Oracle® Communications Billing and Revenue Management

Security Guide

Release 15.2
G35874-01
January 2026

ORACLE

Oracle Communications Billing and Revenue Management Security Guide, Release 15.2

G35874-01

Contents

9 BRM REST Services Manager Security

10 PDC REST Services Manager Security

11 ECE REST API Security

12 Business Operations Center Security

About This Content

This guide provides guidelines and recommendations for managing security in Oracle Communications Billing and Revenue Management (BRM), Oracle Communications Billing and Revenue Management Elastic Charging Engine (ECE), Oracle Communications Pricing Design Center (PDC), Oracle Communications Billing Care, and Business Operations Center.

Audience

This guide is intended for business analysts, developers, and system administrators.

1

BRM Security Overview

Learn about security in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Basic Security Considerations

The following principles are fundamental to using any application securely:

- **Keep software up to date.** This includes the latest product release and any patches that apply to it.

  The system administrator is responsible for ensuring that all installed software dependencies are kept up to date wherever possible. Oracle supports software versions when the software vendor declares backward compatibility with the version certified with BRM, and when the upgrade is a minor version increment (for example, A.B.C to A.B.D).

  System administrators need to adopt this policy for software dependencies for which upgrades are generally related to security, rather than focusing on functionality.

  Review the latest product release documentation for any new guidelines to follow.

- **Restrict network access to critical services.** Keep the BRM Business Process, Data Management, and Data tiers behind a firewall. The firewall provides assurance that access to these systems is restricted to a known network route, which can be monitored and restricted if necessary. Alternatively, a firewall router can substitute for multiple independent firewalls.

  Oracle does not recommend placing a firewall between the Data Management and Data tiers, because the connection between these tiers is persistent; as such, it is vital that a firewall not terminate the connection after a period of time.

  Configure the TNS Listener Valid Node Checking feature, which restricts access based on IP address. However, restricting database access by IP address can cause application client/server programs to fail for DHCP clients. To resolve this, consider using static IP addresses, a software/hardware VPN, or Windows Terminal Services or an equivalent solution.

- **Limit privileges as much as possible.** Give users only as much access as necessary to perform their work. Review user privileges regularly to determine relevance to current work requirements.

- **Monitor system activity.** Ensuring system security requires good security protocols, proper system configuration, and system monitoring. Establish who should access which system components and how often, and monitor those components.

See "Configuring and Using Security Audit" for more information.

- **Install software securely.** For example, use firewalls, secure protocols such as Secure Sockets Layer (SSL), Transport Layer Security (TLS), and secure passwords.

  See "Performing a Secure BRM Installation" for more information.

- **Learn and use the BRM security features.** See "Managing BRM Security".
- **Keep up to date on security information.** Oracle regularly issues security-related patch updates and security alerts. You must install all security patches as soon as possible.

  See *Critical Patch Updates, Security Alerts and Bulletins* on the Oracle website.

About Protecting Data

As you plan your BRM implementation, consider the following:

- **Which resources need to be protected?**

  Many resources in the production environment can be protected, including information in databases accessed by BRM and the availability, performance, applications, and the integrity of the BRM architecture. Consider the resources you want to protect when deciding the level of security you must provide.

  - You need to protect customer data, such as credit card numbers.
  - You need to protect internal data, such as proprietary source code.
  - You need to protect system components from being disabled by external attacks or intentional system overloads.

- **Who are you protecting data from?**

  For all BRM implementations, resources must be protected from everyone on the Internet. However, what data should also be protected from employees on your enterprise intranet? What data should be accessible to a system administrator?

  For example, you need to protect your subscribers' data from other subscribers, but someone in your organization might need to access that data to manage it. You can analyze your workflow to determine who needs access to the data; for example, a system administrator may be able to manage your system components without needing to access the system data.

- **What happens if protections on strategic resources fail?**

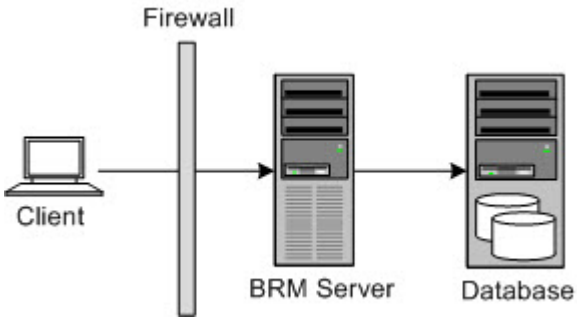
  In some cases, a fault in your security scheme is nothing more than an inconvenience. In other cases, a fault might cause great damage to you or your customers. Understanding the security ramifications of each resource helps you protect it properly.

Recommended Deployment Configurations

This section describes recommended deployment configurations for BRM.

Figure 1-1 shows the general architectural recommendation using the well-known and generally accepted trusted network.

Figure 1-1 Traditional Trusted Network View



Firewalls separating and protecting trusted networks provide two essential functions:

- Blocking any traffic types known to be illegal
- Providing intrusion containment if successful intrusions take over processes or processors.

> **Note**
>
> Oracle does not recommend having a second firewall between the BRM server and the database server.

Operating System Security

To secure your operating system, see the following documents:

- Guide to the Secure Configuration of Red Hat Enterprise Linux 8
- Security Hardening Red Hat Enterprise Linux 8

Oracle Database Security

To secure your database, see Oracle Database Security Guide.

2

Performing a Secure BRM Installation

Learn how to install Oracle Communications Billing and Revenue Management (BRM) securely.

Topics in this document:

For information about installing BRM, see "Installing BRM" in *BRM Installation Guide*.

Preinstallation Tasks

Perform the following preinstallation tasks:

- The target operating system for BRM should have a default configuration with the following differences:
  - Do not disable X Windows. It is required for local administration and is useful for troubleshooting.
  - Do not disable SSH.
  - By default, the application uses the following ports. Ensure that iptables is configured to allow traffic to these ports and that any unused ports are closed:
    - 22 in both directions – used for SSH access
    - 80 in both directions – if using HTTP
    - 443 in both directions – if using HTTPS
  - Additional ports may need to be opened, depending on the ports specified for BRM during the installation process.
- Configure Oracle Database advanced security encryption and integrity algorithms for a secure connection from the installer. See the Oracle Database documentation for advanced security configuration parameters. This is required for the BRM installer to make a secured (encrypted) database connection over the network. For more details, see *Oracle Database Advanced Security Administrator's Guide* at http://docs.oracle.com.
- Install only the required components. This is true of both the BRM components and any third-party software that is required, such as the operating system and the database. This can be achieved by conducting a custom install and selecting only the required components, or by removing any extraneous components as a postinstallation step.
- Install all third-party software according to the security advice provided by the vendor. In particular, avoid default values for data such as user names, passwords, and port numbers whenever possible by selecting different values during installation or immediately changing them as a postinstallation step.

Installing BRM Securely

Follow the steps in *BRM Installation Guide* to install BRM. However, the port numbers, username, password, and database SID should be changed from the default values.

The user name selected must be for an account used only for BRM and must not have unnecessary privileges for any other software. In particular, the account should not have root access privileges.

Postinstallation Tasks

Perform the following tasks after installing BRM:

- Lock and Expire Default User Accounts
- Change Default User Passwords
- Use Strong Passwords for BRM User Schema
- Enable SSL/TLS for SQL*NET
- Use Secure TLS Connections
- Enforce Password Management
- Tighten File Permissions
- Configure Maximum Number of Invalid Login Attempts
- Log Customer Service Representative Activities
- Integrate Paymentech

Lock and Expire Default User Accounts

Oracle Database installs with many default (preset) database server user accounts. Upon the successful creation of a database server instance, the Database Configuration Assistant automatically locks and expires most of the default database user accounts.

> **Note**
>
> If you use Oracle Universal Installer or Database Configuration Assistant, you are prompted for new SYS and SYSTEM passwords.

After the database is installed, lock the SYS and SYSTEM accounts, and use AS SYSDBA for administrator access. Specify administrative passwords individually.

This account (AS SYSDBA) tracks the operating system user name, maintaining accountability. If you need access only for database startup and shutdown, use AS SYSOPER instead. SYSOPER has fewer administrative privileges than SYS, but enough to perform basic operations such as startup, shutdown, mounting, backup, archiving, and recovery.

Change Default User Passwords

Security is most easily compromised when a default database server user account still has its default password after installation. The following steps address this:

- Change the default passwords of administrative users immediately after installing the database server.
- In any Oracle environment (production or test), assign strong, secure passwords to the SYS and SYSTEM user accounts immediately upon successful installation of the database server. Do not allow the passwords for SYS and SYSTEM to retain their default values. Similarly, in production environments, do not use default passwords for any administrative accounts, including SYSMAN and DBSNMP.

Use Strong Passwords for BRM User Schema

BRM requires one or more database users and database schemas to store subscriber data. You must assign unique and complex passwords for each user and grant enough database privileges to perform the required BRM operations.

Enable SSL/TLS for SQL*NET

Configure Oracle Database to communicate over secure sockets layer (SSL) or transport layer security (TLS) channels to secure the data transmitted between the BRM server and the Oracle database.

Use Secure TLS Connections

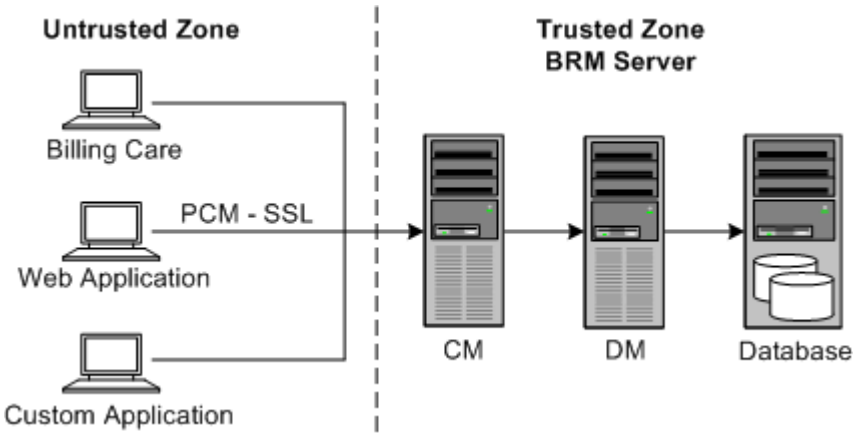
You can configure BRM to communicate between the various components using encrypted TLS sockets by setting the **enable_ssl** entry in the CM's **pin.conf** configuration file. When this configuration is enabled, BRM uses TLS sockets for any communication between its components, such as Oracle Data Manager (DM) (**dm_oracle**), Payload Generator EM (also called the EAI Java Server or **eai_js**), or Paymentech DM (**dm_fusa**).

For example, you can configure BRM client applications or any client tier module, such as Web Services Manager or JCA Resource Adapter, to use encrypted TLS sockets to connect to the BRM server.

BRM provides sample CA certificates and trusted client certificates. You must replace the sample CA certificate with your own CA certificate or use a CA certificate from a third party.

Figure 2-1 shows secure communications between BRM components using TLS.

Figure 2-1 Secure Communications Using TLS



See "Enabling Secure Communication between BRM Components" in *BRM System Administrator's Guide* for more information.

Enforce Password Management

You must apply basic password management rules, such as password length, history, and complexity, to all user passwords.

You can configure complex rules by modifying the PCM_OP_CUST_POL_VALID_PASSWD policy opcode. See "Customizing Passwords" in *BRM Opcode Guide* for more information.

Tighten File Permissions

You must ensure that all installed files have their permissions tightened as much as possible without impacting the operation of the software.

Configure Maximum Number of Invalid Login Attempts

You must set the **MaxLoginAttempts** parameter in the **bus_params_act.xml** configuration file to a value that aligns with your enterprise's internal security policies. Its default value is **5**.

The **pin_bus_params** utility is used to apply any changes to this configuration file.

See "Configuring the Maximum Number of Invalid Login Attempts" in *BRM System Administrator's Guide* for more information.

Log Customer Service Representative Activities

Customer service representatives (CSRs) require special privileges to carry out their roles. It is important to monitor their activities to ensure that they are not abusing those privileges.

CSR activities are logged as part of BRM's session event logging functionality that can be enabled by setting the **login_audit** entry in the CM's **pin.conf** configuration file to **1**. The **pin_notify** configuration file lists all activities that are logged.

The **pin_load_notify** utility is used to apply any changes to this configuration file.

See "Logging Customer Service Representative Activity Events" in *BRM System Administrator's Guide* for more information.

Integrate Paymentech

If the BRM installation is integrated with Paymentech through the **dm_fusa** component, Oracle recommends that the connection between BRM and Paymentech is protected using a VPN. This encrypts the sensitive customer data communicated between the two platforms and protects it from any snooping attempts.

3

Performing a Secure Pricing Design Center Installation

Learn about the recommended deployment configurations for your Oracle Communications Pricing Design Center (PDC) installation that enhance security.

Topics in this document:

Recommended Installation Mode

There are two types of installation modes: silent and secured.

Silent installation is not meant for production environments and should be used only in test environments for quick setup or for backing up properties for use in another test environment.

Secured installation is the only recommended option for production environments.

Operating System Security

PDC is supported on Linux (both Oracle Enterprise Linux and Red Hat Enterprise Linux) and Windows. For the supported versions, see "PDC Software Compatibility" in *BRM Compatibility Matrix*. See the following documents for more information about operating system security:

- *Guide to the Secure Configuration of Red Hat Enterprise Linux*
- *Hardening Tips for the Red Hat Enterprise Linux*

Preinstallation Tasks

Perform the following preinstallation tasks:

- Enable SSL for the target WebLogic server domain, configure the server KeyStore certificate, and then get the client KeyStore trusted certificate. This client KeyStore file should be used in the installer to establish a secure connection during installation.
- If SSL is enabled, ensure that the KeyStore file is created in a secure drive and that access is strictly limited to the user account.
- Configure Oracle Database advanced security encryption and integrity algorithms for a secure connection from the installer. See the Oracle Database documentation for advanced security configuration parameters. This is required for the PDC installer to establish a secure (encrypted) database connection over the network. For more details, see the *Oracle Database Advanced Security Administrator's Guide* documentation.
- Verify that you have the latest supported version of Oracle JDK installed.

Installation Tasks

Perform the following installation tasks:

- During PDC installation, select SSL mode and provide the client KeyStore certificate for connecting to the WebLogic Server over SSL.
- The following logs are generated after the PDC installation:

  Location: *Oracle Inventory***logs/**

```
-rw-r-----  1 user1 eng  480058 Aug 15 09:25 installActions2018-08-15_08-06-57AM.log
-rw-r-----  1 user1 eng    2384 Aug 15 10:33 dbScripts2018-08-15_10-32-00AM.log
-rw-r-----  1 user1 eng  124268 Aug 15 10:33 oraInstall2018-08-15_10-27-07AM.err
```

  The **installActions***xxxxx***.log** and **oraInstall***xxxx***.err** files have details in clear text format that were entered in the PDC installation wizard. Passwords that were entered in the installation wizard are not logged in any of the PDC installation log files. Delete these installation log files if they are not needed for future reference. If they are required, protect them appropriately. By default, these log files are created with file permission 640 (owner can read/write, group can read, others have no permission).

Postinstallation Configuration

- PDC user permissions depend on the group the user belongs to. The following three groups are created in the WebLogic server during PDC installation:
  - Pricing Design Admin
  - Pricing Reviewer
  - Pricing Analyst

  Users belonging to the Pricing Design Admin group have read and write access and can perform any kind of operation using the PDC User Interface.

  Users belonging to the Pricing Analyst group have read and write access to all pricing components and read-only access to setup components.

  Users belonging to the Pricing Reviewer group have read-only access to the pricing and setup components.

  By default, none of the users is authorized to access PDC. The WebLogic server administrator must create an account for each intended user by creating the user in the Oracle WebLogic Remote Console and adding the user to one of the above groups depending on the user role.

- Do not use your browser's remember password feature for the WebLogic Remote Console URL. Always enter the WebLogic server user name and password manually on the login page as a precaution.

Using Secure Cookies

> **Note**
>
> Oracle recommends deploying PDC only on SSL, which encrypts sensitive data, thus eliminating problems like session stealing.

A common web security issue is session stealing, which occurs when an attacker obtains a copy of your session cookie, usually while it is being transmitted over the network. This can only happen when the data is being sent in clear-text; that is, the cookie is not encrypted.

WebLogic Server allows users to securely access HTTPS resources in a session that was initiated using HTTP, without loss of session data.

To use secure cookies:

1. Enable cookie authentication in WebLogic Server. See "Enabling Authentication Cookies".
2. Update your PDC deployment plan to use secure cookies. See "Updating Your PDC Deployment Plan".

Enabling Authentication Cookies

You can enable cookie authentication in two different ways: by editing the **config.xml** file or using the WebLogic Remote Console.

To enable cookie authentication through the **config.xml** file:

1. Open the **config.xml** file.
2. Add **AuthCookieEnabled="true"** to the WebServer element.

```
<WebServer Name="myserver" AuthCookieEnabled="true"/>
```

To enable cookie authentication using the WebLogic Remote Console:

1. Log in to the WebLogic Remote Console.
2. Click **Edit Tree**, then **Environment**, and then **Domain**.
3. Click the **Web Application** tab.
4. Verify that **Auth Cookie Enabled** is turned on.
5. Click **Save**.

By default, **Auth Cookie Enabled** is turned on, but it is not present in the **config.xml** file. If you turn it off, the **<AuthCookieEnabled>** element is added to the **config.xml** file.

Setting **AuthCookieEnabled** to **true**, which is the default setting, causes the WebLogic Server instance to send a new secure cookie, _WL_AUTHCOOKIE_JSESSIONID, to the browser when authenticating through an HTTPS connection. After the secure cookie is set, the session is allowed to access other security-constrained HTTPS resources only if the cookie is sent from the browser.

Oracle recommends keeping cookie settings enabled in the browser. Disabling cookies in the browser also disables several features, such as Help.

Updating Your PDC Deployment Plan

To update your PDC deployment plan to use secure cookies:

1. Open the *PDC_home*/**setup/plan.xml** file in a text editor.
2. Add the following configuration under the **<module-override>** tag:

```
<module-override>
   <module-name>BPA.war</module-name>
   <module-type>war</module-type>
   <module-descriptor external="false">
      <root-element>weblogic-web-app</root-element>
      <uri>WEB-INF/weblogic.xml</uri>
      <variable-assignment>
         <name>secure-cookie</name>
         <xpath>/weblogic-web-app/session-descriptor/cookie-secure</xpath>
      </variable-assignment>
      <variable-assignment>
         <name>url-rewriting-enabled-enable</name>
         <xpath>/weblogic-web-app/session-descriptor/url-rewriting-enabled</xpath>
         <operation>add</operation>
      </variable-assignment>
      <variable-assignment>
         <name>pdc-application-path</name>
         <xpath>/weblogic-web-app/session-descriptor/cookie-path</xpath>
         <operation>add</operation>
      </variable-assignment>
   </module-descriptor>
</module-override>
```

3. Add the following configuration under the **<variable-definition>** tag:

```
<variable>
   <name>secure-cookie</name>
   <value>true</value>
</variable>
<variable>
   <name>pdc-application-path</name>
   <value>/pdc;SameSite=strict</value>
</variable>
<variable>
   <name>url-rewriting-enabled-enable</name>
   <value>false</value>
</variable>
<variable>
   <name>pdc-samesite</name>
   <value>strict</value>
</variable>
```

4. Save and close the file.
5. Redeploy the PDC application with your new **plan.xml** file.

   For more information about updating and deploying your deployment plan, see the "Create and Use a Deployment Plan in Oracle WebLogic Server" tutorial.

Configuring the Session Timeout

The default session timeout in PDC is 10 minutes. Your WebLogic Server administrator can change this value after deployment by doing the following:

1. Log in to WebLogic Remote Console.
2. Click **Monitoring Tree**, then **Deployments**, and then **Application Management**.

   A page with a list of installed Java EE applications and standalone application modules appears.
3. In the table, click **PricingDesignCenter**.

   Information about PricingDesignCenter appears.
4. Click **Configuration** in the tree in the left pane.
5. Click **Session Descriptor** in the tree in the left pane.
6. In the **Session Timeout (in seconds)** field, enter a new timeout value in seconds.
7. Click **Save**.
8. If you do not already have a deployment plan, or if the deployment plan is unavailable, WebLogic Server creates one with these changes and prompt you to save it. Provide a name and path for the new deployment plan and click **OK**.
9. Click **Application Management** in the tree in the left pane.
10. In the table, select the **PricingDesignCenter** application.
11. Click **Update/Redeploy**.
12. Select **Update - Deployment Plan on Server** and set the **Plan Path** field to the deployment plan.
13. Click **Done**.
14. Restart WebLogic Server.
15. Verify your changes by doing the following:
    a. Log in to WebLogic Remote Console.
    b. Click **Monitoring Tree**, then **Deployments**, and then **Application Management**.

       A page with a list of installed Java EE applications and standalone application modules appears.
    c. In the table, select **PricingDesignCenter**.

       The information about PricingDesignCenter page appears.
    d. Click **Configuration** in the tree in the left pane.
    e. Click **Session Descriptor** in the tree in the left pane.
    f. Verify that **Session Timeout (in seconds)** is set to the value you specified.

For more information about deployment plans, including an example of using one while updating session timeout, see "Configuring Applications for Production Deployment" in *Oracle Fusion Middleware Deploying Applications to Oracle WebLogic Server*.

Managing File Permissions

- Following are the default permissions set for the installed files:

  - rw------- 600 (for all non executable files)
  - rwx------ 700 (for all executable files)

  Permissions are set to the lowest possible level, and the WebLogic Server administrator can add or revoke permissions. Oracle recommends keeping the permissions as restrictive as possible, according to your business needs.

- The WebLogic configuration (JMS, JDBC, and so on) file, **config.xml**, in the domain's configuration directory should be protected with proper permissions.
- Output files generated by the export utility should be stored in a protected directory because they may contain sensitive pricing information.

Uninstalling Pricing Design Center

The following files remain in the system after uninstalling PDC:

- Install logs:

  Location: *Oracle Inventory***/logs/**

```
-rw-r-----  1 user1 eng  480058 Aug 15 09:25 installActions2018-08-15_08-06-57AM.log
-rw-r-----  1 user1 eng       0 Aug 15 10:27 oraInstall2018-08-15_10-27-07AM.out
-rw-r-----  1 user1 eng    2384 Aug 15 10:33 dbScripts2018-08-15_10-32-00AM.log
-rw-r-----  1 user1 eng  124268 Aug 15 10:33 oraInstall2018-08-15_10-27-07AM.err
```

- *PDC_home***/oui/data.properties**: This file is used to auto-populate the data during re-installs.

Delete these files manually if you do not need them, or protect them appropriately if they are required for future reference.

By default, these files are created with file permission 640 (owner can read/write, group members can read, others have no permission).

About Changing Passwords in the Wallets

PDC stores the passwords for the WebLogic Server domain, the PDC user, the cross-reference database, and the Oracle Communications Billing and Revenue Management (BRM) database in PDC and BRM Integration Pack wallets.

To change the password in the wallets, you must encrypt the new password manually and update the entry in the appropriate wallet. See "Changing Passwords in the Wallet" in *BRM System Administrator's Guide* for more information.

Implementing Pricing Design Center Security

This section describes how to implement the security capabilities in PDC by using Oracle Identity Management (IDM).

PDC uses IDM for authenticating and authorizing PDC users. Each instance of PDC requires an appropriately configured instance of IDM to enable these functions.

For information about installing PDC, see *PDC Installation Guide*.

> **Note**
>
> If you have configured IDM, you must authorize PDC users by using IDM only.

About Authentication

Within IDM, Oracle Identity Manager (OIM) provides a mechanism for managing user password policies. You must configure OIM to authenticate and authorize PDC users. See *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

About Authorization

Authorization refers to granting appropriate privileges to users and denying access to other functionality based on their job functions. Users with the following roles can access PDC using IDM:

- **Pricing Design Admin**: Can import and export all pricing and setup components in PDC.
- **Pricing Analyst**: Can import only pricing components; however, users with this role can export pricing and setup components.
- **Pricing Reviewer**: Can only export all pricing and setup components.
- **Migration Admin**: Can migrate pricing data from the BRM database to the PDC database.
- **JDGroup**: Can manually trigger the job dispatcher to put transformation jobs in the work item queue.

Configuring Authentication and Authorization by Using OIM

OIM enables enterprises to manage the entire user life cycle across all enterprise resources within and beyond a firewall.

To configure OIM to authenticate and authorize users in PDC:

1. Configure OAM in WebLogic Server. See "Configuring OAM in WebLogic Server".
2. Add users and assign roles in OIM. See "Adding Users and Assigning Roles in OIM".

Configuring OAM in WebLogic Server

To configure Oracle Access Manager (OAM) in WebLogic server:

1. Log in to the WebLogic Remote Console.
2. Click **Edit Tree**, then **Security**, and then **Realms**.

   The Summary of Security Realms page appears.
3. Click the **myrealm** link.

   The myrealm configuration page appears.
4. Click **Authentication Providers** in the tree in the left pane.

   A page with an Authentication Providers table appears.
5. Click **New**.

6. In the **Name** field, enter **OAM Identity Asserter**.
7. From the **Type** list, select **Oracle Access Manager Identity Asserter**.
8. From the **Control Flag** list, select **REQUIRED**.
9. Click **Create**.
10. Click **New**.
11. In the **Name** field, enter **OUD Authenticator**.
12. From the **Type** list, select **Oracle Unified Directory Authenticator**.
13. From the **Control Flag** list, select **SUFFICIENT**.
14. Click **Create**.
15. Click the **Oracle Unified Directory Authenticator Parameters** tab and provide the Oracle Unified Directory (OUD) connection details.
16. Click **Save**.
17. In the Authentication Provider table, arrange the providers in the following order using the **Move Down** and **Move Up** buttons.
    - **OAMIdentityAsserter**
    - **OUD Authenticator**
    - **DefaultAuthenticator**
    - **DefaultIdentityAsserter**
18. Click **DefaultAuthenticator** in the tree in the left pane and modify the **Control Flag** to **SUFFICIENT**.
19. Click **Save**.
20. Click the shopping cart at the top right, and then click **Commit Changes** to commit your changes.
21. Restart WebLogic Server.

Adding Users and Assigning Roles in OIM

To add users and assign roles in OIM to access PDC:

1. Log in to Oracle Identity Self Service.

   The Oracle Identity Self Service home page appears.
2. Create new users (if required) by performing the following steps:
   a. Click **Manage**.
   b. Click **Users**.

      The Users page appears.
   c. Click **+ Create**.

      The Create Users page appears.
   d. Enter the required information.

      For more information on creating users, see the discussion about creating and managing users in the Oracle Identity Manager Administrative and User Console Guide.

3. Select a user.
4. Click **+ Request Roles**.
5. In the **Search** field, enter the name of the role and click **Search**.

   See "About Authentication" for the supported roles.

   The search results appear.
6. Select a role from the list under Categories.
7. Click **+ Add to Cart**.
8. Click **Next** and click **Submit**.

   Now, the users can access PDC.

Verifying OIM Configuration in WebLogic Server

To verify the OIM configuration in the WebLogic server:

1. Log in to the WebLogic Remote Console.
2. Click **Security Data Tree** and then **Realms**.

   The Summary of Security Realms page appears.
3. Click **myrealm**.

   The myrealm configuration page appears.
4. Click **Authentication Providers** in the tree in the left pane.

   A page with an Authentication Provider table appears.
5. In the Authentication Provider table, click **DefaultAuthenticator**.
6. Click **Users** in the tree in the left pane.

   The list of users created in OIM appears.

4

Performing a Secure ECE Installation

Learn how to create a secure Oracle Communications Elastic Charging Engine (ECE) installation.

Topics in this document:

About Deploying ECE into a Secure Environment

Deploy ECE into a secure environment. For example, ensure that:

- ECE is deployed in a closed network environment where all public access is denied.
- All ECE hosts are connected to a single switch or to switches in a parallel configuration.
- No external processes should run on the hosts running ECE and its components.
- Access to the ECE infrastructure is restricted.

You can further harden ECE security by following the instructions in this chapter.

Installing ECE

By default, ECE is installed in a secure mode. ECE uses security measures such as cluster security and host authorization.

When you install ECE, you are prompted to select your preferred security configuration, such as whether to enable secure socket layer (SSL) configuration. Based on the security configuration you select in the installer, ECE sets parameters in the relevant Oracle Coherence and ECE configuration files for enabling the following security levels:

- **JMX security**. Clients require a JMX user name and password to connect to ECE JMX Management servers. For example, Elastic Charging Controller (ECC) can use a JMX user name and password to be authenticated to log in to the cluster.

- **Authorized host list**. A process that joins the Coherence cluster has access to ECE services only if it is running on a host defined in the authorized host list.
- **Coherence node authentication**. ECE nodes are required to authenticate themselves when trying to join the Coherence cluster. The node credentials are stored in a KeyStore file that must be deployed on the ECE nodes.
- **SSL encryption** (intra-cluster communication). Communication across ECE nodes in the Coherence cluster are encrypted.

About ECE Security

Access to ECE files is controlled by creating user accounts and groups and granting specific permissions. The file permissions are granted using Linux commands in a Linux shell. After you have created user accounts and groups and set permissions, users can use ECC to manage ECE files. ECC requires setting up passwordless SSH. You use the ECE user account —a Linux account—for this purpose. See "Setting Up and Managing Elastic Charging Engine Security" in *BRM System Administrator's Guide*.

About Enhanced Security for JAR Files

All ECE-related JAR files are digitally signed with a private key to ensure their authenticity and integrity. This also helps detect any modifications to the files before deployment. You can verify the signatures by using the **jarsigner** utility, provided in the JDK:

```
jarsigner -verify -verbose -certs jarName.jar
```

where *jarName* is the name of the JAR file.

The expected output is:

```
jar verified
```

About Oracle Coherence Security

To restrict access to the ECE Coherence cluster, set up an authorized host list. You can optionally enable SSL for intra-cluster communication, in which case you must also enable Well Known Addresses (WKA). See "Setting Up Cluster Security" in *BRM System Administrator's Guide*.

About Oracle Database Security

If you are using Oracle Database for data persistence, configure Oracle Database advanced security encryption and integrity algorithms for a secure connection from the installer. See the Oracle Database documentation for advanced security configuration parameters. This is required for the ECE installer to make a secure (encrypted) database connection over the network. For more details, see *Oracle Database Advanced Security Administrator's Guide*.

About Oracle NoSQL Database Security

If you are using Oracle NoSQL Database for data persistence, install Oracle NoSQL Database in a secure location where physical and network access to the store is restricted to trusted users. For this reason, Oracle NoSQL Database's security model is designed to prevent accidental data access. However, it is not designed to prevent malicious access or denial-of-service attacks.

You can access the KVStore and its data either through the Java API or via administrative access, which is performed using a command-line interface or a browser-based graphical user interface. System administrators use these interfaces to perform the few administrative actions that are required by Oracle NoSQL Database. You can also monitor the store using these interfaces.

About Cluster Security

ECE uses a file-based credentials store or a KeyStore to keep node credentials that are required to join the cluster and that are used for enabling encryption of cluster communication. The KeyStore is located at *ECE_home***/ocecеserver/config/server.jks**. Although the ECE installer creates a **server.jks** file, you can create your own if required. If you create your own JKS file, ensure it has restrictive permissions to prevent unauthorized access.

ECE uses an Oracle wallet to store passwords required to connect to boundary systems such as Oracle Communications Billing and Revenue Management (BRM) and Oracle Communications Pricing Design Center (PDC).

When you install ECE, you enter the following information:

- The account alias for Coherence cluster security
- The key password for Coherence cluster security (the password for the alias)
- The key password for the boundary system alias
- The password for accessing the KeyStore (the certificate store password)
- DName details

  The DName value specifies user authorizations related to cluster security.

  The DName is used for authorization as defined in *ECE_home***/ocecеserver/config/permissions.xml**.

See *ECE Installation Guide* for more information.

About the KeyStore Files and SSL Considerations

ECE maintains a **server.jks** KeyStore file that stores credentials for cluster node authentication. The file is also used for encrypting intra-cluster communication over SSL.

KeyStore passwords for SSL are stored by default in *ECE_home***/ocecеserver/config/charging-coherence-override-secure-prod.xml**. However, these can be overridden by defining their respective system properties in *ECE_home***/ocecеserver/config/defaultTuningProfile.properties**.

> **Note**
>
> Oracle strongly recommends that you do not override the default *ECE_home***/ocecеserver/config/charging-coherence-override-secure-prod.xml** file.

Installation Settings when SSL Is Enabled

When you select the SSL options during installation, the following are set:

- In *ECE_home***/ocecеserver/config/ece.properties**:

  - tangosol.coherence.override=charging-cache-config-secure-prod.xml
  - The WKA list in the **charging-cache-config-secure-prod.xml** file

    This should contain the WKA host list provided during the installation.
- In *ECE_home***/oceceserver/config/charging-coherence-override-secure-prod.xml**:
  - -Dtangosol.coherence.ssl.keypassword=*keypassword*
  - -Dtangosol.coherence.ssl.storepassword=*storepassword*

  where *keypassword* and *storepassword* are the key and store passwords given during installation.

About Trusted Host Information

ECE caches contain subscriber data. To restrict access, specify the machines or processes you trust and allow to be part of the cluster.

Obtain the IP addresses or host names of all machines or processes that are allowed to access the cluster. Trusted hosts include all server machines where the Coherence cluster is deployed and any other machine that is to be part of the cluster. Include the server machine that runs the Elastic Charging Controller (ECC); if you use Oracle Enterprise Manager, also include the JMX client host running it.

See "Installing Elastic Charging Engine" in *ECE Installation Guide* for more information.

About JMX Security

JMX security is based on standard Java guidelines, as discussed in "Security" in *Java Platform, Standard Edition Java Management Extensions Guide*.

For ECE, JMX can be secured by setting the following system parameters:

- In *ECE_home***/oceceserver/config/ece.properties**:

  ```
  com.sun.management.jmxremote.authenticate=true
  ```

- In *ECE_home***/oceceserver/config/defaultTuningProfile.properties**:

  ```
  -Dcom.sun.management.jmxremote.password.file=../config/jmxremote.password
  ```

The file permissions for **jmxremote.password** must be set to **400**; otherwise, Elastic Charging Server nodes will not start.

ECE includes a **jmxremote.password** file in the *ECE_home***/oceceserver/config** directory, which contains two default accounts for JMX credentials, as defined in *JRE_home***/lib/management/jmxremote.password.template**:

- monitorRole with read-only permissions
- controlRole with read and write permissions

Passwords for these accounts can be set in the **jmxremote.password** file located in *ECE_home***/oceceserver/config**. To add more accounts, simply add them to this file as well. See "Monitoring and Management Using JMX Technology" in *Java Platform, Standard Edition Monitoring and Management Guide* for more information about using the **jmxremote.password** file.

Because the JMX passwords are human-readable in **jmxremote.password**, the file permission must be set to **400**.

> **Note**
>
> The **jmxremote.password** file is used for more than JMX. This file is also used for storing passwords required to authenticate cluster nodes and required to encrypt and decrypt passwords for JMS notification services. See "About Managing External Application Passwords" in *BRM System Administrator's Guide* for more information.

All Elastic Charging Controller (ECC) shell commands are JMX-aware. If JMX is secured, you must provide a user name and password when starting ECE services.

If JMX is secured, commands like **start server** or starting a single node, such as **start ecs1**, **start configLoader**, and so on must provide a user name and password. For example:

```
start server username=controlRole password=password_as_defined
```

In secured mode, Oracle recommends using the ECC shell in interactive mode (all commands are run within the shell rather than as arguments to the ECC script). The ECC command sets the file permissions of the file that saves the history of the commands that have been run to **600**. This prevents unauthorized access to previous commands that might include passwords typed on the command line.

In applications such as JConsole, jVisualVM, or other JMX client applications, you must specify the user name and password when a connection is made.

Postinstallation Security Tasks

The Oracle Universal Installer prompts you to enter security information for postinstallation steps typically required for security.

After installation, verify the following in the *ECE_home*/**oceceserver/config/permissions.xml** file:

- The **principal** section has the same DName information as was defined during the installation process for creating the **server.jks** file.
- Full access to all resources is allowed for an authenticated user.
- If the **secure.access.name** system property is set, then the **tangosol.coherence.security** system property must be set to **true**. If the **tangosol.coherence.security** system property is set to **false**, the **secure.access.name** system property should not be set.

5

Managing BRM Security

Learn how to manage security in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

The Security Model

BRM security requirements arise from the need to protect data: first, from accidental loss and corruption; second, from unauthorized attempts to access or alter that data. Secondary concerns include protection against undue delays in accessing or using data, or interference that may lead to denial of service. The global cost of such security breaches reaches billions of dollars annually, and the cost to individual companies can be severe or even catastrophic.

The critical security features that provide these protections are:

- **Authentication**: Ensures that only authorized individuals have access to the system and data.
- **Authorization**: Ensures access control to system privileges and data. Authorization builds on authentication to ensure that individuals get only appropriate access.
- **Audit**: Allows administrators to detect attempted breaches of the authentication mechanism and attempted or successful breaches of access control.
- **Encryption**: Ensures that data cannot be read without being adequately decrypted.

BRM enables TLS by default to secure communication between the applications. See "Enabling Secure Communication Between BRM Components" in *BRM System Administrator's Guide* for more information.

Configuring and Using Authentication

BRM requires two levels of authentication within its operation:

- Authentication of Applications
- Authentication of Accounts

Both levels of authentication use the Oracle Wallet.

By default, the BRM installer stores configuration entries, including sensitive information (such as TLS certificates and database and account passwords), in the Oracle Wallet. BRM applications retrieve the data from the Oracle Wallet unless the configuration entries are also stored in the **Infranet.properties** and **pin.conf** configuration files. The entries in the configuration files take precedence.

Authentication of Applications

Each component in the application tier must authenticate itself using an account to be allowed to send requests to the BRM server. The user name is stored in the application's configuration file. By default, the password is stored in the Oracle Wallet. However, the application may be configured so that the password is encrypted using AES and stored in the application's configuration file.

Application account information is stored in the BRM database. All passwords are hashed and encrypted before being stored in the database. When the application connects, the password is hashed and the hash is compared with the hashed password in the database.

Authentication of Accounts

Users requesting permission to carry out a transaction must be authenticated against the account information stored in the BRM database. All passwords are hashed and encrypted before being stored in the database. The user name and password are typed in by the user, and then the password is hashed, and the hash is compared with the hashed password in the database.

Configuring and Using Access Control

The following information about configuring and using access control in BRM is available:

- Permissions
- Roles
- Managing CSR Passwords
- Account Lockout
- Automatic Logout
- Access Control in BRM Web Services Manager

Permissions

Permissions determine the tasks a user can perform in BRM applications.

It is possible to restrict activities in applications, such as Customer Center and Pricing Center, by assigning CSRs to a role and setting permissions for that role. For example, it is possible to specify which CSRs can change a password, apply credits, and give refunds. See "Setting Up Permissions in BRM Applications" in *BRM System Administrator's Guide* for more information.

In most cases, only someone with root access, such as a system administrator, has permission to change CSR permissions.

See "Managing ECE Permissions" in *BRM System Administrator's Guide* for more information.

Roles

A role is defined by a set of permissions. A role represents the actions a person with a particular job or position can perform. Roles are used to configure permissions for a group of CSRs based on the tasks they need to perform. For example, it is possible to create different types of CSRs and assign them to different kinds of roles:

- *Manager CSRs* can create new roles, assign CSRs to roles, change permission settings, change credit limits, give refunds, and change account status. A manager can also validate the work that junior CSRs perform, for example, by making sure that new accounts are created correctly and have all the necessary information.
- *Junior CSRs* can check customer account balances, check and change billing information, and answer common customer questions.

For example, CSRs A and B can be assigned to the role Manager, and CSRs C and D can be assigned to the role Lead-CSR, where:

- CSRs A and B have read-write permissions for customer credit card information.
- CSRs C and D have read-only permissions for customer credit card information.

You can also create roles with higher permission levels. For example, you can create roles that include permissions to create and manage roles using Permissioning Center.

Roles can also be configured to allow access to one or more client applications. Additionally, a CSR can be assigned to multiple roles. For example, a CSR can be assigned to a Manager role in Permissioning Center and to a Junior-CSR role in Pricing Center.

Roles can be hierarchical by creating child roles and associating them with a parent role. At each level above the lowest in the hierarchy, child roles can also be parent roles. A child role inherits all permission settings that are associated with its parent role.

See "About Managing Roles" in *BRM System Administrator's Guide* for more information.

Managing CSR Passwords

To improve security features and provide access to BRM client applications, the following password policies are included in Permissioning Center:

- **Ability to set password expiry limits**: The duration of time that a password is valid until the system prevents a user from logging in or forces the password to be changed.
- **Ability to define temporary passwords**: The ability to force CSRs to change their passwords after accessing the application the first time or after a new CSR account has been set up by an administrator.
- **Password content validation**: The ability to validate password contents to ensure certain characters, such as numbers, are included or excluded.

See "Managing CSR Passwords" in *BRM System Administrator's Guide* for more information.

Account Lockout

Users are locked out of the system after a specified number of invalid login attempts. See "Configuring the Maximum Number of Invalid Login Attempts" in *BRM System Administrator's Guide* for instructions for changing the default number of attempts allowed.

Once users are locked out of the system, manual intervention is required to re-enable the accounts. See "Unlocking a Locked CSR Account" in *BRM System Administrator's Guide* for instructions for unlocking the accounts.

Automatic Logout

BRM provides the functionality to force a user to reauthenticate after a given amount of idle time. However, if the password is present in the configuration file, the authentication process is automated. This feature should not be used to allow automated re-authentication of CSR accounts.

You can configure the session timeout interval by setting the **cm_timeout** parameter. See "Setting the CM Time Interval between Opcode Requests" in *BRM System Administrator's Guide* for detailed instructions.

Access Control in BRM Web Services Manager

BRM Web Services Manager enables BRM opcodes to be exposed through Web services.

You can use access control capabilities to restrict Web services to certain user roles. You can create multiple roles, each with a distinct set of privileges. You define access restrictions for Web services using security policies in WebLogic Server. See "Configuring WebLogic Security Policy on BRM Web Services for JAX-WS in WebLogic Server" in *BRM Web Services Manager* for more information.

Web Services Manager uses the OAuth 2.0 protocol to authenticate a client application's identity and authorize it to access BRM Web services. See "Securing Web Services Manager with OAuth2" in *BRM Web Services Manager* for more information.

Configuring and Using Security Audit

BRM provides support for auditing any object in the BRM database so that a record is kept of every version of the object for future reference. This can be used to track changes to customer profiles, customer payment information, and so on. An audit trail can also be used to track internal changes, such as changes to your pricing components.

Specific fields within objects can be requested to be audited. However, because of performance overhead, auditing should only be enabled for fields where there is a security risk.

Monitoring Login Attempts

You can see both successful and unsuccessful login attempts in the Connection Manager (CM) log file, which is located in the *BRM_home***/sys/cm** directory.

Search for the text "login attempt" in the log file. The PIN_FLD_RESULT field in the result from the PCM_OP_ACT_FIND_VERIFY opcode indicates the success or failure of the login attempt.

Encryption

By default, BRM encrypts the passwords stored in the BRM database.

However, encryption can also be extended to fields containing sensitive customer information, such as credit card numbers, to guarantee privacy and prevent unauthorized use. Fields to be encrypted must be in string format. You set up encryption with the BRM Storable Class Editor,

which adds a flag attribute in the metadata defining the field in the BRM data dictionary (PIN_FLD_ENCRYPTABLE).

BRM encrypts the fields marked for encryption when storing them in the database and automatically decrypts the fields when retrieving them from the database.

See "About Encrypting Data" in *BRM System Administrator's Guide* for more information.

Using Oracle ZT Encryption Scheme

Each instance of BRM has a unique root encryption key. This root key is used for all encryption and decryption processes in BRM. You can use an instance-specific root encryption key to assign different keys for development, test, pre-production, and production instances of BRM.

When different root keys are used for each instance, sensitive subscriber data and other credentials, such as subscriber passwords, cannot be copied from one instance to another and decrypted in a different system.

The encryption scheme adds a random initialization vector for each plain block of text to be encrypted. This prevents pattern-based attacks.

See "Generating a Root Encryption Key" in *BRM Developer's Guide* for more information.

Securing Sensitive Customer Data

Protect subscriber data by masking values contained in system responses to clients and logging.

See "Masking Sensitive Customer Data" in *BRM Managing Customers*.

Using Credit Card Tokenization

Credit card tokenization is a secure method of storing credit and debit card data. It replaces the credit and debit card numbers with random identifiers, referred to as tokens. You can use tokens for any BRM-initiated payments instead of the actual card numbers. The actual card numbers and their mapping to the tokens are stored securely in Paymentech. Tokens are valid only between the merchant system and the credit card processor. You can use tokens to transmit payment information safely without the risk of exposing card data.

See "Masking Credit Card Numbers by Using Tokens" in *BRM Configuring and Collecting Payments* for more information.

You can migrate previously stored credit card numbers to tokens using the provided migration application. See "Replacing Credit Card Numbers with Tokens" in *BRM Configuring and Collecting Payments* for more information.

You can use tokens generated outside BRM for credit and debit cards by sending the token to the application through the PCM_OP_CUST_COMMIT_CUSTOMER opcode. See *BRM Opcode Flist Reference* for more information about PCM_OP_CUST_COMMIT_CUSTOMER.

You can prevent users from entering any credit card information in Billing Care by removing the **creditCard** entry from the **paymentTypes** key in the Billing Care **CustomConfigurations.xml** file. See "Editing the Billing Care Configuration File" in *Billing Care SDK Guide* for more information about the **CustomConfigurations.xml** file.

Masking Sensitive Data in Log Files

BRM is preconfigured to store sensitive data in an encrypted format. However, because encryption and decryption are done in the Data Manager to ensure that the business logic has access to the real value. These fields should also be marked as masked to prevent their values from appearing in any BRM log files.

See "Defining Masked Fields" in *BRM Developer's Guide* for more information.

Securing BRM Network Ports

The BRM PCM protocol cannot enforce access control for any command line applications or custom C or Java applications. Therefore, operating system and network security measures must be used to secure access to the BRM network ports:

- Enable TLS to secure the BRM PCM communications. See "Enabling SSL/TLS for C and C++ PCM Clients" and "Enabling SSL/TLS for Java PCM Clients", both in *BRM System Administrator's Guide*, for more information.
- The Connection Manager (CM) port must be blocked to prevent connections from desktops that do not have a business justification to run a BRM client application.
- BRM DM port: The BRM DM port must be blocked to allow connections only from servers running a permitted instance of the CM.

About Managing ECE Security

To manage ECE security, you perform the following tasks:

- Set up user accounts and user groups, and grant permissions. After creating user groups and setting permissions, users can log in to the system, use ECE, and manage the ECE cluster.

  Assign permissions only to users who run and manage ECE processes, rated event files, and the ECE file systems. Restrict permissions as much as possible. You can create a single administrative user with all permissions to manage ECE core processes and directories, or multiple users with specific permissions for these tasks.

  See "Configuring Specific File Permissions" in *BRM System Administrator's Guide* for a list of the files that you need to restrict access to.
- Manage passwords. Linux accounts protected by passwords must be created for ECC. Besides Linux accounts, you need to create non-Linux accounts to access external applications like Oracle Communications Billing and Revenue Management (BRM) and Oracle Communications Pricing Design Center (PDC). BRM and PDC are used to load customer and pricing data, respectively, into ECE. For secure communication between ECE and these systems, credentials stored in ECE are encrypted and stored in the KeyStore.
- Set up cluster security. To restrict access to the ECE Coherence cluster, you must set up an authorized hosts list. You can optionally enable SSL for intra-cluster communication, in which case you must also enable Well Known Addresses (WKA).
- Set up passwordless Secure Shell (SSH). You must set up passwordless Secure Shell (SSH) between driver and server machines, for ECC to work. Passwordless SSH allows servers to connect to the driver and synchronize ECE files.

6

Security Considerations for Developers

Learn how developers can extend Oracle Communications Billing and Revenue Management (BRM) without compromising security.

Topics in this document:

Using the BRM SDK

The frameworks provided in the BRM SDK have the same level of security built into them as exists in the standard BRM product. All extensions developed for BRM should use the framework to ensure the security features detailed in this guide are included in the extension's design.

Security Considerations for ECE Developers

ECE requires that all Java processes joining its cluster have the correct configuration settings. When using ECE secure mode, having only the correct Coherence properties is not sufficient to join the cluster. Developers writing extensions or plug-ins for ECE should not use direct access to Coherence APIs. They must use the Spring and Template framework provided by ECE. Any direct access to Coherence resources, including its caches, causes security exceptions.

7

Billing Care Security

Learn how to install and implement Oracle Communications Billing Care and its components in a secure configuration.

Topics in this document:

About Installing Billing Care Securely

Before installing Billing Care, you must properly install and configure several Oracle products, including Java, Oracle WebLogic Server, and Oracle Communications Billing and Revenue Management (BRM). For Billing Care installation instructions, including all required products and related tasks, such as setting up KeyStores and SSL for WebLogic Server, see "Installing Billing Care" in *Billing Care Installation Guide*.

Oracle Platform Security Services (OPSS) and Oracle Identity Manager provide authentication and authorization capabilities for Billing Care. These products are also required in a Billing Care implementation. See *Oracle Fusion Middleware Integration Guide for Oracle Identity Management Suite* and *Oracle Fusion Middleware Administrator's Guide for Oracle Platform Security Services*.

Encrypting OAP Entries in Infranet.properties

If you are using Oracle Analytics Publisher for invoicing, you must add the BIP user ID, the OAP password, and the OAP URL in the Billing Care wallet. For a secure installation, you must encrypt the OAP password. You use Oracle WebLogic Server to perform the encryption. For more information, see "Storing Configuration Entries in the Billing Care Wallet".

Securing Web Cookies

A common web security problem is the stealing of Web cookies. This happens when an attacker manages to get a copy of your web cookie, generally while it is being transmitted over insecure channels such as HTTP.

You use a WebLogic Server variable to ensure that your Billing Care cookies are sent only through encrypted channels, such as SSL.

To secure your Billing Care Web cookies:

1. Open the *BillingCare_home***/setup/plan.xml** file in a text editor.

   where *BillingCare_home* is the directory in which you installed Billing Care.

2. Set the SECURE_COOKIE variable to **true**:

```
<variable-definition>
   <variable>
      <name>SECURE_COOKIE</name>
      <value>true</value>
   </variable>
</variable-definition>
```

3. Add a variable assignment for SECURE_COOKIE:

```
<module-override>
   <module-name>BillingCare.war</module-name>
   <module-type>war</module-type>
   <module-descriptor external="true">
      <root-element>weblogic-web-app</root-element>
      <uri>WEB-INF/weblogic.xml</uri>
      <variable-assignment>
         <name>SECURE_COOKIE</name>
         <xpath>/weblogic-web-app/session-descriptor/cookie-secure</
xpath>
      </variable-assignment>
   </module-descriptor>
</module-override>
```

4. Save and close the file.
5. Log in to the Oracle WebLogic Remote Console.
6. Click **Monitoring Tree**, then **Deployments**, and then **Application Management**.

   A page with a list of installed Java EE applications and standalone application modules appears.
7. In the table, select the **billingCare.war** checkbox.
8. Click **Update/Redeploy** and do one of the following:
   - If your updated deployment plan is located on the server containing WebLogic Server, select **Update - Deployment Plan on Server** and enter the location of the file in the **Plan Path** field.
   - If your updated deployment plan is located on your local machine, select **Update - Deployment Plan on Local Machine** and click **Choose File** to browse to the file or enter the file name in the **Plan** field.
9. Click **Done**.

Implementing Billing Care Security

Billing Care supports stringent authorization, authentication, and audit requirements. This section describes how to implement the security capabilities supported by Billing Care.

About Identity Management Suite

Oracle Identity Management (IDM) is a primary component for authorization and authentication. Each instance of Billing Care requires a properly configured instance of IDM to enable these functions.

For information about installing Billing Care, see *Billing Care Installation Guide*.

About Authentication

Billing Care supports the following security for authentication:

- Authenticating Billing Care users against an LDAP-based user ID repository
- Enabling Single Sign-On capabilities
- Supporting user's password policies

Oracle Identity Manager manages user password policies. For more information, see *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

About Authorization

Authorization refers to granting users privileges appropriate for their job functions while denying access to other functionality. OPSS handles all authorization tasks for Billing Care. This section provides an overview of setting up and maintaining entitlements for Billing Care plus strategies for mapping enterprise users to those entitlements.

The following terms are used in authorization:

- **Resource Type:** Contains the action definitions, for example, **AdjustmentCurrencyResourceType**.
- **Resource:** Represents a piece of functionality being secured, for example, **AdjustmentResource**. It must always be of a known resource type.
- **Action:** Combined with a resource, defines operations permissible for an application's functionality, for example, **AdjustmentResource** and **make**.
- **Obligation:** Stores transaction limits. Some operations impose transaction limits, such as the maximum payment amount. Obligations are the property of Authorization Policy.
- **Authorization Policy:** Comprises the resources, actions, and obligations that combine to form a logical grouping, for example, an entire set of application functions for the regular CSR.
- **Enterprise (External) Role:** Represents the job functions for the users at your company. You make OPSS aware of roles by mapping them to the Billing Care policies. If you do not map enterprise roles in the authorization policy, you must map to each user.

Billing Care includes an OPSS seed file containing all the resource types, resources, actions, and obligations and few sample authorization policies (Regular CSR, Super CSR, ReadOnly CSR, Auditor, Billing Analyst, and WriteOff).

For instructions on importing the seed file, see *Oracle Fusion Middleware Administering Oracle Platform Security Services*.

> **Note**
>
> Unless you are customizing Billing Care, do not change the seed file.

To deploy the seed file, use the **jps-config.xml** file located in *SDK_home***references\AuthorizationDataModel**.

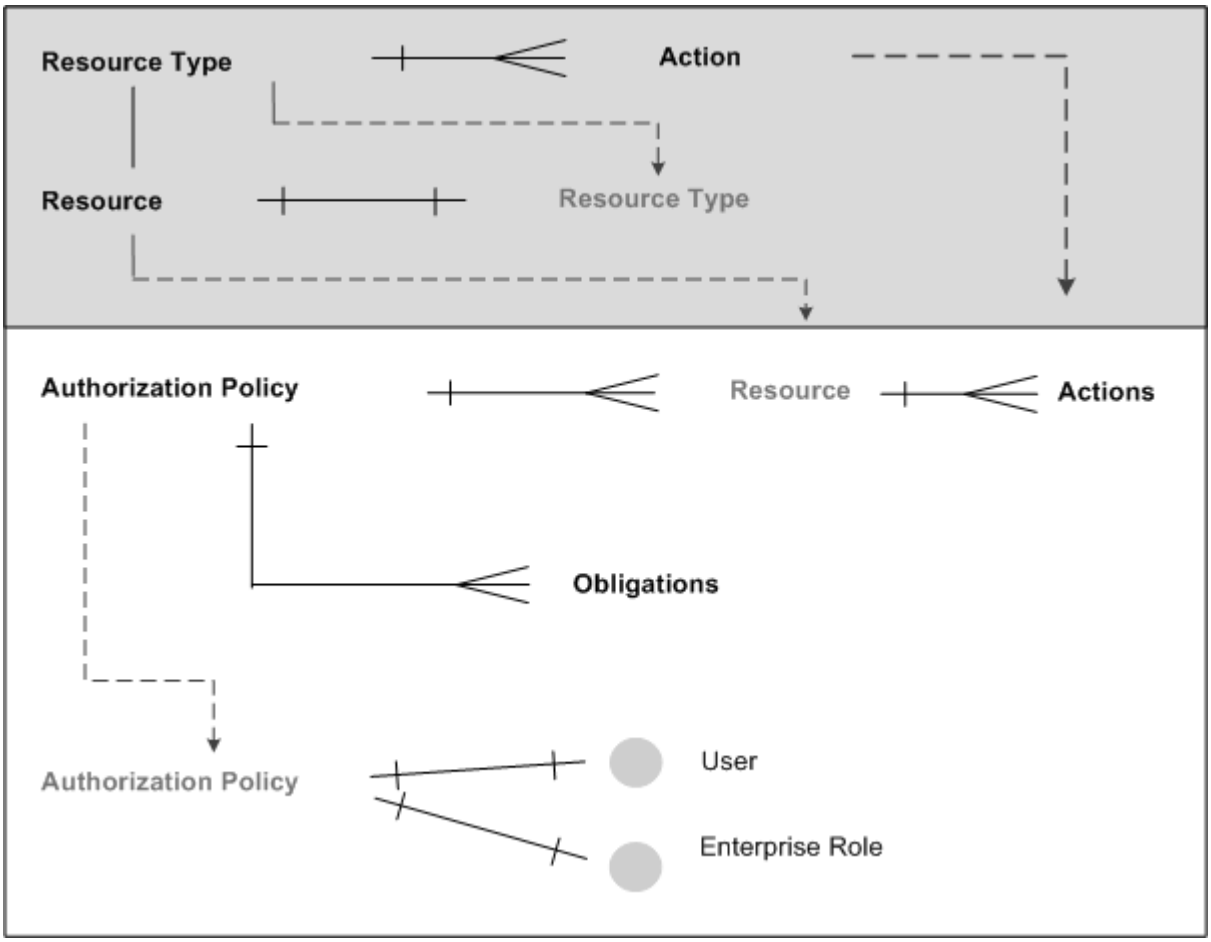
Figure 7-1 describes the authorization flow:

1. The shaded area refers to areas defined in the OPSS seed file. (You load the seed file before performing the configuration).
2. The lower area represents how an authorization policy is mapped to one or more resources in OPSS. A resource may have one or more actions.
3. The authorization policy is mapped to obligations.
4. The authorization policy is associated with a user (individual) or an enterprise role (function).

The authorization policy is mapped to obligations, which are listed in Table 7-2.

Any changes made in OPSS must be redeployed (or distributed).

Figure 7-1 Developing Authorization Policies for User and Enterprise Roles



About Billing Care Authorization Resources

A user who does not have a resource grant is denied access to Billing Care. This behavior is targeted for deployments where a central user identity repository, storing all enterprise users, authenticates Billing Care sign-in requests. The authorization scheme allows access only to users granted resources in OPSS.

Table 7-1 shows the Billing Care Authorization Resources. Resources grant permission to perform general CSR tasks or more advanced A/R tasks.

Table 7-1 Authorization Resources

Resource Type	Resource	Actions	Description
SuperUserType	SuperUserResource	Any	Enables you to create users free of restrictions, including when the user's profile contains other resources. The only exception is the ReadOnlyType, which takes precedence over all other resource types.
ReadOnlyType	ReadOnlyResource	Any	Causes **Save** and **Apply** buttons on overlays to be displayed as read-only. Users are allowed only to read operations even if they have other resources or entitlements.
AccountResourceType	AccountResource	Make, Modify, Search, Transition, View	**Make**: Allows the user to create accounts. **Modify**: Allows the user to add, delete, or save customer contact information. **Search**: Allows the user to access search functionality. **Transition**: Enables changing the account status. **View**: Enables the user to view account profile and other customer information.
AdjustmentCurrencyResourceType	AdjustmentResource	Allocate, Make	**Allocate**: Allows the user to allocate currency adjustments. **Make**: Allows the user to make adjustments. Uses a policy that constrains the maximum payment amount as a function of CSRs access level. See Table 7-2 for transaction limits.

Table 7-1 (Cont.) Authorization Resources

Resource Type	Resource	Actions	Description
AdjustmentNonCurrencyResourceType	AdjustmentNonCurrencyResource	Make	**Make**: Gives noncurrency adjustments their own resource type because they cannot be allocated (unlike currency resources). Policy on the minimum and maximum noncurrency amount applies. See Table 7-2 for transaction limits.
BillResourceType	BillResource	BillNow	**BillNow**: Allows the user to perform Bill Now operations.
BillUnitResourceType	BillUnitResource	Add, BalanceTransfer, Delete, Modify, ModifyValidity, SetLimit	**Add**: Allows the user to create bill units. Allows the user to move services between bill units. **BalanceTransfer**: Allows the user to transfer balances to another bill unit. **Delete**: Reserved for future use. **Modify**: Allows the user to change the bill unit. **ModifyValidity**: Allows the user to modify the validity of noncurrency balances. **SetLimit**: Allows the user to set credit limits and thresholds for noncurrency resources.
CancelInstallmentResourceType	CancelInstallmentResource	Add, Cancel	**Add**: Allows the user to cancel an open installment. **Cancel**: Allows the user to undo an installment cancellation.

Table 7-1 (Cont.) Authorization Resources

Resource Type	Resource	Actions	Description
ChargeSharingResourceType	ChargeSharingResource	Add, Delete, Modify, AddMember, DeleteMember, ModifyMember, ModifyPriority	**Add**: Allows the user to create a charge sharing group. **Delete**: Allows the user to remove a charge sharing group. **Modify**: Allows the user to modify the charge sharing group. **AddMember**: Allows the user to add a member to a charge sharing group. **DeleteMember**: Allows the user to remove members from a charge sharing group. **ModifyMember**: Allows the user to change members in a charge sharing group. **ModifyPriority**: Allows the user to modify the order of priority when a member is part of multiple discount or charge sharing groups.

Table 7-1 (Cont.) Authorization Resources

Resource Type	Resource	Actions	Description
CollectionsResourceType	CollectionsResource	Search, Delete, AddAction, ModifyAction, Replace, Perform, Promise, Exit, Exempt, ReassignHandler, AddMember, DeleteMember	**Search**: Allows the user to access collections. **Delete**: Allows the user to delete a collection group from an owner account. **AddAction**: Allows the user to add a collection action on a bill unit in collections. **ModifyAction**: Allows the user to modify a collections action for a bill unit. **Replace**: Allows the user to replace a scenario with another scenario in collections. **Perform**: Allows the user to perform collections actions. **Promise**: Allows the user to manage promise-to-pay agreements in collections. **Exit**: Allows the user to exit a bill unit of an account from collections. **Exempt**: Allows the user to exempt a bill unit from collections. **ReassignHandler**: Gives the user the collections manager role. It allows the user to assign collections agents to bill units in collections. **AddMember**: Allows the user to add members to a collections group. **DeleteMember**: Allows the user to delete members from a collections group.
ConfigurationsArtifactsType	ConfigurationArtifactsResource	View	**View**: Allows the user to read all configuration-related REST APIs (for example, authorization profiles).

Table 7-1 (Cont.) Authorization Resources

Resource Type	Resource	Actions	Description
CustomerDepositResourceType	CustomerDepositResource	Add, View	**Add**: Allows the user to add and create deposits for a customer. **View**: Allows the user to view your customers' deposits.
DeferredActionResourceType	DeferredActionResource	Edit, Delete, Execute	**Edit**: Allows the user to modify a deferred action. **Delete**: Allows the user to remove a deferred action. **Execute**: Allows the user to run a deferred action.
DepositPaymentResourceType	DepositPaymentResource	Add	**Add**: Allows the user to accept and enter deposit payments for your customers.
DepositRefundResourceType	DepositRefundResource	Add, View	**Add**: Allows the user to refund deposits to your customers. **View**: Allows the user to view deposit refunds.
DepositReleaseResourceType	DepositReleaseResource	Add	**Add**: Allows the user to release deposits back to customers.
DepositResourceType	DepositResource	View	**View**: Allows the user to view customer deposits.
DepositReversalResourceType	DepositReversalResource	Add	**Add**: Allows the user to perform deposit reversals.
DepositSpecProfileResourceType	DepositSpecProfileResource	Add, View, Modify	**Add**: Allows the user to create deposit specification profiles. **View**: Allows the user to view deposit specification profiles. **Modify**: Allows the user to modify deposit specification profiles.

Table 7-1 (Cont.) Authorization Resources

Resource Type	Resource	Actions	Description
DepositSpecResourceType	DepositSpecResource	Add, View, Modify, Delete	**Add**: Allows the user to create deposit specifications. **View**: Allows the user to view deposit specifications. **Modify**: Allows the user to modify deposit specifications. **Delete**: Allows the user to delete deposit specifications.
DepositTransactionResourceType	DepositTransactionResource	View	**View**: Allows the user to view all transactions that have been performed on an account.
DepositTransferResourceType	DepositTransferResource	Add	**Add**: Allows the user to transfer a deposit to another account.
DiscountSharingResourceType	DiscountSharingResource	Add, Delete, Modify, AddMember, DeleteMember, ModifyMember, ModifyPriority	**Add**: Allows the user to create a discount sharing group. **Delete**: Allows the user to remove a discount sharing group. **Modify**: Allows the user to change the member details in a discount sharing group. **AddMember**: Allows the user to add a member to a discount sharing group. **DeleteMember**: Allows the user to delete members from a discount sharing group. **ModifyMember**: Allows the user to change members in a discount sharing group. **ModifyPriority**: Allows the user to modify the order of priority when a member is part of multiple discount sharing groups.

Table 7-1 (Cont.) Authorization Resources

Resource Type	Resource	Actions	Description
DisputeResourceType	DisputeResource	Raise, Settle	**Raise**: Allows the user to raise bill-level, item-level, and event-level disputes. **Settle**: Allows the user to settle bill and item-level disputes. Policy on the maximum dispute amount applies. See Table 7-2 for transaction limits.
InstallmentProposalResourceType	InstallmentProposalResource	Add	**Add**: Allows the user to add installment proposals.
InstallmentResourceType	InstallmentResource	View	**View**: Allows the user to view a customer's individual installments.
InstallmentScheduleResourceType	InstallmentScheduleResource	Add, View	**Add**: Allows the user to add an installment schedule. **View**: Allows the user to view a customer's installment schedules.
InstallmentScheduleSpecResourceType	InstallmentScheduleSpecResource	Add, View, Modify	**Add**: Allows the user to add installment schedule specifications. **View**: Allows the user to view installment schedule specifications. **Modify**: Allows the user to modify installment schedule specifications.
InvoiceImageType	InvoiceImageResource	View	**View**: Allows the user to view invoices.
LoanResourceType	LoanResource	ApplyLoan	**ApplyLoan**: Allows the user to perform a loan for customers.
NoteResourceType	NoteResource	Comment	**Comment**: Allows the user to add comments to notes.
NotificationEventResourceType	NotificationEventResource	View	**View**: Allows the user to view notification events.
NotificationResourceType	NotificationResource	View	**View**: Allows the user to view notifications.

Table 7-1 (Cont.) Authorization Resources

Resource Type	Resource	Actions	Description
NotificationSpecResourceType	NotificationSpecResource	Add, View, Modify	**Add**: Allows the user to create a notification specification. **View**: Allows the user to view notification specifications. **Modify**: Allows the user to change notification specifications.
PaymentMethodResourceType	PaymentMethodResource	Add, Delete, Modify	**Add**: Allows the user to add payment methods. **Delete**: Allows the user to delete payment methods. **Modify**: Allows the user to change payment methods.

Table 7-1 (Cont.) Authorization Resources

Resource Type	Resource	Actions	Description
PaymentResourceType	PaymentResource	Allocate, Audit, BatchProcess, BatchTemplateCreate, BatchTemplateDelete, BatchTemplateModify, Make, noManualAllocationMandatory, ReassignHandler, Reverse, SuspenseAccess, SuspenseAllocate, SuspenseMake, SuspenseMove, SuspenseReverse	**Allocate**: Allows the user to allocate payments. **Audit**: Allows the user to view audit information in the Payment Details dialog box. (Audit information in the **Payment Suspense** screen is always visible). **BatchProcess**: Displays the **Batch Payment** button on the landing page. It allows the user to validate and submit batch payments, batch reversals, and batch refunds. **BatchTemplateCreate**: Allows the user to create batch templates. **BatchTemplateDelete**: Allows the user to delete batch templates. **BatchTemplateModify**: Allows the user to modify batch templates. **Make**: Allows the user to make payments. **noManualAllocationMandatory**: When the user runs validation, it allows validation to pass when a payment amount is less than the total due for the account. **ReassignHandler**: Allows the user to assign an authorized handler to manage suspended payments. **Reverse**: Allows the user to reverse payments. **SuspenseAccess**: Allows the user to access the **Payment Suspense** button on the landing page. **SuspenseAllocate**: Allows the user to allocate suspended payments partially or fully to an account.

Table 7-1 (Cont.) Authorization Resources

Resource Type	Resource	Actions	Description
			SuspenseMake: Allows the user to make suspended payments. **SuspenseMove**: Allows the user to move posted payments into suspended status. **SuspenseReverse**: Allows the user to reverse suspended payments. Returned as an obligation; OPSS returns a number, which is interpreted as a limit. See Table 7-2 for transaction limits.
ProfileResourceType	ProfileResource	View, Add, Delete, Modify	**View**: Allows the user to view an account's profile. **Add**: Allows the user to create an account profile. **Delete**: Allows the user to remove an account profile. **Modify**: Allows the user to change an account profile.
ProfileSharingResourceType	ProfileSharingResource	Add, Delete, Modify, AddMember, DeleteMember, ModifyMember	**Add**: Allows the user to create a profile sharing group. **Delete**: Allows the user to remove a user from a profile sharing group. **Modify**: Allows the user to change the member details in a profile sharing group. **AddMember**: Allows the user to change the member details in a profile sharing group. **DeleteMember**: Allows the user to remove a member from a profile sharing group. **ModifyMember**: Allows the user to modify members in a profile sharing group.
RefundResourceType	RefundResource	Make	**Make**: Allows the user to perform account-level and bill unit-level refunds.

Table 7-1 (Cont.) Authorization Resources

Resource Type	Resource	Actions	Description
RequestInfoResourceType	RequestInfoResource	Add, View, Delete	Use this resource to help record failed REST requests, along with their headers and payloads, in the BRM database. **Add**: Allows the user or client application to use the Billing Care REST API to record failed requests. **View**: Allows the user or client application to use the Billing Care REST API to retrieve failed REST API requests. **Delete**: Allows the user or client to use the Billing Care REST API to delete failed REST API requests.

Table 7-1 (Cont.) Authorization Resources

Resource Type	Resource	Actions	Description
ServiceResourceType	ServiceResource	Cancel, Edit, Inactivate, Make, Reactivate, OfferInactivate, OfferTerminate, OfferReactivate, Terminate, Associate, EditOfferCustomization	**Cancel**: Allows the user to cancel services. **Edit**: Gives the user access to the Asset Details page. **Inactivate**: Allows the user to inactivate services. **Make**: Allows the user to access the **Select** and **Configure** pages of the customer creation wizard. It also exposes the **Purchase** button. **OfferInactivate**: Allows the user to inactivate product and discount offers. **OfferReactivate**: Allows the user to reactivate product and discount offers. **OfferTerminate**: Allows the user to terminate product and discount offers. **Reactivate**: Allows the user to reactivate services. **Terminate**: Allows the user to terminate services. **Associate**: Allows the user to search and associate devices. **EditOfferCustomization** : Allows the user to modify rate customizations.
SubscriptionResourceType	SubscriptionResource	ContractRenew, ContractTerminate, Transit	**ContractRenew**: Allows the user to enable or disable a contract's auto-renewal options. **ContractTerminate**: Allows the user to cancel a customer's contract. **Transit**: Allows the user to transition packages or bundles.

Table 7-1 (Cont.) Authorization Resources

Resource Type	Resource	Actions	Description
TaxExemptionResourceType	TaxExemptionResource	Add, Delete, Modify	**Add**: Allows the user to add tax exemptions whether account has or does not have prior tax exemptions. **Delete**: Allows the user to delete tax exemptions. **Modify**: Allows the user to save changes to tax exemption attributes.
WriteoffResourceType	WriteoffResource	Make	**Make**: Allows the user to write off accounts, bills, and items. Policy on the minimum and maximum write-off amount applies. See Table 7-2 for transaction limits.

Policies on Transaction Limits

Some of the resources listed in Table 7-1 work in combination with transaction limits. For example, a CSR can be authorized to make adjustments but not over a certain amount. System administrators must configure the limits with OPSS.

Table 7-2 lists the attributes that require system administrators to configure transaction limits (values).

Table 7-2 Listing of Transaction Limits (Obligations)

Attribute	Type
Maximum Currency Adjustment Amount	Integer
Minimum Currency Adjustment Amount	Integer
Maximum Noncurrency Adjustment Amount	Integer
Minimum Noncurrency Adjustment Amount	Integer
Maximum Payment Amount	Integer
Maximum Dispute Amount (applies to settle as well)	Integer
Maximum Write-off Amount	Integer
Maximum Refund Issues Amount	Integer
Maximum Refund Settle Amount	Integer

About Auditing

The BRM server software handles auditing of Billing Care activities. The BRM event notification framework captures the audit trail records inside the **/user_activity** storable class. Each audit trail record links the activity with its creator, date, and time. In the audit trail, the identity of the person creating the record is the user name entered in Billing Care at sign-in.

To capture new activity in the audit trail, include the event corresponding to the relevant activity using the **pin_notify** file in BRM. The same instructions apply when excluding events from the audit trail.

Table 7-3 lists all activities preserved in BRM by default. The list is from the **/config/pin_notify** storable class. You can add to or delete from this list.

Table 7-3 Audited List from /config/pin_notify

Task	BRM Event Name (Activity)
Account creation	/event/notification/account/create
Subscription purchase	/event/billing/product/action/purchase
Subscription modification	/event/billing/product/action/modify
Subscription cancellation	/event/billing/product/action/cancel
Updates to bill info (for example, BDOM [billing day of month], billing frequency, accounting type changes)	/event/customer/billinfo/modify
Event adjustment	/event/billing/adjustment/event
Item adjustment	/event/billing/adjustment/item
Account adjustment	/event/billing/adjustment/account
Top up	/event/billing/vouchertopup
Dispute issue	/event/billing/dispute
Dispute settled	/event/billing/settlement/event
Refund	/event/billing/refund
Write-off operation	/event/billing/writeoff
Payment	/event/billing/payment
Credit limit changes	/event/billing/limit, /event/billing/credit
Bill Now	/event/notification/billing/start
Charge sharing group life-cycle operations	/event/group/sharing/charges/create /event/group/sharing/charges/modify /event/group/sharing/charges/delete
Discount sharing group life-cycle operations	/event/group/sharing/discounts/create /event/group/sharing/discounts/modify /event/group/sharing/discounts/delete
Profile (for example, Friends and Family) life-cycle operations	/event/group/sharing/profiles/create /event/group/sharing/profiles/modify /event/group/sharing/profiles/delete
Credit Monitors life-cycle operations	/event/group/sharing/monitor/modify /event/group/sharing/monitor/delete /event/group/sharing/profiles/delete
Account hierarchy operations	/event/group/parent /event/group/member

For information on logging events, including changing the events logged, see "Logging CSR Activity Events" in *BRM System Administrator's Guide*.

Developing Secure Applications for Billing Care

To develop secure applications for Billing Care or extend Billing Care without compromising security, you must control users' access to resources by:

- Adding security controls over new UI features
- Controlling who can access REST services and the limitations of that access

When users sign in, Billing Care calls OPSS and then OPSS provides authorization if appropriate. Additionally, OPSS determines the restraints or obligations of the authorization.

The developer needs to create a web project in NetBeans for the custom Billing Care REST APIs.

Creating a Resource Type with OPSS

To develop secured custom REST APIs or UIs, you need OPSS resource types for authorization. For more information, see "Importing the Billing Care Security Policies to OPSS" in *Billing Care Installation Guide*.

About REST API Authorization

To control the access of custom REST services and operations to authenticated users, define resource types in OPSS as described in "Creating a Resource Type with OPSS".

In custom REST resource operations that require authorization, call **EnforcementUtil.checkAccess()** by passing the required **subject**, **applicationName**, **action**, **resourceType**, **resource**, **Error**, and optional **UIRequestValue** objects as parameters.

UIRequestValue parameters are optional and are used for handling obligations.

For more information, see "Performing Authorization on the REST Framework" in *Billing Care SDK Guide*.

About UI Authorization

After a user successfully signs in to Billing Care, it fetches the grants of all resources and sets it into the global variable **authorizationJSON**.

When opening a page or dialog box, Billing Care gets the grants of resources through the available authorization custom-bindings and then applies the bindings in the respective view model or overlay view model.

For more information, see "Performing Authorization on the UI" in *Billing Care SDK Guide*.

Adding New Resource Types

To add new resource types:

1. In the **CustomConfigurations.xml** file, add the new OPSS resource types:

   In this example, the new resource type **CreditProfileResourceType** is added.

> **Note**
>
> Do not change key values.

```
<keyvals>
  <key>authorizationResourceTypes/key>
  <value>CreditProfileResourceType</value>
    <desc>Add comma separated OPSS Resource Types(values)for authorization.
          Also these resource types must be defined in OPSS.
          Do not change the keys here.
    </desc>
</keyvals>
```

2. Redeploy the customization.

For more information, see "Packaging and Deploying Customizations" in *Billing Care SDK Guide*.

Storing Billing Care Passwords in Oracle Wallet

By default, the Billing Care installer stores sensitive information such as passwords in the Oracle wallet, and the Billing Care application retrieves the passwords from the Oracle wallet. However, if the passwords are also stored in configuration files, the Billing Care application retrieves the passwords from the configuration files. The Billing Care application automatically decrypts the encrypted passwords when retrieving them from the configuration files.

By default, passwords in configuration files are encrypted in the Oracle ZT PKI format. For more information, see "Encrypting Data" in *BRM Developer's Guide*.

> **Note**
>
> To encrypt passwords that are associated with customizations, use the **pin_crypt_app** utility. For details, see "About Encrypting Passwords" in *BRM Developer's Guide*.

Storing Configuration Entries in the Billing Care Wallet

To store a configuration entry for the Billing Care wallet:

1. Go to the *SDK_home*/**BillingCareSDK/samples/Wallet** directory, where *SDK_home* is the Billing Care SDK installation directory.
2. Do one of the following:
   - On Linux, run the following command:

     ```
     java -cp
     '.:oraclepkiLocation:osdtCertLocation:osdtCoreLocation:cetLocation:'com.portal.cet.ConfigEditor -setconf -wallet clientWalletLocation -parameter configEntry -value value
     ```

     where:

     – *oraclepkiLocation* is the path to the **oraclepki.jar** file which contains the APIs that are required for the **wallet.oraclepki.jar** is stored in the *SDK_home*/**BillingCareSDK/samples/Wallet** directory.

- *osdtCertLocation* is the path to the **osdt_cert.jar** file, which contains the JARs that are used by the JAVA PCM library for establishing a TLS connection to BRM. The **osdt_cert.jar** file is stored in the *SDK_home*/**BillingCareSDK/samples/Wallet** directory.
- *osdtCoreLocation* is the path to the **osdt_core.jar** file, which contains the JARs that are used by the JAVA PCM library for establishing a TLS connection to BRM. The **osdt_core.jar** file is stored in the *SDK_home*/**BillingCareSDK/samples/Wallet** directory.
- *cetLocation* is the **cet.jar** file, which contains the APIs that are required for the wallet. The **cet.jar** file is stored in the *SDK_home*/**BillingCareSDK/samples/Wallet** directory.
- *clientWalletLocation* is the path to the Billing Care wallet.
- *configEntry* is the configuration entry in the Billing Care wallet.
- *value* is the appropriate value for the respective entry in the Billing Care wallet.

For example, running the following command with the -**value** parameter stores the **infranet.log.level** as **1** in the Billing Care wallet. If the value exists in the wallet, it is overwritten:

```
java -cp
'.:oraclepki.jar:osdt_cert.jar:osdt_core.jar:cet.jar:'com.portal.cet.ConfigEditor
  -setconf -wallet "/scratch/pin11/wallet" -parameter infranet.log.level -value 1
```

If you run the command without the **-value** parameter, it prompts for the values for the **infranet.connection** entries and stores them in the Billing Care wallet. At the command prompt, enter the values listed in Table 7-4.

Table 7-4 BRM Connection Information

Field	Description
User Name	The user name for connecting to BRM.
Password	The BRM user's password.
Host Name	The IP address or the host name of the machine on which the primary BRM Connection Manager (CM) or CM Master Process (CMMP) is running.
Port Number	The TCP port number of the CM or CMMP on the host computer.
Service Type	The BRM service type.
Service POID Id	The POID of the BRM service.

- On Windows, run the following command:

```
java -cp ".;oraclepkiLocation:osdtCertLocation:osdtCoreLocation:cetLocation"
com.portal.cet.ConfigEditor -setconf -wallet clientWalletLocation -parameter
configEntry -value value
```

For example, running the following command with the **-value** parameter stores the **infranet.log.level** as **1** in the Billing Care wallet:

```
java -cp ".;C:\Program Files (x86)\Portal
Software\BillingCare\lib\oraclepki.jar;C:\Program Files (x86)\Portal
Software\BillingCare\lib\osdt_cert.jar;C:\Program Files (x86)\Portal
Software\BillingCare\lib\osdt_core.jar;C:\Program Files (x86)\Portal
Software\BillingCare\lib\cet.jar" com.portal.cet.ConfigEditor  -setconf -wallet
```

```
"C:\Program Files (x86)\Portal Software\BillingCare\wallet\client" -parameter
infranet.log.level -value 1
```

If you run the command without the **-value** parameter, it prompts for the values for the **infranet.connection** entries and stores them in the Billing Care wallet. At the command prompt, enter the values listed in Table 7-4.

3. Enter the Billing Care client wallet password.

   The value is stored in the Billing Care wallet.

For retrieving stored configuration entries, see "About Oracle Wallet" in *BRM System Administrator's Guide*.

8

Billing Care REST API Security

Learn how to implement the security capabilities supported by Oracle Communications Billing Care REST API. The Billing Care REST API supports stringent authorization and authentication requirements.

Topics in this document:

- About Authentication and Authorization
- Setting Up OAuth with Oracle Identity Cloud Service
- Setting Up OAuth with Oracle Access Management

For more information, see *REST API Reference for Billing Care*.

About Authentication and Authorization

The Billing Care REST API uses the OAuth 2.0 protocol to authenticate a client application's identity and to authorize the client application to access its REST API. It does this by validating an OAuth access token that is passed in the header of the client's HTTP/HTTPS request to the Billing Care REST API.

Your client must pass this OAuth access token in the header of every HTTP/HTTPS request sent to the Billing Care REST API.

To set up authentication and authorization for your client, you can use either Oracle Identity Cloud Service (IDCS) or Oracle Access Management.

Setting Up OAuth with Oracle Identity Cloud Service

You can set up your client application to use OAuth authentication to access the Billing Care REST API. The client application uses the resource owner's credentials to access the resource server.

The high-level steps for setting up OAuth authentication using Oracle Identity Cloud Service (IDCS) includes the following:

1. Creating a Confidential OAuth Application for the Resource Server
2. Creating a Confidential OAuth Application for Your Client Application
3. Encoding the Client's Credentials in Base64 Format
4. Configuring OAuth Settings Using IDCS
5. Storing the Resource Server's Credentials in the Wallet

Afterward, you must request an access token for each client application that submits REST requests to the Billing Care REST API. See "Requesting an OAuth Access Token".

> **Note**
>
> Authentication is required for production systems only. In test systems, you can submit requests without configuring authentication.

Creating a Confidential OAuth Application for the Resource Server

When you create a confidential OAuth application in Oracle Identity Cloud Service (IDCS), it provides you with a client ID and client secret. Your client needs the client ID and client secret to request OAuth access tokens for accessing the Billing Care REST API.

To create a confidential OAuth application in IDCS:

1. In the Identity Cloud Service console, expand the **Navigation Drawer** and then click **Applications**.
2. Click **Add application**.

   The Add Application page appears.
3. Select **Confidential Application** and then click **Launch workflow**.

   The Add Confidential Application page appears.
4. In the **Name** field, enter a name for your application, such as **BCREST**.
5. Click **Next**.
6. In the Resource server configuration section, do the following:
   a. Select the **Configure this application as a resource server now** option.
   b. In the **Access token expiration** field, enter how long in seconds the access token remains valid.
   c. In the **Primary audience** field, enter the primary recipient where the access token of your confidential application is processed.
   d. Select the **Add scopes** option.
   e. In the Scopes table, click **Add**.

      The Add Scope dialog box appears.
   f. In the **Scope** field, enter **BillingCare**.
   g. Click **Add**.
7. In the Client configuration section, select the **Configure this application as a client now** option.
8. In the Authorization section, do the following:
   a. In the **Allowed Grant Types** field, select the **Client credentials** option.
   b. In the **Client Type** field, select the **Confidential** option.
   c. In the **Allowed Operations** field, select the **Introspect** check box.
9. Click **Next**.
10. Under Web tier policy, select the **Skip and do later** option.
11. Click **Finish**.

12. In the Application Added pop-up window, make note of the client ID and client secret. Provide this to the person who needs to generate the OAuth access token.
13. Click **Activate** and then click **Activate application** to confirm the activation.

Creating a Confidential OAuth Application for Your Client Application

When you create a confidential OAuth application in Oracle Identity Cloud Service (IDCS), it provides you with a client ID and client secret. Your client needs the client ID and client secret to request OAuth access tokens for accessing the Billing Care REST API.

To create a confidential OAuth application in IDCS for your client application:

1. In the Identity Cloud Service console, expand the **Navigation Drawer** and then click **Applications**.
2. Click **Add application**.

   The Add Application page appears.
3. Select **Confidential Application** and then click **Launch workflow**.

   The Add Confidential Application page appears.
4. In the **Name** field, enter a name for your application, such as **BCRESTClient1**.
5. Click **Next**.
6. In the Resource server configuration section, select the **Skip for later** option.
7. In the Client configuration section, select the **Configure this application as a client now** option.
8. In the Authorization section, do the following:
   a. In the **Allowed Grant Types** field, select the **Client credentials** option.
   b. In the **Client Type** field, select the **Confidential** option.
   c. Select the **Add resources** option.
   d. In the Resources table, click **Add scope**.

      The Add Scope dialog box appears.
   e. In the **Resource** field, enter **ResourceServerBillingCare**.
   f. Click **Add**.
9. Click **Next**.
10. Under Web tier policy, select the **Skip and do later** option.
11. Click **Finish**.
12. In the Application Added pop-up window, make note of the client ID and client secret. Provide this to the person who needs to generate the OAuth access token.
13. Click **Activate** and then click **Activate application** to confirm the activation.

Encoding the Client's Credentials in Base64 Format

To request OAuth access tokens for accessing the Billing Care REST API, your client application needs the client ID and client secret encoded in Base64 format. Generate a Base64-encoded value of your client ID and client secret joined by a single colon (*ClientID*:*ClientSecret*).

You pass the Base64-encoded value in the header of your HTTP/HTTPS request for an OAuth access code.

Configuring OAuth Settings Using IDCS

To configure OAuth to connect to your BRM server using IDCS:

1. Open the *domain_home***/Infranet.properties** file in a text editor, where *domain_home* is the directory which contains the configuration for the WebLogic Server domain into which Billing Care was installed.
2. Add this entry:

```
verifyWithCertificate=true
idp.vendor=IDCS
idp.url=http(s)://hostname:port/oauth2/rest
idp.resource_server.scope=ResourceServer.ScopeName
idp.certificate.path=PubKeyPath
```

   where:

   - **verifyWithCertificate** is set to true if you want to use the local token validation instead of the network validation.
   - *hostname:port* is the host and port number of the Oracle Identity Cloud Service manager that is running the **oauth2** application.
   - *ResourceServer***.***ScopeName* is the name of the resource server followed by the name of the OAuth2 scope.
   - *PubKeyPath*: The filesystem path to the public key PEM file for signature verification. Required for local validation.
3. Save and close the file.

Storing the Resource Server's Credentials in the Wallet

IDCS needs your resource server's client credentials to validate OAuth authentication requests. To improve security, store the resource server's client ID and client secret in the Oracle wallet so IDCS can retrieve it when needed.

To store the resource server's client credentials in the Oracle wallet:

1. Go to the *BRM_home***/bin** directory.
2. Store the client ID by running this command:

```
java -cp "JarLocationUnderInstallHome/oraclepki.jar:JarLocationUnderInstallHome/
osdt_cert.jar:JarLocationUnderInstallHome/osdt_core.jar:JarLocationUnderInstallHome/
cet.jar" com.portal.cet.ConfigEditor -setconf -wallet WalletLocation -parameter
idp.client.id -value "ClientIdValue"
```

   where:

   - *JarLocationUnderInstallHome* is the directory in which Java is installed in your Billing Care REST API installation.
   - *WalletLocation* is the directory in which the Oracle wallet resides.
   - *ClientIdValue* is the client ID.
3. At the **Enter password for the wallet** prompt, enter your client wallet password.

4. Store the client secret by running this command:

```
java -cp "JarLocationUnderInstallHome/oraclepki.jar:JarLocationUnderInstallHome/
osdt_cert.jar:JarLocationUnderInstallHome/osdt_core.jar:JarLocationUnderInstallHome/
cet.jar" com.portal.cet.ConfigEditor -setconf -wallet WalletLocation -parameter
idp.client.secret [-pwd]
```

5. At the **Enter password for the wallet** prompt, enter your client wallet password.
6. At the **Enter the value** prompt, enter the client secret to store in the wallet.

Requesting an OAuth Access Token

To request an OAuth access token, use cURL to send an HTTP/HTTPS request to the Billing Care REST API authorization endpoint. For example:

```
curl --location --request GET 'http://hostname:port/bcws/webresources/v1.0/
authentication/queryAccessToken' \
     --header 'Authorization: Basic credentials'
```

where:

- *hostname:port* is the IP address or host name and port of the Billing Care REST API server.
- *credentials* is the Base64-encoded value of your IDCS administrator user name and password joined by a single colon (*username*:*password*).

If the request is successful, the token type and access token are returned. For example:

```
{
    "token_type": "Bearer",
    "access_token": "accessToken"
}
```

Your client must pass this OAuth access token in the header of every HTTP/HTTPS request sent to the Billing Care REST API.

Setting Up OAuth with Oracle Access Management

The Oracle Communications Billing Care REST API uses OAuth 2.0 to authenticate requests from clients. Perform the following one-time tasks to configure OAuth and Oracle WebLogic Server for authentication:

1. Configuring OAuth Services
2. Configuring WebLogic Server to Access LDAP Server
3. Configuring OAuth Settings Using Oracle Access Management
4. Requesting OAuth Access Tokens for Client Applications
5. Creating a WebLogic User

Afterward, your client applications can start submitting requests to the Billing Care REST API. See *REST API Reference for Billing Care*.

> **Note**
>
> Authentication is required for production systems only. In test systems, you can submit requests without configuring authentication.

Configuring OAuth Services

To use OAuth 2.0 for authentication, configure your Billing Care REST API services and then register your client application as a trusted client on Oracle Access Manager.

> **Caution**
>
> After you set up OAuth services, you must request an access token for each client application that submits REST requests to the Billing Care REST API. See "Requesting OAuth Access Tokens for Client Applications".

To configure OAuth services using Oracle Access Manager:

1. Create an identity domain, which controls the authentication and authorization of your client applications. It also controls which features your client application can access in relation to the service.

   To create an identity domain, use cURL to send an HTTP/HTTPS request to the Oracle Access Management URL. For example:

   ```
   curl -i
        -H "Content-Type: application/json"
        -H "Accept: application/json"
        -H "Authorization:Basic credentials"
        -X POST
        http(s)://hostname:port/oam/services/rest/ssa/api/v1/oauthpolicyadmin/
   oauthidentitydomain
        -d '{
            "name": "identityDomain",
            "description": "Description for Billing Care REST API Identity
   Domain",
            "tokenSettings":[
                {
                    "tokenType": "ACCESS_TOKEN",
                    "tokenExpiry": tokenExpiry
                }
            ]
        }'
   ```

   where:

   - *credentials* is the Base64-encoded value of your Oracle Access Manager administrator user name and password joined by a single colon (*username*:*password*).
   - *hostname*:*port* is the host and port of the Oracle Access Manager Administration Server.

- *identityDomain* is the name of the Oracle Access Manager identity domain that you want to create.
- *tokenExpiry* is the number of seconds before the access token expires, such as 3600 for one hour. After the token expires, you must request a new access token. See "Requesting OAuth Access Tokens for Client Applications".

See "Add a new OAuth Identity Domain" in *REST API for OAuth in Oracle Access Manager* for more information.

2. Create a resource server, which hosts protected resources and accepts and responds to protected resource requests using access tokens.

   To create and configure your resource server, use cURL to send an HTTP/HTTPS request to the Oracle Access Management URL. For example:

```
curl -i
     -H "Content-Type: application/json"
     -H "Authorization:Basic credentials"
     -X POST
     http(s)://hostname:port/oam/services/rest/ssa/api/v1/oauthpolicyadmin/
application
     -d '{
         "name": "resourceServer",
         "idDomain": "identityDomain",
         "description": "Billing Care REST API Resource Server",
         "scopes":[
             {
                 "scopeName":"scopeName",
                 "description":"All Access"
             }
         ]
     }'
```

   where:

   - *resourceServer* is the name of your resource server, such as BillingCare.
   - *scopeName* is the name of the scope, such as All.

   See "Add a new Resource Server" in *REST API for OAuth in Oracle Access Manager* for more information.

3. Create a client application that makes protected resource requests on behalf of the resource owner and with the resource owner's authorization. Billing Care REST API clients are web applications with an OAuth 2.0 client type of Confidential Client. Clients must use a grant type of Client Credentials for requesting access to Billing Care REST API resources.

   To create a client application, use cURL to send an HTTP/HTTPS request to the Oracle Access Management URL. For example:

```
curl -i
     -H "Content-Type:application/json"
     -H "Authorization:Basic credentials"
     -X POST
     http(s)://hostname:port/oam/services/rest/ssa/api/v1/oauthpolicyadmin/
client
     -d '{
         "secret": "client_secret",
```

```
            "id": "client_id",
            "name": "clientName",
            "scopes": [
                "resourceServer.scopeName"
            ],
            "clientType": "CONFIDENTIAL_CLIENT",
            "idDomain": "identityDomain",
            "description": "Description of client of Billing Care REST API
Server",
            "grantTypes":[
                "CLIENT_CREDENTIALS"
            ],
            "defaultScope": "resourceServer.scopeName",
            "redirectURIs": [
                {
                    "url":"http(s)://BillingCareHost:BillingCarePort/bcws",
                    "isHttps": isHttps
                }
            ]
        }'
```

where:

- *client_secret* is the password for your client.
- *client_id* is the client ID for your client. It is generated automatically if not specified.
- *clientName* is the name of your client.
- *BillingCareHost*:*BillingCarePort* is the host and port of the Billing Care REST API Server.
- *isHttps* is a Boolean value that specifies whether the URL is accessed over HTTPS (**true**) or HTTP (**false**).

See "Add a new OAuth Client" in *REST API for OAuth in Oracle Access Manager* for more information.

Configuring WebLogic Server to Access LDAP Server

To configure your WebLogic Server to access user details on the LDAP server:

1. Update your Java policy store configuration for the Billing Care REST API to access the LDAP server.

   You deploy Billing Care on a JRF-enabled Oracle WebLogic Server domain configured as an Oracle Platform Security Services (OPSS) client domain. Create a JRF-enabled domain in Oracle WebLogic Server and configure an application domain policy that controls access to the Billing Care application.

   You deploy Billing Care on a JRF-enabled Oracle WebLogic Server domain configured as an Oracle Platform Security Services (OPSS) client domain. Create a JRF-enabled domain in Oracle WebLogic Server and configure an application domain policy that controls access to the Billing Care application.

   For creating the domain and configuring the domain policy, see "Configuring the Oracle Fusion Middleware Infrastructure Domain" in *Oracle Fusion Middleware Installing and Configuring the Oracle Fusion Middleware Infrastructure*.

2. In the **config.xml** file, set this entry to **false**. This configures WebLogic Server to skip basic authentication.

```
<domain>
    <security-configuration>
        <!-- Other configurations -->
        <enforce-valid-basic-auth-credentials>false</enforce-valid-basic-
auth-credentials>
    </security-configuration>
</domain>
```

Configuring OAuth Settings Using Oracle Access Management

To configure OAuth to connect to your BRM server using Oracle Access Management:

1. Open the *domain_home*/**Infranet.properties** file in a text editor, where *domain_home* is the directory which contains the configuration for the WebLogic Server domain into which Billing Care was installed.
2. Add these entries:

```
verifyWithCertificate=true
idp.vendor=OAM
idp.url=https://hostname:port/oauth2/rest
idp.resource_server.scope=ResourceServer.ScopeName
idp.identity_domain=IdentityDomain
idp.certificate.path=PubKeyPath
```

   where:

   - **verifyWithCertificate** is set to true if you want to use the local token validation instead of the network validation.
   - *hostname:port* is the host and port number of the Oracle Access Manager Managed Server that is running the **oam_server** application.
   - *ResourceServer*.*ScopeName* is the name of the resource server followed by the name of the OAuth2 scope.
   - *IdentityDomain* is the name of your identity domain.
   - *PubKeyPath*: The filesystem path to the public key PEM file for signature verification. Required for local validation.
3. Save and close the file.

Requesting OAuth Access Tokens for Client Applications

To request OAuth access tokens for your client applications, use cURL to send an HTTP/HTTPS request to the Billing Care REST API authorization endpoint. For example:

```
curl --location --request GET 'http://hostname:port/bcws/webresources/v1.0/
authentication/queryAccessToken' \
     --header 'Authorization: Basic credentials'
```

where:

- *hostname:port* is the IP address or host name and port of the Billing Care REST API server.

- *credentials* is the Base64-encoded value of your IDCS administrator user name and password joined by a single colon (*username*:*password*).

If the request is successful, the token type, expiry time, and access token are returned. For example:

```
{
    "expires_in": 3600,
    "token_type": "Bearer",
    "access_token": "accessToken"
}
```

The token expires after the number of seconds shown in **expires_in**. This value is set while setting up the OAuth identity domain as described in "Configuring OAuth Settings Using Oracle Access Management". After the token expires, you must request a new one.

You use the token in the Authorization header for each REST request with the following format:

```
-H  "Authorization: Bearer accessToken"
```

Creating a WebLogic User

Create a user in Oracle WebLogic Server, setting the user name to the client ID and the password to the client secret.

For more information, see "Create Users" in the *Oracle WebLogic Administration Console Online Help*.

9

BRM REST Services Manager Security

Learn how to implement the security capabilities supported by BRM REST Services Manager. Oracle Communications Billing and Revenue Management (BRM) REST Services Manager supports stringent authorization and authentication requirements.

Topics in this document:

For more information, see *REST Services Manager API for Billing and Revenue Management*.

About Authentication and Authorization

BRM REST Services Manager uses the OAuth 2.0 protocol to authenticate a client application's identity and to authorize the client application to access its REST API and SDK. It does this by validating an OAuth access token that is passed in the header of the client's HTTP/HTTPS request to BRM REST Services Manager.

Your client must pass this OAuth access token in the header of every HTTP/HTTPS request sent to BRM REST Services Manager.

To set up authentication and authorization for your client, you can use either Oracle Identity Cloud Service or Oracle Access Management.

Setting Up OAuth with Oracle Identity Cloud Service

You can set up your client application to use two-legged or three-legged OAuth authentication to access the BRM REST Services Manager API. Three-legged OAuth authentication allows the resource owner to give the client application access to the resource server without sharing its credentials. In two-legged authentication, the client application uses the resource owner's credentials to access the resource server.

To set up three-legged OAuth authentication using Oracle Identity Cloud Service, perform all of the following steps. To set up two-legged OAuth authentication, skip the "3-Legged OAuth Only" sections.

1. Creating Confidential OAuth Applications
2. Creating Roles (Groups) (3-Legged OAuth Only)
3. Assigning Users to Groups (3-Legged OAuth Only)
4. Assigning Roles and Scopes to Application Functions
5. Generating an Authorization Code (3-Legged OAuth Only)
6. Encoding the Client ID and Client Secret in Base64 Format
7. Requesting an OAuth Access Token

If you are using three-legged OAuth authentication, you can refresh your access token after it expires. See "Refreshing OAuth Access Tokens (3-Legged OAuth Only)".

Creating Confidential OAuth Applications

When you create a confidential OAuth application in Identity Cloud Service, it provides you with a client ID and client secret. Your client needs the client ID and client secret to request OAuth access tokens for accessing the BRM REST Services Manager API.

To create a confidential OAuth application in Identity Cloud Service:

1. In the Identity Cloud Service console, expand the **Navigation Drawer** and then click **Identity & Security**.
2. Click **Domains** and select your identity domain,
3. Click **Integrated applications**. and then **Add Application**.
4. In the **Add Application** page, click **Confidential Application**.
5. In the **App Details** page, enter a name for your application, such as **MyClient**, and then click **Submit**.
6. In the Integrated Applications page, click your application name (for example, **MyClient**).
7. In the Application Details page, click **OAuth configuration**.
8. In the OAuth configuration page, click **Edit OAuth Configuration**.
9. In the Edit OAuth configuration page, under Client Configuration, select **Configure this application as a client now**.
10. In the Authorization section, enter the following information:
    - In the **Allowed Grant Types** field, select **Client Credentials**, **JWT Assertion**, **Refresh Token**, and **Authorization Code**.
    - Select the **Allow non-HTTPS URLs** option.
    - In the **Redirect URL** field, enter the application URL where the user is redirected after authentication. Use the format:

      https://*hostname***:***port***/oidc/redirect**

      > **Note**
      >
      > If you are intending to install Collections Configuration Center, this will be the URL for the Collections Configuration Center.

    - If you intend to install Collections Configuration Center, in the **Logout URL** field, enter the application URL where the user is redirected after logging off. Use the format:

      https://*hostname***:***port***/oauth2/v1/userlogout**

    - If you intend to install Collections Configuration Center, in the **Post Logout Redirect URL** field, enter the application URL where the user is redirected after logging off. Use the format:

      https://*hostname***:***port***/**

    - In the **Client Type** field, select **Confidential**.
    - In the **Allowed Operations** field, select **Introspect**.
11. In the Token Issuance Policy section, do the following:

a. In the **Authorized Resources** field, select **Specific**.

b. Select the **Add app roles allowed** option

c. In the Add App Role section, select the following:

- **Application Administrator**
- **Identity Domain Administrator**
- **User Administrator**

d. Click **Add**.

12. In the Resource server configuration section, select **Configure this application as a resource server now**.

    Click **Next**.

13. In the Configure application APIs that need to be OAuth protected section, do the following:

a. In the **Access Token Expiration** field, enter how long (in seconds) the access token remains valid. For example, enter **3600** for 1 hour.

b. Select the **Is Refresh Token Allowed** option.

c. In the **Refresh Token Expiration** field, enter how long (in seconds) the refresh token, which is returned with your access token and is associated with your confidential application, remains valid.

d. In the **Primary Audience** field, enter the primary recipient where the access token of your confidential application is processed.

e. Select the **Add scopes** option, and then in the Scopes section, click **Add**.

f. In the Add Scope dialog box, add the scopes you intend to use. The default scopes are **BillingAgent** and **BillingViewer**. If you intend to use different or additional scopes, you should map them to endpoints in the "Assigning Roles and Scopes to Application Functions" for instructions.

14. Click **Submit**.

15. In the **Application Added** pop-up window, make note of the client ID and client secret. Provide this to the person who needs to generate the OAuth access token.

16. Click **Close**.

17. At the top right of the application page, select the application you created and from the Actions menu, select **Activate** and then click **OK** to confirm the activation.

18. Provide the following to the person who needs to generate the OAuth access token:

- The Identity Cloud Service URLs for generating authorization codes and requesting access tokens. For example:

  `https://idcs_hostname/oauth2/v1/authorize`

  `https://idcs_hostname/oauth2/v1/token`

  where *idcs_hostname* is the host name of the server of your Identity Cloud Service instance

- The redirect URL to send authorization codes and access tokens to
- The client ID and client secret

Creating Roles (Groups) (3-Legged OAuth Only)

If your client application is using three-legged OAuth authentication, users are granted access to the BRM REST Services Manager API through Oracle Identity Cloud Service groups. To grant users access, you first create the groups in Identity Cloud Service as described below and then assign users to those groups as described in "Assigning Users to Groups (3-Legged OAuth Only)".

The sample groups (roles) are **Billing Agent** and **Billing Viewer**. You can create additional roles according to your business requirements. If you create new roles, you must configure those roles to have access to endpoints. See "Assigning Roles and Scopes to Application Functions" for more information.

To create a group in Identity Cloud Service:

1. In the Identity Cloud Service console, expand the **Navigation Drawer** and then click **Identity & Security**.
2. Click **Domains** and select your identity domain,
3. Click **User management**, scroll down to the Groups section, and click **Create group**.
4. In the Add Group dialog box, in the **Name** field, enter the name of the new group. If you are using the default configuration, create both the **Billing Agent** and **Billing Viewer** groups. If you are using custom groups, ensure that you create all of them.
5. Click **Create**.

Assigning Users to Groups (3-Legged OAuth Only)

You can assign users to either the default groups or to any custom groups you create.

To assign users to the appropriate group for accessing the BRM REST Services Manager API:

1. In the Identity Cloud Service console, expand the **Navigation Drawer** and then click **Identity & Security**.
2. Click **Domains** and select your identity domain,
3. Click **User management** and scroll down to the Users section.
4. Select a user that needs access to the BRM REST Services Manager API.
5. In the user's page, select **Groups** at the top.
6. In the Groups page, click **Assign user to group**.
7. In the Assign Groups dialog box, select all of the groups to which the user should belong, for example **Billing Agent** and **Billing Viewer**. For more information about the default groups, see "Scopes and Roles for Accessing REST Services Manager".
8. Click **Assign User**.

Assigning Roles and Scopes to Application Functions

Users are granted access to BRM REST Services Manager through Oracle Identity Cloud Service groups. Groups and Scopes are then assigned to endpoints in the **authorization-policy.yaml** file. It is not necessary to make changes to this file if you use only the default roles and scopes. See "Scopes and Roles for Accessing REST Services Manager" for information about the default roles.

To change the assignment of roles to the REST endpoints:

1. Edit the *BRM_home*/**scripts/authorization-policy.yaml** file.

   The paths and endpoints are listed in the file, along with the for example:

   ```
   - path: "/brm/accountManagement/v5/billingCycleSpecification[{*}]"
     sockets: []
     methods: ["get"]
     action: GetBillingCycleSpecification
     abac:
       policy-validator:
         statement: "${((inRole(user, 'Billing Agent') || inRole(user, 'Billing
   Viewer'))) || ((inScope(subject, 'BillingAgent') || inScope(subject,
   'BillingViewer') || inScope(subject, 'all')))}"
   ```

   In the statement above, the **Billing Agent** and **Billing Viewer** roles (for 3-legged authorization) and **BillingAgent** and **BillingViewer** scopes (for 2-legged authorization) can both access the endpoint.

2. Locate the endpoint definitions that you wish to change and update them with your new logic. You can use the following operators:

   - || (OR)
   - && (AND)
   - ! (NOT)
   - == (equal)
   - != (not equal)

3. Save and close the **authorization-policy.yaml** file.
4. Restart BRM REST Services Manager.

Generating an Authorization Code (3-Legged OAuth Only)

If you are using three-legged OAuth authentication, you must include an authorization code in your request for an OAuth access token. To generate an authorization code, use a browser to send an HTTP/HTTPS request to the Identity Cloud Service URL:

https://*idcs_hostname***/oauth2/v1/authorize?client_id=***client_id***&response_type=code&redirect_uri=***redirect_url***&scope=Billing Agent BillingView all offline_access**

where:

- *idcs_hostname* is the hostname of your Identity Cloud Service instance.
- *client_id* is the client ID generated by Identity Cloud Service when you create a confidential application.
- *redirect_url* is the URL for your application.

The authorization code is returned with the parameter named “code” in the redirect URL. Make a note of the authorization code so it can be used to request OAuth access tokens.

For more information, see "Generate Authorization Code and Identity Token (3-legged OAuth Flow)" in *REST API for Oracle Identity Cloud Service*.

Encoding the Client ID and Client Secret in Base64 Format

Before you can request an OAuth access code, you must encode your client ID and client secret in Base64 format. Generate a base64-encoded value of your client ID and client secret joined by a single colon (*ClientID*:*ClientSecret*).

You pass the base64-encoded value in the header of your HTTP/HTTPS request for an OAuth access code.

Requesting an OAuth Access Token

To request an OAuth access token, use cURL to send an HTTP/HTTPS request to the Identity Cloud Service URL. The cURL syntax you use depends on whether you are using two-legged or three-legged OAuth authentication.

- For two-legged OAuth authentication, use this cURL syntax:

```
curl -i
-H "Authorization: Basic encoded_client"
-H "Content-Type: application/x-www-form-urlencoded;charset=UTF-8"
--request POST https://idcs_hostname/oauth2/v1/token
-d 'grant_type=client_credentials&scope=BillingAgent BillingView all'
```

- For three-legged OAuth authentication, include the authorization code in your HTTP/HTTPS request:

```
curl -i
-H "Authorization: Basic encoded_client"
-H "Content-Type: application/x-www-form-urlencoded;charset=UTF-8"
--request POST https://idcs_hostname/oauth2/v1/token
-d
"redirect_uri=redirect_url&grant_type=authorization_code&code=authorization_code"
```

where:

- *encoded_client* is the base64-encoded client ID and client secret that you created in "Encoding the Client ID and Client Secret in Base64 Format".
- *idcs_hostname* is the hostname of your Identity Cloud Service instance.
- *redirect_url* is the URL for your application.
- *authorization_code* is the authorization code returned in "Generating an Authorization Code (3-Legged OAuth Only)".

After you submit the request, Identity Cloud Service returns an OAuth access token.

Identity Cloud Service also returns a refresh token if the following are true:

- You are using three-legged OAuth authentication.
- Your OAuth client allows the **Refresh Token** grant type.
- You included the **offline_access** scope in your request for an authorization code.

The OAuth access token and refresh token returned from the request follows this syntax:

```
{
   "access_token":"access_token",
   "token_type":"Bearer","expires_in":3600,
   "refresh_token":"refresh_token"
}
```

For more information, see "Generate Access Token and Other OAuth Runtime Tokens to Access the Resource" in *REST API for Oracle Identity Cloud Service*.

Refreshing OAuth Access Tokens (3-Legged OAuth Only)

If you are using three-legged OAuth authentication, you can refresh your access token. To do so, use cURL to send an HTTP/HTTPS request to the Identity Cloud Service URL:

```
curl -H 'Authorization: Basic encoded_client' -H "Content-Type:application/x-
www-form-urlencoded;charset=UTF-8"
--request POST https://idcs_hostname/oauth2/v1/token -d
'grant_type=refresh_token&refresh_token=refresh_token
```

where:

- *encoded_client* is the base64-encoded client ID and client password that you created in "Encoding the Client ID and Client Secret in Base64 Format".
- *idcs_hostname* is the hostname of your Identity Cloud Service instance.
- *refresh_token* is the refresh token that was returned when you requested an access token in "Requesting an OAuth Access Token".

For more information, see "Generate Access Token and Other OAuth Runtime Tokens to Access the Resource" in *REST API for Oracle Identity Cloud Service*.

Setting Up OAuth using Oracle Access Management

Setting up OAuth using Oracle Access Management involves these high-level steps:

1. Installing the Oracle Access Management software. For the list of supported versions, see "Additional BRM Software Requirements" in *BRM Compatibility Matrix*.
   For more information about installing the Oracle Access Management software, see Oracle Fusion Middleware Installing and Configuring Oracle Identity and Access Management.
2. Installing the Oracle Unified Directory software with the HTTP port enabled. For the list of supported versions, see "Additional BRM Software Requirements" in *BRM Compatibility Matrix*.
   For more information about installing Oracle Unified Directory, see Oracle Fusion Middleware Installing Oracle Unified Directory.
3. Enabling OAuth Services
4. Creating an OAuth Identity Domain
5. Creating a Resource Server
6. Creating an OAuth Client
7. Create an access token using either two-legged or three-legged OAuth authentication:

- Using Two-Legged OAuth to Create an Access Token
- Using Three-Legged OAuth to Create an Access Token

> **Note**
>
> If you use both BRM REST Services Manager and PDC REST Services Manager, you must set up separate OAuth identity domains, resource servers, and clients for each component.

Enabling OAuth Services

To enable OAuth services in Oracle Access Manager:

1. In the Oracle Access Manager Console, click **Configuration** at the top of the window.
2. Click **Available Services**.
3. Confirm that the **Access Manager** service is enabled.
4. In the **OAuth Service** row, click **Enable Service**.
5. In the **OpenIDConnect Service** row, click **Enable Service**.

For more information, see "Managing Common Services and Certificate Validation" in *Oracle Fusion Middleware Administering Oracle Access Management*.

Creating an OAuth Identity Domain

You create an OAuth identity domain to control the authentication and authorization of users who can sign in to BRM REST Services Manager, and what features they can access. You create all artifacts, such as the resource server and OAuth client, under the identity domain.

To create an identity domain, use cURL to send an HTTP/HTTPS request to the Oracle Access Management URL:

```
curl -i --header 'Content-Type: application/json'
--header 'Authorization:Basic encoded_admin'
--header 'Cookie:
JSESSIONID=Ax_wYZQ2svzaTYpH5Gwz4KTwqfD2toL1tEi2hzkuTSAK8KVuf0aw!642164469'
--request POST http//:oam_host:oam_port/oam/services/rest/ssa/api/v1/
oauthpolicyadmin/oauthidentitydomain
--data-raw
'{"name":"identity_domain","identityProvider":"identity_store","description":"
My Identity Domain",
{"tokenType":"AUTHZ_CODE","tokenExpiry":3600,"lifeCycleEnabled":false,"refresh
TokenEnabled":false,"refreshTokenExpiry":86400,"refreshTokenLifeCycleEnabled":
false},
{"tokenType":"SSO_LINK_TOKEN","tokenExpiry":3600,"lifeCycleEnabled":false,"ref
reshTokenEnabled":false,"refreshTokenExpiry":86400,"refreshTokenLifeCycleEnabl
ed":false}'
```

where:

- *encoded_admin* is the base64-encoded format of the Oracle Access Management administrator user name and password.

- *oam_host:oam_port* is the host name and port for the Oracle Access Management server.
- *identity_domain* is the name of the Oracle Access Management identity domain that you want to create.
- *identity_store* is set to your Oracle Unified Directory.

The following shows an example cURL command for creating an identity domain named **TMF_ID_Domain** with the identity store set to **OUD**:

```
curl -i --header 'Content-Type: application/json'
--header 'Authorization: Basic encoded_admin'
--header 'Cookie:
JSESSIONID=Ax_wYZQ2svzaTYpH5Gwz4KTwqfD2toL1tEi2hzkuTSAK8KVuf0aw!642164469'
--request POST 'http://oam_host:oam_port/oam/services/rest/ssa/api/v1/
oauthpolicyadmin/oauthidentitydomain'
--data-raw
'{"name":"TMF_ID_Domain","identityProvider":"OUD","description":"My Identity
Domain",
"tokenSettings":
[{"tokenType":"ACCESS_TOKEN","tokenExpiry":3600,"lifeCycleEnabled":false,"refr
eshTokenEnabled":false,"refreshTokenExpiry":86400,"refreshTokenLifeCycleEnable
d":false},
{"tokenType":"AUTHZ_CODE","tokenExpiry":3600,"lifeCycleEnabled":false,"refresh
TokenEnabled":false,"refreshTokenExpiry":86400,"refreshTokenLifeCycleEnabled":
false},
{"tokenType":"SSO_LINK_TOKEN","tokenExpiry":3600,"lifeCycleEnabled":false,"ref
reshTokenEnabled":false,"refreshTokenExpiry":86400,"refreshTokenLifeCycleEnabl
ed":false}],
"errorPageURL":"/oam/pages/servererror.jsp","consentPageURL":"/oam/pages/
consent.jsp"}'
```

If the identity domain was created successfully, you see a response similar to this:

```
Successfully created entity - OAuthIdentityDomain, detail - OAuth Identity
Domain :: Name - TMF_ID_Domain, Id - 19f85bc53b49561ea52f039474c2c4b,
Description - My
Identity Domain, TrustStore Identifiers - TMF_ID_Domain,Identity Provider -
OUD, TokenSettings -
[{"tokenType":"ACCESS_TOKEN","tokenExpiry":3600,"lifeCycleEnabled":false,"refr
eshTokenEnabled":false,"refreshTokenExpiry":86400,"refreshTokenLifeCycleEnable
d":false},
{"tokenType":"AUTHZ_CODE","tokenExpiry":3600,"lifeCycleEnabled":false,"refresh
TokenEnabled":false,"refreshTokenExpiry":86400,"refreshTokenLifeCycleEnabled":
false},
{"tokenType":"SSO_LINK_TOKEN","tokenExpiry":3600,"lifeCycleEnabled":false,"ref
reshTokenEnabled":false,"refreshTokenExpiry":86400,"refreshTokenLifeCycleEnabl
ed":false}],
ConsentPageURL - /oam/pages/consent.jsp, ErrorPageURL -  /oam/pages/
servererror.jsp, CustomAttrs - null
```

Creating a Resource Server

A resource server hosts the protected resources. It must be capable of accepting and responding to resource requests using OAuth access tokens.

To create a resource server, use cURL to send an HTTP/HTTPS request to the Oracle Access Management URL:

```
curl -k -u wls_admin:password -H 'Content-Type: application/json' 'http://
oam_host:oam_port/oam/services/rest/ssa/api/v1/oauthpolicyadmin/application'
-d '{"name":"resource_server","description":"TestResourceServer",
"scopes":[{"scopeName":"scope1","description":"Scope1 description"},
{"scopeName":"scope2","description":"Scope2 description"},
{"scopeName":"scope3","description":"Scope3 description"}],
"tokenAttributes":
[{"attrName":"sessionId","attrValue":"$session.id","attrType":"DYNAMIC"},
{"attrName":"resSrvAttr","attrValue":"RESOURCECONST","attrType":"STATIC"}],"id
Domain":"TestDomain1","audienceClaim":{"subjects":["ab0"]}}'
```

where:

- *wls_admin:password* is the Admin user name and password for WebLogic Server.
- *resource_server* is the name of the resource server that you want to create.
- *scopeN* is the name of a scope. After the scopes are defined under the resource server, refer to them as *resource_server*.*scope*. For example: TMFResourceServer.BillingAgent. For information about the scopes, see "Scopes and Roles for Accessing REST Services Manager".

The following shows an example of creating a resource server named **TMFResourceServer** with the **BillingAgent** and **BillingViewer** scopes, an identity domain named **TMF_ID_Domain**, and static and dynamic customer attributes:

```
curl -k -u wls_admin:password -H 'Content-Type: application/json' 'http://
oam_host:oam_port/oam/services/rest/ssa/api/v1/oauthpolicyadmin/application'
-d '{"name":"TMFResourceServer","description":"Resource Server",
"scopes":[{"scopeName":"BillingAgent","description":"Scope for CSR with
BillingAgent Role"},{"scopeName":"BillingViewer","description":"Scope for CSR
with BillingViewer Role"},
"tokenAttributes":
[{"attrName":"sessionId","attrValue":"$session.id","attrType":"DYNAMIC"},
{"attrName":"resSrvAttr","attrValue":"RESOURCECONST","attrType":"STATIC"}],"id
Domain":"TMF_ID_Domain","audienceClaim":{"subjects":["ab0"]}}'
```

If the resource server is created successfully, you see a response similar to this:

```
Successfully created entity - OAuthResourceServer, detail -
IdentityDomain="TMF_ID_Domain",Name="TMFResourceServer",Description="Resource
Server",
resourceServerId="4953a4f4-8c3f-41fd-99b5-837cfa9f9ecb",resourceServerNameSpac
ePrefix="TMFResourceServer.",audienceClaim="{"subjects":["ab0"]}",
resServerType="CUSTOM_RESOURCE_SERVER",Scopes="[{"scopeName":"BillingAgent","d
escription":"Scope for CSR with BillingAgent Role"},
{"scopeName":"BillingAgent","description":"Scope for CSR with  BillingAgent
Role"}, {"scopeName":"BillingViewer","description":"Scope for CSR with
BillingViewer Role"},}]",
tokenAttributes="[{"attrName":"sessionId","attrValue":"$session.id","attrType"
:DYNAMIC},
{"attrName":"resSrvAttr","attrValue":"RESOURCECONST","attrType":STATIC}]
```

Creating an OAuth Client

To create an OAuth client, use cURL to send an HTTP/HTTPS request to the Oracle Access Management URL:

```
curl -k -u wls_admin:password -H 'Content-Type: application/json' 'http://
oam_host:oam_port/oam/services/rest/ssa/api/v1/oauthpolicyadmin/client'
-d {"attributes":
[{"attrName":"custom_attribute","attrValue":"custom_value","attrType":"static"
}],
"secret":"client_secret","id":"clientID","scopes":
["resource_server.scope1"],"clientType":"CONFIDENTIAL_CLIENT",
"idDomain":"identity_domain","description":"Client
Description","name":"client_name","grantTypes":
["PASSWORD","CLIENT_CREDENTIALS","JWT_BEARER","REFRESH_TOKEN","AUTHORIZATION_C
ODE"],
"defaultScope":"resource_server.scope1","redirectURIs":[{"url":"http://
redirect_host:redirect_port/Sample.jsp","isHttps":true}]}
```

where:

- *custom_attribute* and *custom_value* are custom attribute names and values. You can optionally define a set of static and dynamic custom attributes, which are then added as custom attributes to the OAuth access token.
- *client_id* and *client_secret* are the client ID and client secret.
- *client_name* is the name of the OAuth client that you want to create.
- *redirect_host:redirect_port* is the URL for your client application.

The following shows an example cURL request for creating a confidential OAuth client named **TMFClient** with the **TMFResourceServer:BillingAgent** (default) and **TMFResourceServer:BillingViewer** scopes, and an identity domain named **TMF_ID_Domain**.

```
curl -k -u wls_admin:password -H 'Content-Type: application/json' 'http://
oam_host:oam_port/oam/services/rest/ssa/api/v1/oauthpolicyadmin/client'
-d'{"attributes":[{"attrName":"customAttribute1","attrValue":"Custom
Value1","attrType":"static"},
{"attrName":"customAttribute2","attrValue":"Custom
Value2","attrType":"static"}],
"secret":"client_secret","id":"client_id","scopes":
["TMFResourceServer.BillingAgent","TMFResourceServer.BillingViewer"],"clientTy
pe":"CONFIDENTIAL_CLIENT",
"idDomain":"TMF_ID_Domain","description":"Client entry for identity
domain","name":"TMFClient","grantTypes":
["PASSWORD","CLIENT_CREDENTIALS","JWT_BEARER","REFRESH_TOKEN","AUTHORIZATION_C
ODE"],
"defaultScope":"TMFResourceServer.BillingAgent","redirectURIs":
[{"url":"http://redirect_host:redirect_port/oauth/callback","isHttps":true}]}'
```

If the client is created successfully, the response is similar to this:

```
Successfully created entity - OAuthClient, detail - OAuth Client - uid =
4b37dd63-08dd-45b5-b5a5-c1e788cb2ff2, name = TMFClient, id =  TMFClientId,
```

```
identityDomain = TMF_ID_Domain, description = Client  entry for TMF OAuth
Domain,secret = TMFPassword, clientType = CONFIDENTIAL_CLIENT, grantTypes =
[PASSWORD,
CLIENT_CREDENTIALS, JWT_BEARER, REFRESH_TOKEN, AUTHORIZATION_CODE],
attributes =
[{"attrName":"customAttribute1","attrValue":"Custom
Value1","attrType":STATIC},
{"attrName":"customAttribute2","attrValue":"Custom
Value2","attrType":STATIC},
{"attrName":"sessionId","attrValue":"session.id","attrType":DYNAMIC},
{"attrName":"resSrvAttr","attrValue":"RESOURCECONST","attrType":STATIC}],
scopes =
[TMFResourceServer.BillingAgent, TMFResourceServer.BillingViewer],
defaultScope = TMFResourceServer.BillingAgent, redirectURIs = [{"url":"http://
redirect_host:redirect_port/oauth/callback","isHttps":true}]
```

Using Two-Legged OAuth to Create an Access Token

You create an access token for two-legged OAuth authentication by using the Oracle Access Management OAuth REST API. You submit a request by using its **Create Access Token Flow** endpoint. For more information, see *REST API for OAuth in Oracle Access Manager*.

To request an OAuth access token, use cURL to send an HTTP/HTTPS request to the Oracle Access Management URL:

```
curl -i --header 'Authorization: Basic encoded_admin' \
        --header "Content-Type: application/x-www-form-
urlencoded;charset=UTF-8" \
        --header "X-OAUTH-IDENTITY-DOMAIN-NAME: identity_domain" \
        --request POST http://oam_host:oam_port/oauth2/rest/token \
        --data-
urlencode "grant_type=CLIENT_CREDENTIALS&scope=resource_server.scope"
```

The following shows an example cURL request for creating a creating OAuth access token for the **TMF_ID_Domain** identity domain, **TMFResourceServer** resource server, and **BillingAgent** scope:

```
curl --location --header 'Authorization: Basic encoded_admin' \
     --header "Content-Type: application/x-www-form-urlencoded;charset=UTF-8"
\
     --header "X-OAUTH-IDENTITY-DOMAIN-NAME: TMF_ID_Domain" \
     --request POST http://oam_host:oam_port/oauth2/rest/token \
     --data-urlencode
"grant_type=CLIENT_CREDENTIALS&scope=TMFResourceServer.BillingAgent"
```

If the request is successful, Oracle Access Management returns something similar to this:

```
{
   "access_token":"access_token",
   "token_type":"Bearer",
   "expires_in":3600
}
```

Your client must pass this OAuth access token in the header of every HTTP/HTTPS request sent to the BRM REST Services Manager.

Using Three-Legged OAuth to Create an Access Token

You create an access token for three-legged OAuth authentication by doing the following:

1. Creating protected resources in the Oracle Access Management server. See "Creating Resources in the Oracle Access Management Server".
2. Modifying the WebGate **mod_wl_ohs.conf** file so the Oracle HTTP server can connect to the WebLogic Server. See "Configuring the Oracle HTTP Server".
3. Generating an OAuth access token. See "Generating the OAuth Access Token".

Creating Resources in the Oracle Access Management Server

In three-legged OAuth authentication, you must create resources in the Oracle Access Management server for protecting the consent page and approval page. If you customize the consent page, it must be protected by a WebGate.

To create resources on your Oracle Access Management server:

1. In the Oracle Access Management Console, click **Application Security**.
2. In the Application Security console, click **Application Domains**.
3. Search for and open your application domain.
4. Click the **Resources** tab.
5. Create a protected resource named **/oauth2/rest/approval**. This has to be protected by WebGate.

   Click **Create**, do the following in the **Create Resource** screen, and then click **Apply**:

   - **Resource URL**: Enter **/oauth2/rest/approval**.
   - **Operations Available**: Select **POST**.
   - **Protection Level**: Select **Protected**.
   - **Authentication Policy**: Select **Protected HigherLevel Policy**.
   - **Authorization Policy**: Select **Protected Resource Policy**.
6. Create a protected resource named **/oam/pages/consent.jsp**. If you are using a custom consent page, it needs to be protected by WebGate and the appropriate resource has to be added here.

   Click **Create**, do the following in the **Create Resource** screen, and then click **Apply**:

   - **Resource URL**: Enter **/oam/pages/consent.jsp**.
   - **Operations Available**: Select **GET**.
   - **Protection Level**: Select **Protected**.
   - **Authentication Policy**: Select **Protected Resource Policy**.
   - **Authorization Policy**: Select **Protected Resource Policy**.
7. Create an excluded resource named **/oauth2/rest/****.

   Click **Create**, do the following in the **Create Resource** screen, and then click **Apply**:

   - **Resource URL**: Enter **/oauth2/rest/****.

- **Protection Level**: Select **Excluded**.

8. Create an excluded resource named **/oam/****.

   Click **Create**, do the following in the **Create Resource** screen, and then click **Apply**:

   - **Resource URL**: Enter **/oam/****.
   - **Operations Available**: Select **ALL**.
   - **Protection Level**: Select **Excluded**.

Configuring the Oracle HTTP Server

To configure the Oracle HTTP Server for WebLogic Server, modify the WebGate **mod_wl_ohs.conf** file:

1. Open the *OHS_home***/user_projects/domains/base_domain/config/fmwconfig/components/OHS/***OHS_InstanceName***/mod_wl_ohs.conf** file in a text editor.

   where:

   - *OHS_home* is the path to the Oracle HTTP Server directory.
   - *OHS_InstanceName* is the name of your Oracle HTTP Server instance.

2. Add the following entries to the file:

```
<Location /oauth2>
   SetHandler weblogic-handler
   WebLogicHost  ManagedServerHostName
   WebLogicPort  ManagedServerPort
   ErrorPage  http:/WEBLOGIC_HOME:WEBLOGIC_PORT/
</Location>

<Location /oam>
   SetHandler weblogic-handler
   WebLogicHost  ManagedServerHostName
   WebLogicPort  ManagedServerPort
   ErrorPage  http:/WEBLOGIC_HOME:WEBLOGIC_PORT/
</Location>
```

3. Save and close the file.

Generating the OAuth Access Token

Generating an OAuth access token is a two-step process. First, you retrieve an authorization code from Oracle Access Manager. Then, you include the authorization code in a request to the Oracle Access Manager REST API for an OAuth access token.

To generate an OAuth access token:

1. Open the following URL in a browser:

   http://*OHS_host:OHS_port***/oauth2/rest/authorize?response_type=code&domain=***identity_domain*&client_id=*client_name*&scope=*resource_server*.BillingAgent&state=code1234&redirect_uri=http://***redirect_host:redirect_port***/oauth/callback**

2. Enter your user credentials in the Oracle Access Manager login screen.
3. Click **Allow**.

4. Copy the authorization code from the browser URL.
5. Generate the OAuth access token by submitting a cURL request to the **Create Access Token Flow** endpoint in the Oracle Access Manager OAuth REST API. For example:

```
curl --location --request POST 'http://oam_host:oam_port/14100/oauth2/rest/
token' \
     --header 'X-OAUTH-IDENTITY-DOMAIN-NAME: identity_domain' \
     --header 'Authorization: Basic encoded_admin' \
     --header 'Content-Type: application/x-www-form-urlencoded' \
     --data-urlencode 'grant_type=AUTHORIZATION_CODE' --data-urlencode
'code=authorization_code' \
     --data-urlencode 'redirect_uri=http://redirect_host:redirect_port/
oauth/callback'
```

   For more information, see REST API for OAuth in Oracle Access Manager.

Scopes and Roles for Accessing REST Services Manager

Table 9-1 lists the scopes provided by default to control access to the REST Services Manager API and SDK for two-legged authentication. You can add custom scopes as needed.

Table 9-1 Scopes for BRM REST Services Manager

Scope Name	Description
BillingAgent	Authorizes access to all BRM REST Services Manager APIs.
BillingViewer	Authorizes only GET requests for all BRM REST Services Manager endpoints.

Table 9-2 lists the roles (groups) provided by default to control access to the REST Services Manager API and SDK for three-legged authentication. You can add custom roles as needed.

Table 9-2 Roles for BRM REST Services Manager

Role Name	Description
Billing Agent	Authorizes access to all BRM REST Services Manager APIs.
Billing Viewer	Authorizes only GET requests for all BRM REST Services Manager endpoints.

10

PDC REST Services Manager Security

Learn how to set up security for Oracle Communications Pricing Design Center (PDC) REST Services Manager.

Topics in this document:

For more information, see *PDC REST Services Manager Integration Guide*.

About PDC REST Services Manager Security

PDC REST Services Manager uses the following security protocols to secure inbound and outbound requests:

- **OAuth 2.0**: Authenticates your enterprise product catalog's identity and authorizes it to access the PDC REST Services Manager API by validating an OAuth access token that is passed in the header of every HTTP/HTTPS request to the PDC REST Services Manager API.
  You can enable OAuth for PDC REST Services Manager using either Oracle Identity Cloud Service or Oracle Access Management.
- **TLS**: Secures communication from your enterprise product catalog to PDC REST Services Manager.
- **T3S**: Secures communication from PDC REST Services Manager to PDC.

Setting up PDC REST Services Manager security involves these tasks:

1. One of the following, depending on your OAuth provider:
   - Setting Up OAuth for PDC REST Services Manager with Oracle Identity Cloud Service
   - Setting Up OAuth for PDC REST Services Manager with Oracle Access Management
2. Securing Inbound Communications
3. Securing Outbound Requests to PDC

You can also encrypt sensitive data, such as passwords, by using the **RestServicesManager.sh** script. See "Encrypting Sensitive Data".

Setting Up OAuth for PDC REST Services Manager with Oracle Identity Cloud Service

Setting up OAuth for PDC REST Services Manager with Oracle Identity Cloud Service involves these tasks:

1. Creating Confidential OAuth Applications for PDC REST Services Manager
2. Setting Up Security with Oracle Identity Cloud Service in the PDC REST Services Manager Configuration File
3. Requesting an OAuth Access Token from Oracle Identity Cloud Service

Creating Confidential OAuth Applications for PDC REST Services Manager

Add your enterprise product catalog as a confidential application to IDCS by following the instructions in "Add a Confidential Application" in *Administering Oracle Identity Cloud Service*. When adding the confidential application, ensure that you:

- Select **Confidential** for the **Client Type** option.
- Add a scope named **pubevent** for accessing the Publish Event endpoint in PDC REST Services Manager.
- Add a scope named **metrics** for accessing the Metrics endpoint in PDC REST Services Manager.

After you add the confidential application, Oracle Identity Cloud Service provides you with the following information. You need it when requesting an OAuth access token and when configuring inbound communication to PDC REST Services Manager:

- The Oracle Identity Cloud Service URL for requesting OAuth access tokens. For example:

  ```
  https://idcs_hostname/oauth2/v1/token
  ```

  where *idcs_hostname* is the host name of your Oracle Identity Cloud Service instance
- The primary audience URL
- The client ID and client secret. Encode these in Base64 before using them to request OAuth access tokens.

Setting Up Security with Oracle Identity Cloud Service in the PDC REST Services Manager Configuration File

To set the Oracle Identity Cloud Service details in the PDC REST Services Manager **application.yaml** file:

1. Open the *PDC_RSM_home*/**apps/conf/application.yaml** file in a text editor, where *PDC_RSM_home* is the directory in which you installed PDC REST Services Manager.
2. Set the keys under **security** as shown in Table 10-1.

Table 10-1 Security Keys in the application.yaml File

Key	Description
config.require-encryption	Controls whether requests require encryption using **client_id** and **client_secret**. Set this to **true**.
enabled	Enables or disables security. Enable security in production environments by setting this to **true**.
properties.idcs-uri	The base URL of your Oracle Identity Cloud Service instance in this format: **`https://idcs-`*`TenantID`*`.identity.oraclecloud.com`**
properties.idcs-client-id	The client ID for your confidential application.
properties.idcs-client-secret	The Base64-encrypted client secret obtained from your Oracle Identity Cloud Service application. For security purposes, do not store the client secret in plain text. To encrypt the client secret, see "Encrypting Sensitive Data".
properties.frontend-uri	The base URL of your confidential application at runtime. For example: `http://localhost:8080`
properties.audience	The primary audience as provisioned for the PDC REST Services Manager application in Oracle Identity Cloud Service. For example: `http://localhost:8080/` **Note**: Ensure that you include the trailing slash in the URL.
properties.proxy-host	The host name of the proxy server, if required.
web-server.paths.<0>.abac.scopes	The scope defined in Oracle Identity Cloud Service for protecting the TMF620 **publishEvent** endpoint.
web-server.paths.<1>.abac.scopes	The scope defined in Oracle Identity Cloud Service for protecting the **metrics** endpoint.

3. In the Providers section, ensure that the **oidc** and **abac** providers are not commented out. Comment out the **oamoidc** provider.
4. In the **app.httpClients.security** section, set the keys based on the type of authentication required by your enterprise product catalog. These keys allow you to secure outbound requests from PDC REST Services Manager to your enterprise product catalog. See:
   - OAuth Configuration Properties for Outbound Requests
   - Basic Authentication Configuration Properties for Outbound Requests
5. Save and close the **application.yaml** file.
   See "Example application.yaml Security Configuration with Oracle Identity Cloud Service" for a sample file showing the appropriate properties.
6. Restart PDC REST Services Manager by running the following command from the *PDC_RSM_home*`/apps/bin` directory:
   `./RestServicesManager.sh restart`

Requesting an OAuth Access Token from Oracle Identity Cloud Service

Request an OAuth access token from Oracle Identity Cloud Service to include in requests to the PDC REST Services Manager APIs. For more information, see "Generate Access Token

and Other OAuth Runtime Tokens to Access the Resource" in *REST API for Oracle Identity Cloud Service*.

To request an OAuth access token using cURL, use the following format for your HTTP/HTTPS request to the Oracle Identity Cloud Service URL:

```
curl -i
     -H "Authorization: Basic encoded_credentials" \
     -H "Content-Type: application/x-www-form-urlencoded;charset=UTF-8" \
     --request POST https://idcs_hostname/oauth2/v1/token \
     -d 'grant_type=client_credentials&scope=https://primaryAudience/scope'
```

where:

- *encoded_credentials* is either the client ID and client secret (*clientID*:*clientSecret*) or user name and password (*username*:*password*) in Base64-encoded format.
- *idcs_hostname* is the host name of your Oracle Identity Cloud Service instance.
- *primaryAudience* is the host name and port of your confidential application.
- *scope* is one of the following:
  - **pubevent**: Authorizes access to the Publish Event endpoint.
  - **metrics**: Authorizes access to the Metrics endpoint.

After you submit the request, Oracle Identity Cloud Service returns an OAuth access token. Your client must pass this OAuth access token in the header of every HTTP/HTTPS request sent to the PDC REST Services Manager.

Setting Up OAuth for PDC REST Services Manager with Oracle Access Management

Setting up OAuth for PDC REST Services Manager using Oracle Access Management involves these high-level steps:

1. Installing the Oracle Access Management software. For the list of supported versions, see "Additional BRM Software Requirements" in *BRM Compatibility Matrix*.
   For more information about installing the Oracle Access Management software, see *Oracle Fusion Middleware Installing and Configuring Oracle Identity and Access Management*.
2. Installing the Oracle Unified Directory software with the HTTP port enabled. For the list of supported versions, see "Additional BRM Software Requirements" in *BRM Compatibility Matrix*.
   For more information about installing Oracle Unified Directory, see *Oracle Fusion Middleware Installing Oracle Unified Directory*.
3. Enabling OAuth Services for PDC REST Services Manager
4. Creating an OAuth Identity Domain for PDC REST Services Manager
5. Creating a Resource Server for PDC REST Services Manager
6. Creating an OAuth Client for PDC REST Services Manager
7. Setting Up Security with Oracle Access Management in the PDC REST Services Manager Configuration File
8. Requesting an OAuth Access Token from Oracle Access Management

> **Note**
>
> If you use both BRM REST Services Manager and PDC REST Services Manager, you must set up separate OAuth identity domains, resource servers, and clients for each component.

Enabling OAuth Services for PDC REST Services Manager

Enable OAuth services in Oracle Access Management as described in "Managing Common Services and Certificate Validation" in *Oracle Fusion Middleware Administering Oracle Access Management*. Ensure that the following services are enabled:

- Access Manager
- OAuth
- OpenID Connect

Creating an OAuth Identity Domain for PDC REST Services Manager

You create an OAuth identity domain to control the authentication and authorization of users who can sign in to PDC REST Services Manager, and what features they can access. You create all artifacts, such as the resource server and OAuth client, under the identity domain.

To create an identity domain, submit a request to the **Add a new OAuth Identity Domain** endpoint of the Oracle Access Manager OAuth REST API. See "Add a new OAuth Identity Domain" in *REST API for OAuth in Oracle Access Manager* for more information about this endpoint.

The following shows an example cURL command for creating an identity domain named **PDC_RSM_Domain**, with the **OUD** identity provider (for Oracle Unified Directory):

```
curl -i --header 'Content-Type: application/json'
--header 'Authorization:Basic encoded_admin'
--request POST http://oam_host:oam_port/oam/services/rest/ssa/api/v1/
oauthpolicyadmin/oauthidentitydomain
--data-raw
'{"name":"PDC_RSM_Domain","identityProvider":"OUD","description":"Identity
Domain for PDC REST Services Manager","tokenSettings":[
{"tokenType":"ACCESS_TOKEN","tokenExpiry":3600,"lifeCycleEnabled":false,"refre
shTokenEnabled":false,"refreshTokenExpiry":86400,"refreshTokenLifeCycleEnabled
":false}]'
```

where:

- *encoded_admin* is the Base64-encoded format of the Oracle Access Management administrator user name and password.
- *oam_host:oam_port* is the host name and port for the Oracle Access Management server.

If the identity domain was created successfully, you see a response similar to this:

```
Successfully created entity - OAuthIdentityDomain, detail - OAuth Identity
Domain :: Name - PDC_RSM_Domain, Id - 19f85bc53b49561ea52f039474c2c4b,
Description - Identity Domain for PDC REST Services Manager, TrustStore
```

```
Identifiers - PDC_RSM_Domain,Identity Provider - OUD, TokenSettings -
[{"tokenType":"ACCESS_TOKEN","tokenExpiry":3600,"lifeCycleEnabled":false,"refr
eshTokenEnabled":false,"refreshTokenExpiry":86400,"refreshTokenLifeCycleEnable
d":false}],
ConsentPageURL - /oam/pages/consent.jsp, ErrorPageURL -  /oam/pages/
servererror.jsp, CustomAttrs - null
```

Creating a Resource Server for PDC REST Services Manager

A resource server hosts the protected resources. It must be capable of accepting and responding to resource requests using OAuth access tokens.

To create a resource server, submit a request to the **Add a new Resource Server** endpoint of the Oracle Access Management OAuth REST API. See "Add a new Resource Server" in *REST API for OAuth in Oracle Access Manager* for more information about this endpoint.

The following shows an example of creating a resource server named **PDCRSMResourceServer** with the **all** and **read** scopes, an identity domain named **PDC_RSM_Domain**, and static and dynamic customer attributes:

```
curl -k -u wls_admin:password -H 'Content-Type: application/json' 'http://
oam_host:oam_port/oam/services/rest/ssa/api/v1/oauthpolicyadmin/application'
-d '{"name":"PDCRSMResourceServer","description":"Resource server for PDC
REST Services Manager",
"scopes":[{"scopeName":"all","description":"All permissions"},
{"scopeName":"read","description":"Read permissions"}],
"tokenAttributes":
[{"attrName":"sessionId","attrValue":"$session.id","attrType":"DYNAMIC"},
{"attrName":"resSrvAttr","attrValue":"RESOURCECONST","attrType":"STATIC"}],"id
Domain":"PDC_RSM_Domain","audienceClaim":{"subjects":["ab0"]}}'
```

where:

- *wls_admin:password* is the administrator user name and password for Oracle WebLogic Server.
- *resource_server* is the name of the resource server that you want to create.
- *scopeN* is the name of a scope.

After the scopes are defined under the resource server, refer to them as *resource_server***.***scope* for subsequent tasks, such as creating the OAuth client and requesting an OAuth token. For example, **PDCRSMResourceServer.all**.

If the resource server is created successfully, you see a response similar to this:

```
Successfully created entity - OAuthResourceServer, detail -
IdentityDomain="PDC_RSM_Domain",Name="PDCRSMResourceServer",Description="Resou
rce server for PDC REST Services Manager",
resourceServerId="4953a4f4-8c3f-41fd-99b5-837cfa9f9ecb",resourceServerNameSpac
ePrefix="PDCRSMResourceServer.",audienceClaim="{"subjects":["ab0"]}",
resServerType="CUSTOM_RESOURCE_SERVER",Scopes="[{"scopeName":"all","descriptio
n":"All permissions"},{"scopeName":"read","description":"Read permissions"}]",
tokenAttributes="[{"attrName":"sessionId","attrValue":"$session.id","attrType"
:DYNAMIC},
{"attrName":"resSrvAttr","attrValue":"RESOURCECONST","attrType":STATIC}]
```

Creating an OAuth Client for PDC REST Services Manager

You create an OAuth client for PDC REST Services Manager to authenticate requests.

To create an OAuth client, submit a request to the **Add a new OAuth Client** endpoint of the Oracle Access Management OAuth REST API. See "Add a new OAuth Client" in *REST API for OAuth in Oracle Access Manager* for more information about this endpoint.

The following shows an example cURL request for creating a confidential OAuth client named **PDCRSMClient** with the **PDCRSMResourceServer.all** and default **PDCRSMResourceServer.read** scopes, an identity domain named **PDC_RSM_Domain**, and some custom attributes.

```
curl -k -u wls_admin:password -H 'Content-Type: application/json' 'http://
oam_host:oam_port/oam/services/rest/ssa/api/v1/oauthpolicyadmin/client'
-d'{"attributes":[{"attrName":"customAttribute1","attrValue":"Custom
Value1","attrType":"static"},
{"attrName":"customAttribute2","attrValue":"Custom
Value2","attrType":"static"}],
"secret":"client_secret","id":"client_id","scopes":
["PDCRSMResourceServer.all","PDCRSMResourceServer.read"],"clientType":"CONFIDE
NTIAL_CLIENT",
"idDomain":"PDC_RSM_Domain","description":"PDC RSM OAuth
client","name":"PDCRSMClient","grantTypes":["CLIENT_CREDENTIALS"],
"defaultScope":"PDCRSMResourceServer.read","redirectURIs":[{"url":"http://
redirect_host:redirect_port/oauth/callback","isHttps":true}]}'
```

where:

- *client_id* and *client_secret* are the client ID and client secret.
- *redirect_host:redirect_port* is the URL for your client application.

If the client is created successfully, the response is similar to this:

```
Successfully created entity - OAuthClient, detail - OAuth Client - uid =
4b37dd63-08dd-45b5-b5a5-c1e788cb2ff2, name = PDCRSMClient, id =
PDCRSMClientId,
identityDomain = PDC_RSM_Domain, description = PDC RSM OAuth client, secret =
PDCRSMPassword, clientType = CONFIDENTIAL_CLIENT, grantTypes =
[CLIENT_CREDENTIALS],
attributes = [{"attrName":"customAttribute1","attrValue":"Custom
Value1","attrType":STATIC},
{"attrName":"customAttribute2","attrValue":"Custom
Value2","attrType":STATIC},
{"attrName":"sessionId","attrValue":"session.id","attrType":DYNAMIC},
{"attrName":"resSrvAttr","attrValue":"RESOURCECONST","attrType":STATIC}],
scopes =
[PDCRSMResourceServer.all, PDCRSMResourceServer.read], defaultScope =
PDCRSMResourceServer.read, redirectURIs = [{"url":"http://
redirect_host:redirect_port/oauth/callback","isHttps":true}]
```

Setting Up Security with Oracle Access Management in the PDC REST Services Manager Configuration File

To set the Oracle Access Management details in the PDC REST Services Manager **application.yaml** file:

1. Open the *PDC_RSM_home*/**apps/conf/application.yaml** file in a text editor, where *PDC_RSM_home* is the directory in which you installed PDC REST Services Manager.
2. Set the keys under **security** as shown in Table 10-2.

Table 10-2 Security Keys in the application.yaml File

Key	Description
enabled	Enables or disables security. Enable security in production environments by setting this to **true**.
config.require-encryption	Controls whether requests require encryption using **client_id** and **client_secret**. Set this to **false**.
properties.token-endpoint-uri	The URL for requesting an OAuth token from Oracle Access Management. For example, **http://***oam_host*:*oam_port***/oauth2/rest/token**
properties.introspect-endpoint-uri	The URL for validating an OAuth token from Oracle Access Management. For example, **http://***oam_host*:*oam_port***/oauth2/rest/token/info**
properties.oauth-identity-domain-name	The name of the OAuth identity domain that you created in "Creating an OAuth Identity Domain for PDC REST Services Manager". For example, **PDC_RSM_Domain**.
properties.authorization-endpoint-uri	The URL for authorizing role-based access. PDC REST Services Manager does not support role-based access, so this is not used. For example, **http://***oam_host*:*oam_port***/oauth2/authorize**
properties.frontend-uri	The URL for the OAuth client you created in "Creating an OAuth Client for PDC REST Services Manager". For example, **http://***oam_host*:*oam_port*
properties.proxy-host	The URL for your proxy server, if needed.
properties.audience	The name of the OAuth resource server that you created in "Creating a Resource Server for PDC REST Services Manager". For example, **PDCRSMResourceServer**.
properties.scope-audience	The primary audience for PDC REST Services Manager in the Oracle Access Management resource, used for error handling. This is the same as **properties.frontend-uri**, ending with **/**. For example, **http://***oam_host*:*oam_port***/**
providers.oamoidc.validate_with_jwk	Whether to validate with JSON Web Keys. Set this to **false**.
providers.oamoidc.token-endpoint-uri	The URL for requesting an OAuth token from Oracle Access Management. For example, **http://***oam_host*:*oam_port***/oauth2/rest/token**
providers.oamoidc.authorization-endpoint-uri	The URL for authorizing role-based access. PDC REST Services Manager does not support role-based access, so this is not used. For example, **http://***oam_host*:*oam_port***/oauth2/authorize**

Table 10-2 (Cont.) Security Keys in the application.yaml File

Key	Description
providers.oamoidc.introspect-endpoint-uri	The URL for validating an OAuth token from Oracle Access Management. For example, **http://***oam_host*:*oam_port***/oauth2/rest/token/info**
providers.oamoidc.scope-audience	The primary audience for PDC REST Services Manager in the Oracle Access Management resource. Set this to **"${ALIAS=security.properties.scope-audience}"**.
providers.oamoidc.audience	The name of the OAuth resource server that you created in "Creating a Resource Server for PDC REST Services Manager". For example, **PDCRSMResourceServer**.
providers.oamoidc.proxy-host	The URL for your proxy server, if needed. Set this to **"${ALIAS=security.properties.proxy-host}"**.
providers.oamoidc.frontend-uri	The URL for your application. Set this to **"${ALIAS=security.properties.frontend-uri}"**.
providers.oamoidc.cookie-use	Whether to use cookies. Set this to **false**.
providers.oamoidc.header-use	Whether to use headers. Set this to **true**.
providers.oamoidc.redirect	Whether to use a redirect URL. Set this to **false**.
providers.oamoidc.oidc-metadata-well-known	Whether to use OpenID Connect Discovery metadata. Set this to **false**.
providers.oamoidc.oauth-identity-domain-name	The name of the OAuth identity domain that you created in "Creating an OAuth Identity Domain for PDC REST Services Manager." For example, **PDC_RSM_Domain**.
web-server.paths.methods	The methods allowed for the endpoint. • For the projectPublishEvent endpoint, set this to **["get", "post"]**. • For the metrics endpoint, set this to **["get"]**.
web-server.paths.authenticate	Whether authentication is enabled for the endpoint. Set this to **true**.
web-server.paths.authorize	Whether authorization is enabled for the endpoint. Set this to **true**.
web-server.paths.abac.scopes	The scopes that control access to the endpoint. Use the scopes that you configured in "Creating a Resource Server for PDC REST Services Manager", without the resource server name. For example, **read** or **all**.

3. In the providers section, ensure that the **oamoidc** and **abac** providers are not commented out. Comment out the **oidc** provider.
4. In the **app.httpClients.security** section, set the keys based on the type of authentication required by your enterprise product catalog. These keys allow you to secure outbound requests from PDC REST Services Manager to your enterprise product catalog. See:
   - OAuth Configuration Properties for Outbound Requests
   - Basic Authentication Configuration Properties for Outbound Requests
5. Save and close the **application.yaml** file.
   See "Example application.yaml Security Configuration with Oracle Access Management" for a sample file showing the appropriate properties.

6. Restart PDC REST Services Manager by running the following command from the *PDC_RSM_home*/**apps/bin** directory:
   `./RestServicesManager.sh restart`

Requesting an OAuth Access Token from Oracle Access Management

You create an access token for OAuth authentication by submitting a request to the **Create Access Token Flow** endpoint of the Oracle Access Management OAuth REST API. For more information, see "Create Access Token Flow" in *REST API for OAuth in Oracle Access Manager*.

To request an OAuth access token, use cURL to send an HTTP/HTTPS request to the Oracle Access Management URL:

```
curl -i --header 'Authorization: Basic encoded_admin' \
     --header "Content-Type: application/x-www-form-urlencoded;charset=UTF-8"
\
     --header "X-OAUTH-IDENTITY-DOMAIN-NAME: identity_domain" \
     --request POST http://oam_host:oam_port/oauth2/rest/token \
     --data-
urlencode "grant_type=CLIENT_CREDENTIALS&scope=resource_server.scope"
```

where:

- *encoded_admin* is the base64-encoded format of the Oracle Access Management administrator user name and password.
- *identity_domain* is the name of the OAuth identity domain created in Oracle Access Management for PDC REST Services Manager.
- *oam_host:oam_port* is the host name and port for the Oracle Access Management server.
- *resource_server* is the name of the Oracle Access Management resource server created for PDC REST Services Manager.
- *scope* is the name of a scope.

The following shows an example cURL request for creating an OAuth access token for the **PDC_RSM_Domain** identity domain, **PDCRSMResourceServer** resource server, and **all** scope:

```
curl --location --header 'Authorization: Basic encoded_admin' \
     --header "Content-Type: application/x-www-form-urlencoded;charset=UTF-8"
\
     --header "X-OAUTH-IDENTITY-DOMAIN-NAME: PDC_RSM_Domain" \
     --request POST http://oam_host:oam_port/oauth2/rest/token \
     --data-urlencode
"grant_type=CLIENT_CREDENTIALS&scope=PDCRSMResourceServer.all"
```

If the request is successful, Oracle Access Management returns something similar to this:

```
{
   "access_token":"access_token",
   "token_type":"Bearer","expires_in":3600
}
```

Your client must pass this OAuth access token in the header of every HTTP/HTTPS request sent to PDC REST Services Manager.

Securing Inbound Communications

You secure communications sent from your enterprise product catalog to the PDC REST Services Manager APIs by enabling TLS in PDC REST Services Manager.

To secure inbound communications to PDC REST Services Manager:

1. Create a PKCS12 certificate file.
2. Copy the PKCS12 certificate file to a location that is accessible by PDC REST Services Manager, such as **~/certs**.
3. Edit the following entries in the *PDC_RSM_home***/apps/conf/application.yaml** file:
   - **server.ssl.private-key.keystore-path**: Set this to the file system path of the PKCS12 file containing the X.509 certificate and private key.
   - **server.ssl.private-key.keystore-passphrase**: Set this to the password for the PKCS12 file. For example, if you used OpenSSL to create the PKCS12 certificate file, set it to the export password. For security, encrypt the password so it is not stored in plain text. See "Encrypting Sensitive Data" for more information.

     > **Note**
     >
     > Set the **server.ssl.private-key.keystore-passphrase** key only if the PKCS12 file was created using a password.

     For example:

     ```
     server:
       ...
       ssl:
         private-key:
           keystore-path: "/scratch/ri-user-1/certs/certificate.p12"
           keystore-passphrase: "${passPhrase}"
     ```

4. Restart PDC REST Services Manager by running the following command from the *PDC_RSM_home***/apps/bin** directory:

   ```
   ./RestServicesManager.sh restart
   ```

Securing Outbound Requests to PDC

During installation, the PDC REST Services Manager installer prompts you for the information required to connect PDC REST Services Manager to PDC. To secure the communications from PDC REST Services Manager to PDC, enable the T3S protocol in PDC REST Services Manager.

To enable T3S in PDC REST Services Manager:

1. Go to the *PDC_RSM_home***/apps/conf** directory.
2. In the **application.yaml** file, set the **app.pdc.url** key to the T3S protocol and a secure PDC port.

For example:

```
app:
  pdc:
    url: "t3s://pdc.example.com:8002"
```

3. Restart PDC REST Services Manager by running the following command from the *PDC_RSM_home*/**apps/bin** directory:

   ```
   ./RestServicesManager.sh restart
   ```

If you want to change to the insecure T3 protocol, set the **app.pdc.url** key to the T3 protocol and an insecure PDC port. For example:

```
app:
  pdc:
    url: "t3://pdc.example.com:8001"
```

Encrypting Sensitive Data

You can encrypt sensitive data, such as passwords, by using the **RestServicesManager.sh** script.

To encrypt sensitive data:

1. Go to the *PDC_RSM_home*/**apps/bin** directory, where *PDC_RSM_home* is the directory in which you installed PDC REST Services Manager.
2. Run the following command:

   ```
   ./RestServicesManager.sh hash
   ```

   The **Enter value to hash** prompt appears.
3. Enter the sensitive information that you want to encrypt.

   The encrypted value is displayed.

PDC REST Services Manager Security Configuration Reference Information

The following topics contain reference information about PDC REST Services Manager security configuration properties and sample **application.yaml** configuration files:

- OAuth Configuration Properties for Outbound Requests
- Basic Authentication Configuration Properties for Outbound Requests
- Example application.yaml Security Configuration with Oracle Identity Cloud Service
- Example application.yaml Security Configuration with Oracle Access Management

OAuth Configuration Properties for Outbound Requests

Table 10-3 describes the keys to configure when your enterprise product catalog uses an OAuth 2.0 authentication type. All keys are nested under **app.httpClients.security.oauth2**.

Table 10-3 OAuth 2.0 Keys

Key	Description
tokenEndpoint	The URL for requesting an OAuth token. For example, **http://***host*:*port***/oauth2/rest/token**.
clientId	The client ID used to authenticate the request from PDC REST Services Manager.
clientSecret	The encrypted client secret used to authenticate the request from PDC REST Services Manager. To encrypt the client secret, see "Encrypting Sensitive Data".
scope	The scopes required by the enterprise product catalog. If you are using Oracle Access Management, use the format *resourceServerName.scope*. For example, **ResourceServer.read**. If you are using Oracle Identity Cloud Service, use the format **urn:opc:resource:consumer::***scope*.
grantType	The grant type to be used for the OAuth flow: **client_credentials** or **password**. If you are using Oracle Access Management, only **client_credentials** is supported.
username	The user name required for accessing the enterprise product catalog. Set this only when **grantType** is **password**.
password	The encrypted password required for accessing the enterprise product catalog. To encrypt the password, see "Encrypting Sensitive Data". Set this only when **grantType** is **password**.
domainId	The Oracle Access Management Identity domain. Set this only when using Oracle Access Management.

The following shows an example configuration when **grantType** is **client_credentials**.

```
app:
  httpClients:
    - urlRegex: "local.*:8889"
      security:
        oauth2:
          tokenEndpoint: "http://host:port/oauth2/v1/token"
          clientId: "ClientID"
          clientSecret: "EncryptedClientSecret"
          scope: "https://hostnameurn:opc:resource:consumer::all"
          grantType: "client_credentials"
```

The following shows an example configuration when **grantType** is **password**:

```
app:
  httpClients:
    - urlRegex: "local.*:8889"
      security:
        oauth2:
          tokenEndpoint: "http://host:port/oauth2/v1/token"
          clientId: "ClientID"
          clientSecret: "EncryptedClientSecret"
          scope: "https://hostnameurn:opc:resource:consumer::all"
          grantType: "password"
```

```
        username: "ApplicationUsername"
        password: "EncryptedApplicationPassword"
```

Basic Authentication Configuration Properties for Outbound Requests

Table 10-4 describes the keys to configure when your enterprise product catalog uses a Basic authentication type. All keys are nested under **app.httpClients.security.basicAuth**.

Table 10-4 basicAuth Keys

Key	Description
username	The user name required for accessing the enterprise product catalog.
password	The password required for accessing the enterprise product catalog.

The following shows an example configuration for Basic authentication:

```
app:
  httpClients:
    - urlRegex: "local.*:8889"
      security:
        basicAuth:
          username: "ApplicationUsername"
          password: "Base64EncodedApplicationPassword"
```

Example application.yaml Security Configuration with Oracle Identity Cloud Service

The following shows sample entries in the **application.yaml** file for configuring PDC REST Services Manager OAuth security with Oracle Identity Cloud Service:

```
security:
  config.require-encryption: true
  enabled: true
  properties:
    idcs-uri: "idcsURI"
    idcs-client-id: "clientId"
    idcs-client-secret: ${clientSecret}
    frontend-uri: "http://localhost:8080"
    audience: "http://localhost:8080/"
    proxy-host: ""
  providers:
    - abac:
      # Adds ABAC Provider - it does not require any configuration
    - oidc:
        validate-with-jwk: false
        client-id: "${ALIAS=security.properties.idcs-client-id}"
        client-secret: "${ALIAS=security.properties.idcs-client-secret}"
        identity-uri: "${ALIAS=security.properties.idcs-uri}"
        realm: "pdcrsm"
        audience: "${ALIAS=security.properties.audience}"
        proxy-host: "${ALIAS=security.properties.proxy-host}"
        redirect: false
```

```
        cookie-use: false
        header-use: true
    #- oamoidc:
    #    validate-with-jwk: false
    #    token-endpoint-uri: "http://oam_host:oam_port/oauth2/rest/token"
    #    authorization-endpoint-uri: "http://oam_host:oam_port/oauth2/
authorize"
    #    introspect-endpoint-uri: "http://oam_host:oam_port/oauth2/rest/token/
info"
    #    scope-audience: "${ALIAS=security.properties.scope-audience}"
    #    audience: "PDCRSMResourceServer"
    #    proxy-host: "${ALIAS=security.properties.proxy-host}"
    #    frontend-uri: "${ALIAS=security.properties.frontend-uri}"
    #    redirect: false
    #    cookie-use: false
    #    header-use: true
    #    oidc-metadata-well-known: false
    #    oauth-identity-domain-name: "PDC_RSM_Domain"
  # Comment/Uncomment/Override for protection of resources
  web-server:
    paths:
      - path: "/productCatalogManagement/v1/projectPublishEvent[/{*}]"
        methods: ["get", "post"]
        authenticate: true
        authorize: true
        abac:
          scopes: ["pubevent"]
      - path: "/metrics[/{*}]"
        methods: ["get"]
        authenticate: true
        authorize: true
        abac:
          scopes: ["metrics"]
...
app:
  httpClients:
    - urlRegex: "http://catalog_host:catalog_port/*"
      security:
        oauth2:
          tokenEndpoint: "http://hostname/oauth2/v1/token"
          clientId: "ClientID"
          clientSecret: "EncryptedClientSecret"
          scope: "https://hostnameurn:opc:resource:consumer::all"
          grantType: "client_credentials"
  pdc:
    url: "t3s://pdc_host:secure_pdc_port"
...
server:
  ...
  ssl:
    private-key:
      keystore-path: "file_path/certificate.p12"
      keystore-passphrase: "${passPhrase}"
```

Example application.yaml Security Configuration with Oracle Access Management

The following shows sample entries in the **application.yaml** file for configuring PDC REST Services Manager OAuth security with Oracle Access Management:

```
security:
  config.require-encryption: false
  enabled: true
  properties:
    token-endpoint-uri: "http://oam_host:oam_port/oauth2/rest/token"
    introspect-endpoint-uri: "http://oam_host:oam_port/oauth2/rest/token/info"
    oauth-identity-domain-name: "PDC_RSM_Domain"
    authorization-endpoint-uri: "http://oam_host:oam_port/oauth2/authorize"
    frontend-uri: "http://localhost:8080"
    proxy-host: ""
    audience: "PDCRSMResourceServer"
    scope-audience: "http://localhost:8080/"
  providers:
    - abac:
      # Adds ABAC Provider - it does not require any configuration
    #- oidc:
    #    validate-with-jwk: false
    #    client-id: "${ALIAS=security.properties.idcs-client-id}"
    #    client-secret: "${ALIAS=security.properties.idcs-client-secret}"
    #    identity-uri: "${ALIAS=security.properties.idcs-uri}"
    #    realm: "pdcrsm"
    #    audience: "${ALIAS=security.properties.audience}"
    #    proxy-host: "${ALIAS=security.properties.proxy-host}"
    #    redirect: false
    #    cookie-use: false
    #    header-use: true
    - oamoidc:
       validate-with-jwk: false
    token-endpoint-uri: "http://oam_host:oam_port/oauth2/rest/token"
    authorization-endpoint-uri: "http://oam_host:oam_port/oauth2/authorize"
    introspect-endpoint-uri: "http://oam_host:oam_port/oauth2/rest/token/info"
    scope-audience: "${ALIAS=security.properties.scope-audience}"
       audience: "PDCRSMResourceServer"
       proxy-host: "${ALIAS=security.properties.proxy-host}"
    frontend-uri: "${ALIAS=security.properties.frontend-uri}"
       redirect: false
       cookie-use: false
       header-use: true
    oidc-metadata-well-known: false
    oauth-identity-domain-name: "PDC_RSM_Domain"
  # Comment/Uncomment/Override for protection of resources
  web-server:
    paths:
      - path: "/productCatalogManagement/v1/projectPublishEvent[/{*}]"
        methods: ["get", "post"]
        authenticate: true
        authorize: true
        abac:
```

```
          scopes: ["read", "all"]
      - path: "/metrics[/{*}]"
        methods: ["get"]
        authenticate: true
        authorize: true
        abac:
          scopes: ["read", "all"]
...
app:
  httpClients:
    - urlRegex: "http://catalog_host:catalog_port/*"
      security:
        oauth2:
          tokenEndpoint: "http://oam_host:oam_port/oauth2/rest/token"
          clientId: "EncryptedClientID"
          clientSecret: "EncryptedClientSecret"
          scope: "ResourceServer.all"
          grantType: "client_credentials"
       domainId: "OAM_Domain"

  pdc:
    url: "t3s://pdc_host:secure_pdc_port"
...
server:
  ...
  ssl:
    private-key:
      keystore-path: "file_path/certificate.p12"
      keystore-passphrase: "${passPhrase}"
```

11

ECE REST API Security

Learn how to enable OAuth 2.0 for 5G services and set up security for the Oracle Communications Elastic Charging Engine (ECE) REST API.

Topics in this document:

For more information, see *REST API for Elastic Charging Engine*.

About ECE REST API Security

The ECE REST API uses the OAuth 2.0 security protocol to authenticate a 5G client's identity and authorize it to access the ECE REST API. It does this by validating an OAuth access token that is passed in the header of every HTTP/HTTPS request to the ECE REST API.

The 5G client must pass this OAuth access token in the header of every HTTP/HTTPS request sent to the ECE REST API.

To set up authentication and authorization for the 5G client, you use Oracle Access Management.

About the OAuth 2.0 Flow in HTTP Gateway

The Charging Function (CHF) acts as a resource server for other Network Functions (NF), such as Policy Control Function (PCF) and Session Management Function (SMF). When you enable OAuth 2.0 support, the CHF manages the authentication of incoming requests from the NF.

When PCF and SMF want to communicate with CHF, they must first obtain an authentication token from the OAuth 2.0 authorization server (Network Repository Function). This authorization server provides a JSON Web Token (JWT) with a private key.

The claims in the private key should include:

- NF Instance ID of NRF (the issuer)
- NF Instance ID of the NF Service consumer (subject)
- One or more NF Instance IDs for the requested NF Service producers (audience)
- Expected service names (scope)
- Optionally, additional "scope" information.

Once the NF obtains a token and makes a call to CHF, the JWT should be present in the header. When CHF receives the request, it authenticates the token by verifying the key and the claims in the JWT payload.

The CHF sends the following responses after validating the token:

- If the token validation fails, the CHF will return a **401 Unauthorized** error or a **403 Forbidden** error. This means that the token is missing, invalid, expired, or the scope is wrong.
- If the token is valid, the CHF will return a **200 OK** code with an appropriate response.

Setting Up OAuth for the ECE REST API with Oracle Access Management

Setting up OAuth for the ECE REST API using Oracle Access Management involves these high-level steps:

1. Installing the Oracle Access Management software. For the list of supported versions, see *BRM Compatibility Matrix*.

   For more information about installing the Oracle Access Management software, see *Oracle Fusion Middleware Installing and Configuring Oracle Identity and Access Management*.
2. Installing the Oracle Unified Directory software with the HTTP port enabled. For the list of supported versions, see *BRM Compatibility Matrix*.

   For more information about installing Oracle Unified Directory, see *Oracle Fusion Middleware Installing Oracle Unified Directory*.
3. Enabling OAuth Services for ECE REST API
4. Creating an OAuth Identity Domain for ECE REST API
5. Creating a Resource Server for ECE REST API
6. Creating an OAuth Client for the ECE REST API
7. Requesting an OAuth Access Token from Oracle Access Management

Enabling OAuth Services for ECE REST API

Enable OAuth services in Oracle Access Management as described in "Managing Common Services and Certificate Validation" in *Oracle Fusion Middleware Administering Oracle Access Management*. Ensure that the following services are enabled:

- Access Manager
- OAuth

Creating an OAuth Identity Domain for ECE REST API

You create an OAuth identity domain to control the authentication and authorization of users who can sign in to the ECE REST API, and what features they can access. You create all artifacts, such as the resource server and OAuth client, under the identity domain.

To create an identity domain, submit a request to the **Add a new OAuth Identity Domain** endpoint of the Oracle Access Manager OAuth REST API. See "Add a new OAuth Identity Domain" in *REST API for OAuth in Oracle Access Manager* for more information about this endpoint.

The following shows an example cURL command for creating an identity domain named **ECE_Domain**, with the **OUD** identity provider (for Oracle Unified Directory):

```
curl -i --header 'Content-Type: application/json'
--header 'Authorization:Basic encoded_admin'
--request POST http://oam_host:oam_port/oam/services/rest/ssa/api/v1/
oauthpolicyadmin/oauthidentitydomain
--data-raw
'{"name":"ECE_Domain","identityProvider":"OUD","description":"Identity Domain
for ECE REST API","tokenSettings":[
{"tokenType":"ACCESS_TOKEN","tokenExpiry":3600,"lifeCycleEnabled":false,"refre
shTokenEnabled":false,"refreshTokenExpiry":86400,"refreshTokenLifeCycleEnabled
":false}]'
```

where:

- *encoded_admin* is the Base64-encoded format of the Oracle Access Management administrator user name and password.
- *oam_host:oam_port* is the host name and port for the Oracle Access Management server.

If the identity domain was created successfully, you will see a response similar to this:

```
Successfully created entity - OAuthIdentityDomain, detail - OAuth Identity
Domain :: Name - ECE_Domain, Id - 19f85bc53b49561ea52f039474c2c4b,
Description - Identity Domain for ECE REST API, TrustStore Identifiers -
ECE_Domain,Identity Provider - OUD, TokenSettings -
[{"tokenType":"ACCESS_TOKEN","tokenExpiry":3600,"lifeCycleEnabled":false,"refr
eshTokenEnabled":false,"refreshTokenExpiry":86400,"refreshTokenLifeCycleEnable
d":false}],
```

Creating a Resource Server for ECE REST API

A resource server hosts the protected resources. It must be capable of accepting and responding to resource requests using OAuth access tokens.

To create a resource server, submit a request to the **Add a new Resource Server** endpoint of the Oracle Access Management OAuth REST API. See "Add a new Resource Server" in *REST API for OAuth in Oracle Access Manager* for more information about this endpoint.

The following shows an example of creating a resource server named **NetworkFunctionServiceProducer** with **all** and **read** scopes, an identity domain named **ECE_Domain**, and static and dynamic customer attributes:

```
curl -k -u wls_admin:password -H 'Content-Type: application/json' 'http://
oam_host:oam_port/oam/services/rest/ssa/api/v1/oauthpolicyadmin/application'
-d '{"name":"NetworkFunctionServiceProducer","description":"Resource server
for ECE REST API",
"scopes":[{"scopeName":"all","description":"All permissions"},
{"scopeName":"read","description":"Read permissions"}],
"tokenAttributes":
[{"attrName":"sessionId","attrValue":"$session.id","attrType":"DYNAMIC"},
{"attrName":"resSrvAttr","attrValue":"RESOURCECONST","attrType":"STATIC"}],"id
Domain":"ECE_Domain","audienceClaim":{"subjects":["ab0"]}}'
```

where:

- *wls_admin:password* is the administrator user name and password for Oracle WebLogic Server.
- *resource_server* is the name of the resource server that you want to create.
- *scopeN* is the name of a scope.

After the scopes are defined under the resource server, refer to them as resource_server.scope for subsequent tasks, such as creating the OAuth client and requesting an OAuth token. For example, **NetworkFunctionServiceProducer.all**.

If the resource server is created successfully, you will see a response similar to this:

```
Successfully created entity - OAuthResourceServer, detail -
IdentityDomain="ECE_Domain",Name=" NetworkFunctionServiceProducer.all
",Description="Resource server for ECE REST API",
resourceServerId="4953a4f4-8c3f-41fd-99b5-837cfa9f9ecb",resourceServerNameSpac
ePrefix=" NetworkFunctionServiceProducer.",audienceClaim="{"subjects":
["ab0"]}",
resServerType="CUSTOM_RESOURCE_SERVER",Scopes="[{"scopeName":"all","descriptio
n":"All permissions"},{"scopeName":"read","description":"Read permissions"}]",
tokenAttributes="[{"attrName":"sessionId","attrValue":"$session.id","attrType"
:DYNAMIC},
{"attrName":"resSrvAttr","attrValue":"RESOURCECONST","attrType":STATIC}]
```

Creating an OAuth Client for the ECE REST API

You create an OAuth client for the ECE REST API to authenticate requests.

To create an OAuth client, submit a request to the **Add a new OAuth Client** endpoint of the Oracle Access Management OAuth REST API. See "Add a new OAuth Client" in *REST API for OAuth in Oracle Access Manager* for more information about this endpoint.

The following shows an example cURL request for creating a confidential OAuth client named **NetworkFunctionServiceConsumer** with the **NetworkFunctionServiceProducer.all** and default **NetworkFunctionServiceProducer.read** scopes, an identity domain named **ECE_Domain**, and some custom attributes.

```
curl -k -u wls_admin:password -H 'Content-Type: application/json''http://
oam_host:oam_port/oam/services/rest/ssa/api/v1/oauthpolicyadmin/client'
-d'{"attributes":[{"attrName":"customAttribute1","attrValue":"Custom
Value1","attrType":"static"},
{"attrName":"customAttribute2","attrValue":"Custom
Value2","attrType":"static"}],
"secret":"client_secret","id":"client_id","scopes":["
NetworkFunctionServiceProducer.all "," NetworkFunctionServiceProducer. read
"],"clientType":"CONFIDENTIAL_CLIENT",
"idDomain":"ECE_Domain","description":"ECE RSM OAuth
client","name":"ECERSMClient","grantTypes":["CLIENT_CREDENTIALS"],
"defaultScope":" NetworkFunctionServiceProducer. read ","redirectURIs":
[{"url":"http://redirect_host:redirect_port/oauth/callback","isHttps":true}]}'
```

where:

- *client_id* and *client_secret* are the client ID and client secret.
- *redirect_host:redirect_port* is the URL for your client application.

If the client is created successfully, the response will be similar to this:

```
Successfully created entity - OAuthClient, detail - OAuth Client - uid =
4b37dd63-08dd-45b5-b5a5-c1e788cb2ff2, name = ECERSMClient, id =
ECERSMClientId,
identityDomain = ECE_RSM_Domain, description = ECE RSM OAuth client, secret =
ECERSMPassword, clientType = CONFIDENTIAL_CLIENT, grantTypes =
[CLIENT_CREDENTIALS],
attributes = [{"attrName":"customAttribute1","attrValue":"Custom
Value1","attrType":STATIC},
{"attrName":"customAttribute2","attrValue":"Custom
Value2","attrType":STATIC},
{"attrName":"sessionId","attrValue":"session.id","attrType":DYNAMIC},
{"attrName":"resSrvAttr","attrValue":"RESOURCECONST","attrType":STATIC}],
scopes =
[NetworkFunctionServiceProducer.all, NetworkFunctionServiceProducer. read],
defaultScope = ECERSMResourceServer.read, redirectURIs = [{"url":"http://
redirect_host:redirect_port/oauth/callback","isHttps":true}]
```

Requesting an OAuth Access Token from Oracle Access Management

You create an access token for OAuth authentication by submitting a request to the **Create Access Token Flow** endpoint of the Oracle Access Management OAuth REST API. For more information, see "Create Access Token Flow" in *REST API for OAuth in Oracle Access Manager*.

To request an OAuth access token, use cURL to send an HTTP/HTTPS request to the Oracle Access Management URL:

```
curl -i --header 'Authorization: Basic encoded_admin'
--header "Content-Type: application/x-www-form-urlencoded;charset=UTF-8"
--header "X-OAUTH-IDENTITY-DOMAIN-NAME: identity_domain"
--request POST http://oam_host:oam_port/oauth2/rest/token
--data-urlencode "grant_type=CLIENT_CREDENTIALS&scope=resource_server.scope"
```

where:

- *encoded_admin* is the Base64-encoded format of the Oracle Access Management administrator user name and password.
- *identity_domain* is the name of the OAuth identity domain created in Oracle Access Management for ECE REST API.
- *oam_host:oam_port* is the host name and port for the Oracle Access Management server.
- *resource_server* is the name of the Oracle Access Management resource server created for ECE REST API.
- *scope* is the name of a scope.

The following shows an example cURL request for creating an OAuth access token for the **ECE_domain** identity domain, **NetworkFunctionServiceProducer** resource server, and **all** scope:

```
curl --location --header 'Authorization: Basic encoded_admin'
--header "Content-Type: application/x-www-form-urlencoded;charset=UTF-8"
--header "X-OAUTH-IDENTITY-DOMAIN-NAME: ECE_Domain"
```

```
--request POST http://oam_host:oam_port/oauth2/rest/token
--data-urlencode
"grant_type=CLIENT_CREDENTIALS&scope=NetworkFunctionServiceProducer.all"
```

If the request is successful, Oracle Access Management returns something similar to this:

```
{"access_token":"access_token",
"token_type":"Bearer","expires_in":3600}
```

Your client must pass this OAuth access token in the header of every HTTP/HTTPS request sent to the ECE REST API.

Enabling OAuth 2.0 Authentication in HTTP Gateway

You can enable HTTP Gateway in the application to enable or disable OAuth 2.0 support. You can enable or disable OAuth 2.0 in both on-premises and cloud-native environments.

> **Note**
>
> If you have enabled the OAuth 2.0 functionality, an SSL connection must be present between the Network Functions and the Network Repository Function.

Enabling OAuth 2.0 in On-Premises Systems

You can enable OAuth 2.0 in your HTTP Gateway on-premises system during the ECE installation process or at runtime. To enable it during the ECE installation process, see "Installing a Standalone ECE System" in *ECE Installation Guide*.

To enable OAuth 2.0 in an on-premises environment while ECE is running:

1. Access the ECE configuration MBeans in a JMX editor, such as JConsole.

   See "Accessing ECE Configuration MBeans" in *ECE Implementing Charging*.
2. Expand the **ECE Configuration** node.
3. Expand **charging.httpGatewayConfigurations**.
4. Expand **Attributes**.
5. Specify values for the following fields:
   - **nrfPublicKeyLocation**: Enter the path to the public key of the Network Repository Function.
   - **nrfJwtAlgorithm**: Enter the algorithm used by the Network Repository Function to create the JSON Web Token (JWT).
6. Expand **charging.HttpGatewayConfigurations.***name*, where *name* is the name of the HTTP Gateway instance.
7. Expand **Attributes**.
8. Set the **oauth2Enabled** attribute to **true**. If **OAuth2enabled** is set to **true**, you must send an access token in the header.

> **Note**
>
> When you make access tokens by asymmetric cryptography, the following JWT Algorithms are supported:
>
> - RSA
> - ECDSA

Enabling OAuth 2.0 in Cloud Native Deployments

To enable OAuth 2.0 in cloud native deployments, do the following:

1. Move the public key to the **oc-cn-ece-helm-chart/secrets/httpgateway** directory.

   After deployment, the public key is mounted in the **httpgateway** pod as a Kubernetes Secret.

2. Set the following keys in your **override-values.yaml** file for **oc-cn-ece-helm-chart**:

   - **httpgateway.httpgatewayList.httpGatewayConfiguration.oauth2Enabled**: Set this to **true**. If **oauth2Enabled** is set to **true**, you must ensure that all incoming requests have an access token in the header. If the access token is missing, you will receive an error message.
   - **httpgateway.nrfPublicKeyLocation**: Set this to the name of the public key file.
   - **httpgateway.nrfJwtAlgorithm**: Set this to the algorithm used to create the Network Repository Function public key. The default is **RSA**.

3. Run the **helm install** command to deploy the ECE Helm chart:

   ```
   helm install EceReleaseName oc-cn-ece-helm-chart --namespace BrmNameSpace
   --values OverrideValuesFile
   ```

   where:

   - *EceReleaseName* is the release name for **oc-cn-ece-helm-chart** and is used to track this installation instance. It must be different from the one used for the BRM Helm chart.
   - *BrmNameSpace* is the namespace in which BRM Kubernetes objects reside for the BRM Helm chart.
   - *OverrideValuesFile* is the path to a YAML file that overrides the default configurations in the **oc-cn-ece-helm-chart/values.yaml** file.

12

Business Operations Center Security

Learn how to install and implement Oracle Communications Business Operations Center and its components in a secure configuration.

Topics in this document:

About Installing Business Operations Center

Before installing Business Operations Center, you must properly install and configure several Oracle products, including Java, Oracle WebLogic Server, Oracle Identity and Access Management components, and Oracle Communications Billing and Revenue Management.

For installation instructions, including all the required products and related tasks, such as setting up KeyStores and SSL for WebLogic Server, see "Installing Business Operations Center" in *Business Operations Center Installation Guide*.

About Implementing Business Operations Center Security

Business Operations Center supports stringent authorization and authentication requirements. This section describes how to implement the security capabilities supported by Business Operations Center.

About Identity and Access Management

To authenticate users when they log in and to control user access to functionality, Business Operations Center uses the following Oracle Identity and Access Management components in a production environment:

- Oracle Identity Cloud Service (IDCS)
- Oracle Identity Manager for authentication
- Oracle Platform Security Services (OPSS) for authorization

These components are required in a Business Operations Center implementation.

For more information, see the following documentation:

- Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager
- Oracle Fusion Middleware Administrator's Guide for Oracle Platform Security Services

About Authentication

Authentication is the process of verifying the identity of a user. The Business Operations Center authentication scheme is designed for deployments in which a central user identity repository, storing all enterprise users, authenticates Business Operations Center sign-in requests.

Business Operations Center supports the following authentication options:

- Authenticating users against an LDAP-based user ID repository
- Enabling single sign-on (SSO) capabilities
- Supporting users' password policies

Oracle Identity Manager manages user password policies. For more information, see *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Manager*.

About Authorization

Authorization is the process of granting users access to privileges (entitlements) appropriate for their job functions while denying access to other functionality. Oracle Platform Security Services handles all authorization tasks for Business Operations Center.

A user who has not been granted any entitlements in Oracle Platform Security Services is denied access to Business Operations Center.

To grant entitlements, you use authorization policies, which contain a collection of the following components combined to form a logical entitlement:

- **Resource type:** Specifies the full scope of traits for a resource, such as job execution history, and defines all actions that can be performed on the resource.
- **Resource:** Represents the aspect of an application's functionality being secured, such as billing, payment collection, and invoicing. Each resource must belong to a resource type.
- **Action:** Represents an operation that can be performed on a resource, such as view, create, modify, delete, history, and timeline.

You map authorization policies to enterprise (external) roles, which represent job functions for the users in your company. If you do not map enterprise roles to authorization policies, you must map each user to an authorization policy.

For more information about authorization policies and enterprise roles, see *Oracle Fusion Middleware Administrator's Guide for Oracle Platform Security Services*.

Business Operations Center includes an authorization policy component file (**system-jazn-data.xml**), which defines all the resource types, resources, and actions available for Business Operations Center authorization policies (see Table 12-1).

Table 12-1 Business Operations Center Authorization Policy Components

Resource Type	Resource	Action	Description
Metrics	Subscribers	View	Permits users to view subscriber metrics.
Metrics	Subscriptions	View	Permits users to view subscription metrics.

Table 12-1 (Cont.) Business Operations Center Authorization Policy Components

Resource Type	Resource	Action	Description
Metrics	Billed Revenue	View	Permits users to view billed-revenue metrics.
Metrics	Payments Received	View	Permits users to view payments-received metrics.
Metrics	AR	View	Permits users to view accounts receivable in the dashboard.
Job	Billing GL Invoicing PaymentCollection PricingSync Refund Workflow	Create View Modify Delete Timeline History	**Create**: You can create a new job. **View**: You can view job categories and the jobs for this category. **Modify**: You can edit, deactivate, or reactivate jobs. **Delete**: You can delete jobs. **Timeline**: You can view the timeline. **History**: You can view jobs in history.
PaymentFailures	PaymentFailures	View Resolve	**View**: You can view real-time checkpoints, unresolved batches and unresolved payments, and failure report for Payment Collections jobs. **Resolve**: You can resolve unresolved payments.
BlackoutPeriod	BlackoutPeriod	View Create Delete	**View**: You can view the blackout period in the timeline. **Create**: You can create the blackout period in the timeline. **Delete**: You can remove the blackout period in the timeline.
Job	Custom	Create View Modify Delete	**Create**: You can create custom categories in Business Operations Center. **View**: You can only view custom categories in Business Operations Center.
Job	VirtualTime	View Modify	**View**: You can view the modified pin virtual time in the Manage Virtual Time banner. **Modify**: You can change the pin virtual time and date from the jobs actions menu or manage the virtual time banner.
Any	Any	Any	Permits users to perform all operations.

Table 12-1 (Cont.) Business Operations Center Authorization Policy Components

Resource Type	Resource	Action	Description
Job	**category**_*customcategory1*_**resource** Where: • **category** is a prefix you should add for each resource name. • *customcategory1* is the name of the resource of the custom category entered when you create a new category. • **resource** is the suffix that you should add to the resource name. For example, category_custom_billing_resource. For information about creating a resource name, see "Defining a Custom Category" in *Business Operations Center Online Help*.	Create View Modify Delete Timeline History	**Create**: You can create a new job. **View**: You can view job categories and the jobs for this category. **Modify**: You can edit, deactivate, or reactivate jobs. **Delete**: You can delete jobs. **Timeline**: You can view the timeline. **History**: You can view jobs in history.

The **system-jazn-data.xml** file also includes the following sample authorization policies:

- OperationsAdminPolicy
- FinancialsAdminPolicy
- FullAdminPolicy

The file is located in the *Domain_home***/lib/oes_config** directory, where *Domain_home* is the WebLogic Server domain home directory location of the Oracle Platform Security Services client domain in which Business Operations Center is deployed.

> **Note**
>
> Do not change the **system-jazn-data.xml** file.

Creating Authorization Policies for Business Operations Center

To create authorization policies for Business Operations Center:

1. Import the Business Operations Center authorization policy component file: *Domain_home***/lib/oes_config/system-jazn-data.xml**.

   For detailed instructions, see "Importing the Business Operations Center Operations Security Policies into OPSS" in *Business Operations Center Installation Guide*.

2. In Oracle Platform Security Services (OPSS), map an authorization policy to one or more resources, which may have one or more actions.

For more information, see *Oracle Fusion Middleware Administrator's Guide for Oracle Platform Security Services*.

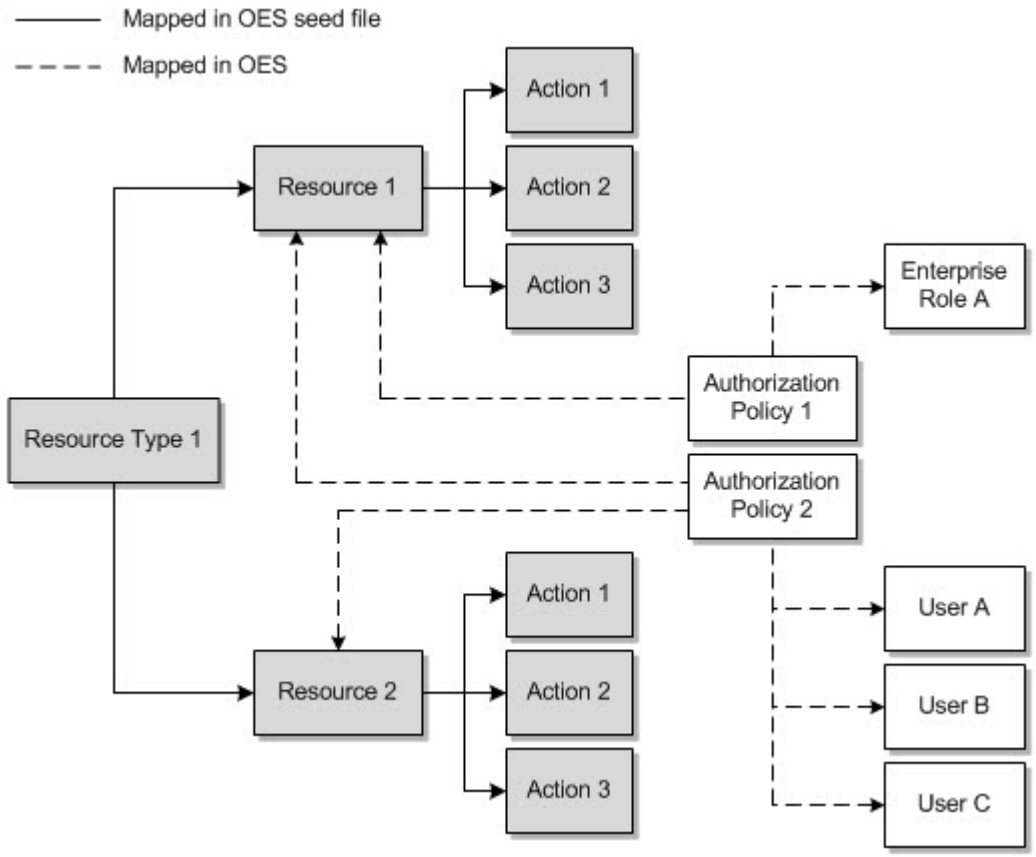
3. Associate the authorization policy with a user or an enterprise role.

   For more information, see *Oracle Fusion Middleware Administrator's Guide for Oracle Platform Security Services*.

4. Redeploy all changes made in OPSS.

Figure 12-1 shows how authorization policies are mapped to resources and enterprise roles or users.

Figure 12-1 Mapping Authorization Policies to Resources and Enterprise Roles or Users



Custom Job Resource Authorization

To grant a user access to a custom job resource in Business Operations Center:

1. Modify the following entries in *Domain_home*/**lib/bocauth-config.properties**:

   ```
   JOB_BOC_ADMIN_HAS_ACCESS_TO_RESOURCE = category_customName_resource

   ACTIONS_GRANTED_FOR_category_customName_resource =
   View,Create,Modify,Delete,Timeline,History
   ```

   where *customName* is the name of the custom category that has been created or is yet to be created, displayed in Business Operations Center.

2. Restart the server to apply the authorization changes.

Storing Business Operations Center Passwords in Oracle Wallet

By default, the Business Operations Center installer stores sensitive information such as passwords in the Oracle wallet and the Business Operations Center application retrieves the passwords from the Oracle wallet. However, if the passwords are also stored in the configuration files, the Business Operations Center application retrieves the passwords from the configuration files. The Business Operations Center application automatically decrypts the encrypted passwords when retrieving them from the configuration files.

By default, the passwords in the configuration files are encrypted in the Oracle ZT PKI format. For more information, see "Encrypting Data" in *BRM Developer's Guide*.

> **Note**
>
> To encrypt passwords that are associated with customizations, use the **pin_crypt_app** utility. For details, see "About Encrypting Passwords" in *BRM Developer's Guide*.

Storing Configuration Entries in the Business Operations Center Wallet

To store a configuration entry for the Business Operations Center wallet:

1. Go to the *BOC_home***/wallet/client** directory, where *BOC_home* is the directory in which Business Operations Center is installed.
2. Do one of the following:
   - On Linux, run the following command:

     ```
     java -cp
     '.:oraclepki.jar_location:osdt_cert.jar_location:osdt_core.jar_location:cet.jar_l
     ocation' com.portal.cet.ConfigEditor -setconf -wallet clientWalletLocation -
     parameter configEntry -value value
     ```

     where:

     - *oraclepki.jar_location* is the path to the **oraclepki.jar** file, which contains the APIs that are required for the wallet. The **oraclepki.jar** file is stored in the *BOC_home***/lib** directory.
     - *osdt_cert.jar_location* is the path to the **osdt_cert.jar** file, which contains the JARs that are used by the Java PCM library for establishing a TLS connection to BRM. The **osdt_cert.jar** file is stored in the *BOC_home***/lib** directory.
     - *osdt_core.jar_location* is the path to the **osdt_core.jar** file, which contains the JARs that are used by the Java PCM library for establishing a TLS connection to BRM. The **osdt_core.jar** file is stored in the *BOC_home***/lib** directory.
     - *cet.jar_location* is the path to the **cet.jar** file, which contains the APIs that are required for the wallet. The **cet.jar** file is stored in the *BOC_home***/lib** directory.
     - *clientWalletLocation* is the path to the Business Operations Center wallet.
     - *configEntry* is the configuration entry in the Business Operations Center wallet.

– *value* is the appropriate value for the respective entry in the Business Operations Center wallet.

For example, running the following command with the -**value** parameter stores the **infranet.log.level** as **1** in the Business Operations Center wallet. If the value exists in the wallet, it is overwritten:

```
java -cp '.:oraclepki.jar:osdt_cert.jar:osdt_core.jar:cet.jar:'
com.portal.cet.ConfigEditor  -setconf -wallet "/scratch/pin11/wallet" -parameter
infranet.log.level -value 1
```

If you run the command without the **-value** parameter, it prompts for the values for the **infranet.connection** entries and stores them in the Business Operations Center wallet. At the command prompt, enter values listed in Table 12-2.

Table 12-2 BRM Connection Information

Field	Description
User Name	The user name for connecting to BRM.
Password	The BRM user's password.
Host Name	The IP address or the host name of the machine on which the primary BRM Connection Manager (CM) or CM Master Process (CMMP) is running.
Port Number	The TCP port number of the CM or CMMP on the host computer.
Service Type	The BRM service type.
Service POID Id	The POID of the BRM service.

- On Windows, run the following command:

```
java -cp
".;oraclepki.jar_location:osdt_cert.jar_location:osdt_core.jar_location:cet.jar_l
ocation" com.portal.cet.ConfigEditor -setconf -wallet clientWalletLocation -
parameter configEntry -value value
```

For example, running the following command with the **-value** parameter stores the **infranet.log.level** as **1** in the Business Operations Center wallet:

```
java -cp ".;C:\Program Files (x86)\Portal
Software\BOC_HOME\lib\oraclepki.jar;C:\Program Files (x86)\Portal
Software\BOC_HOME\lib\osdt_cert.jar;C:\Program Files (x86)\Portal
Software\BOC_HOME\lib\osdt_core.jar;C:\Program Files (x86)\Portal
Software\BOC_HOME\lib\cet.jar" com.portal.cet.ConfigEditor  -setconf -wallet
"C:\Program Files (x86)\Portal Software\BOC_HOME\wallet\client" -parameter
infranet.log.level -value 1
```

If you run the command without the **-value** parameter, it prompts for the values for the **infranet.connection** entries and stores them in the Business Operations Center wallet. At the command prompt, enter values listed in Table 12-2.

3. Enter the Business Operations Center client wallet password.

   The value is stored in the Business Operations Center wallet.

To retrieve stored configuration entries, see "About Oracle Wallet" in *BRM System Administrator's Guide*.

13

Collections Configuration Center Security

Learn how to install and implement Oracle Communications Collections Configuration Center and its components in a secure configuration.

Topics in this document:

About Installing Collections Configuration Center

Before installing Collections Configuration Center, ensure that Java, Oracle Identity and Access Management components, and Oracle Communications Billing and Revenue Management (BRM) REST Services Manager are installed and configured.

For installation instructions, including all required products and related tasks, such as setting up KeyStores and SSL, see *Collections Configuration Center Installation Guide*.

About Implementing Collections Configuration Center Security

Collections Configuration Center adheres to strict authorization and authentication requirements. This section outlines how to implement its supported security features.

> **Note**
>
> This section provides information specific to implementations that use Oracle Identity Cloud Service (IDCS). You can use another OpenID Connect (OIDC) provider. If you do, consult your provider's documentation for implementation information.

Topics in this section:

About Identity and Access Management

To authenticate users when they log in and to control user access to functionality, Collections Configuration Center uses the following Oracle Identity and Access Management components in a production environment:

- Oracle Identity Cloud Service (IDCS)
- BRM REST Services Manager for authentication and authorization enforcement

These components are required for a Collections Configuration Center implementation.

For more information, see the following documentation:

- *Oracle Fusion Middleware Administering Oracle Identity Governance*
- *REST Services Manager API for Billing and Revenue Management*

About Authentication

Authentication verifies a user's identity. The authentication scheme used by Collections Configuration Center is designed for environments where a central user identity repository, containing all enterprise users, authenticates sign-in requests.

Collections Configuration Center supports Single sign-on (SSO).

About Authorization

Authorization grants users privileges (entitlements) appropriate for their job functions while denying access to other functionality. BRM REST Services Manager handles authorization for Collections Configuration Center. Users without entitlements are denied access to Collections Configuration Center. For information about the default roles available, see "About Roles for Accessing Collections Configuration Center Functions".

Storing Collections Configuration Center Passwords in Oracle Wallet

By default, the Collections Configuration Center installer stores sensitive information, such as passwords, in the Oracle wallet. The Collections Configuration Center application retrieves these passwords from the wallet. If passwords are also stored in configuration files, the application retrieves them from configuration files by default.

When retrieving passwords from configuration files, the application automatically decrypts them.

Typically, passwords in configuration files are encrypted by using the Oracle ZT PKI format. For more information, see "Encrypting Data" in *BRM Developer's Guide*.

> **Note**
>
> To encrypt passwords associated with customizations, use the **pin_crypt_app** utility. For more details, see "About Encrypting Passwords" in *BRM Developer's Guide*.

Setting Up OAuth with Oracle Identity Cloud Service

Collections Configuration Center uses the OAuth 2.0 protocol to authenticate a user's identity and to authorize the user to access its features. Authentication is handled through BRM REST Services Manager.

To set up authentication and authorization for your client, you must use Oracle Identity Cloud Service to perform the following high-level steps::

1. Creating Roles (Groups)
2. Assigning Users to Groups
3. Assigning Roles to Application Functions
4. Encoding the Client ID and Client Secret in Base64 Format

Creating Roles (Groups)

Users are granted access to the Collections Configuration Center through Oracle Identity Cloud Service groups. To grant users access, you first create the groups in Identity Cloud Service as described below and then assign users to those groups as described in "Assigning Users to Groups".

The sample groups (roles) are **Billing Viewer**, **CreateAccess**, **DeleteAccess**, and **UpdateAccess**. You can create additional roles according to your business requirements. If you create new roles, you must configure those roles to have access to endpoints. See "Assigning Roles to Application Functions" for more information.

To create a group in Identity Cloud Service:

1. In the Identity Cloud Service console, expand the **Navigation Drawer** and then click **Identity & Security**.
2. Click **Domains** and select your identity domain,
3. Click **User management**, scroll down to the Groups section, and click **Create group**.
4. In the Add Group dialog box, in the **Name** field, enter the name of the new group. If you are using the default configuration, create the **Billing Viewer**, **CreateAccess**, **DeleteAccess**, and **UpdateAccess** groups. If you are using custom groups, ensure that you create all of them.
5. Click **Create**.

Assigning Users to Groups

You can assign users to either the default groups or to any custom groups you create.

To assign users to the appropriate group for accessing Collections Configuration Center:

1. In the Identity Cloud Service console, expand the **Navigation Drawer** and then click **Identity & Security**.
2. Click **Domains** and select your identity domain.
3. Click **User management** and scroll down to the Users section.
4. Select a user that needs access to Collections Configuration Center.
5. In the user's page, select **Groups** at the top.
6. In the Groups page, click **Assign user to group**.
7. In the Assign Groups dialog box, select all groups the user should belong to. For more information about the default groups, see "About Roles for Accessing Collections Configuration Center Functions".
8. Click **Assign User**.

Assigning Roles to Application Functions

Users are granted access to Collections Configuration Center through Oracle Identity Cloud Service groups. You then assign group to endpoints in the **authorization-policy.yaml** file. You do not need to change this file if you use only the default roles. See "About Roles for Accessing Collections Configuration Center Functions" for information about the default roles.

To change the assignment of roles to the application functions:

1. Edit the *BRM_home*/**scripts/authorization-policy.yaml** file.

   The paths and endpoints are listed in the file. For example:

   ```
   - path: "/brm/collections/configuration/v5/aging-bucket[{*}]"
     sockets: [ ]
     methods: [ "post" ]
     action: createAgingBucket
     abac:
       policy-validator:
         statement: "${(inRole(user, 'CreateAccess') && inRole(user, 'Billing
   Viewer'))}"
   ```

   In the statement above, to create an aging bucket, a user must belong to both **CreateAccess** and **Billing Viewer** groups.

2. Locate the application functions that you want to change and update them with your new logic. You can use the following operators:
   - || (OR)
   - && (AND)
   - ! (NOT)
   - == (equals)
   - != (not equal)
3. Save and close the **authorization-policy.yaml** file.
4. Restart BRM REST Services Manager.

Encoding the Client ID and Client Secret in Base64 Format

Before you can request an OAuth access token, you must encode your client ID and client secret in Base64 format. Generate a Base64-encoded value of your client ID and client secret joined by a single colon (*ClientID*:*ClientSecret*).

You pass the Base64-encoded value in the header of your HTTP/HTTPS request for an OAuth access token.

About Roles for Accessing Collections Configuration Center Functions

Table 13-1 lists the roles (groups) provided by default to control access to Collections Configuration Center functions. You can add custom roles as needed.

Table 13-1 Roles for Collections Configuration Center

Role Name	Description
Billing Viewer	This role allows users to view information in the UI. By default, it is also required, in addition to other roles, for access to other functions.
CreateAccess	This role allows users to create objects in the UI.
DeleteAccess	This role allows users to delete objects in the UI.
UpdateAccess	This role allows users to update objects in the UI.

A

Secure Deployment Checklist

Learn how to use checklists to install Oracle Communications Billing and Revenue Management (BRM) and Oracle Communications Pricing Design Center (PDC) securely.

Topics in this appendix:

BRM Checklist

The following security checklist lists guidelines to help you secure BRM and its components.

1. Install only what is required.
2. Lock and expire default user accounts.
3. Enforce password management.
4. Practice the principle of least privilege.
   - Grant only the necessary privileges.
   - Revoke unnecessary privileges from the PUBLIC user group.
   - Restrict permissions on run-time facilities.
5. Enforce access controls effectively and authenticate clients stringently.
6. Restrict network access.
   - Use a firewall.
   - Never poke a hole through a firewall.
   - Monitor who accesses your systems.
   - Check network IP addresses.
7. Apply all security patches and workarounds.
8. Contact Oracle Security Products if you come across a vulnerability in Oracle Database.

PDC Checklist

Follow this checklist to deploy PDC securely.

1. Preinstallation steps:
   a. Enable SSL for the target Oracle WebLogic Server domain.
   b. Configure the server KeyStore certificate and get the client KeyStore trusted certificate.
   c. Configure Oracle Database advanced security encryption and integrity algorithms for a secure connection from the installer.

d. Ensure that the latest supported version of Oracle JDK is installed and configured with your PDC or WebLogic installation.

2. Installation steps:
   - Select SSL mode and provide the client KeyStore certificate (**.jks** file) for connecting to a WebLogic server over SSL.
3. Postinstallation steps:
   a. If you do not need the installation log files, make sure to delete them.
   b. The WebLogic Server administrator needs to create PDC users based on the roles and privileges.
   c. Do not use your browser's remember password feature for the WebLogic Remote Console URL.
   d. Enable secure cookies.
   e. Verify that file permissions for the installed files are 600 for all nonexecutable files and 700 for all executable files.
4. Un-installation steps:
   - Delete the log files in *OracleInventory***/logs/** manually if you do not need them or protect them appropriately if they are required for further reference. These log files have file permission 640 (owner can read/write, group members can read, others cannot do anything) by default.