_id%20oracle.oauth.svc_p_n**
- *scopePrefix* is your resource name. The resource name is a default prefix for all the available scopes of a certain resource. For example, if the resource scope is created as **BRMWSM.BRMBalServices_v2**, **BRMWSM** is the scope prefix.

4. Save and close the file.
5. Restart the WebLogic server.

> **Note:** When you restart the WebLogic server, ensure that the **libportal.so** BRM library is set in **LD_LIBRARY_PATH**.
>
> For JRE on 64-bit environments, rename **libportal64.so** to **libportal.so**.

Enabling OAuth Validation for Web Services Manager

By default, OAuth validation is disabled.

To enable OAuth validation for BRM web services:

1. Open the *local_dir*/**BrmWebServices.war** file in a text editor, where *local_dir* is the directory that contains the **BrmWebServices.war** file.
2. Uncomment the following text:

```
<filter>
      <filter-name>OAuthTokenValidationFilter</filter-name>
      <filter-class>com.portal.jax.OAuthTokenValidationFilter</filter-class>
</filter>
<filter-mapping>
   <filter-name>OAuthTokenValidationFilter</filter-name>
   <servlet-name>BrmWebServices</servlet-name>
   <url-pattern>/BrmWebServices/*</url-pattern>
   <url-pattern>/BRMPricingServices_v2/</url-pattern>
   <url-pattern>/BRMBalServices_v2/</url-pattern>
```

```
        <url-pattern>/BRMARServices_v2</url-pattern>
        <url-pattern>/BRMBillServices_v2</url-pattern>
        <url-pattern>/BRMCustServices_v2</url-pattern>
        <url-pattern>/BRMCustcareServices_v2</url-pattern>
        <url-pattern>/BRMInvServices_v2</url-pattern>
        <url-pattern>/BRMPymtServices_v2</url-pattern>
        <url-pattern>/BRMCollectionServices_v2</url-pattern>
        <url-pattern>/BRMReadServices_v2</url-pattern>
        <url-pattern>/BRMActServices_v2</url-pattern>
        <url-pattern>/BRMSubscriptionServices_v2</url-pattern>
        <dispatcher>FORWARD</dispatcher>
        <dispatcher>REQUEST</dispatcher>
    </filter-mapping>
```

3. Save and close the file.
4. Regenerate the **BrmWebServices.war** file.

5 Customizing Web Services

This chapter contains information about customizing Oracle Communications Billing and Revenue Management (BRM) Web Services Manager.

Before reading this chapter, you should be familiar with implementing web services.

Setting Up Web Services Manager to Support Custom Opcodes

To expose custom opcodes as web services, enable Web Services Manager to support custom opcodes. For more information on custom opcodes, see "Using Custom Opcodes" in *BRM Developer's Guide*.

Note: Before you customize an opcode for a web service in Web Services Manager, implement the custom opcode in BRM.

To enable Web Services Manager to support custom opcodes:

1. Do one of the following:
   - Create the **CustomOp.java** file by entering the following command:

     ```
     parse_custom_ops_fields -L pcmjava -I input -O output -P java_package
     ```

     where:

     – *input* is the header file you create for your custom opcodes and fields.

     – *output* is the memory-mapped file or directory for the output of the script. *output* must be a directory having some correspondence with the Java package. For example, if *java_package* is in **com.portal.classFiles**, *output* must be **f:/mysource/com/portal/classFiles**.

     – *java_package* is the Java package in which to put the generated classes.

     For more information, see the discussion about the **parse_custom_ops_fields** utility in *BRM Developer's Guide*.
   - Manually create the **CustomOp.java** file.
2. Verify that the **CustomOp.java** file contains the following:
   - The opcode-name-to-opcode-number mapping for all the custom opcodes in the file.

Important: Verify that the mapping includes the full name of each opcode. If any opcode name is truncated, replace the truncated name with the full name.

- The **opToString** method, which converts opcode numbers to opcode names.
- The **stringToOp** method, which converts opcode names to opcode numbers.

The following is a sample **CustomOp.java** file:

```
public class CustomOp {
   public static final int CUSTOM_OP_ACT_INFO= 100000;
   public static final int CUSTOM_OP_READ_ACT_PRODUCT = 100001;

   public static String opToString( int op ) {
           try {
                 java.lang.reflect.Field[] flds =
CustomOp.class.getFields();
           for( int i = 0; i < flds.length; i++ ) {
                 try {
                    int val = flds[i].getInt(null);
                    if( val == op ) {
                        return flds[i].getName();
                    }
            } catch( IllegalAccessException e ) { continue;
            } catch( IllegalArgumentException e ) { continue; }
                    }
             } catch( SecurityException e ) {}

              return null;
}

public static int stringToOp( String op ) {
              try {
                    java.lang.reflect.Field[] flds =
CustomOp.class.getFields();
                 for( int i = 0; i < flds.length; i++ ) {
                    try {
                        String name = flds[i].getName();
                        if( name.equals(op) ) {
                          return flds[i].getInt(null);
                         }
                     } catch( IllegalAccessException e ) { continue;
}
                        catch( IllegalArgumentException e ) { continue; }
               }
          } catch( SecurityException e ) {}
       return -1;
     }
}
```

3. Compile the **CustomOp.java** file into the **CustomOp.class** file by entering the following command:

```
javac -d . path/CustomOp.java
```

For example:

```
javac -d . com/portal/classFiles/CustomOp.java
```

4. Package the **CustomOp.class** file into the **CustomFields.jar** file by entering the following command:

   > **Note:** Make sure the JRE version that was used to generate the **CustomFields.jar** file is the same or lower than the version of the WebLogic Server JRE.

   ```
   jar -cvf CustomFields.jar path.CustomOp.class
   ```

   For example:

   ```
   jar cvf CustomFields.jar com.portal.classFiles.CustomOp.class
   ```

5. Make the **CustomFields.jar** file available to Web Services Manager by doing one of the following:

   - If you have not deployed Web Services Manager, do the following:

     a. Copy the *path*/**CustomFields.jar** file to the *local_dir*/**WEB-INF/lib** directory, where *path* is the path to the **CustomFields.jar** file (for example, **com/portal/classFiles**).

     b. Open the *BRM_home*/**deploy/web_services/Infranet.properties** file in a text editor.

     c. Add or modify the following entry:

        ```
        infranet.custom.field.package = package
        ```

        where *package* is the name of the package that contains the **CustomOp.java** file; for example, **com.portal.classFiles**.

     d. Add all the custom fields to the **Infranet.properties** file.

     e. Save and close the file.

     f. Copy the *BRM_home*/**deploy/web_services/Infranet.properties** file to the *local_dir*/**WEB-INF/classes** directory or the home directory on the machine on which you installed WebLogic Server.

   - If you have deployed Web Services Manager, do the following:

     a. Copy the *path*/**CustomFields.jar** file to the *local_dir*/**WEB-INF/lib** directory.

        where *local_dir* is the directory in which you deployed Web Services Manager on your application server.

     b. Open the *Webservices_deployment_dir*/**WEB-INF/classes/Infranet.properties** file in a text editor.

     c. Add or modify the following entry:

        ```
        infranet.custom.field.package = package
        ```

        where *package* is the name of the package that contains the **CustomOp.java** file; for example, **com.portal.classFiles**.

     d. Add all the custom fields to the **Infranet.properties** file.

     e. Save and close the file.

Generating Schema Files for Your System

Web Services Manager uses schema files to validate data it sends to or receives from BRM.

To generate schema files for your system:

1. If you modified any opcodes, generate schemas for the opcodes in your BRM system. See "Generating Schema for Your Opcodes".
2. Generate schemas for the storable classes and subclasses in your BRM system. See "Generating the Schema for Your Storable Classes and Subclasses" in *BRM JCA Resource Adapter*.
3. In your opcode schema files, specify the location of your storable class schema files. See "Specifying the Location of the Storable Class Schema Files in the Opcode Schema Files" in *BRM JCA Resource Adapter*.

Note: After generating the opcode and storable class schema files, copy the schema files to a location that is accessible to the Web Services Manager. Make sure that this location is the same as the location that is specified in the **include** section of the opcode schema files and in the opcode schema **InteractionSpec** attribute in the WSDL files. See "Specifying the Location of the Storable Class Schema Files in the Opcode Schema Files" and "Generating the WSDL Files for Your System" in *BRM JCA Resource Adapter*.

Generating Schema for Your Opcodes

The Web Services Manager package includes all the opcode schemas and flist specifications you need for a default integration.

If you customized any of the opcodes that are supported by Web Services Manager or if you added support for new opcodes, you must generate XSD schema files for the opcodes.

Note: Before you customize an existing opcode specification, ensure that you update the opcode specification in the BRM system.

After you customize web services, copy the customized schema files and the WSDL files to the **infranetwebsvc.war** file.

Generating Schema for an Existing Opcode

To generate schema files for an opcode that you customized and Web Services Manager already supports:

1. Modify the opcode's XML specification file. By default, the opcode specification XML files are installed in the *BRM_home***/apps/brm_integrations/opspecs** directory, where *BRM_home* is the directory in which you installed the BRM components.
2. Do one of the following:
   - For web services that take payload as XML string:
     – Run the **pin_opspec_to_schema** utility. See "Creating Opcode Specification Schema Files".

   - Copy the customized XSD files to the *BRM_home*/**deploy**/**web_services**/**schemas** directory.

- For web services that take payload as XML element:
   - Run the **pin_opspec_to_schema_v2** utility. See "Creating Opcode Specification Schema Files".
   - Copy the customized XSD files to the **BrmWebServices/WEB-INF/wsdll** directory.

Creating Opcode Specification Schema Files

You must create opcode flist specification files for opcodes that you customize or add to the Web Services Manager. Create the specification XML files by following the *BRM_home*/**apps/brm_integrations/stylesheets/opspec.xsd** file.

You then convert the opcode flist specification XML files into XSD schema by using the **pin_opspec_to_schema** and **pin_opspec_to_schema_v2** utilities.

To convert opcode flist specification XML files into XSD schema, go to the *BRM_home*/**apps/brm_integrations** directory and do the following:

- For web services that supports XML string payload, run the following command:

```
pin_opspec_to_schema -i input_file [-o output_file]
```

- For web services that supports XML element payload, run the following command:

   Note: Before you run the following command, specify the BRM installation directory in the **pin_opspec_to_schema_v2** utility by replacing $PIN_HOME with *BRM_home*.

```
pin_opspec_to_schema_v2 -i input_file > output_file
```

   where:

   - *input_file* specifies the name and location of the opcode's XML flist specification. By default, the utility looks for the file in the current directory.
   - *output_file* creates the XSD schema output file using the name you specify. By default, the utility creates a file named *opcodename***.xsd** in the directory from which you run the utility.

You can also create XSD schema for web services that take payload as XML element by using the **pin_opspec_to_schema_v2** XSD generator utility that is located in the *BRM_home*/**bin** directory.

To create the XSD schema file by using the **pin_opspec_to_schema_v2** utility, run the following command using Groovy:

```
groovy pin_opspec_to_schema_v2 -i input.xml > output.xsd
```

where:

- *input.xml* specifies the name of the opcode's XML flist specification
- *output.xsd* creates the XSD schema output file using the name you specify

Specifying the XSL Rules to Create the Opcode Schema

The **pin_opspec_to_schema** utility uses the *BRM_home*/**brm_integrations/stylesheets/pin_opspec_to_schema.xsl** style sheet to generate the schema for BRM opcodes. If your opcode references custom fields, you must customize the **pin_opspec_to_schema.xsl** style sheet to handle your custom fields.

For a list of the supported BRM data types, see "Understanding the BRM Data Types" in *BRM Developer's Guide*.

Creating a Custom Web Service

You can extend Web Services Manager to support custom web services. Before you create a custom web service or customize an existing web service in Web Services Manager, implement your custom opcodes in the BRM system. For more information on custom opcodes, see "Using Custom Opcodes" in *BRM Developer's Guide*.

To create a custom web service, do one of the following:

- If you have deployed **infranetwebsvc.war,** create a custom web service that supports XML string payload. See "Creating a Custom Web Service that Supports XML String Payload."
- If you have deployed **BrmWebServices.war,** create a custom web service that supports XML element payload. See "Creating a Custom Web Service that Supports XML Element Payload."

Creating a Custom Web Service that Supports XML String Payload

To create a custom web service that supports XML string payload:

1. If you created an opcode with custom fields for your custom web service, configure BRM to recognize the custom fields.
2. Create the XML specifications for your custom opcodes. See "Creating Opcode Specification Schema Files".
3. Create a **/service** storable class for your custom web service. See the *BRM_home*/**deploy/web_services/sample/SampleWebService.java.template** sample.

   Note: When you compile the **/service** storable class, use the libraries from the **infranetwebsvc/WEB-INF/lib** directory in addition to the JDK and JAX-WS reference implementation (RI).

4. Build and generate custom web services jar file in an integrated development environment IDE or by using an Ant build file as follows:

   a. Create the following directory structure in a local directory (*local_dir*) on the machine:

   ```
   /src
   /classes
   /jar
   ```

   b. Copy your custom service classes into the *local_dir*/**src** directory.

   c. Create **custom_services.xml** as an Ant build file.

   The following is a sample **custom_services.xml** file:

   ```
   <?xml version="1.0"?>
   ```

```
<project name="Custom WebServices build file" default="all" basedir=".">
<property name="buildDir"     value="classes"/>
<property name="srcDir"value="src"/>

<!-- define the classpath -->
<path id="classpath">
        <pathelement path="${buildDir}"/>
        <pathelement path="jar/web_services.jar"/>
        <pathelement path="jar/webServicesUtils.jar"/>


</path>

<target name="all" custom_jar" description="build everything" />

<!-- compile task -->
<target name="compile"  description="compile source files" >
        <echo>" Compiling JAX-WS impl classes"</echo>
        <javac  srcdir="${srcDir}"
                destdir="${buildDir}"
                classpathref="classpath"
                debug="on"
                source="1.5"
        />
</target>
<!--Create custom service jar -->
<target name="custom_jar" depends="custom_service_gen, compile"
description="generate jar file" >
        <jar jarfile="custom_services.jar" basedir="${buildDir}" >
        </jar>
</target>
</project>
```

5. Generate and build your custom web services by running the following command:

   ant -file custom_services.xml

6. Add all the custom field **enum** constants to the **Infranet.properties** file. See "Connecting Web Services Manager to the BRM System" for more information.

7. Package your custom web service with the **infranetwebsvc.war** file by doing the following:

   a. Extract the **infranetwebsvc.war** file to a local directory (*local_dir*) on the machine.

   b. Do one of the following:

      (WebLogic server) Add your custom service URL mapping to the *local_dir*/**WEB-INF/Web.xml** file. It should be similar to existing URL mapping.

      (Apache Tomcat and GlassFish servers) Add your custom service implementation class to the *local_dir*/**WEB-INF/sun.jaxws.xml** file.

   c. Copy your **custom_services.jar** file into the *local_dir*/**WEB-INF/lib** directory.

   d. Copy your **CustomFields.jar** file into the *local_dir*/**WEB-INF/lib** directory.

   e. Delete the existing **infranetwebsvc.war** file.

   f. Create a new **infranetwebsvc.war** file by running the following command:

      jar -cvf infranetwebsvc.war *

Creating a Custom Web Service that Supports XML Element Payload

To create a custom web service that supports XML element payload:

1. If you created an opcode with custom fields for your custom web service, configure BRM to recognize the custom fields.
2. Create a WSDL file for the web service. See "Generating WSDL Files for Web Services" in *BRM JCA Resource Adapter*.
3. Create a WSDL file for your custom web service. See the *BRM_home*/**deploy/web_services/BrmWebServices.war/WEB-INF/wsdl** sample.
4. Create the XML specifications for your custom opcodes. See "Creating Opcode Specification Schema Files".
5. Build and generate custom web service classes for your custom service as follows:

   a. Create the following directory structure in a local directory (*local_dir*) on the machine on which BRM is installed.

   ```
   /wsdl
   /src
   /classes
   /jar
   ```

   b. Copy your custom WSDL files and schema (XSD) files into the *local_dir*/**wsdl** directory.

   c. Copy the **BrmWebServices.war/WEB-INF/wsdl/BRMWebServiceException.xsd** file into the *local_dir*/**wsdl** directory.

   d. Copy the mentioned files from **BrmWebServices.war/WEB-INF/lib/**file into the *local_dir*/**jar** directory.

   ```
   web_services.jar
   webServicesUtils.jar
   xmlparserv2.jar
   ```

   e. Create **custom_services.xml** as an Ant build file.

   The following is a sample **custom_services.xml** file:

   ```
   <?xml version="1.0"?>
   <project name="Custom BRM WebServices build file" default="all"
   basedir=".">
   <property name="buildDir"     value="classes"/>
   <property name="srcDir" value="src"/>
   <property name="wsdlDir" value="wsdl"/>
   <property name="pinwsgen" value="pin_wsgen"/>

   <!-- define the classpath -->
   <path id="classpath">
           <pathelement path="${buildDir}"/>
           <pathelement path="jar/web_services.jar"/>
           <pathelement path="jar/webServicesUtils.jar"/>
           <pathelement path="jar/xmlparserv2.jar"/>


   </path>

   <!-- create Source files from WSDL and XSDs -->
   <target name="custom_service_gen" description="Create java source files
   ```

```
from wsdl"  >
        <exec executable="BRM_home/deploy/web-services/pin_wsgen/pin_
wsgen" failonerror="true">
            <arg value="-s"/>
            <arg value="src"/>
            <arg value="-d"/>

<arg value="${buildDir}"/>

<arg value="-p"/>
<arg value="com.portal.jax.'YourPackageSubdirName' "/>
<arg value="${wsdlDir}/'YourCustomServices_v2.wsdl'/>
        </exec>
</target>

<target name="all" depends="custom_service_gen, custom_jar"
description="build everything" />

<!-- compile task -->
<target name="compile" depends="custom_service_gen" description="compile
source files" >
        <echo>" Compiling JAX-WS impl classes"</echo>
        <javac  srcdir="${srcDir}"
                destdir="${buildDir}"
                classpathref="classpath"
                debug="on"
                source="1.5"
        />
</target>
<!--Create custom service jar -->
<target name="custom_jar" depends="custom_service_gen, compile"
description="generate jar file" >
        <jar jarfile="custom_services.jar" basedir="${buildDir}" >
        </jar>
</target>
<!--ant clean task -->
<target name="clean" description="remove derived objects" >

<delete dir="classes/com"/>

<delete dir="custom_service.jar"/>

</target>

</project>
```

where:

BRM_home is the directory in which BRM is installed.

YourCustomServices_v2 is the custom service WSDL file name.

YourPackageSubdirName is the package directory for your custom service.

6. Generate and build your custom web services by running the following command:

   `ant -file custom_services.xml`

7. Add all the custom field **enum** constants to the **Infranet.properties** file. See "Connecting Web Services Manager to the BRM System".

8. Package your custom web service with the **BrmWebServices.war** file as follows:

a. Extract the **BrmWebServices.war** file to a local directory (*local_dir*) on the machine on which you installed your application server.

b. Do one of the following:

   (WebLogic server) Add your custom service URL mapping to the *local_dir***/WEB-INF/Web.xml** file. It should be similar to existing URL mapping.

   (Apache Tomcat and GlassFish servers) Add your custom service implementation class to the *local_dir***/WEB-INF/sun.jaxws.xml** file.

c. Copy your custom WSDL files and schema (XSD) files into the *local_dir***/WEB-INF/wsdl/** directory.

d. Copy your **custom_services.jar** and all custom com.portal.classFiles into the *local_dir***/WEB-INF/**classes and *local_dir***/WEB-INF/**classes/com/portal/jax/classFiles directory respectively.

e. Delete the existing **BrmWebServices.war** file.

f. Create a new **BrmWebServices.war** file by running the following command:

```
jar -cvf BrmWebServices.war *
```

6 Using Web Services

This chapter contains information about using Oracle Communications Billing and Revenue Management (BRM) Web Services Manager.

Before reading this chapter, you should be familiar with implementing web services.

About WSDL Files and BRM Opcodes

Web Services Manager exposes BRM opcodes as operations through different web services.

The web services included in Web Services Manager define the opcodes that can be called as Web service APIs and the attributes required to call a specific opcode. The Web service APIs (opcodes) are grouped by functional area into a Web service. For example, the **BRMBillServices** Web service defines the billing Web service APIs, and the **BRMPymtServices** Web service defines the payment Web service APIs. Web Services Manager includes one WSDL file for each Web service.

Web Services Manager contains different WSDL files for web services that supports XML string payload and for web services that supports XML element payload.

For example:

- The **BRMBalService** web service defines balances web service APIs that supports XML string payload.
- The **BRMBalService_v2** web service defines balances web service APIs that supports XML element payload. The file names of WSDL files for web services that supports XML element payload contain **_v2** as a suffix.

Note: The WSDL files and schema (XSD) files for web services that supports XML string payload are included in the **infranetwebsvc.war** file. If you customize any web services, copy the customized schema files and WSDL files to the **infranetwebsvc.war** file. The WSDL files and schema (XSD) files for web services that supports XML element payload are included in the **BrmWebServices.war** file. If you customize any web services, copy the customized schema files and WSDL files to the **BrmWebServices.war** file.

WSDLs with XML String Payload

Table 6–1 describes the web services included in Web Services Manager that supports XML string payload.

Table 6–1 Web Services Included in Web Services Manager that Supports XML String Payload

Web Service Name	Description
BRMARServices	Defines the accounts receivable web service, which includes the following opcodes: ▪ PCM_OP_AR_ACCOUNT_ADJUSTMENT ▪ PCM_OP_AR_BILL_ADJUSTMENT ▪ PCM_OP_AR_GET_ACCT_ACTION_ITEMS ▪ PCM_OP_AR_GET_ACCT_BAL_SUMMARY ▪ PCM_OP_AR_GET_ACCT_BILLS ▪ PCM_OP_AR_GET_BAL_SUMMARY ▪ PCM_OP_AR_GET_BILL_ITEMS ▪ PCM_OP_AR_ITEM_ADJUSTMENT ▪ PCM_OP_AR_EVENT_ADJUSTMENT ▪ PCM_OP_AR_GET_ACTION_ITEMS ▪ PCM_OP_AR_GET_BILLS ▪ PCM_OP_AR_RESOURCE_AGGREGATION See "Accounts Receivable FM Standard Opcodes" in *BRM Developer's Reference* for more information.
BRMBalServices	Defines the balances web service, which includes the following opcodes: ▪ PCM_OP_BAL_GET_BALANCES ▪ PCM_OP_BAL_GET_BAL_GRP_AND_SVC ▪ PCM_OP_BAL_GET_ACCT_BAL_GRP_AND_SVC ▪ PCM_OP_BAL_GET_ACCT_BILLINFO See "Balance FM Standard Opcodes" in *BRM Developer's Reference* for more information.
BRMBillServices	Defines the billing web service, which includes the following opcodes: ▪ PCM_OP_BILL_GET_ITEM_EVENT_CHARGE_ DISCOUNT ▪ PCM_OP_BILL_GROUP_MOVE_MEMBER ▪ PCM_OP_BILL_MAKE_BILL_NOW ▪ PCM_OP_BILL_DEBIT ▪ PCM_OP_BILL_GROUP_GET_PARENT See "Billing FM Standard Opcodes" in *BRM Developer's Reference* for more information.
BRMCollectionsServices	Defines the collections web service, which includes the following opcode: ▪ PCM_OP_COLLECTIONS_SET_ACTION_STATUS See "Collections Manager FM Standard Opcodes" in *BRM Developer's Reference* for more information.
BRMCustcareServices	Defines the customer care web service, which includes the following opcode: ▪ PCM_OP_CUSTCARE_MOVE_ACCT See "Customer Care FM Standard Opcodes" in *BRM Developer's Reference* for more information.

Table 6–1 (Cont.) Web Services Included in Web Services Manager that Supports XML String Payload

Web Service Name	Description
BRMCustServices	Defines the customer web service, which includes the following opcodes: ■ PCM_OP_CUST_COMMIT_CUSTOMER ■ PCM_OP_CUST_MODIFY_CUSTOMER ■ PCM_OP_CUST_UPDATE_CUSTOMER ■ PCM_OP_CUST_UPDATE_SERVICES ■ PCM_OP_CUST_DELETE_ACCT ■ PCM_OP_CUST_DELETE_PAYINFO ■ PCM_OP_CUST_CREATE_PROFILE ■ PCM_OP_CUST_MODIFY_PROFILE ■ PCM_OP_CUST_DELETE_PROFILE See "Customer FM Standard Opcodes" in *BRM Developer's Reference* for more information.
BRMInvServices	Defines the invoicing web service, which includes the following opcode: ■ PCM_OP_INV_VIEW_INVOICE **Important:** You must configure your client application to convert the invoice data received from the PCM_OP_INV_VIEW_INVOICE opcode into the appropriate format. See "About Invoicing Output XML Data" in *BRM JCA Resource Adapter*. See "Invoicing FM Standard Opcodes" in *BRM Developer's Reference* for more information.
BRMPricesServices	Defines the prices web service, which includes the following opcodes: ■ PCM_OP_PRICE_COMMIT_PRODUCT ■ PCM_OP_PRICE_COMMIT_DISCOUNT ■ PCM_OP_PRICE_SET_PRICE_LIST ■ PCM_OP_PRICE_GET_PRICE_LIST See "Price List FM Standard Opcodes" in *BRM Developer's Reference* for more information.
BRMPymtServices	Defines the payment web service, which includes the following opcode: ■ PCM_OP_PYMT_COLLECT See "Payment FM Standard Opcodes" in *BRM Developer's Reference* for more information.
BRMReadServices	Defines the read web service, which includes the following opcodes: ■ PCM_OP_READ_FLDS ■ PCM_OP_READ_OBJ ■ PCM_OP_SEARCH See "LDAP Base Opcodes" in *BRM Developer's Reference* for more information.

Table 6–1 (Cont.) Web Services Included in Web Services Manager that Supports XML String Payload

Web Service Name	Description
BRMSubscriptionServices	Defines the subscription web service, which includes the following opcodes: ▪ PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT ▪ PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT ▪ PCM_OP_SUBSCRIPTION_CANCEL_SUBSCRIPTION ▪ PCM_OP_SUBSCRIPTION_CHANGE_DEAL ▪ PCM_OP_SUBSCRIPTION_PURCHASE_DEAL ▪ PCM_OP_SUBSCRIPTION_SET_BUNDLE ▪ PCM_OP_SUBSCRIPTION_SET_DISCOUNT_STATUS ▪ PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO ▪ PCM_OP_SUBSCRIPTION_SET_PRODINFO ▪ PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS ▪ PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION ▪ PCM_OP_SUBSCRIPTION_GET_PURCHASED_ OFFERINGS See "Subscription Management FM Standard Opcodes" in *BRM Developer's Reference* for more information.

WSDLs with XML Element Payload

Web services that support XML element payload describe the input in a well-defined structure. Any standards-compliant SOAP development application can generate a client stub. The WSDL files of these web services describe the input as a well-defined message.

Table 6–2 describes the web services included in Web Services Manager that support XML element payload.

Table 6–2 Web Services Included in Web Services Manager that Support XML Element Payload

Web Service Name	Description
BRMACTServices_v2	Defines the activity web service, which includes the following opcodes: ▪ PCM_OP_ACT_ACTIVITY ▪ PCM_OP_ACT_CALC_MAX_USAGE ▪ PCM_OP_ACT_FIND ▪ PCM_OP_ACT_LOAD_SESSION See "Activity FM Standard Opcodes" in *BRM Developer's Reference* for more information.

Table 6–2 (Cont.) Web Services Included in Web Services Manager that Support XML Element Payload

Web Service Name	Description
BRMARServices_v2	Defines the accounts receivable web service, which includes the following opcodes: ■ PCM_OP_AR_ACCOUNT_ADJUSTMENT ■ PCM_OP_AR_ACCOUNT_WRITEOFF ■ PCM_OP_AR_BILL_ADJUSTMENT ■ PCM_OP_AR_BILL_DISPUTE ■ PCM_OP_AR_BILL_SETTLEMENT ■ PCM_OP_AR_BILL_WRITEOFF ■ PCM_OP_AR_BILLINFO_WRITEOFF ■ PCM_OP_AR_EVENT_ADJUSTMENT ■ PCM_OP_AR_EVENT_DISPUTE ■ PCM_OP_AR_EVENT_SETTLEMENT ■ PCM_OP_AR_GET_ACCT_ACTION_ITEMS ■ PCM_OP_AR_GET_ACCT_BAL_SUMMARY ■ PCM_OP_AR_GET_ACCT_BILLS ■ PCM_OP_AR_GET_ACTION_ITEMS ■ PCM_OP_AR_GET_BAL_SUMMARY ■ PCM_OP_AR_GET_BILLS ■ PCM_OP_AR_GET_BILL_ITEMS ■ PCM_OP_AR_GET_DISPUTES ■ PCM_OP_AR_GET_DISPUTE_DETAILS ■ PCM_OP_AR_GET_ITEMS ■ PCM_OP_AR_GET_ITEM_DETAILS ■ PCM_OP_AR_ITEM_ADJUSTMENT ■ PCM_OP_AR_ITEM_DISPUTE ■ PCM_OP_AR_ITEM_SETTLEMENT ■ PCM_OP_AR_ITEM_WRITEOFF ■ PCM_OP_AR_RESOURCE_AGGREGATION See "Accounts Receivable FM Standard Opcodes" in *BRM Developer's Reference* for more information.
BRMBALServices_v2	Defines the balances web service, which includes the following opcodes: ■ PCM_OP_BAL_CHANGE_VALIDITY ■ PCM_OP_BAL_GET_BALANCES ■ PCM_OP_BAL_GET_BAL_GRP_AND_SVC ■ PCM_OP_BAL_GET_ACCT_BAL_GRP_AND_SVC ■ PCM_OP_BAL_GET_ACCT_BILLINFO See "Balance FM Standard Opcodes" in *BRM Developer's Reference* for more information.

Table 6–2 (Cont.) Web Services Included in Web Services Manager that Support XML Element Payload

Web Service Name	Description
BRMBILLServices_v2	Defines the billing web service, which includes the following opcodes: ■ PCM_OP_BILL_DEBIT ■ PCM_OP_BILL_FIND ■ PCM_OP_BILL_GET_ITEM_EVENT_CHARGE_DISCOUNT ■ PCM_OP_BILL_GROUP_GET_PARENT ■ PCM_OP_BILL_GROUP_MOVE_MEMBER ■ PCM_OP_BILL_ITEM_EVENT_SEARCH ■ PCM_OP_BILL_ITEM_REFUND ■ PCM_OP_BILL_MAKE_BILL_NOW ■ PCM_OP_BILL_REVERSE ■ PCM_OP_BILL_SET_LIMIT_AND_CR ■ PCM_OP_BILL_VIEW_INVOICE See "Billing FM Standard Opcodes" in *BRM Developer's Reference* for more information.
BRMCOLLECTIONSServices_v2	Defines the collections web service, which includes the following opcode: ■ PCM_OP_COLLECTIONS_SET_ACTION_STATUS See "Collections Manager FM Standard Opcodes" in *BRM Developer's Reference* for more information.
BRMCUSTCAREServices_v2	Defines the customer care web service, which includes the following opcode: ■ PCM_OP_CUSTCARE_MOVE_ACCT See "Customer Care FM Standard Opcodes" in *BRM Developer's Reference* for more information.

Table 6–2 (Cont.) Web Services Included in Web Services Manager that Support XML Element Payload

Web Service Name	Description
BRMCUSTServices_v2	Defines the customer web service, which includes the following opcodes: ■ PCM_OP_CUST_COMMIT_CUSTOMER ■ PCM_OP_CUST_CREATE_PROFILE ■ PCM_OP_CUST_DELETE_ACCT ■ PCM_OP_CUST_DELETE_PAYINFO ■ PCM_OP_CUST_DELETE_PROFILE ■ PCM_OP_CUST_FIND ■ PCM_OP_CUST_FIND_PAYINFO ■ PCM_OP_CUST_FIND_PROFILE ■ PCM_OP_CUST_GET_NOTE ■ PCM_OP_CUST_MODIFY_CUSTOMER ■ PCM_OP_CUST_MODIFY_PROFILE ■ PCM_OP_CUST_SET_NOTE ■ PCM_OP_CUST_SET_STATUS ■ PCM_OP_CUST_SET_TAXINFO ■ PCM_OP_CUST_UPDATE_CUSTOMER ■ PCM_OP_CUST_UPDATE_SERVICES See "Customer FM Standard Opcodes" in *BRM Developer's Reference* for more information. ■ PCM_OP_CUST_POL_GET_PLANS ■ PCM_OP_CUST_POL_GET_DEALS ■ PCM_OP_CUST_POL_GET_PRODUCTS ■ PCM_OP_CUST_POL_READ_PLAN See "Customer FM Policy Opcodes" in *BRM Developer's Reference* for more information.
BRMINVServices_v2	Defines the invoicing web service, which includes the following opcode: ■ PCM_OP_INV_VIEW_INVOICE **Important:** You must configure your client application to convert the invoice data received from the PCM_OP_INV_VIEW_INVOICE opcode into the appropriate format. See "About Invoicing Output XML Data" in *BRM JCA Resource Adapter*. See "Invoicing FM Standard Opcodes" in *BRM Developer's Reference* for more information.

Table 6–2 (Cont.) Web Services Included in Web Services Manager that Support XML Element Payload

Web Service Name	Description
BRMPRICESServices_v2	Defines the prices web service, which includes the following opcodes: ■ PCM_OP_PRICE_COMMIT_PRODUCT ■ PCM_OP_PRICE_COMMIT_DISCOUNT ■ PCM_OP_PRICE_SET_PRICE_LIST ■ PCM_OP_PRICE_GET_DISCOUNT_INFO ■ PCM_OP_PRICE_GET_PRICE_LIST ■ PCM_OP_PRICE_GET_PRODUCT_INFO See "Price List FM Standard Opcodes" in *BRM Developer's Reference* for more information.
BRMPYMTServices_v2	Defines the payment web service, which includes the following opcode: ■ PCM_OP_PYMT_COLLECT See "Payment FM Standard Opcodes" in *BRM Developer's Reference* for more information.
BRMREADServices_v2	Defines the read web service, which includes the following opcodes: ■ PCM_OP_READ_FLDS ■ PCM_OP_READ_OBJ ■ PCM_OP_SEARCH ■ PCM_OP_TEST_LOOPBACK See "LDAP Base Opcodes" in *BRM Developer's Reference* for more information.

Table 6–2 (Cont.) Web Services Included in Web Services Manager that Support XML Element Payload

Web Service Name	Description
BRMSUBSCRIPTIONServices_v2	Defines the subscription web service, which includes the following opcodes: ■ PCM_OP_SUBSCRIPTION_CANCEL_DEAL ■ PCM_OP_SUBSCRIPTION_CANCEL_PRODUCT ■ PCM_OP_SUBSCRIPTION_CANCEL_DISCOUNT ■ PCM_OP_SUBSCRIPTION_CANCEL_SUBSCRIPTION ■ PCM_OP_SUBSCRIPTION_CHANGE_DEAL ■ PCM_OP_SUBSCRIPTION_GET_HISTORY ■ PCM_OP_SUBSCRIPTION_PURCHASE_DEAL ■ PCM_OP_SUBSCRIPTION_PURCHASE_FEES ■ PCM_OP_SUBSCRIPTION_READ_ACCT_PRODUCTS ■ PCM_OP_SUBSCRIPTION_SERVICE_BALGRP_TRANSFER ■ PCM_OP_SUBSCRIPTION_SET_BUNDLE ■ PCM_OP_SUBSCRIPTION_SET_DISCOUNT_STATUS ■ PCM_OP_SUBSCRIPTION_SET_DISCOUNTINFO ■ PCM_OP_SUBSCRIPTION_SET_PRODINFO ■ PCM_OP_SUBSCRIPTION_SET_PRODUCT_STATUS ■ PCM_OP_SUBSCRIPTION_TRANSFER_SUBSCRIPTION ■ PCM_OP_SUBSCRIPTION_TRANSITION_DEAL ■ PCM_OP_SUBSCRIPTION_TRANSITION_PLAN ■ PCM_OP_SUBSCRIPTION_GET_PURCHASED_OFFERINGS See "Subscription Management FM Standard Opcodes" in *BRM Developer's Reference* for more information.

Finding WSDL V2 Files

The file names of WSDL files for web services that supports XML element payload contain **_v2** as a suffix.

To locate the WSDL V2 files:

1. Go to the *BRM_home*/**deploy/web_services** directory.
2. Unpack the **infranetwebsvc.war** file:

   ```
   jar -xvf infranetwebsvc.war
   ```

   The **WEB-INF/services** directory is created.
3. Go to the *BRM_home*/**deploy/web_services/WEB-INF/services** directory.
4. Unpack the **InfranetWebservices.aar** file:

   ```
   jar -xvf InfranetWebservices.aar
   ```

   All WSDL V2 files are unpacked into the *BRM_home*/**deploy/web_services/WEB-INF/services/MET-INF** directory.

About Validating Input and Output XML Data

Web Services Manager validates the input and output XML by comparing the XML fields and values against the opcode XML schema.

The opcode specifications, schemas, and WSDL files are packaged along with Web Services Manager. The package includes the **opspec.xsd** file and the **pin_opspec_to_schema** utility. Use the **opspec.xsd** file to write opcode specifications for custom opcodes that need to be exposed as a Web service. Use the **pin_opspec_to_schema** utility to generate the schema files from the opcode specification files.

To configure Web Services Manager to validate the input and output XML against the target opcode XML schema:

1. Open the *local_dir*/**WEB-INF/classes/Infranet.properties** file.
2. Add the following entries to the file:
   - **webservices.input.validation.enabled=true**
   - **webservices.output.validation.enabled=true**
3. Do one of the following:
   - If you are using Oracle WebLogic Server, copy the schema files packaged as a part of Web Services Manager installation from the *BRM_home*/**deploy/web_services/schemas** directory to the *local_dir*/**common/lib** directory.
   - If you are using any supported server, copy the schema files from the *BRM_home*/**deploy/web_services/schemas** directory to the *local_dir*/**WEB-INF/classes** directory.

About Developing and Testing Client Applications with Web Services Manager

You can develop custom applications that interact with BRM through Web Services Manager. Use a SOAP development environment that supports importing WSDL files, for example SoapUI, to develop and test your custom Web service applications. SOAP development applications may have minor differences in product configuration. Consult your SOAP development application documentation for configuration information.

In general, do the following to develop and test your web services applications:

1. Download and install a SOAP development application.
2. Configure a new project in your SOAP development application.
3. Write a client application that communicates with web services using the SOAP protocol.
4. Import the Web service definitions using the WSDL files. See "About WSDL Files and BRM Opcodes" for more information on accessing WSDL files.
5. Run the required commands to set up your application server environment.
6. Configure the properties of the web services operations in your SOAP development environment with valid credentials.
7. Send a Web service request to BRM from the SOAP development environment client.
8. View the Web service response in the SOAP development environment.

Example of a Testing a Web Services Implementation Using a Client Application

To test your web services implementation, write a client application that communicates with the Web service using the SOAP protocol.

The sample procedures use WebLogic Server, but you can apply the concepts to any other supported application server.

This sample procedure demonstrates how to use the **TestClient.java** sample code with the PCM_OP_TEST_LOOPBACK opcode to verify communication between BRM and the Web service.

To test your web services implementation using a client application:

1. Set up the WebLogic Server environment by running one of the following commands:
   - On UNIX: *WebLogic_home*/**wlserver/server/bin/setWLSEnv.sh**
   - On Windows: *WebLogic_home*/**server/bin/setenv.exe**

     where *WebLogic_home* is the directory in which WebLogic Server is installed.
2. Create an XML file (*some_name_1*.**xml**) using the following text:

```
<project name="buildWebservice" default="all">
<property name="buildDir" value="./myapps" />
<property name="jarFiles" value="jars" />
<target name="all" depends="jar" description="builds everything">
</target>
<target name="generate-client">
<clientgen
wsdl="http://198.51.100.1:7001/infranetwebsvc/services/Infranet?wsdl"
packageName="test_client"
destDir= "./myapps"/>
</target>
<target name="compile" depends="generate-client" description="compile source
files">
    <echo> Compiling adapter files</echo>
    <javac  destdir="${buildDir}">
        <src path="${buildDir}"/>
    </javac>
</target>
<target name="jar" depends="compile" description="generate jar file(s)">
    <jar jarfile="clientStub.jar"  basedir="${buildDir}">
    <exclude name = "**/*.java"/>

    </jar>
</target>
<target name="clean" description="remove files created by target prepare">
    <delete dir="${buildDir}"/>
</target>
</project>
```

   This XML file uses the WebLogic Server clientgen task to automatically generate a utility library that provides low-level SOAP communication (client stubs).
3. Run the following command, which creates the client stubs:

```
ant -file some_name_1.xml
```

   This process generates the **clientstubs.jar** file, which contains stubs used by the client. The test client code (*Source_home*/**TestClient.java**, where *Source_home* is the directory in which your source code files are stored) then creates an flist, converts it to XML, and calls the PCM_OP_TEST_LOOPBACK opcode.

The following is a sample listing of **TestClient.java**:

```
import java.io.IOException;
import test_client.*;  // corresponds to package name clientgen generated

public class TestClient {

    public static void main(String[] args) {
        try {

            String wsdlUrl =
"http://198.51.100.1:7001/infranetwebsvc/services/Infranet?wsdl";
            InfranetWebServiceService service = new InfranetServiceService_
Impl( wsdlUrl );
            InfranetWebService port = service.getInfranet();

            // convert flist to XML representation
            String XMLInput="<flist
xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"><POID>0.0.0.1 /account
80408 8</POID></flist>";
            System.out.println("Input: " + XMLInput);
            // invoke web service 'opcode' method
            String result = port.opcode("TEST_LOOPBACK", XMLInput);

            System.out.println("result: "+ result);

        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

4. Create another XML file (*some_name_2*.**xml**) using the following text:

   Note: Replace the paths for the JAR files as required.

```
<project name="test_client" default="all">

<target name="all" depends="run"/>

<path id="classpath">
    <pathelement path="clientstubs.jar"/>
    <pathelement path=".\classes"/>
    <pathelement path="D:\WebServices\webman\lib\jaxrpc.jar"/>
    <pathelement path="D:\bea\wlserver_10.3\server\lib\wseeclient.jar"/>
    <pathelement path="D:\bea\wlserver_10.3\server\lib\weblogic.jar"/>
</path>

c
<target name="compile">
    <mkdir dir="classes"/>
    <javac  srcdir="src"
        destdir="classes"
        classpathref="classpath"
    />
```

```
</target>

<target name="run" depends="compile">
    <java classname="TestClient"
            fork="yes"
            classpathref="classpath">
    </java>
</target>

</project>
```

5. Build and run the test with the *some_name_2***.xml** file using regular Ant tasks:

   ```
   ant -file some_name_2.xml
   ```

Testing the Web Service

To test your web services implementation, use a SOAP development application that supports importing WSDL files: for example, SoapUI. SOAP development applications may have minor differences in project configuration. Consult your SOAP development environment documentation for configuration information.

To test your web services implementation, write a client application that communicates with the Web service using the SOAP protocol.

Note: You can test web services using SoapUI version 4.6.3.

The sample procedures use WebLogic Server, but you can apply the concepts to any other supported server.

To test your Web service:

1. Set up the WebLogic Server environment by running one of the following commands:

   - On UNIX: *WebLogic_home***/wlserver/server/bin/setWLSEnv.sh**
   - On Windows: *WebLogic_home***/server/bin/setenv.exe**

     where *WebLogic_home* is the directory in which WebLogic Server is installed.

2. Create an XML file (*some_name_1***.xml**) using the following text:

   ```
   <project name="buildWebservice" default="all">
   <property name="buildDir" value="./myapps" />
   <property name="jarFiles" value="jars" />
   <target name="all" depends="jar" description="builds everything">
   </target>
   <target name="generate-client">
   <clientgen
   wsdl="http://198.51.100.1:7001/infranetwebsvc/services/Infranet?wsdl"
   packageName="test_client"
   destDir= "./myapps"/>
   </target>
   <target name="compile" depends="generate-client" description="compile source
   files">
       <echo> Compiling adapter files</echo>
       <javac  destdir="${buildDir}">
           <src path="${buildDir}"/>
       </javac>
   </target>
   ```

```
<target name="jar" depends="compile" description="generate jar file(s)">
    <jar jarfile="clientStub.jar"  basedir="${buildDir}">
    <exclude name = "**/*.java"/>

    </jar>
</target>
<target name="clean" description="remove files created by target prepare">
    <delete dir="${buildDir}"/>
</target>
</project>
```

This XML file uses the WebLogic Server clientgen task to automatically generate a utility library that provides low-level SOAP communication (client stubs).

3. Run the following command, which creates the client stubs:

   ```
   ant -file some_name_1.xml
   ```

   This process generates the **clientstubs.jar** file, which contains stubs used by the client. The test client code (**src\TestClient.java**) then creates an flist, converts it to XML, and calls the PCM_OP_TEST_LOOPBACK opcode.

   The following is a sample listing of **TestClient.java**:

```
import java.io.IOException;
import test_client.*;  // corresponds to package name clientgen generated

public class TestClient {

    public static void main(String[] args) {
        try {

            String wsdlUrl =
"http://198.51.100.1:7001/infranetwebsvc/services/Infranet?wsdl";
            InfranetWebServiceService service = new InfranetServiceService_
Impl( wsdlUrl );
            InfranetWebService port = service.getInfranet();

            // convert flist to XML representation
            String XMLInput="<flist
xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"><POID>0.0.0.1 /account
80408 8</POID></flist>";
            System.out.println("Input: " + XMLInput);
            // invoke web service ‘opcode’ method
            String result = port.opcode("TEST_LOOPBACK", XMLInput);

            System.out.println("result: "+ result);

        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

4. Create another XML file (*some_name_2*.**xml**) using the following text:

   Note: Replace the paths for the JAR files as required.

```
<project name="test_client" default="all">
```

```
<target name="all" depends="run"/>

<path id="classpath">
    <pathelement path="clientstubs.jar"/>
    <pathelement path=".\classes"/>
    <pathelement path="D:\WebServices\webman\lib\jaxrpc.jar"/>
    <pathelement path="D:\bea\wlserver_10.3\server\lib\wseeclient.jar"/>
    <pathelement path="D:\bea\wlserver_10.3\server\lib\weblogic.jar"/>
</path>

c
<target name="compile">
    <mkdir dir="classes"/>
    <javac  srcdir="src"
        destdir="classes"
        classpathref="classpath"
    />
</target>

<target name="run" depends="compile">
    <java classname="TestClient"
            fork="yes"
            classpathref="classpath">
    </java>
</target>

</project>
```

5. Build and run the test with the *some_name_2*.**xml** file using regular Ant tasks:

   ```
   ant -file some_name_2.xml
   ```

Using Sample Web Services Programs

Web Services Manager includes sample programs that demonstrate how to write code for various tasks when customizing web services. For example, the sample program **InfranetBALTestClient.java** creates an flist, converts it to XML, and calls the PCM_OP_BAL_GET_BALANCES opcode.

Example of Creating an Account in BRM Using Web Services

This section describes an example of creating a customer account using web services. The example shows a sample SOAP request and a response message for creating a customer account in BRM by calling a Web service in Web Services Manager.

To create an account in BRM using web services, call the **pcmOpCustCommitCustomer** Web service API which maps to the PCM_OP_CUST_COMMIT_CUSTOMER opcode. The **pcmOpCustCommitCustomer** Web service API is included in the BRMCUSTServices_v2 Web service. The BRMCUSTServices_v2 Web service contains Web service APIs that are related to customer accounts. See "About WSDL Files and BRM Opcodes" for more information about the web services included in the Web Services Manager package.

You use URLs to create SOAP clients for web services. The URL to create a SOAP client is generated by the JAX-WS in WebLogic Server.

To generate the URL for a Web service:

1. Log in to WebLogic Server Administration Console.

2. Go to the **Deployments** section.
3. Click the **BrmWebServices** link.

   The Settings page for the deployment appears and all the web services are listed in the **Modules and Components** section.
4. In the **Modules and Components** section, click the Web service.
5. Click the **Testing** tab.
6. WebLogic Server Administration Console displays a URL for the BRM web services.

A sample URL for the BRMCUSTServices_v2 Web service is as follows:

```
http://hostIPAddress:port/BrmWebServices/BRMCUSTServices_v2?wsdl
```

To call a Web service, you are required to authenticate using a valid user name and a password. Users can call only those web services that they are authorized to call.

Sample SOAP Request Input XML File

The following sample shows a SOAP request for the **pcmOpCustCommitCustomer** Web service API.

```
- <soapenv:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"xmlns:xsd="http://www.w3.org/
2001/XMLSchema"
   xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:bus="http://xmlns.oracle.com/BRM/schemas/BusinessOpcodes">
   <soapenv:Header />
 - <soapenv:Body>
  - <bus:pcmOpCustCommitCustomer>
   - <PCM_OP_CUST_COMMIT_CUSTOMER_Request xsi:type="bus:PCM_OP_CUST_COMMIT_
CUSTOMER_RequestType">
       <flags xsi:type="xsd:int">1</flags>
     - <PCM_OP_CUST_COMMIT_CUSTOMER_inputFlist
xmlns="http://xmlns.oracle.com/BRM/schemas/BusinessOpcodes">
       - <ACCTINFO elem="0">
           <ACCOUNT_NO>a022020202011992</ACCOUNT_NO>
           <BAL_INFO />
           <BUSINESS_TYPE>1</BUSINESS_TYPE>
           <CURRENCY>840</CURRENCY>
           <POID>0.0.0.1 /account -1 0</POID>
         </ACCTINFO>
     - <BAL_INFO elem="0">
         <BILLINFO />
       - <LIMIT elem="840">
           <CREDIT_LIMIT>"0"</CREDIT_LIMIT>
         </LIMIT>
       <NAME>Account Level Balance Group</NAME>
       <POID>0.0.0.1 /balance_group -1 0</POID>
     </BAL_INFO>
 - <BILLINFO elem="0">
     <BAL_INFO />
     <BILL_WHEN>1</BILL_WHEN>
     <BILLINFO_ID>88-CYZZ5</BILLINFO_ID>
     <CURRENCY>840</CURRENCY>
     <PAY_TYPE>10001</PAY_TYPE>
     <PAYINFO />
     <POID>0.0.0.1 /billinfo -1 0</POID>
```

```
      </BILLINFO>
      <END_T>2010-02-17T22:37:49</END_T>
      <FLAGS>0</FLAGS>
    - <LOCALES elem="1">
        <LOCALE>en_US</LOCALE>
     </LOCALES>
    - <NAMEINFO elem="1">
        <ADDRESS>123 Hollywood Boulevard</ADDRESS>
        <CITY>Los Angeles</CITY>
        <CONTACT_TYPE>Account holder</CONTACT_TYPE>
        <COUNTRY>USA</COUNTRY>
        <EMAIL_ADDR>test_001</EMAIL_ADDR>
        <FIRST_NAME>Chetn3457</FIRST_NAME>
        <LAST_NAME>Chet8905</LAST_NAME>
        <STATE>NJ</STATE>
        <ZIP>90001</ZIP>
      </NAMEINFO>
    - <PAYINFO elem="0">
         - <INHERITED_INFO>
             - <INV_INFO elem="0">
                  <ADDRESS>123 Hollywood Boulevard</ADDRESS>
                  <CITY>Los Angeles</CITY>
                  <COUNTRY>USA</COUNTRY>
                  <DELIVERY_DESCR>test_001</DELIVERY_DESCR>
                  <DELIVERY_PREFER>0</DELIVERY_PREFER>
                  <EMAIL_ADDR />
                <INV_TERMS>0</INV_TERMS>
              <NAME>Chet3457 Chet8905</NAME>
            <STATE>NJ</STATE>
          <ZIP>90001</ZIP>
         </INV_INFO>
        </INHERITED_INFO>
        <INV_TYPE>0</INV_TYPE>
        <PAY_TYPE>10001</PAY_TYPE>
        <POID>0.0.0.1 /payinfo/invoice -1 0</POID>
        </PAYINFO>
        <POID>0.0.0.1 /plan -1 0</POID>
       </PCM_OP_CUST_COMMIT_CUSTOMER_inputFlist>
      </PCM_OP_CUST_COMMIT_CUSTOMER_Request>
     </bus:pcmOpCustCommitCustomer>
    </soapenv:Body>
  </soapenv:Envelope>
```

Sample SOAP Response Output XML File

The following sample shows a SOAP response message for the **pcmOpCustCommitCustomer** Web service API.

```
- <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
 - <soapenv:Body>
  - <brm:PCM_OP_CUST_COMMIT_CUSTOMER_outputFlist
xmlns:brm="http://xmlns.oracle.com/BRM/schemas/BusinessOpcodes">
     <brm:ACCOUNT_OBJ>0.0.0.1 /account 225757 0</brm:ACCOUNT_OBJ>
   - <brm:ACCTINFO elem="0">
      <brm:ACCOUNT_NO>a022020202011992</brm:ACCOUNT_NO>
      <brm:BAL_INFO elem="0" />
      <brm:BUSINESS_TYPE>1</brm:BUSINESS_TYPE>
      <brm:CURRENCY>840</brm:CURRENCY>
      <brm:POID>0.0.0.1 /account -1 0</brm:POID>
    </brm:ACCTINFO>
```

```
- <brm:BAL_INFO elem="0">
   <brm:ACCOUNT_OBJ>0.0.0.1 /account 225757 0</brm:ACCOUNT_OBJ>
   <brm:BILLINFO_OBJ>0.0.0.1 /billinfo 226269 0</brm:BILLINFO_OBJ>
   - <brm:LIMIT elem="840">
      <brm:CREDIT_LIMIT />
    </brm:LIMIT>
    <brm:NAME>Account Level Balance Group</brm:NAME>
    <brm:POID>0.0.0.1 /balance_group 225341 0</brm:POID>
    <brm:SERVICE_OBJ>0.0.0.0 0 0</brm:SERVICE_OBJ>
    </brm:BAL_INFO>
  - <brm:BILLINFO elem="0">
      <brm:BAL_GRP_OBJ>0.0.0.1 /balance_group 225341 0</brm:BAL_GRP_OBJ>
      <brm:BILLINFO_ID>88-CYZZ5</brm:BILLINFO_ID>
      <brm:BILL_WHEN>1</brm:BILL_WHEN>
      <brm:CURRENCY>840</brm:CURRENCY>
      <brm:CURRENCY_SECONDARY>0</brm:CURRENCY_SECONDARY>
      <brm:EFFECTIVE_T>2010-02-17T22:37:49Z</brm:EFFECTIVE_T>
      <brm:PAYINFO_OBJ>0.0.0.1 /payinfo/invoice 226781 0</brm:PAYINFO_OBJ>
      <brm:PAY_TYPE>10001</brm:PAY_TYPE>
      <brm:POID>0.0.0.1 /billinfo 226269 0</brm:POID>
    </brm:BILLINFO>
    <brm:END_T>2010-02-17T22:37:49Z</brm:END_T>
    <brm:FLAGS>0</brm:FLAGS>
    <brm:GROUP_INFO />
  - <brm:HOST elem="1">
      <brm:HOSTNAME>XXX.XXX.XXX.XXX</brm:HOSTNAME>
      <brm:TYPE>1</brm:TYPE>
    </brm:HOST>
  - <brm:HOST elem="2">
      <brm:HOSTNAME>XXX.XXX.XXX.XXX</brm:HOSTNAME>
      <brm:TYPE>1</brm:TYPE>
   </brm:HOST>
 - <brm:HOST elem="3">
      <brm:HOSTNAME>XXXXXXXXX.XXX</brm:HOSTNAME>
      <brm:PORT>0</brm:PORT>
      <brm:TYPE>2</brm:TYPE>
   </brm:HOST>
 - <brm:HOST elem="4">
      <brm:HOSTNAME>XXXX.XXX</brm:HOSTNAME>
      <brm:TYPE>3</brm:TYPE>
  </brm:HOST>
- <brm:HOST elem="5">
      <brm:HOSTNAME>XXXX.XXX</brm:HOSTNAME>
      <brm:TYPE>4</brm:TYPE>
  </brm:HOST>
  <brm:HTTP_URL>XXXXXXXXXXXXXXX</brm:HTTP_URL>
- <brm:LOCALES elem="1">
    <brm:LOCALE>en_US</brm:LOCALE>
  </brm:LOCALES>
- <brm:NAMEINFO elem="1">
    <brm:ADDRESS>123 Hollywood Boulevard</brm:ADDRESS>
    <brm:CANON_COUNTRY>US</brm:CANON_COUNTRY>
    <brm:CITY>Los Angeles</brm:CITY>
    <brm:COMPANY />
    <brm:CONTACT_TYPE>Account holder</brm:CONTACT_TYPE>
    <brm:COUNTRY>USA</brm:COUNTRY>
    <brm:ELEMENT_ID>1</brm:ELEMENT_ID>
    <brm:EMAIL_ADDR>test_001</brm:EMAIL_ADDR>
    <brm:FIRST_NAME>Chetn3457</brm:FIRST_NAME>
    <brm:LAST_NAME>Chet8905</brm:LAST_NAME>
```

```
      <brm:MIDDLE_NAME />
      <brm:SALUTATION />
      <brm:STATE>NJ</brm:STATE>
      <brm:TITLE />
      <brm:ZIP>90001</brm:ZIP>
     </brm:NAMEINFO>
   - <brm:PAYINFO elem="0">
    - <brm:INHERITED_INFO>
     - <brm:INV_INFO elem="0">
        <brm:ADDRESS>123 Hollywood Boulevard</brm:ADDRESS>
        <brm:CITY>Los Angeles</brm:CITY>
        <brm:COUNTRY>USA</brm:COUNTRY>
        <brm:DELIVERY_DESCR>test_001</brm:DELIVERY_DESCR>
        <brm:DELIVERY_PREFER>0</brm:DELIVERY_PREFER>
        <brm:EMAIL_ADDR />
        <brm:INV_TERMS>0</brm:INV_TERMS>
        <brm:NAME>Chet3457 Chet8905</brm:NAME>
        <brm:STATE>NJ</brm:STATE>
        <brm:ZIP>90001</brm:ZIP>
       </brm:INV_INFO>
      </brm:INHERITED_INFO>
    <brm:INV_TYPE>0</brm:INV_TYPE>
    <brm:PAY_TYPE>10001</brm:PAY_TYPE>
    <brm:POID>0.0.0.1 /payinfo/invoice 226781 0</brm:POID>
    </brm:PAYINFO>
    <brm:POID>0.0.0.1 /plan -1 0</brm:POID>
    <brm:START_T>2014-05-07T06:00:09Z</brm:START_T>
    <brm:SUPPORT_PHONE>XXXXXXXXXXXXXXX</brm:SUPPORT_PHONE>
   </brm:PCM_OP_CUST_COMMIT_CUSTOMER_outputFlist>
  </soapenv:Body>
</soapenv:Envelope>
```

About Masked Fields in Web Services Responses

SOAP output response XML files may contain masked fields as configured by your BRM implementation. Subscriber fields, including payment information and user credentials, may be hidden in responses for securing sensitive subscriber data.

See "About Securing Sensitive Customer Data with Masking" in *BRM Managing Customers* for more information on configuring data masking.