Oracle® Communications Order and Service Management

Security Guide

Release 7.5
F60010-02
June 2024

Oracle Communications Order and Service Management Security Guide, Release 7.5

F60010-02

Contents

A Secure Deployment Checklist

Preface

This document describes security considerations and procedures for a traditional deployment of Oracle Communications Order and Service Management (OSM).

For details about security considerations for an OSM cloud native deployment, see *OSM Cloud Native Deployment Guide*.

Audience

This document is intended for system administrators, system integrators, database administrators, and other individuals who are responsible for managing OSM and ensuring that the software is operating in a secure manner. This guide assumes that you have a working knowledge of OSM, the relevant operating system, Oracle Database, Oracle WebLogic Server, and Java J2EE software.

Documentation Accessibility

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

Access to Oracle Support

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

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

Overview

This chapter provides an overview of Oracle Communications Order and Service Management (OSM) security.

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.
- **Limit privileges as much as possible.** Users should be given only the access necessary to perform their work. User privileges should be reviewed periodically to determine relevance to current work requirements.
- **Monitor system activity.** Establish who should access which system components, and how often, and monitor those components.
- **Install software securely.** For example, use firewalls, secure protocols (such as Transport Layer Security (TLS) and Secure Sockets Layer (SSL)), and secure passwords. See "Performing a Secure OSM Installation " for more information.
- **Learn about and use the OSM security features.** See "Implementing OSM Security " for more information.
- **Use secure development practices.** For example, take advantage of existing database security functionality instead of creating your own application security. See "Security Considerations for Developers " for more information.
- **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 the Critical Patch Updates and Security Alerts website:

  `http://www.oracle.com/technetwork/topics/security/alerts-086861.html`

Overview of OSM Security

OSM security is designed to protect application users, modules, solution data, order data, logs, and interfaces.

- **OSM application security:** Users of the application are authenticated using the Oracle WebLogic Server authentication framework.
- **OSM solution data security:** Solution data, such as deployed cartridges, is stored in the Oracle Database, which requires database credentials to access.
- **OSM order data security:** Application access to solution and order data is authorized and validated by the OSM role-based authorization model. Order data is stored in the Oracle Database, which requires database credentials to access.
- **OSM interface security:** Users of the application are authorized at the interface level (for example, web client, web service, or deployment) using the WebLogic Server security roles. Interfaces security is further configured using interface-specific means; the OSM

Web Service is secured by a WebLogic Server security policy, for example. Credentials for accessing external systems are stored securely.

- **Application log security:** Application log content is configured by users of the WebLogic Server **Administrators** group. The application distribution, settings and properties, and logs are protected by the user authorization and authentication procedures of the host operating system. Only the user who starts OSM has access to the files, based on file permissions.
- **Database security:** The OSM database credentials are stored securely inside the Java Database Connectivity (JDBC) data source in the WebLogic server.

Understanding the OSM Environment

When planning your OSM implementation, consider the following:

- **Which resources need to be protected?**
  - You need to protect customer data.
  - You need to protect internal data.
  - You need to protect system components from being disabled by external attacks or intentional system overloads.
- **Who are you protecting data from?**

  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 workflows to determine who needs access to the data; for example, it is possible that a system administrator can manage your system components without needing to access the system data.
- **What will happen if protections on strategic resources fail?**

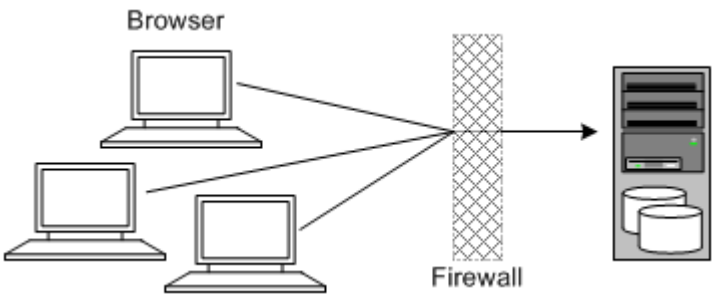
  In some cases, a fault in your security scheme is nothing more than an inconvenience. In other cases, a fault might cause great damage to you or your customers. Understanding the security ramifications of each resource will help you protect it properly.

Recommended Deployment Topologies

This section describes recommended deployment configurations for OSM.

Figure 1-1 shows a single-computer installation topology: the simplest OSM deployment architecture.

Figure 1-1 Single-Server Deployment Topology



In this topology, all the application components and data are kept on a single server, protected from external attacks by a firewall. The firewall can be configured to block known illegal traffic types. There are fewer resources to secure because all the components are on a single server and all of the communication is local. Fewer ports have to be opened through the firewall.

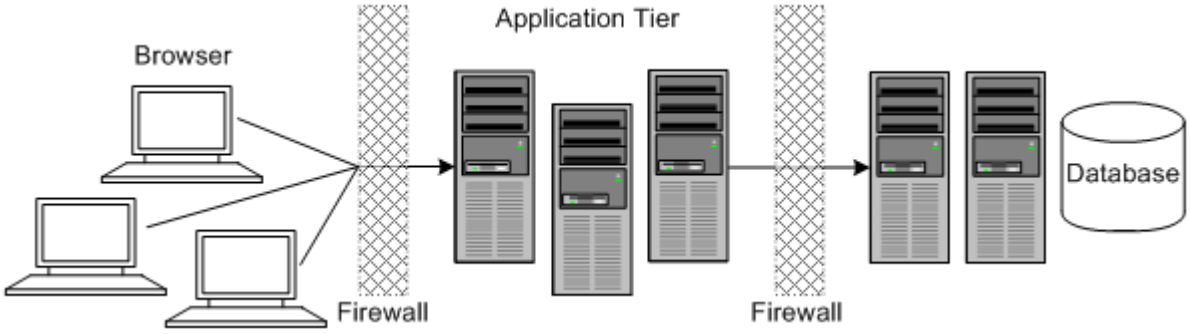
Conversely, there are fewer points of attack, and if security is compromised, an attacker would have access to the entire system and data.

A single-server installation topology is best suited for test and lab environments.

A single-server deployment is cost effective for small organizations but does not provide high availability because all components are stored on a single server.

Figure 1-2 shows a tiered installation deployment: a scalable OSM deployment offering greater security and high availability.

Figure 1-2 Tiered Deployment Topology



In this topology, the application tier is isolated by firewalls from both the Internet and the intranet. The database and servers are protected from potential attacks by two layers of firewall. Both firewalls can be configured to block known illegal traffic types. The two layers of firewall also provide intrusion containment. Although there are more components to secure, and more ports must be opened to allow secure communication between the tiers, the attack surface is spread out.

Operating System Security

This section describes operating system security topics that are specific to OSM. See the documentation for your operating system for general information.

Operating System Releases and Patch Levels

For information about the supported operating system releases and patch levels, see the system requirements in *OSM Installation Guide*. OSM depends on Oracle Database and WebLogic Server. For all operating systems, check the Oracle Database and WebLogic Server documentation for any additional operating system patches required to support those applications.

Restricting Permissions for OSM Directories

Oracle recommends keeping the permissions as restrictive as possible for your business needs. When installing on UNIX or Linux, consider using `umask 066` to deny read and write permission to all users except the user that installed the software. OSM creates files in the

directories listed in Table 1-1. Examine these directories to ensure they have the appropriate permissions.

Table 1-1 OSM Directories

Name	Description
OSM Config home	The directory in which OSM configuration related files may be generated or modified by OSM. This directory contains various configuration related files for OSM DB schema and Weblogic domain.
Domain home	The directory containing the WebLogic Server domain for OSM. The default location is *Fusion_Middleware_installation_directory*/**user_projects/domains/***domain_name*, but it is frequently set to something else during installation.
VFS home	The directory in which OSM stores various solution files. The default location is **/tmp/vfs_cache**. If it is not using the default location, you will see the location in the value of the **-Djava.io.tmpdir=***new_path* argument to the OSM WebLogic Server startup scripts (where *new_path* is the parent directory of **vfs_cache**).

Port Security

OSM communicates over a limited number of ports. Depending on your solution requirements, additional ports may be required, especially if OSM is deployed to a WebLogic Server cluster.

Close all unused ports, especially non-SSL ports. Opt for SSL-enabled ports for all communications (for example: HTTPS or t3s) when possible.

The types of ports OSM uses are listed in Table 1-2.

Table 1-2 OSM Ports

Port	Port Description
Listen port for the administration server	The default value is 7001, but a different value can be set during domain creation.
SSL listen port for the administration server	The default value is 7002, but a different value can be set during domain creation.
WebLogic Server administration port (SSL-only)	The default value is 9001 if the administration port is enabled. By default, this port is disabled, but Oracle recommends enabling the port.
Coherence cluster port	The default value is 17001, but a different value can be set during OSM installation.
Database listener port	The default is 1521, but a different value can be set during database creation.

Oracle Database Security

This section describes database security topics specific to OSM. For more information about securing Oracle Database, see *Oracle Database Security Guide* and *Oracle Database Advanced Security Guide*. Some OSM database changes in this section must be made by an Oracle database administrator (DBA).

Oracle Database Administrator Roles and Permissions

The following roles and permissions are required for the account used by the DBA:

- Required permissions
  - Connect to a resource with admin option
  - Execute on dbms_lock with grant option
  - Execute on dbms_redefinition with grant option
  - Select on dba_jobs with grant option
  - Execute on exp_full_database and imp_full_database with admin option
  - Create table with admin option
  - Create materialized view with admin option
  - Query rewrite with admin option
  - Select on v_$parameter with grant option
- Required roles
  - sysdba role

Transparent Data Encryption

You can encrypt the OSM tablespace and schema, at the expense of system performance, using Oracle Database Transparent Data Encryption (TDE). Encrypting the schema and tablespace enforces data-at-rest encryption in the database layer and therefore prevents would-be attackers from bypassing the database and reading sensitive information from storage.

If you choose to encrypt the tablespace and schema, you must configure TDE on the tablespace before creating the OSM schema. See *Oracle Database Advanced Security Guide* for more information.

Dependent Schemas

Before creating the WebLogic Server domain, you must create certain database schemas using the Oracle Fusion Middleware Repository Creation Utility (RCU). For information about RCU, see *Oracle Fusion Middleware Creating Schemas with the Repository Creation Utility*. For information about the RCU schemas required for creating an OSM WebLogic Server domain, see the information about installing and configuring WebLogic Server in *OSM Installation Guide*.

WebLogic Server Security

This section contains WebLogic Server security information relevant to OSM. For additional information about WebLogic Server security, see *Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server* and *Oracle Fusion Middleware Administering Security for Oracle WebLogic Server*.

When planning your WebLogic Server domain installation, keep the following recommendations in mind:

- **Secure the WebLogic Server host:** WebLogic Server domain and server configuration files should be accessible only by the operating system users who configure or run

WebLogic Server. No other operating system user (apart from the system administrators) should have read, write, or execute access to WebLogic Server product files or your domain files.

- **Set file access permissions for data stored in the persistent store:** Set operating system file access permissions to restrict access to data stored in the persistent store. When using the synchronous write policy of Direct-Write-With-Cache, limit access to the cache directory, especially if there are customized user access limitations on the primary directory. For more information about the WebLogic Server services and subsystems that can create connections to the persistent store, see the information about using the persistent store in *Administering Server Environments for Oracle WebLogic Server*. The default persistent store maintains its data in the *domain_home*/**servers**/*server_name*/**data**/**store**/**default** directory, where *domain_home* is the home directory for the OSM domain, and *server_name* is the name of the relevant WebLogic server.
- **Do not run WebLogic Server in development mode in a production environment:** Production mode sets the server to run with settings that are more secure and appropriate for a production environment. For more information about development mode and production mode, see the information about domain modes in *Understanding Domain Configuration for Oracle WebLogic Server*.
- **Use appropriate encryption:** WebLogic Server includes a set of demonstration private keys, digital certificates, and trusted certificate authorities that are for development only; do not use the demonstration identity and trust in a production environment. See the topic on configuring keystores in the *Oracle WebLogic Server Administration Console Online Help* and the information about configuring SSL in *Administering Security for Oracle WebLogic Server* for more information about encryption.

  To prevent sensitive data from being compromised, secure data transfers by using HTTPS.

2

Performing a Secure OSM Installation

This chapter presents planning information for your Order and Service Management (OSM) installation.

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

Pre-Installation Configuration

OSM depends on a database instance and Oracle WebLogic Server domain that have been properly configured. See "Operating System Security," "Oracle Database Security," and "WebLogic Server Security" for details on secure file system, database, and WebLogic Server domain configuration.

About the O7_DICTIONARY_ACCESSIBILITY Parameter

If you intend to use the **sys** user when the OSM installer prompts for database administrator credentials, you have two options:

- Ensure that the O7_DICTIONARY_ACCESSIBILITY parameter is set to **TRUE** in the database before running the OSM installer. If you choose this option, you should consider setting the parameter back to the default of **FALSE** after you have finished installing OSM.
- When prompted for the database administrator user name in the OSM installer, append **as sysdba** to the user name.

For more information, see the information about installing and configuring the database in *OSM Installation Guide*.

Installing OSM Securely

This section describes ways of ensuring that OSM is installed securely and information that you can use to secure installed components after installation.

Password Policies

The OSM installer creates database schema and WebLogic Server domain application user accounts. The installer requires you to specify the password (and in some cases, user name) for these users. Oracle recommends the following password policies for these users, as well as users you create in the future:

- The password should be between eight and 24 characters long.
- The password should contain at least one letter, one number, and one special character.
- The password should not contain the user name.
- The user's account should be temporarily disabled after five login failures.

These recommended password policies are not implemented by default in the OSM installer, but should be configured manually after the installation. See *Oracle Database Security Guide* for information about implementing user security for Oracle Database users. For information

about configuring the policies for WebLogic Server users, see the information about customizing the default security configuration in *Oracle Fusion Middleware Administering Security for Oracle WebLogic Server*.

Users and Groups

The OSM installer creates various users and groups in the database and in WebLogic Server. For information about the users and groups created by the OSM installer, see the information about installed components in *OSM System Administrator's Guide*.

The OSM installer creates the WebLogic Server users and groups listed in *OSM System Administrator's Guide*. If you intend to use another security implementation, such as LDAP, you must manually create those users and groups and assign the users to the groups.

Security-Relevant Installation Steps

Some steps in the installation process have security implications that you should keep in mind.

- In the WebLogic Server Connection Information window of the OSM installer, you can choose to connect to WebLogic Server over SSL, which encrypts all communications between the installer and the WebLogic administration server, including the user names and passwords that the installer creates.
- In the Order and Service Management Session Information window of the OSM installer, you can set the **Session Timeout** value for the OSM web clients. It is a security risk to leave a session active for an extended period of time. Oracle recommends updating this setting to the lowest value that meets your business needs.
- All the sensitive information like password fields gathered during the interactive discovery phase of OSM Installer are encrypted in the generated **configuration.properties** file inside *OSM_Config_Home* directory using a user-provided passphrase. The same passphrase is required to be available to the subsequent installer scripts in order to read the secure information for OSM install or upgrade.

3

Implementing OSM Security

This chapter provides a synopsis of the Order and Service Management (OSM) security features. For additional details, see the information about setting up OSM security in *OSM System Administrator's Guide*.

Secure Credential Management

OSM provides two distinct secure credential management solutions, each appropriate to the type of credential to be secured:

- **EncryptPasswords utility**: This utility is used to secure the credentials required to run other OSM utilities, such as the XML Import/Export tool. Oracle recommends that you use it when you run utilities unattended. If you run the utilities attended, Oracle recommends that you provide the required credentials interactively when prompted by the utility, rather than using EncryptPasswords. Because EncryptPasswords is assumed to run unattended and therefore the transformation is automated, the output of EncryptPasswords should be considered obfuscated rather than encrypted. Secure the files containing the output with appropriate file-system-level restrictions.
- **Credential Store**: This utility is used to secure credentials required to access systems with which your OSM solution interacts. It builds on the Credential Store Framework (CSF), adding OSM-specific features.

For information about how to secure credentials using these methods, see *OSM System Administrator's Guide*.

Secure Solution Data Storage

As a fulfillment system, OSM does not need a fixed data model, and so is not required or typically used to store sensitive data other than that used for OSM user authentication.

You can secure OSM user credentials as described in "Secure Credential Management," but if your implementation requires OSM to store or log other sensitive data that appears on orders, Oracle recommends that you encrypt the data outside of OSM. Because the encryption happens outside of OSM, you are responsible for developing and maintaining the encryption method.

Using the WebLogic Scripting Tool

Several OSM features make use of the WebLogic Scripting Tool (WLST). When connecting to a WebLogic Server service instance, Oracle recommends that you connect to the service instance through the administration port. By default, this port is not enabled, but Oracle recommends that you enable the administration port in a production environment. The administration port requires all communication to be secured using SSL. By default, all servers in a WebLogic Server domain use demonstration certificate files for SSL, but these certificates are not appropriate for a production environment. For information about configuring the administration port, see the information about the administration port and administrative channel in *Administering Server Environments for Oracle WebLogic Server*. For more information about WLST, see *Oracle Fusion Middleware Understanding the WebLogic Scripting*

Tool. For more information about connecting to WLST for OSM, see the information about managing and monitoring OSM in *OSM System Administrator's Guide*.

Secure Logging

OSM can be configured to suppress stack trace information in log output. A stack trace is a list of the method calls that an application is in the middle of at the time an exception is thrown. Running OSM with stack-trace logging enabled can be important for debugging the application during run time. However, in certain cases, suppressing this information can improve application security.

You can enable or disable logging stack traces using the **oms-config.xml** parameter **enable_log_stacktraces**. By default, this parameter is enabled. If there is a security concern about having log stack traces enabled, you can disable this parameter. For more information about setting this parameter, see the information about configuring OSM with **oms-config.xml** in *OSM System Administrator's Guide*.

4

About OSM Authentication and Authorization Methods

This chapter describes the authentication and authorization methods for OSM interfaces. This includes human users and APIs.

The following table describes abbreviations and terms used across this chapter.

Table 4-1 Abbreviations and Terms Used in This Chapter

Term	Description
SAML	Security Assertion Markup Language
SAML Assertion Token	SAML XML document that the identity provider sends to the service provider containing data about authorization status and user's information.
SSO	Single Sign-On
OIDC	OpenID Connect Protocol
ID Token	OIDC JSON document in JWT format that the identity provider sends to the service provider containing data about authorization status and user's information.
OAuth	Open Authorization Protocol
Access Token	Represents the authorization to access resources on behalf of the enduser. OAuth 2.0 does not define a specific format for Access Tokens. In OSM context, the JSON Web Token (JWT) format is used.
JWT	JSON Web Token
SLO	Single Log Out
IdP	Identity provider
SP	Service Provider

About Authentication in OSM

OSM supports multiple authentication schemes for humans and APIs based on its interfaces. These are:

- Basic Authentication
- SAML
- OIDC

This section lists the relevancy of the above schemes in the context of OSM interfaces, usages, and their respective identity stores.

The following table shows OSM-supported authentication schemes on each OSM interface:

Table 4-2 OSM-supported Authentication Schemes

Authentication Scheme	OSM Web UIs	Web Services API	XML API	TMF API	Fallout API
Basic Authentication	Yes	Yes	Yes	No	No
SAML	Yes	No	No	No	No
OIDC	No	No	No	Yes	Yes

The following table shows OSM-supported Identity stores on each authentication scheme:

Table 4-3 OSM-supported Identity Stores on each Authentication Scheme

Authentication Scheme	Weblogic Embedded LDAP	External LDAP (Weblogic User Federation)	IdP User Federation	IdP Embedded User Store
Basic Authentication	Yes	Yes	No	No
SAML	No	No	Yes	Yes
OIDC	No	No	Yes	Yes

The following table shows recommended authentication schemes based on the deployment environments:

Table 4-4 OSM Authentication Scheme Best Practices

Environment	SAML	Basic Authentication (Human Users)
Development	No	Yes
Production	Yes	No

Using Basic Authentication

With basic authentication, the web browser opens a login screen in response to an OSM resource request. The login screen prompts the user for a username and password. OSM continues to support basic authentication for backward compatibility for web pages as well as its legacy APIs. OSM supports embedded LDAP and external LDAP via Weblogic Authentication providers for the basic authentication method.

While basic authentication is widely supported, it is considered less secure compared to other authentication methods. Therefore, it is recommended to use more secure authentication schemes like SAML, OIDC or token-based authentications for your deployments.

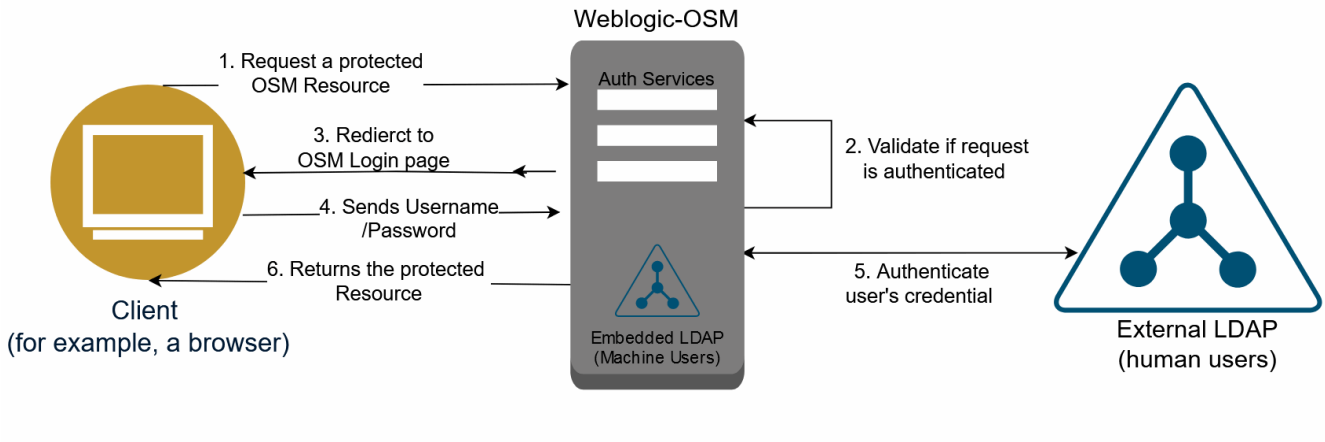
The following interfaces are supported by OSM for the basic authentication scheme:

- OSM Web UIs (human users)
- XML API (machine users)
- Web Services (machine users)

The following diagram shows OSM Basic Authentication flow for human users.

> **Note:**
>
> The same flow applies to machine users except that Embedded LDAP is used.

Figure 4-1 OSM Basic Authentication Flow for Human Users



About Using Embedded LDAP

OSM Weblogic uses the default authentication provider, which is embedded LDAP. Weblogic Embedded LDAP is not a global user store. It is created implicitly on the creation of the Weblogic domain. Hence, every OSM domain instance maintains its dedicated and private user store. OSM recommends using the WebLogic default authenticator for the following use cases:

- Setting up lightweight OSM in development or test environments.
- OSM uses this authentication provider for its Web services and XML API. OSM relies on the users (machine users) to set up these credentials.

> **Note:**
>
> OSM uses the term "machine users" for XML API and Webservices credentials.

For more details on how to configure machine users for the APIs in OSM cloud native, see "Provisioning Machine User Accounts in OSM Cloud Native".

The WebLogic default authenticator is not suitable for production environments for human users. In such cases, OSM recommends using an external LDAP server as the authentication provider. This allows for centralized user management and enhanced security features.

Provisioning Machine User Accounts in OSM Cloud Native

This section describes how to use the OSM CNTK sample script to create machine users for XML API and Web Services API in an OSM cloud native environment.

A sample text file **$OSM_CNTK/samples/credentials/osm_users.txt** is provided in cloud native toolkit to describe the details required to provision the user accounts properly. Each user is captured in one line and has the following format:

```
osm:_sysgen_:__REPLACE_WITH_USERNAME__:secret[:osm-groups]
```

To create machine users for XML API and Web Services API:

1. Copy the text file provided in the toolkit to your private specification repository under the instance-specific directory.
2. Rename it to something meaningful. For example, you can rename the file as **repo/machine_users.txt**.
3. Add the following to **machine_users.txt**.

   ```
   osm:_sysgen_:osm_api_user:secret:OMS_xml_api,OSM_automation,OMS_ws_api
   ```

   > **Note:**
   >
   > For this example, the machine username is **osm_api_user**. You can use any name of your choice, but this user should be part of OSM predefined roles **OMS_xml_api** and **OMS_ws_api**. The additional role **OSM_automation** for automation is also added in this example.

   In the above example, the first line creates a user **osm_api_user** against embedded LDAP and associates that user with the groups listed. It also creates a Kubernetes secret entry for this user in the "osm" credential secret. The entry contains a username, password, and the group associations.

   The secrets that the **manage-cartridge-credentials.sh** script creates are named `project-instance-osmcn-cred-osm`, as per the naming conventions required by OSM. The script prompts for passwords interactively.

   To create the credential store secret, run the following script:

   ```
   $OSM_CNTK/samples/credentials/manage-cartridge-credentials.sh \
   -p project \
   -i instance \
   -c create \
   -f fileRepo/machine_users.txt
   ```

   You will get the following output:
   secret/project-instance-osmcn-cred-osmcreated
4. In the **project.yaml**, under the **cartridgeUsers** section, add all the machine users (only those from the prefix/map name osm).
5. If OSM instance is running, then upgrade the instance to take into account the new machine users or create an OSM instance.

> **Note:**
>
> During the creation of the OSM server instance, for all the machine users listed, an account is created in embedded LDAP with the same username, password, and groups as the Kubernetes secret. For example:
>
> ```
> cartridgeUsers: - osm_api_user
> ```

6. Users can be verified by logging into the Weblogic Admin console.
   To verify the users:
   a. Log into Weblogic Admin console.
   b. Select **Security Realms** from **Domain Structure**.
   c. Select **myrealm**.
   d. Go to **Users**.
      The new user **osm_api_user** would be available with the corresponding users.

About Using External LDAP

External LDAP provides a global user store when OSM intends to use the Basic Authentication scheme. The external LDAP integration allows OSM customers to use their LDAP servers as the user store for OSM instances. In the Basic Authentication scheme, the external LDAP server needs to be configured directly in Weblogic via the Weblogic LDAP Authentication Provider. For more details on how to configure external LDAP (Basic Authentication) in OSM cloud native, refer to the OSM Cloud Native Deployment Guide.

The configuration process involves specifying the LDAP server details. Additionally, you will need to provide the distinguished name (DN) and password for binding to the LDAP server. OSM also supports SSL-based communication with the LDAP server, ensuring secure transmission of user authentication data. For cases where SSL is enabled, you will be prompted to enter the path to the trust store file used for verifying the LDAP server's certificate. Once the configuration is complete, OSM can authenticate users against the external LDAP server using Basic Authentication.

Using SAML for SSO and SLO

OSM adopts SAML2 as its chosen protocol to facilitate Single Sign-On (SSO) and Single Logout (SLO) functionalities for human users (Web UIs).

> **Note:**
>
> OSM takes advantage of the authorization capabilities offered by identity providers (IdPs), such as logical grouping and roles for human users.

IdPs are capable of integration with various identity stores, such as LDAP. OSM SSO deployments can effectively eliminate the need for cumbersome WebLogic LDAP integration. This integration provides a streamlined and efficient approach to managing user authentication and authorization within OSM.

With the adoption of standard protocols like SAML2, OSM enables seamless interaction with IdPs, allowing for centralized control over authentication and authorization processes. This shift towards standard methods not only simplifies user management but also enhances security and improves the overall user experience.

For the standardization of authentication and authorization methods, IdPs play a crucial role. OSM does not prescribe any specific IdP. You are free to choose your desired IdP for the standard protocols such as SAML and OIDC. OSM's flexibility in supporting various identity providers allows you to align with your existing infrastructure and choose the provider that best suits your needs. This approach ensures a seamless integration with your preferred IdP, enabling a smooth experience while maintaining stringent security measures.

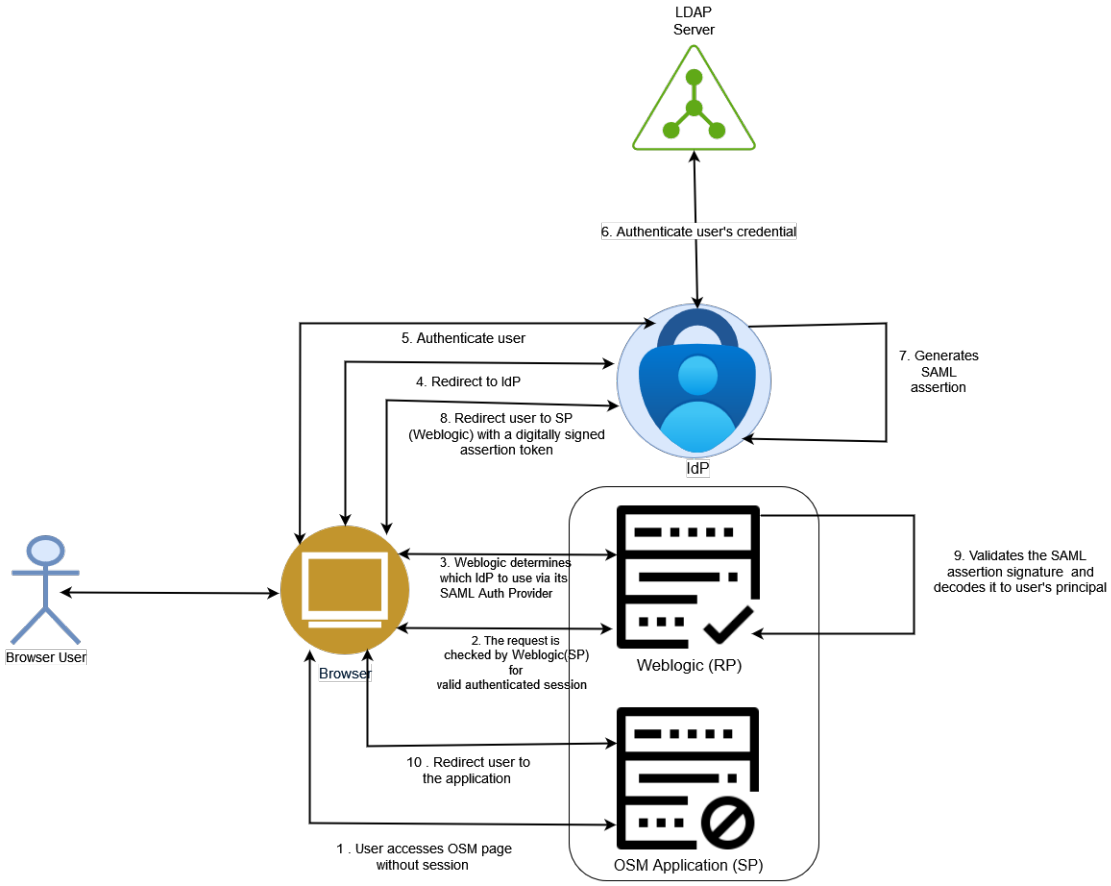
By leveraging standard protocols like SAML2 and OIDC, OSM empowers you to centralize authentication and authorization processes, simplifying user management and bolstering overall system security.

> **Note:**
>
> You are free to choose your desired IdP. OSM SAML and OIDC schemes are not dependent on a specific IdP. OSM choses Keycloak as a reference to test out these capabilities. Hence, any referenced Keycloak documentation within this document is only meant as an example and more importantly is not meant for production.

The following diagram shows the SAML flow when OSM as a participant application, initiates an SSO session. It begins when you try to access an OSM web page.

The steps involved in this type of process are outlined in the following diagram. In this diagram, IdP receives users from external LDAP (User Federation).

Figure 4-2 SAML Flow When OSM Initiates an SSO Session



There are two typical SAML flows:

- IdP initiated flow
- SP initiated flow

OSM out-of-the-box SAML support uses "SP initiated flow".

The flow of SAML in OSM is as follows:

1. You attempt to access the OSM application instance.
2. The Weblogic (SP), where OSM client application is hosted, redirects you back to the IdP to be authenticated and provides you with the OSM page URL to access OSM after you are authenticated.
3. The Weblogic as SP determines, which IdP it should use to authenticate you. This is configured via Weblogic SAML Authentication Provider.
4. The Weblogic (SP) SAML Authentication Provider redirects you to the appropriate IdP.
5. The IdP authenticates your identity.
6. IdP leverages User Federation with LDAP connector to delegate the authentication to an external identity store such as LDAP server.
7. The IdP creates and signs an XML-based SAML assertion that includes information about the your identity, along with any other attribute information that the IdP and SP agreed to share to authenticate you.

8. The IdP either sends the assertion to the SP through a browser that the SP can use to securely retrieve the assertion or sends a reference to the assertion that the SP can securely retrieve the assertion.
9. The SP validates the signature to ensure the SAML assertion came from its trusted IdP and that none of the values in the assertion have been modified. This is done using SAML signing public certificate stored in Weblogic trust store. It also extracts the identity, attribute, and authorization information it needs to determine whether access should be granted and which privileges you will have. Only users with **OMS_Client** role can access the page.
10. You access the application's page.

> **Note:**
>
> Users can be created and managed in IdPs via your embedded user stored such as database or federated from external sources. The above diagram shows federated users from external LDAP.

OSM SAML Support for Web Applications

OSM SAML is only meant to provide authentication and authorization for browser users or humans. It does not provide any support for applications that access resources through the API.

- In OSM cloud native, the OSM Legacy APIs such as XML API and Webservices API are protected using Basic Authentication backed by Weblogic-embedded LDAP.
- OSM uses OIDC for the new TMF REST APIs and Fallout APIs as described later in this section. Similar to OSM SAML SSO, OSM relies on OIDC-backed Realm Client in IdP.

Setting up SSO Using SAML2 in OSM Cloud Native

OSM CNTK relies on JKS key store to be created and populated with public certificates and private keys, which are required by OSM Weblogic to decrypt SAML Assertions and validate SAML signatures.

This section describes how to configure SSO, along with Single Logout (SLO) using SAML2 protocol in an OSM cloud native environment.

An Identity Provider (IdP) is required for user authentication and authorization. There are multiple IdPs in the market. In this document, KeyCloak has been used as an example of an Identity Providers (IdP).

> **Note:**
>
> The keystore in JKS format must be created by Java 8. Ensure that you have the latest Java 8 version when setting up the keystore file. OSM CNTK does not accept a keystore that is built by a Java 8 version older than B301.

Preparing the Keystore

Two pairs of keys and certificates are required in SAML2 SSO in WebLogic to exchange SAML SSO and SLO messages against the IdP. One is used to sign the SAML messages, while the

other one is used for encryption. In an OSM cloud native environment, the default SSL key and certificate are used for SAML message encryption. Hence, two pairs of keys and certificates need to be contained in the keystore.

See "Creating Keystore for OSM SAML2 SSO" for more information on keystore.

Creating the "wlstls" Secret

When SAML2 SSO is enabled in an OSM cloud native environment, the **$PROJECT-$INSTANCE-keystore** secrets are required. The `$OSM_CNTK/scripts/manage-instance-credentials.sh` script with the "wlstls" option is used to create the two secrets, which are **$PROJECT-$INSTANCE-truststore** and **$PROJECT-$INSTANCE-keystore**.

> **Note:**
>
> The **keystore.jks** file should be the keystore described in "Preparing the Keystore".

Generating Service Provider Metadata

To generate the OSM service provider metadata dynamically, run the `$OSM_CNTK/scripts/create-service-provider-metadata.sh` script.

The service provider metadata will be used in the Keycloak realm to set up the *client* entity. For more information, see the step about importing service provider metadata file in "Setting Up Keycloak for SAML SSO".

In an OSM cloud native environment, the SSL key and certificate will be used as the SSO encrypting key and certificate. When using the `$OSM_CNTK/scripts/create-service-provider-metadata.sh` script to generate the service provider metadata file, you need to assign the SSL key alias and passphrase for the encrypt key (for instance, -e ssl_key).

> **Note:**
>
> The **keystore.jks** file should be the keystore described in the "Preparing the Keystore" section.

The OSM service provider metadata is generated at **/tmp/spmetadata.xml**. Back up this file, and it will be used to register OSM WebLogic SSO and SLO service in the IdP. See "Setting Up Keycloak for SAML SSO" for more information.

Setting Up IdP and Acquiring IdP Metadata File

An IdP is required for OSM SAML2 SSO. The IdP could typically generate an IdP metadata file, which would contain details on how to communicate with the IdP endpoint. This IdP metadata file is also used to register the IdP with the service provider (OSM WebLogic domain). The service provider (OSM WebLogic domain) also needs to provide the service provider metadata file, and be registered in the IdP. For OSM cloud native, this service provider metadata file is generated as described in "Generating Service Provider Metadata."

There are multiple IdPs in the market. In this document, Keycloak is used as an example of an IdP.

See "Setting Up Keycloak for SAML SSO" for more information on setting up Keycloak as your IdP. In the same section, see the step about acquiring IdP metadata file from the Keycloak realm.

Creating SSO Archive Secret

When SAML2 SSO is enabled in an OSM cloud native environment, the **$PROJECT-$INSTANCE-ssosaml-archive** secret is required. The `$OSM_CNTK/scripts/manage-instance-credentials.sh` script with the **samlsso** option is used to create the **$PROJECT-$INSTANCE-ssosaml-archive** secret.

When running `$OSM_CNTK/scripts/manage-instance-credentials.sh -p ${PROJECT} -i {INSTANCE} create samlsso`, it would prompt "The path to SAML2 XML Metadata file:" Provide the path to the SAML2 XML Metadata file, which is described in "Setting Up IdP and Acquiring IdP Metadata File".

> **Note:**
>
> There is a limitation to the size of the SAML2 XML Metadata file exported from the IdP; It can not be more than 1MB when compiling into an archive file. The `$OSM_CNTK/scripts/manage-instance-credentials.sh` script will report an error message if the size of the compiled archive file is more than 1MB. This is not likely to happen, but depends on the IdP you use. If you get the error message while running `$OSM_CNTK/scripts/manage-instance-credentials.sh`, try to edit the SAML2 XML Metadata file by removing the comment and annotation elements, and retry. If you still run into this issue, you can refer to the SAML2 XML metadata file at: https://docs.oasis-open.org/security/saml/v2.0/saml-schema-metadata-2.0.xsd, and remove the elements for the SP roles. For instance, the **md:SPSSODescriptor** element.

Activating SAML2 SSO

If SAML2 SSO is enabled, the `sso.enabled` parameter is shown in the project specification file. The default value for this parameter is `false`. You need to set the value to `true` if you want to activate SAML2 SSO.

```
sso:
            # if SAML SSO is enabled, default value is false
            enabled: true
```

Creating an OSM instance

Run the `$OSM_CNTK/scripts/create-instance.sh -p $PROJECT -i $INSTANCE -s $SPEC_PATH` command to create the OSM cloud native instance.

Planning Certificate Rotation

Certificate rotation should be planned well before the expiry of the certificates. This could happen to both the service provider signing certificate and encryption certificate that are shared by the service provider (OSM) within the service provider metadata XML file, and identity provider signing certificate within the IdP metadata XML file. Changing either of them requires you to run the following procedure.

Identity Provider Signing Certificate Rotation

When the identity provider signing certificate is changed, the IdP metadata XML file that contains the IdP signing certificate is changed accordingly. The IdP needs to regenerate the IdP metadata XML file and share it with the OSM cloud native instance. OSM cloud native needs to:

1. Regenerate the SSO archive secret with the newly provided IdP metadata file by following the steps described in "Creating SSO Archive Secret".
2. Recreate the OSM cloud native instance.

Service Provider Signing and Encryption Certificate Rotation
The service provider signing certificate and encryption certificate are extracted from the JKS keystore as described in "Preparing the Keystore". When the service provider signing or encryption certificate rotation is rotated, an updated JKS keystore file should be provided. The updated JKS keystore should contain the updated signing or encryption certificates and keys. OSM cloud native needs to:

1. Create the "wlstls" secret using the updated JKS keystore file by following the steps described in "Creating the "wlstls" Secret".
2. Generate Service Provider metadata using the updated JKS keystore file by following the steps described in "Generating Service Provider Metadata."
3. Share the generated service provider metadata with the IdP, and update the setting in IdP accordingly. Different IdPs could have different settings. Here, Keycloack is being used as an example. See "Setting Up Keycloak for SAML SSO" for details.
4. Recreate the OSM cloud native instance.

Enabling SSO for OSM Cloud Native

OSM cloud native automatically picks the `sso_enabled` parameter from the OSM instance and propagates it to the **osm-config.xml** file.

Setting up SSO Using SAML2 in OSM Traditional Deployments

This section is about configuring Single Sign On (SSO), along with Single LogOut (SLO) using SAML2. You can use SAML 2.0 to enable SSO in OSM. SSO allows you to log in to applications using a single user name and password combination.

> **Note:**
>
> The OSM installer does not support SSO using SAML.

An Identity Provider (IdP) is required for user authentication and authorization. The IdP used here is KeyCloak.

Environment Details
SAML2 SSO is only enabled when you access an OSM Web UI, such as the OSM Task Web UI, the OSM Order Management Web UI, or the Order Operations and the Fallout Management UI. OSM XML APIs and OSM Web Service APIs are excluded.

When you access an OSM Web UI, you will be authenticated against the IdP associated with the OSM instance. The OSM XML APIs and OSM Web Service APIs will be authenticated against the default authenticator, which is the WebLogic internal embedded LDAP server.

> **Note:**
>
> Currently this works only on traditional OSM with Fusion Middleware 12.2.1.4 applied with October 2023 patches.

Configuring WebLogic to Use Keys from an External Keystore for SAML SSO

There are two pairs of keys and certificates used in SAML2 SSO in WebLogic to exchange SAML SSO and SLO messages against the IdP. One is used to sign the messages, while the other one is used for encryption. You need to configure the WebLogic managed servers that host OSM to use an external provided JKS keystore. OSM SAML2 SSO would use keys and certificates from the keystore for the SAML messages signing and encrypting.

Configuring OSM Managed Server to Use a Specified Keystore
To configure an OSM managed server to use a specified keystore:

1. Access the WebLogic Server Console as an administrator.
2. Click **Lock & Edit**.
3. Click **Environment**, then select **Servers** from the drop-down list.
4. Select the managed server that is hosting the OSM application (for example, ms1).

> **Note:**
>
> In a clustered environment, these steps need to be performed on each managed server that hosts OSM. The JKS keystore used in all these managed servers also needs to be exactly the same.

5. Click **Configuration**, then select **Keystores** from the drop-down list. This page is used to define the keystore and trust store used by this managed server.
6. Click **Change** to modify the keystore type used by the managed server.
7. Select the type of identity and trust depending on whether a trust store is used along with the keystore or not.
   - If only a keystore is assigned, choose **Custom Identity and Standard Trust**.
   - If both a keystore and a trust store are assigned, choose **Custom Identity and Custom Trust**.

> **Note:**
>
> Since an external keystore is always used here for SSO, a custom identity should always be used.

8. In the **Custom Identity Keystore** field, enter the path and file name of the provided external keystore.
9. In the **Custom Identity Keystore Type** field, enter JKS.
10. In the **Custom Identity Keystore Passphrase** field, enter the passphrase of the keystore.
11. In the **Confirm Custom Identity Keystore Passphrase** field, enter the passphrase of the keystore again to confirm that the right passphrase has been typed in.

12. If an external trust store is assigned to the managed server, repeat steps 8 to 11 for the trust store.
13. Click **Save**.

> **Note:**
>
> When a WebLogic server is configured to utilize an externalized keystore, as indicated above, you need to assign a default private key for the server. Use the following procedure to configure the default private key alias for each managed server that hosts OSM:
>
> 1. Click **Environment**, then, from the drop-down list, select **Servers**.
> 2. Select the managed server that is hosting the OSM application (for example, ms1).
> 3. Click **Configuration**, then select **SSL** from the drop-down list.
> 4. In the **Private Key Alias** field, enter the alias of the default private key in the keystore.
> 5. In the **Private Key Passphrase** field, enter the passphrase of the default private key.
> 6. In the **Confirm Private Key Passphrase** field, enter the passphrase of the defaut private key again to confirm that the right passphrase has been typed in.
> 7. Click **Save**.
>
> All the managed servers that host OSM should be assigned the same private key alias.

Configuring SAML for SSO Using WebLogic

When SSO is enabled in OSM, human users that access OSM through OSM UIs (Task Web Client, Order Management Web Client, or the Order Operations and Fallout Management UI) are authenticated and authorized against the integrated IdP (or SAML Authenticator). Machine users that access OSM through XML APIs or Web Service APIs are authenticated and authorized against the default WebLogic embedded LDAP server (or Default Authenticator).

No other Authentication Providers such as LDAP are supported when SSO is enabled.

To configure SAML for SSO using WebLogic:

1. Create SAML2 assertion provider and SAML authenticator. See "Creating SAML2 Assertion Provider and SAML Authenticator" for more information.
2. Specify general information. See "Specifying General Information" for more information.
3. Configure the SAML service provider. See "Configuring the SAML Service Provider" for more information.
4. Publish the service provider metadata. See "Publishing the Service Provider Metadata" for more information.
5. Register the IdP in WebLogic. See "Registering IdP in WebLogic" for more information.
6. Update the OSM deployment plan. See "Updating the OSM Deployment Plan" for more information.

Creating SAML2 Assertion Provider and SAML Authenticator

To create a SAML2 assertion provider and SAML authenticator:

1. Access the WebLogic Server Console as administrator.
2. Click **Lock & Edit**.
3. Open the **Providers** page, and then click **New**.
4. Enter a name in the **Name** field, select **SAML2IdentityAsserter** in the **Type** field, and click **OK**.
   The SAML2IdentityAsserter is displayed in the **Authentication Providers** table.

   > **Note:**
   >
   > It is found in some IdPs (for instance, KeyCloak) that the name of the SAML2 Identity Assertion provider used among the Service Providers that are registered to the same IdP security service needs to be unique. Consult the IdP administrator for the name to use if that is the case.

5. Select the newly created SAML2IdentityAsserter.
6. Click **Configuration**, then click **Provider Specific**, and then click the check box beside **Replicated Cache Enabled** to enable SAML replicated cache.
7. Navigate back to the **Providers** page and click **New** again.
8. Enter a name in the **Name** field, select **SAMLAuthenticator** in the **Type** field, and click **OK**.
   The SAMLAuthenticator is displayed in the **Authentication Providers** table.
9. Click **Reorder**.
10. Select and reorder the providers in the following order:
    a. SAML2IdentityAsserter
    b. SAMLAuthenticator
    c. DefaultAuthenticator
    d. DefaultIdentityAsserter
11. Click **OK**.
12. Select the SAMLAuthenticator.
13. Set the **Control Flag** to **Sufficient** and then click **Save**.
14. Return to the **Providers** page.
15. Select the DefaultAuthenticator.
16. Set the **Control Flag** to **Sufficient** and then click **Save**.
17. Click **Activate Changes**.
18. Restart the server.

Specifying General Information

To specify general information:

1. Access the WebLogic Server Console as administrator.

2. Click **Lock & Edit**.
3. Click **Environment** and then select **Servers** from the drop-down list.
4. Select the managed server that is hosting the OSM application (for example, ms1).

> **Note:**
>
> In a clustered environment, the steps below need to be performed on each managed server that hosts OSM. The **Single Sign-on Signing Key Alias** and the **Transport Layer Security Key Alias** values used in all managed servers that host OSM need to be exactly the same.

5. Click **Federation Services** and then select **SAML 2.0 General**.

> **Note:**
>
> This page is used to define the site information and additional settings for the SAML assertion, and to generate the service provider metadata file.

6. Click the **Replicated Cache Enabled** checkbox to enable SAML Replicated Cache.
7. Modify the given fields under **General Settings** as per your information:
   - **Published Site URL**: http://*hostname*:*port*/saml2 (or https://*hostname*:*httpsport*/saml2 when HTTPS is enabled)

     > **Note:**
     >
     > The hostname and port portion of the URL should be the hostname and port at which the server is visible externally; It may not be the same as the hostname and port by which the server is known locally. If you are configuring SAML 2.0 services in a cluster, the hostname and port may correspond to the load balancer or proxy server that distributes client requests to servers in the cluster.

   - **Entity ID**: The SAML2IdentityAsserter created in the section "Creating SAML2 Assertion Provider and SAML Authenticator".
   - **Recipient Check Enabled**: Clear the checkbox.
8. Configure the Single Sign-on signing key.
   - In the **Single Sign-on Signing Key Alias** field, enter the alias of the signing key in the keystore.
   - In the **Single Sign-on Signing Key Pass Phrase** field, enter the passphrase of the signing key.
   - In the **Confirm Single Sign-on Signing Key Pass Phrase** field, enter the passphrase of the signing key again to confirm the correct passphrase is typed.

> **Note:**
>
> The Single Sign-on Signing Key Alias value used in all managed servers that host OSM needs to be exactly the same.

9. Configure the Single Sign-on encryption key.
   - In the **Transport Layer Security Key Alias** field, enter the alias of the encryption key in the keystore.
   - In the **Transport Layer Security Key Passphrase** field, enter the passphrase of the encryption key.
   - In the **Confirm Transport Layer Security Key Passphrase** field, enter the passphrase of the encryption key again to confirm the correct passphrase is typed.

> **Note:**
>
> The Transport Layer Security Key Alias value used in all managed servers that host OSM needs to be exactly the same.

10. Click **Save**.

Configuring the SAML Service Provider

To configure the SAML service provider:

1. Access the WebLogic Server Console as administrator.
2. Click **Lock & Edit**.
3. Click **Environment** and then select **Servers** from the drop-down list.
4. Select the managed server that is hosting the OSM application (for example, ms1).

> **Note:**
>
> In a clustered environment, the steps below need to be performed on each managed server that hosts OSM.

5. Click **Configuration**, then **Federation Services**, and then **SAML 2.0 Service Provider**.
6. Select **Enabled**.
7. Modify the following fields for SLO:
   - Select **Single Logout Enabled**.
   - Set **Preferred Binding** to **POST**.
   - In the **Allowed redirect URIs** field, type in /OrderManagement.
8. In the SAML 2.0 Service Provider settings, set **Preferred Binding** to **POST**.
9. Click **Save**.
10. Click **Activate Changes**.
11. Restart the server.

Publishing the Service Provider Metadata

The service provider metadata will be used in the KeyCloak realm to set up the "client" entity. See "Setting Up Keycloak for SAML SSO".

To publish the service provider metadata:

1. Access the WebLogic Server Console as administrator.
2. Click **Lock & Edit**.
3. Click **Environment**, then select **Servers** from the drop-down list.
4. Select the managed server that is hosting the OSM application (for example, ms1).

> **Note:**
>
> In a clustered environment, the steps below need to be performed on each managed server that hosts OSM.

5. Click **Federation Services**, then select **SAML 2.0 General**.
6. Click **Publish Meta Data**.
   The Publish SAML 2.0 Meta Data page appears.
7. In the **Path** field, enter the full path and filename of the metadata file.

> **Note:**
>
> Back up this metadata file which is also used in configuring clients in KeyCloak.

8. Click **Lock & Edit**.

Registering IdP in WebLogic

This section describes how to register KeyCloak Service as a SAML Identity Provider (IdP) in WebLogic.

> **Note:**
>
> The IdP metadata XML file is to be exported from the KeyCloak realm. See step 4 in "Setting Up Keycloak for SAML SSO" for more details.

To register an IdP in WebLogic:

1. Upload the IdP metadata xml file obtained from the IdP (in this case, KeyCloak) to the server hosting WebLogic.
2. Access the WebLogic Administration Server Console as administrator.
3. Click **Security Realm** and then click **myRealm**.
4. Open the **Providers** page, then click the SAML2IdentityAsserter created in the section "Creating SAML2 Assertion Provider and SAML Authenticator".

5. Click **Management**, then click **New** and select **New Web Single Sign-On Identity Provider Partner** from the drop-down list.
   The **Create a Web Single Sign-On Identity Provider Partner** page appears.
6. In the **Name** field, enter the name (for example, WebSSO-IdP-Partner-1).
7. In the **Path** field, enter the path to the XML file that contains the identity provider's metadata.
8. Click **OK**.
9. Click the Web Single Sign-On Identity Provider Partner created above.
10. Ensure that the identity provider details are displayed in the **Site Info** and **Single SignOn Signing Certificate** tabs.
11. In the **General** tab, check the **Enabled**, **Virtual User**, and **Process Attributes** check boxes.
12. In the **Redirect URIs** field, enter the following:
    /OrderManagement/*.jsp

    /OrderManagement/*.html

    /OrderManagement/*.htm

    /OrderManagement/control/*

    /OrderManagement/control/*.jsp

    /OrderManagement/control/*.html

    /OrderManagement/control/*.htm

    /OrderManagement/orchestration/*

    /OrderManagement/orchestration/*.jsp

    /OrderManagement/orchestration/*.html

    /OrderManagement/orchestration/*.htm

    /OrderManagement/osmweb/*

    /OrderManagement/osmweb/*.jsp

    /OrderManagement/osmweb/*.html

    /OrderManagement/osmweb/*.htm

    /OrderManagement/processhistory/*

    /OrderManagement/processhistory/*.jsp

    /OrderManagement/processhistory/*.html

    /OrderManagement/processhistory/*.htm

    /OrderManagement/fallout-runtimeui/*

    /OrderManagement/fallout-runtimeui/*.jsp

    /OrderManagement/fallout-runtimeui/*.html

    /OrderManagement/fallout-runtimeui/*.htm

    /OrderManagement/osm-landingpage/*

    /OrderManagement/osm-landingpage/*.jsp

    /OrderManagement/osm-landingpage/*.html

    /OrderManagement/osm-landingpage/*.htm

13. Click **Save**.
    The WebLogic server console displays a confirmation message.
14. Sign out of WebLogic server and close your browser.

Updating the OSM Deployment Plan

Update the OSM deployment plan as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<deployment-plan xmlns="http://xmlns.oracle.com/weblogic/deployment-plan"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:schemaLocation="http://xmlns.oracle.com/weblogic/
deployment-plan http://xmlns.oracle.com/weblogic/deployment-plan/1.0/
deployment-plan.xsd"
                 global-variables="false">
    <application-name>oms.ear</application-name>
    <variable-definition>
        <variable>
            <name>cookieName</name>
            <value>JSESSIONID</value>
        </variable>
        <variable>
            <name>cookiePath</name>
            <value>/</value>
        </variable>
    </variable-definition>
    <module-override>
        <module-name>oms.ear</module-name>
        <module-type>ear</module-type>
        <module-descriptor external="false">
            <root-element>weblogic-application</root-element>
            <uri>META-INF/weblogic-application.xml</uri>
            <variable-assignment>
                <name>cookieName</name>
                <xpath>/weblogic-application/session-descriptor/cookie-name</
xpath>
                <operation>add</operation>
            </variable-assignment>
        </module-descriptor>
    </module-override>
    <module-override>
        <module-name>oms.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>cookiePath</name>
                <xpath>/weblogic-web-app/session-descriptor/cookie-path</
xpath>
                <operation>remove</operation>
            </variable-assignment>
        </module-descriptor>
    </module-override>
    <module-override>
        <module-name>osmwebui.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>cookiePath</name>
                <xpath>/weblogic-web-app/session-descriptor/cookie-path</
xpath>
                <operation>remove</operation>
            </variable-assignment>
        </module-descriptor>
    </module-override>
    <module-override>
        <module-name>osm-ui-web.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>cookiePath</name>
                <xpath>/weblogic-web-app/session-descriptor/cookie-path</
xpath>
                <operation>remove</operation>
            </variable-assignment>
        </module-descriptor>
    </module-override>
</deployment-plan>
```

1. Log in to the Oracle WebLogic Server Administration Console.
2. In the **Domain Structure** section, click **Deployments**.
3. In the **Configuration** tab, select the **oms** checkbox and then click **Update**.
   The **Update Application Assistant** screen appears.
4. In **Deployment Plan Path**, click **Change Path** and enter the path to your OSM deployment file.
5. Click **Finish**.

About KeyCloak Configuration

A KeyCloak realm needs to be created and utilized for the OSM SAML SSO. Also, the users needed for logging into OSM should have been created (or imported) into the realm; the users should also have their roles set properly to log into OSM. The "user federation" functionality of KeyCloak provides access to external databases and directories, such as LDAP and Active Directory, and can extract user account data.

> **Note:**
>
> Users and roles created are identical to what is available in Weblogic console as users and groups. SAML2 SSO is enabled for human user access through OSM web UIs. So the Identity Provider only needs to manage human users that access OSM through OSM UIs. Internal users like oms-automation and oms-internal are managed by the embedded LDAP.

Examples of roles used in OSM are:

> **Note:**
>
> Roles in KeyCloak are equivalent to groups in WebLogic console.

- OMS_client
- OMS_designer
- OMS_user_assigner
- OMS_workgroup_manager
- OSM_automation
- OMS_xml_api
- OMS_ws_api
- OMS_ws_diag
- OMS_log_manager
- OMS_cache_manager
- Administrators
- Cartridge_Management_WebService

See "Setting Up Keycloak for SAML SSO" for more information on setting up KeyCloak.

About Single Logout (SLO)

Since WebLogic does not support back channel logout, you must adopt front channel logout. This means that you need to configure the client in KeyCloak to front channel logout.

Configuring the KeyCloak client to front channel logout
To configure the KeyCloak client to front channel logout:

1. Log in to the KeyCloak Admin URL as an administrator.
2. Select the realm.
3. Click **Clients**.
4. Select the required client.
   The client setting screen appears.
5. Open the **Settings** tab, go to **Logout Settings**, select **Front channel logout**, and set its value to **On**.
6. Click **Save**.

Enabling SSO for OSM Traditional

To enable SSO for OSM traditional deployments:

1. Append the following code to **$DOMAIN_HOME/oms-config.xml**:

```
<oms-parameter>
     <oms-parameter-name>sso_enabled</oms-parameter-name>
```

```
        <oms-parameter-value>true</oms-parameter-value>
</oms-parameter>
```

2. Restart the WebLogic server that OSM runs.

Verifying SAML SSO

For verifying SAML SSO in OSM, you ideally need to have multiple OSM instances. All of these OSM instances need to have SAML SSO enabled, and be registered in the same KeyCloak realm. Assume there are two OSM instances: OSM1 and OSM2. To verify SAML SSO in OSM:

1. Go to the URL of the OSM1 instance (http://*IPaddress1*:*port1*/OrderManagement/) in your web browser.
   The login page of the identity provider is displayed.
2. Enter your login credentials.
   The OSM home page appears.
3. In the same web browser, open a new tab and enter the URL of the OSM2 instance (http://*IPaddress2*:*port2*/OrderManagement/).
   The browser automatically logs you in to the OSM home page of the OSM2 instance without asking for login credentials.

Verifying SAML SLO

Assume there are two OSM instances: OSM1 and OSM2. Log in to both instances in your web browser by following the steps in "Verifying SAML SSO."

To verify SAML SLO:

1. Click the **Logout *Username*** link in the corresponding web browser tab to log out from the OSM1 instance.
   The IdP logout screen appears. The web browser tab then redirects to the IdP login page.
2. Go to the web browser tab that has the OSM2 instance logged in.
3. Click any link or menu in the GUI.
   The web browser tab automatically redirects to the IdP login page. This indicates that the OSM2 instance is also logged out.

Choosing and Configuring an IdP

OSM SSO is built on vanilla SAML, which allows it to work with any Identity Provider (IdP) that supports SAML2. OSM has been utilizing Oracle IDCS and Keycloak to test and validate its SAML capability. Keycloak is an open-source identity and access management platform that allows organizations to securely manage user authentication, authorization, and user account management. It provides features such as single sign-on, social login integration, multi-factor authentication, user federation, and centralized user management. Keycloak helps organizations to secure their applications and services by providing a secure and centralized authentication and authorization system.

See the following sections for details:

- Setting Up Oracle IDCS for OSM SAML2 SSO
- Setting Up Keycloak for SAML SSO

Setting Up Oracle IDCS for OSM SAML2 SSO

This section describes how to use Oracle Identity Cloud Service (IDCS) as the IdP. Oracle IDCS is an Identity-as-a-Service (IDaaS) solution available in Oracle Public Cloud (OPC). OSM uses IDCS to authenticate users who would need access to the OSM Task Web client and the OSM Order Management Web client.

For more details about installing Oracle IDCS and the initial configuration, see: https://docs.oracle.com/en/cloud/paas/identity-cloud/uaids.

Setting Up OSM Predefined Groups in the IDCS Domain

You can use groups in an IDCS domain to give users access and permissions to use applications. A group is typically applicable to one type of user. However, you can assign a user to more than one group, if required.

To be synchronized to OSM, which is a WebLogic-based application, the groups in the IDCS domain need to be named exactly the same as the groups defined in the WebLogic security realm in the WebLogic domain.

Groups defined in the WebLogic domain can be found within the **Security Realms** in the OSM WebLogic domain.

To navigate to groups:

1. Click **Security Realms** in the drop-down list,
2. Select **myrealm**.
3. From the new drop-down list that appears, select **Users and Groups** ad then click **Groups**.

Typically, the list of groups defined for OSM are listed in "About KeyCloak Configuration." These groups are the ones that need to be defined as groups in the IDCS domain.

To define a group in the IDCS domain:

1. Click **Groups** in the IDCS domain menu.
   The **Groups in OSM Domain** page appears.
2. Click **Create group**.
3. In the **Create group** page that appears, enter the name and description of the group in the **Name** and **Description** fields.
4. Use the **Users** search bar to search for and select users to assign to the group.
5. Click **Create**.

Setting Up Users in the IDCS Domain

To set up a user in the IDCS domain:

1. Click **Users** in the IDCS domain menu.
   The **Users in OSM Domain** page appears.
2. Click **Create user**.
3. In the **Create user** page that appears, enter the first name, last name, and user name of the user in the **First name**, **Last name** and **Username** fields.
4. From the **Groups** list, select groups to assign the new user to. Use the **Groups** search bar to search for specific groups.

5. Click **Create**.

Registering OSM in the IDCS Domain

You can register OSM in the IDCS domain by adding a SAML application in the domain.

To add a SAML application in the IDCS domain:

1. Navigate to the **OSM domain** and click **Integrated applications** in the IDCS domain menu.
   The **Integrated applications in OSM Domain** page appears.
2. Click **Add application**.
3. In the **Add application** page, select **SAML Application** and click **Launch workflow**.
4. In the **Add SAML Application** page, enter the name of the application in the **Name** field, and click **Next**.
5. In the **General** section of the **Configure single sign-on** page, fill in the following fields:
   - **Entity ID**: For a traditional OSM environment, this should be the same as the **Entity ID** value provided while configuring SAML for SSO in WebLogic. See "Specifying General Information" for more details. For an OSM cloud native environment, its value is set to **$PROJECT-$INSTANCE-OsmSaml2IdentityAsserter**. In the service provider metadata file, this value can be found in the **entityID** attribute.

     > **Note:**
     >
     > Refer to "Publishing the Service Provider Metadata" for generating the service provider metadata file in a traditional OSM environment, and to "Generating Service Provider Metadata" for generating it in an OSM cloud native environment.

   - **Assertion consumer URL**: For a traditional OSM environment, this should be the same as the **Published Site URL** value set while configuring SAML for SSO in WebLogic, appended with **/sp/acs/post**. See "Specifying General Information" for more details. For an OSM cloud native environment, its value is set to **$PROTOCOL://$PROJECT.$INSTANCE.$DOMAIN_NAME:$PORT/saml2/sp/acs/post**. In the service provider metadata file, this value can be found at the **md:AssertionConsumerService** with the **Binding** attribute value as **urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST**.
   - **Name ID format**: Choose **Unspecified**.
   - **Name ID value**: Choose **Username**.
   - **Signing certificate**: To generate the signing certificate from the key store, utilize the command line `keytool -exportcert -keystore ./keystore.jks -storepass $STORE_PASSPHRASE -alias $SAML_SIGNING_KEY -file $EXOPORTED_SIGNING_CERT -rfc`. The file **./keystore.jks** is the keystore file used by the OSM WebLogic managed servers. The exported certificate is saved to **$EXOPORTED_SIGNING_CERT**. Upload the signing certificate into this field.
6. In the **Additional configurations** section of the **Configure single sign-on** page, fill in the following fields:
   - **Single logout URL**: For a traditional OSM environment, this should be the same as the **Published Site URL** value set while configuring SAML for SSO in WebLogic, appended with **/sp/slo**. See "Specifying General Information" for more details. For an OSM cloud native environment, its value is set

to **$PROTOCOL://$PROJECT.$INSTANCE.$DOMAIN_NAME:$PORT/saml2/sp/slo**. In the service provider metadata file, this value can be found at the **md:SingleLogoutService** with the **Binding** attribute value as **urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST**.

- **Logout response URL**: Fill in the same value entered in the **Single logout URL** field above.

7. In the **Attribute configuration** section, click **Additional attribute**.
8. Add an attribute by filling in the following fields:
   - **Name**: Enter **Groups**.
   - **Format**: Choose **Basic**.
   - **Type**: Choose **User attribute**.
   - **Type value**: Choose **Group membership**.
   - **Condition**: Choose **All groups**.
9. Click **Finish** to generate the integrated application in the IDCS domain.
10. Review the application configuration.
11. Click **Activate** to activate the application.

Exporting the IdP Metadata XML File

The IdP metadata XML file contains details that you can use to connect to the SAML application configured in the IDCS domain. IDCS exports this file and shares it with you, so that you can communicate SAML messages to the IDCS SAML application safely.

To export the IdP metadata XML file:

1. Navigate to the application configuration page of the SAML application created earlier.
2. Click **Download identity provider metadata** to retrieve the file.

The IdP metadata XML file is used by WebLogic to configure the SAML2 Assertion Provider in the OSM WebLogic domain. For OSM cloud native environments, the file is used in the process of creating the SSO archive secret. See "Creating SSO Archive Secret" for more details.

For traditional OSM environments, the file is used in the process of registering an IdP in WebLogic. See "Registering IdP in WebLogic" for more details.

Assigning Users to the SAML Application

To assign users to the SAML application:

1. Navigate to the application configuration page of the newly created SAML application.
2. Select **Users** from the **Resources** menu.
3. Click **Assign users**.
4. In the **Assign users** page, select the required users from the list. You can also use the search bar to search for specific users.
5. Click **Assign**.

Setting Up Keycloak for SAML SSO

An IdP is required for user authentication and authorization to support SAML2 SSO in OSM. There are multiple IdPs available, from which you can choose. In this document, we are using Keycloak as the IdP.

For more information on installing and configuring Keycloak, see the guides at: https://www.keycloak.org/guides#server

Setting Up Keycloak SAML Realm and SAML Client
A realm in Keycloak is equivalent to a tenant. Each realm allows an administrator to create isolated groups of applications and users. Keycloak comes with a single realm called `master`. Use this realm only for managing Keycloak, and not for managing any applications.

It is recommended to create a separate Keycloak realm for your OSM SAML2 SSO integration.

To set up Keycloak SAML realm and SAML client:

1. Create a security realm by following the steps given below:

> **Note:**
>
> If you are using an existing realm for OSM SAML2 SSO, you are not required to perform the following steps. You can directly register OSM in the KeyCloak realm by adding it as a client.

   a. Open the Keycloak admin console, and log in with an administrator user.
   b. Select **master** from the top-left corner and then click **Create Realm**.
   c. Enter the realm name in the **Realm name** field, for example OSM1.
   d. Set **Enabled** to **On**.
   e. Click **Create**.
   f. Configure the **Realm Settings** as needed.

> **Note:**
>
> The **Require SSL** field needs to be changed according to the OSM endpoints it connects to. If requires SSL, you need to set it to **All requests**, otherwise, set it to **None**.

2. Register OSM in the Keycloak realm by adding it as a client.
   You need to register the OSM instance in the Keycloak realm to enable Keycloak to exchange SAML messages with the OSM managed server.

   To register OSM in the keycloak realm by adding it as a client, import the service provider metadata file from OSM.

   The service provider metadata file here is an XML file that contains details of how the IdP (Keycloak) is used to connect to the service provider (OSM instance).

   For OSM Cloud native environments, see "Generating Service Provider Metadata" for details on how to acquire this service provider metadata file.

For traditional OSM environment, see "Publishing the Service Provider Metadata for details on how to acquire this service provider metadata file from the OSM WebLogic domain."

To import the service provider metadata file from OSM:

a. Select **OSM1** from the top-left corner.

b. Click **Clients** from the menu.

c. In the **Clients lists** tab, click **Import client**.

d. In the **Resource file** field, click **Browse**.

e. Select the required XML file.

f. Make sure that **Front channel logout** is set to **On** and then click **Save**.

3. Configure a role list mapper upon the client:

> **Note:**
>
> In the **Client Scope** page, make sure there is no **role_list** assigned client scope created for this client. If it has been created, you need to delete it by selecting the **Remove** option in the sub menu.

a. Select the client you added in step 2.

b. Go to **Client scopes** tab and click the link in the **Assigned client scope**.

c. In the **Mappers** tab, click **Add mapper** and then select **By configuration**.

d. Select **Role list** from the **Mapper**.

e. Click **Created role list mapper**.

f. In the **Role list** tab, set the **Role attribute name** value to "Groups".

g. Make sure to set the value of **Single Role Attribute** to **On**.

h. Click **Save**.

4. Export the IdP metadata XML file.
The IdP metadata XML file contains details for the service provider used to connect to the Keycloak realm. Keycloak exports this file, and shares it with the service providers, so that service providers can communicate SAML messages with the Keycloak realm safely.

For OSM cloud native environments, this IdP metadata file is used in "Creating SSO Archive Secret."

For traditional OSM environments, this IdP metadata file is used in step "Registering IdP in WebLogic"

To export the IdP metadata file:

a. Click **Realm settings** under **OSM1** realm.

b. Under the **General** tab, from the **Endpoints** field, click **SAML 2.0 Identity Provider Metadata** to retrieve the IdP metadata XML file. This file would be used by WebLogic to configure the SAML2 assertion provider in the OSM WebLogic domain.

Setting Up Users and OSM Predefined Roles in the Security Realm
To set up users and OSM predefined roles in the security realm:

1. Set up OSM predefined roles.

Roles in a Keycloak realm are used to give users access and permission to use applications. While a role is typically applied to one type of user, a user can be assigned with one or multiple roles. To be synchronized with OSM, which is a WebLogic-based application, the roles in the Keycloak realm need to be named exactly the same as the groups defined in the WebLogic security realm in the WebLogic domain.

Groups defined in the WebLogic domain can be found within the **Security Realms** in the OSM Weblogic Domain.

To navigate to groups, click **Security Realms** in the drop down list, and select **myrealm**. From the drop down list that opens, select **Users and Groups** and then click **Groups**.

The following are the groups that are usually defined for OSM:

- OMS_client
- OMS_designer
- OMS_user_assigner
- OMS_workgroup_manager
- OSM_automation
- OMS_xml_api
- OMS_ws_api
- OMS_ws_diag
- OMS_log_manager
- OMS_cache_manager
- Administrators
- Cartridge_Management_WebService

> **Note:**
>
> The groups can be different in your OSM WebLogic domain.

These groups need to be defined as roles in the Keycloak realm.

To define a group (OMS_client) as a role in Keycloak:

a. Select **Realm roles** from the menu in the Keycloak realm.
b. Click **Create role**.
c. In the **Create role** page, type in the **Role name**.
d. Click **Save**.

2. Set up users in the security realm.
   a. Select **Users** from the menu in the Keycloak realm.
   b. Click **Add user**.
   c. In the **Create user** page, type in the **Username**.
   d. Click **Create**.
   e. Select the newly created user.
   f. In the **User details** page, select the **Role mapping** tab.

g. Click **Assign role**.

h. Select the roles that you want to assign to the user.

i. Click **Assign**.

3. Set up a password for the user.

a. Select **Users** from the menu in the Keycloak realm.

b. Select the specific user to which you want to set up a password.

c. In the **User details** page, select **Credentials** tab and then click **Set password**.

d. Type in the password in **Password** and **Password confirmation** fields.

e. Make sure that the value for **Temporary** is set to **Off**.

f. Click **Save**.

About Single Logout (SLO)

SLO allows you to log out of all applications within your current identity provider login session. It ensures that when you initiate a logout, you are logged out of all connected applications simultaneously. SLO can be either service provider-initiated or identity provider-initiated.

- Service Provider-initiated SLO: The SP initiates the logout process, which is the front-channel.
- Identity Provider-initiated SLO: The IdP initiates the logout process, which is the back-channel.

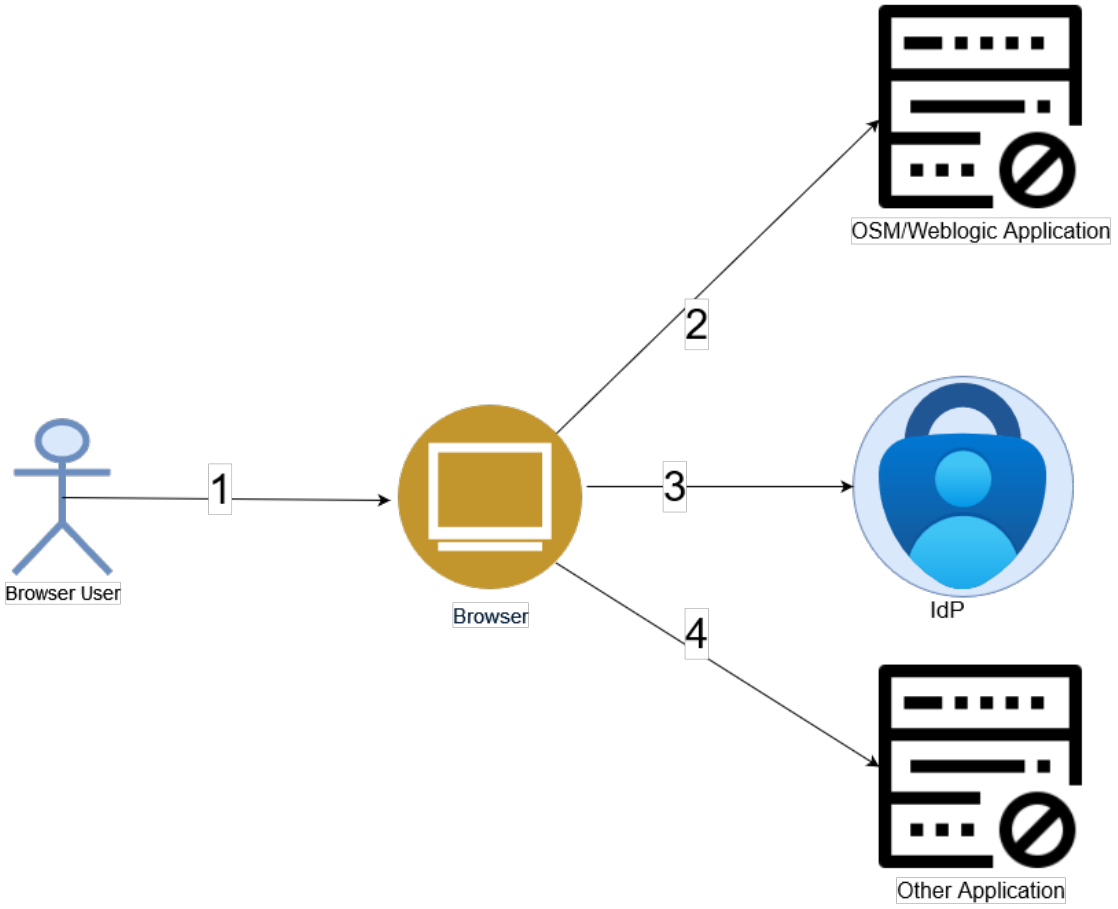
> **Note:**
>
> Not all IdPs support both methods. When you log out, the IdP ensures logout across all participating applications in a security realm. Proper configuration of SLO URLs is essential during application registration with the IdP.

About Front-channel Logout

OSM SAML SLO only supports front-channel configuration on the IdP side. Before proceeding, you should understand the difference between a back-channel and a front-channel logout. A front-channel logout relies on the browser with an iframe in the browser window that contains the application's logout URL. A front-channel logout is triggered when you initiate a logout from an application. The application sends a logout request to the IdP, which then broadcasts the request to all SPs where you have an active session.

The following diagram illustrates the front-channel flow in OSM Weblogic SAML configuration.

Figure 4-3 Front-channel Flow in OSM Weblogic SAML Configuration



> **Note:**
>
> OSM Weblogic does not support back-channel SLO flow. A back-channel logout takes place between IdP and its SPs. IdP detects your logout and sends a request containing a logout token to all registered SPs, via SAML client within a security realm, where you are logged in. The important aspect of back-channel is that it does not rely on a browser to broadcast logout requests to other SP applications.

About IdP Security Realm

A realm is a fundamental concept in every IdP architecture. It allows you to isolate and organize resources, permissions, and configurations. Think of a realm as a logical grouping where you manage various objects, including the following:

- **Users**: Individuals who belong to and log into a specific realm.
- **Applications**: Services or client applications that interact with the authentication provider.
- **Roles**: Permissions or access levels assigned to users and applications.
- **Groups**: Organizational units for users and applications within the realm.

Realms can represent different organizations, departments, or projects. OSM recommends that you configure policies and settings specific to each realm, controlling access to resources within that realm. Realms provide a powerful way to organize and secure OSM instances and their users in IdP. Security Realm can be used to define the SSO scope and partition users. You can define and organize security realm to meet your business requirements.

About Using User, Group, and Role Federation

User federation allows seamless integration with external user stores and partners. User federation enables IdP to connect to external user stores, such as LDAP, Active Directory, or custom databases. It provides a unified way to federate different user account systems into IdP.

When you log in, IdP searches its internal user store first. If the user is not found there, it iterates over configured user storage providers (federation providers) until it finds a match. IdP can store and manage users but often, organizations already have LDAP or Active Directory services that store user and credential information for their organizations.

You can point your IdPs to validate credentials from those external user stores and pull in identity information. Additionally, the roles and groups in a security realm or its clients can be sourced from external LDAP providers. For details, refer to the documentation of your IdP.

> **Note:**
>
> OSM does not provide configuration details on Users, Roles and Groups federation. You are responsible for the desired configuration meeting both business and security requirements.

Considerations When Using OSM SAML

This topic provides information about the things that you need to consider when setting up OSM SAML. In the sections that follow, you will find more information about:

- Security Considerations
- Development and Production Environments
- Additional Considerations

Security Considerations

This topic provides details about security considerations.

Message Confidentiality
OSM-SAML requires TLS 1.2 configuration to guarantee message confidentiality and integrity during transport. This helps to counter attacks such as eavesdropping, theft of user authentication information, and man-in-the-middle attacks. You should digitally sign messages with a certified key to ensure integrity and authentication.

SAML Assertion and Response Encryption
Sensitive attributes within assertions can be encrypted to prevent disclosure after transportation. IdP encrypts the assertion or response and SP decrypts it. This mitigates the risk of theft of user authentication information. Encryption is done using private keys, (not public keys) when asymmetrical cryptography is selected, which is IdP dependent. If the SAML assertion needs to be encrypted, the IdP must use the public key or the public certificate to

encrypt the assertion. In this case, SP uses a private key and the public certificate containing its public key to decrypt the encrypted assertion.

> **Note:**
>
> OSM cloud native provides two sets of private keys and their corresponding public certificates for encryption and SAML signing.

OSM relies on java keystore that contains these keys along with the other required keys. For more details on how to create a key store to be used for SAML in OSM, refer to "Creating Keystore for OSM SAML2 SSO".

Signing SAML Requests, Responses, and Assertions
Signing is done using private keys. Verifying the signature needs a public key published by the signer. In SAML flow, the IdP provides its certificate to the SP to validate the signed SAML assertion or responses by the IdP using its private key. If the SAML request needs to be signed by OSM, then OSM must use its private key for this purpose and publish a public certificate for IdP verifying the signed request. This cryptographic method ensures secure communication between the SP which are OSM and IdP, maintaining the integrity and authenticity of SAML tokens during authentication.

OSM requires its users to form a Java keystore file containing certificates and private keys for signing and encryption. OSM will form a Kubernetes secret with all required data for Weblogic to be able to use the externally supplied keystore (such as a passphrase). This data is stored in Java key stores used by Weblogic containers in cloud native. Here, they need to decrypt the SAML assertion, verify the signature of the SAML response or assertion (signed by IdP) and sign the SAML request where applicable. For more details, refer to "Creating Keystore for OSM SAML2 SSO".

To prevent assertion replay attacks in SAML, consider the following measures:

- Secure Communication: Use TLS-secured connections between the Identity Provider (IdP) and the SP to prevent man-in-the-middle attacks.
- Assertion Signing and Encryption:
  - Sign the assertions: Properly signed assertions ensure their integrity and authenticity. If an assertion is altered, the signature verification will fail, preventing replay attacks.
  - Encrypt the assertions: Encryption adds an extra layer of security, ensuring that sensitive information within the assertion remains confidential.

> **Note:**
>
> Replay attacks involve the unauthorized reuse of valid intercepted SAML messages. By implementing these precautions, you can enhance the security of your SAML-based single sign-on (SSO) system.

Creating Keystore for OSM SAML2 SSO

There are two pairs of keys and certificates used in SAML2 SSO in WebLogic to exchange SAML SSO and SLO messages against the IdP. One pair is used to sign the messages, while the other is used for encryption. This section explores the use of externally provided keys and certificates for signing and encrypting the SAML messages.

You need to import the externally provided private keys and certificates to the keystore first. WebLogic loads the private keys and certificates from the keystore.

See the following topics:

- Preparing the Keystore
- Creating Keystore with Private Keys and Self-signed Certificates

Preparing the Keystore

Besides the SAML2 SSO signing key and certificate and the SAML2 SSO encryption key and certificate required by SAML SSO, there can be other keys and certificates required by the OSM WebLogic domain for other purposes. One example is the key and certificate used by the OSM managed server to establish SSL connections when SSL is enabled in a traditional OSM environment. This pair of key and certificate will be used by the managed server as default.

All the pairs of keys and certificates used by the OSM managed servers must be imported to the same keystore. Typically, in a production environment, a certificate needs to be signed by a CA (certificate authority).

For each key and certificate pair:

1. Create the private key and public key.
2. Have the public key signed and get the certificate.
3. Import the private key and the certificate to the keystore.

There are various ways to create keystores, generate private keys and public keys, and add keys to the JKS keystore. The following section provides an example that uses the "keytool" to create a keystore with three pairs of private keys and self-signed certificates in the keystore.

Creating Keystore with Private Keys and Self-signed Certificates

You can only use self-signed certificates for development environments. They should not be used in a production environment. For production environments, use CA (certificate authority) signed certificates.

You can create a keystore with private keys and self-signed certificates in both traditional and cloud native instances of OSM. The following section describes keystore creation for both instances.

Creating Keystore for OSM Traditional
For a traditional OSM instance, there are three pairs of keys and certificates that should be acquired for SAML signing, SAML encryption and SSL. You need to create three pairs of keys and certificates in the keystore.

> **Note:**
>
> The key and certificate pair acquired for SSL are also used as the default of the managed server.

The following is an example to create the keystore and keys:

```
keytool -genkeypair -alias saml_sign_key -keyalg RSA -keysize 2048 -dname
"CN=quick.sr.com,OU=Support,O=Oracle,L=Kanata,ST=Ontario,C=CA" -keystore
keystore.jks
keytool -genkeypair -alias saml_encrypt_key -keyalg RSA -keysize 2048 -dname
```

```
"CN=quick.sr.com,OU=Support,O=Oracle,L=Kanata,ST=Ontario,C=CA" -keystore
keystore.jks
keytool -genkeypair -alias ssl_key -keyalg RSA -keysize 2048 -dname
"CN=quick.sr.com,OU=Support,O=Oracle,L=Kanata,ST=Ontario,C=CA" -keystore
keystore.jks
```

The keystore is created at **keystore.jks**, and self-signed certificates are also created and added to the keystore for:

- **ssl_key**, for SSL key and certificate.
- **saml_sign_key**, for SAML SSO signing key and certificate.
- **saml_encrypt_key**, for SAML SSO encrypting key and certificate.

Collect the following information when creating the keystore, which would be used in configuring the OSM WebLogic domain:

Table 4-5 Information Required When Creating Keystore for OSM Traditional

Description	Example of Value	Comment
Keystore JKS file path and name	**/path/keystore.jks**	The JKS keystore file contains the three key and certificate pairs.
Keystore passphrase	*Keystore passphrase*	The passphrase to access the keystore.
SSL key alias	**ssl_key**	The key alias used for SSL in the keystore.
SSL key passphrase	*Key passphrase*	The passphrase to access the SSL key in the keystore.
SAML SSO signing key alias	**saml_sign_key**	The key alias used for SAML SSO signing in the keystore.
SAML SSO signing key passphrase	*Key passphrase*	The passphrase to access the SAML SSO signing key in the keystore.
SAML SSO encrypting key alias	**saml_encrypt_key**	The key alias used for SAML SSO encrypting in the keystore.
SAML SSO encrypting key passphrase	*Key passphrase*	The passphrase to access the SAML SSO encrypting key in the keystore.

Creating Keystore for OSM Cloud Native

In an OSM cloud native environment, the SSL key and certificate pair is used as the SSO encrypting key and certificate pair. You do not need to have a separate SAML SSO encrypting key and certificate. This means that only two pairs of keys and certificates are needed in the keystore.

The following is an example to create the keystore and keys:

```
keytool -genkeypair -alias saml_sign_key -keyalg RSA -keysize 2048 -dname
"CN=quick.sr.com,OU=Support,O=Oracle,L=Kanata,ST=Ontario,C=CA" -keystore
keystore.jks
keytool -genkeypair -alias ssl_key -keyalg RSA -keysize 2048 -dname
"CN=quick.sr.com,OU=Support,O=Oracle,L=Kanata,ST=Ontario,C=CA" -keystore
keystore.jks
```

The keystore is created at **keystore.jks** and self-signed certificates are also created and added to the keystore for:

- **ssl_key**, for SSL key and certificate.

- **saml_sign_key**, SAML SSO signing key and certificate.

Collect the following information when creating the keystore, which would be used in configuring the OSM WebLogic domain:

Table 4-6 Information Required When Creating Keystore for OSM Cloud Native

Description	Example of Value	Comment
Keystore JKS file path and name	**/path/keystore.jks**	The JKS keystore file that contains the two key and certificate pairs.
Keystore passphrase	*Keystore passphrase*	The passphrase to access the keystore.
SSL key alias	**ssl_key**	The key alias used for SSL in the keystore.
SSL key passphrase	*Key passphrase*	The passphrase to access the SSL key in the keystore.
SAML SSO signing key alias	**saml_sign_key**	The key alias used for SAML SSO signing in the keystore.
SAML SSO signing key passphrase	*Key passphrase*	The passphrase to access the SAML SSO signing key in the keystore.

Development and Production Environments

OSM does not recommend using the SAML scheme in development environments. Development and test phases of OSM solution development can benefit from the simplicity of Basic Authentication schemes using Weblogic-embedded LDAP. OSM SAML schemes are more applicable in pre-production and production environments.

OSM cartridge roles as well as OSM system-defined Weblogic roles such as **OMS_Client** must be sourced in IdP when SSO is enabled. You are responsible for having the OSM cartridge roles in sync with the user's roles defined in IdP. Whether these roles are defined in IdP or mapped from the LDAP directory is a deployment choice and shall be configured in IdP.

> **Note:**
>
> OSM recommends these roles sourced in IdP and defined as realm or client roles and use LDAP for user identities and their working groups.

Additional Considerations

The following are some additional things for you to consider when using OSM SAML:

- OSM does not own or provide login pages when OSM is configured with SSO. The login page is outsourced to IdP. You can configure and customize the login page in your chosen IdP.
- OSM SAML relies on IdP session management via cookie-based session Id when SSO interoperates with applications that are using OIDC.
- Due to WebLogic constraints, OSM SLO is limited to logout initiation from the origin applications (SP) and not from IdP. Thus, in OSM 7.5.0.0.1 session revocation and idle session timeout capabilities within IdP are not supported.

About Configuring Session Timeout

OSM relies on Weblogic to parse and validate the SSO session and its associated SAML Assertion Token. Weblogic SSO does not take the following configurations into account:

- SSO idle session timeout
- SSO Session revocation
- In the SAML2 specification, there are several places in the assertion token where it is possible to specify a **lifetime**:
  - The *SubjectConfirmationData* element contains a `NotOnOrAfter` attribute.
  - The *Conditions* element contains a `NotOnOrAfter` attribute.
  - The *AuthnStatement* element contains a `SessionNotOnOrAfter` attribute.

To mitigate all of the above, configure the session timeout on the OSM side equal to your desired corporate idle session timeout value on the IdP side.

About Using OpenID Connect (OIDC)

OpenID Connect (OIDC) is an identity layer built on top of the OAuth 2.0 protocol. It allows clients (applications) to verify the identity of end-users based on authentication performed by an authorization server. In other words, OIDC provides a way to ensure that a user is who they claim to be. OIDC achieves this through the use of JSON Web Tokens (JWTs), which are digitally signed and contain claims about the user. By verifying the signature of the JWT, clients can be confident in the authenticity of the user's identity. This makes OIDC a powerful tool for building secure and user-friendly authentication systems.

OSM recommends the Client Credentials flow in OIDC. The benefit of using Client Credentials in contrast to merely basic authentication using API keys is two-fold:

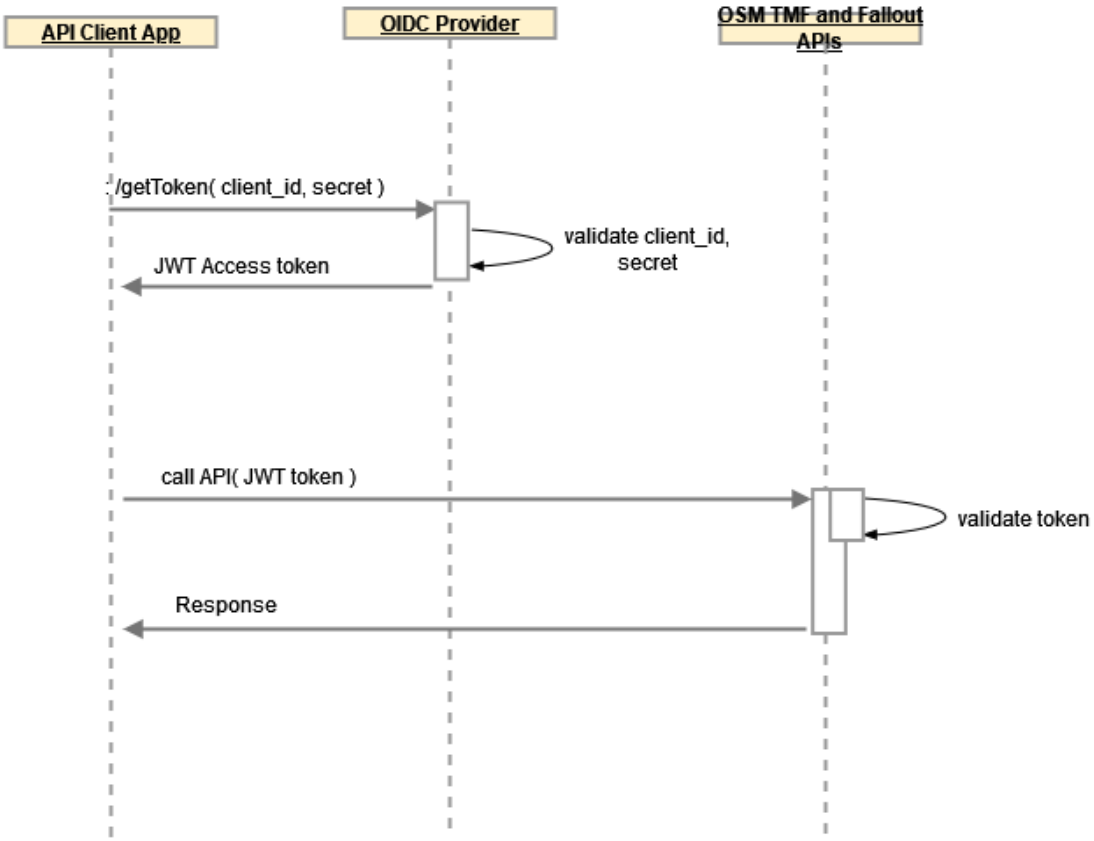
- Firstly, your API infrastructure can be made uniform, no matter if the request comes from an authenticated user or from an application with a machine credential, the authentication in the API can be reused.
- Secondly, by using Client Credentials, you limit the parties whom you send the credentials to. In OIDC (OAuth2), the client calls the token endpoint to retrieve an access token and then uses that token to call the API.

OSM uses OIDC to authenticate its TMF API and Fallout users only. OIDC is the only Authentication scheme provided by OSM for these APIs. OSM continues to support basic authentication schemes for other OSM public APIs, such as XML API and Web Services. OSM recommends using Weblogic Embedded LDAP (Default Auth Provider) for these APIs.

OSM does not provide SSO via OIDC for human users and web pages. You can configure your IdP for having interoperable OIDC and SAML clients. IdPs often provide this capability via the IdP user's session management. OSM does not provide this support out of the box.

OSM does not provide OIDC or OAuth2 based authorization (using roles or scopes) for TMF and Fallout APIs and their associated resources. In the following diagram, you can see the flow of client credentials in OIDC.

Figure 4-4 OSM OIDC Client Credentials



OSM requires JWT tokens obtained through authorization code or client credential flow, which is recommended, to secure TMF and Fallout APIs. Like SAML, OSM does not prescribe any specific IdP as OIDC or OAuth provider. You can choose your own desired IdP.

Here are the highlights of the Client Credentials flow:

- Applications authenticate as themselves (not on behalf of a user). Hence API callers authenticate directly without user involvement.
- They programmatically and securely obtain the OIDC access token to an API.

OIDC simplifies authentication and ensures secure access to APIs. OSM continues using this method for its new future APIs. The OSM term "machine users" is only applicable in the Basic Authentication scheme.

> **Note:**
>
> Given TMF and Fallout APIs are only meant for machine users:
>
> - OSM recommends using the access token, instead of ID token, if provided in the form of JWT by your IdP.
> - OSM recommends using **grant_type=client_credentials** to obtain the access token. Hence, no user needs to be created or federated into IdP. **Password** grant type requires a machine user to be created in the realm. This grant type allows obtaining an access token by providing a username and password.
> - OSM does not provide fine-grained authorization for these APIs via vanilla scopes or roles. You can rely on the Authorization policies which are dependent on your IdP to restrict and authorize access to an OSM OIDC client within a security realm.

About JSON Web Tokens (JWTs)

JSON Web Tokens (JWTs) are widely used for exchanging authentication information securely on the web. The following are some key security considerations when dealing with JWT:

- **Token Structure**: JWTs consist of three parts: the header, the payload (claims), and the signature. The header contains information about the token, such as the algorithm used for signing. The payload holds claims, which include user identity and permissions. The signature ensures the integrity of the token.
- **Statelessness and Session Management**: JWTs enable stateless session management, eliminating the need for session cookies. Once configured, the backend does not need to communicate with the authorization server repeatedly.

The following are some advantages of using JWTs:

- **Compact Representation**: Simplifies communication between services.
- **Security and Trust**: JWTs are digitally signed, ensuring their integrity. IdPs and SPs can validate the token signature before trusting it.
- **Avoiding User Credential Sharing**: OIDC avoids sharing user credentials (such as passwords) with services. Instead, it relies on tokens, enhancing security.

The following are some best practices related to security:

- **JWT storage**: Avoid storing JWTs in local browser or session storage due to lack of XSS protections (HttpOnly and secure flags). Consider memory-only handling.
- **Crypto keys**: Protect the cryptographic keys on the IdP side.
- **Secure Channels**: Always send JWTs over secure channels such as HTTPS to prevent eavesdropping or interception.

> **Note:**
>
> JWTs are digitally signed, and not encrypted necessarily, which is why the emphasis is on signed tokens. Signed tokens can verify the integrity of the claims contained within them, while encrypted tokens hide those claims from other parties. When tokens are signed using public or private key pairs, the signature also certifies that only the party holding the private key is the one that signed it. Normally, JWTs can be signed using a public or private key pair using **RSA** or **ECDSA**. While JWTs guarantee data ownership, they are not encrypted. Always use JWTs over HTTPS for enhanced security.

Configuring OIDC for OSM TMF APIs and Fallout APIs

The following sections provide you with details about configuring OIDC for OSM TMF APIs and Fallout APIs:

- For details about configuring OIDC for OSM, see "Configuring OIDC for OSM".
- For details about configuring OIDC in Keycloak, see "Configuring OIDC in Keycloak ".

Configuring OIDC for OSM

This section provides information about configuring OpenID Connect (OIDC) for OSM microservices.

As a prerequisite, you need to create a unified authentication service that supports the standard OIDC protocol such as Keycloak. For more information about setting that up for OSM, refer to "Configuring OIDC in Keycloak ".

The following table lists the OIDC parameters required to configure OIDC to access the TMF APIs and the Fallout API:

Table 4-7 OIDC Configuration Parameters for TMF APIs and Fallout API

Parameters	Description
audience	Audience of issued tokens.
Auth URL	URI of the identity server, base used to retrieve OIDC metadata. such as http://*host*:*port*/realms/osm
client_id	Client ID as generated by OIDC server.
client_secret	Client secret as generated by OIDC server. Used to authenticate this application with the server when requesting JWT based on a code.
scope	openid
Token URL	This is the URL where you exchange the authorization code for an access token and ID token. This step typically occurs after you have successfully authenticated and granted permissions such as http://*host*:*port*/realms/osm/protocol/openid-connect/token

To obtain the access token IdP, run the following command:

```
curl --noproxy '*' -i -H 'Authorization: Basic
base64Encodedclient_id:client_secret' -H 'Content-Type: application/x-www-
form-urlencoded' -XPOST token_url -d
'grant_type=client_credentials&scope=openid'
```

OSM cloud native uses a secret for the administrator to provide the required OIDC information to secure its REST APIs.

Using the OIDC parameters defined in the above table, the Kubernetes secret can be created using the `manage-instance-credentials.sh` script from the OSM cloud native toolkit, with the OIDC option.

Run the following script to create the required secret:

```
$OSM_CNTK/scripts/manage-instance-credentials.sh -p sr -i quick create oidc
```

Configuring OIDC in Keycloak

This section provides information about configuring OIDC in Keycloak. This section does not cover the installation of the Keycloak IdP instance.

> **Note:**
>
> This configuration is an example and is not meant to reflect the actual configuration in production deployments.

The two steps you need to complete to configure OIDC in Keycloak IdP are:

1. Creating a Realm
2. Creating a Client

Creating a Realm
To create a new realm to manage applications:

1. Open the Keycloak console.
2. In the top left corner, click **Master**, and then click **Create Realm**.
3. Fill the form that opens by adding the realm name, for example, OSM.
4. Click **Create**. This creates a new realm.

To verify that your realm is created, in the top-left corner where you could see **Master** previously, you should now be able to see the name of the realm you created, or **OSM** if you followed the example.

Click on the realm name **OSM** to change the current realm.

> **Note:**
>
> Make sure all configurations or modifications are saved before changing the current realm or you will be subject to losing your configuration.

Creating a Client
To create a new client:

1. Open the Keycloak console.
2. Make sure the current realm, **OSM** is selected on the top-left corner in the dropdown menu.

3. Navigate to the **Manage** section and then click **Clients**. This window displays a clients list from the realm.
4. Click **Create client** and fill in the form that comes up:
   - **Client Type** as OpenID Connect.
   - **Client ID**. For example **tmf_client**.
   - **Name**. For example, **tmf**.
   - Turn on **Always Display in Console**.
5. Click **Next**.
6. On the Client Capability Config page, turn on the following configurations:
   - Client authentication
   - Authorization
   - Under Authentication flow, select the following checkboxes
     - Standard flow
     - Direct access grants
     - Implicit flow
7. Click on **Save**.
8. Navigate to the **Manage** section on the left and click **Clients**.
9. On the **Clients** page, select the **Credentials** tab.
10. Select **Client Authenticator: Client ID and Secret**.
11. Copy the client secret or click the regenerate box to regenerate the client secret.

Configuring OpenID Endpoint

1. Navigate to the **Config** section in the left menu and click **Realm Settings**.
2. On the **Realm** page, under the **General** tab, click the OpenID Endpoint Configuration link to get the JSON file. You can find `token_endpoint`.

To obtain the access token IdP, run the following command:

```
curl --noproxy '*' -i -H 'Authorization: Basic base64Encoded client_id:client_secret' -H
'Content-Type: application/x-www-form-urlencoded' -XPOST token_url -d
'grant_type=client_credentials&scope=openid'
```

OIDC in Keycloak is now configured and ready to use.

About Authorization in OSM

OSM has two levels of authorization:

- OSM Predefined Roles
- User-defined Roles in OSM cartridges

The following sections describe these two levels of authorization.

About OSM Predefined Authorization Roles

By default, OSM has a set of Authorization roles. These roles are mapped to Weblogic groups and used in OSM to provide authorization on various OSM operations and APIs. The following list describes these OSM Authorization roles.

- **OMS_client**: This role is required for human users to be able to access OSM Web UIs.
- **OSM_automation**: This role is required for automated users associated with OSM plugins.
- **OMS_ws_api**: This role is required for Web Service API users.
- **OMS_xml_api**: This role is required for XML API users.
- **osmRestApiGroup**: This role is required for REST API users such as process history and OLM users.
- **omsMetricsGroup**: This is an internal role used for OSM application metrics.
- **OMS_log_manager**: This is required for you to manage log levels of the OSM core application dynamically.
- **OMS_cache_manager**: This is an internal role used for debugging. This role allows you to retrieve order data cache via HTTP.
- **OMS_ws_diag**: This is an internal role used for debugging. This role allows you to diagnose compensation-related data on an order in the Amending state.
- **OMS_user_assigner**: This role is required for an Admin user to be able to associate users to the cartridge roles in workgroups. This is not applicable for those Authentication schemes, where roles are sourced in IdP, that is, in SSO or SAML.
- **OMS_workgroup_manager**: This role is required for an Admin user to manage OSM cartridge roles in workgroups. This is not applicable for those Authentication schemes, where roles are sourced in IdP, that is, in SSO or SAML.
- **Cartridge_Management_WebService**: This is an internal role used by OSM. It receives the cartridge operation request from Design Studio, which is a Webservice endpoint.

The following sections describe these roles in the context of OSM Authentication schemes.

OSM Predefined Roles in Basic Authentication Scheme
In Basic Authentication (embedded LDAP or External LDAP), the aforementioned roles are not defined in IdP and they are sourced in Weblogic or external LDAP.

- Weblogic Embedded LDAP: OSM cloud native and traditional instances create these roles as Weblogic groups. In these cases these roles are sourced and scoped to the Weblogic domain instance and not externalized.
- External LDAP: You can optionally create these roles in external LDAP and have them mapped to the Weblogic domain for OSM to consume. By default, OSM does not provide this configuration. For more details, refer to *Administering Security for Oracle WebLogic Server*.

OSM Predefined Roles in SAML Authentication Scheme
When OSM is configured for SSO using IDP and SAML 2.0 protocol, you should create these roles in your own chosen IdP. There are various ways to define these roles in IdP:

In general, roles can be defined as following:

- **Realm Roles**: These are global roles within the entire realm. When OSM roles are scoped to a realm, all OSM application instances can leverage the roles if they are associated with the realm's users.

- **Client Roles**: These are specific to a particular client (application) within the realm. If OSM roles are defined as client roles, these roles are only available to the specific application or OSM instance where roles are defined.
- **Federated Roles**: These roles are defined in external LDAP and federated into the IdP realm or realm's client.

OSM does not provide scripts to configure these required roles in IdP. That said, for more information on how these roles are created in Keycloak as an example, refer to the following section.

OSM Predefined Roles in IdP (KeyCloak)
Human users who are not associated with the predefined role **OMS_client** have no access to OSM user interfaces, despite successful authentication.

OSM predefined roles should be available in IdP and users should be assigned to these roles accordingly. For user interface, **OMS_client** role alone is sufficient.

Create OSM predefined role **OMS_client** in IdP (Keycloak) to access OSM user interfaces. See "About Cartridge Roles in SAML Client IdP (Keycloak)" for more information on how to create a role in IdP (Keycloak).

Make sure that the OSM Predefined role **OMS_client** is present in IdP (Keycloak). These roles should be federated in IdP from LDAP. Also, make sure that the OSM predefined role **OMS_client** is assigned to a user in IdP (Keycloak).

About User-Defined Roles in OSM Cartridges

OSM has a proprietary ACL model as part of its metadata, allowing cartridges to define roles for OSM functions and operations. OSM order and task and other sensitive operations are protected. Only users of a specified role can perform these functions. The following are the list of OSM-protected operations:

- Change task or order data
- Change task state or completion
- View order data
- Worklist
- Perform order lifecycle operations defined in OSM Order State Policy, for example, suspend and cancel.

Cartridge Roles in SAML Authentication Scheme
In these schemes, the cartridge roles are sourced from IdP, despite the definition in the cartridge. Therefore, they either need to be directly defined in IdP or mapped from external LDAP. IdP as SAML provider encodes your associated roles into SAML assertion token.

Thus, OSM runtime honors these roles only when they are found with SAML token. This requires synchronization of these cartridge roles with roles that come from IdP when the SAML Authentication scheme is selected. OSM DS, deployment and runtime are not aware of the source of these cartridge roles. Hence, they do not validate if the roles are synchronized. For more details, refer to "About User-Defined Cartridge Roles in IdP (Keycloak)".

Cartridge Roles in Basic Authentication Scheme
There is no IdP for this scheme. OSM continues to work with respect to these cartridge roles as the prior releases of OSM. These roles are sourced directly in the OSM cartridge.

About User-Defined Cartridge Roles in IdP (Keycloak)

Cartridge Roles can be created for cartridges in Design Studio. A Cartridge Designer can define any number of roles for a cartridge in any combination of the following permissions. Based on role permissions, users with that role can have these permissions.

Table 4-8 Cartridge Roles and Permissions

Field	Use
Search View	Enables users to access the order Query function.
Worklist Viewer	Enables users to access the Worklist function.
Exception Processing	Enables users to alter the flow of a process by applying exception statuses at any time throughout the process.
Task Assignment	Enables users to assign tasks to others.
Create Versioned Orders	Enables users to create orders for different versions of cartridges. If this permission is not granted, users can create orders only for the default version of the cartridge.
Order Priority Modification	Enables users to modify the priority of a task in an order.
Reference Number Modification	Enables users to modify the reference number of an order.
Online Reports	Enables users to view summarized reports on all orders and tasks on the system.

Additionally, OSM cartridge roles protect OSM functions and operations and control the access model for OSM tasks and orders:

- OMS Order lifecycle operations are defined in the Order State Policy.
- Order Data Changes
- Task Data Changes
- User's Worklist
- User's Notification List

For more information, see "Working with Roles " in *Design Studio Modeling OSM Processes Guide*.

About Cartridge Roles in SAML Client IdP (Keycloak)

These cartridge roles should be available in your IdP (Keycloak). Cartridge Roles can be assigned to a user in IdP. For example, you can create an OSM user-defined cartridge role in IdP (Keycloak) to access OSM user interfaces.

To create OSM user-defined role in Keycloak:

1. Select **Realm roles** from Keycloak realm.
2. In the **Realm roles** page, click **Create role**.
3. In the **Create role** page, enter **Role name** and **Description**.
4. Click **Save**.

If human user **osm-human-user1** is assigned with both OSM predefined role **OMS_client** and cartridge role **COM_FulfillmentOrder_OLM_Role** in IdP itself, these roles will be presented to OSM as SAML response for an authenticated user.

About Restrictive SAML/OIDC Clients

Some IdPs allow you to configure the client roles to restrict authentication. These capabilities are IdP-dependent and not tied to vanilla OIDC or SAML. Hence, you need to determine the configuration details based on your business requirements within your IdP.

By default, OSM does not support this capability via vanilla OIDC and SAML. In the absence of these capabilities, you may have to consider limiting the Authentication scope by partitioning users via separated security realms. For example, for human users, each security realm needs to get its users federated for their respective LDAP subtree.

Alternatives
Any other alternatives should be looked at in the context of the IdP. For example, in KeyCloak, you can install an extension to restrict authentication.

- You can use this extension when you need to restrict human users' access to OSM application instances, for example, Dev or Prod within a single security realm. For example, in a corporate SSO realm, you can use multiple SAML clients for various OSM instances such as Dev and Prod. Only users with restricted access can access the Prod client's resources, which could include OSM UIs.
- You can rely on this extension when you need to restrict OIDC service accounts, such as Client Credentials, from accessing APIs offered by OSM-Gateway microservice within a single security realm. This way, OSM TMF APIs, and the Fallout API for OSM instances like Dev and Prod in a security realm can be protected from unauthorized access. Unauthorized access could include an access token obtained from the Dev instance, but cannot be used against the Prod instance.

> **Note:**
>
> OSM is not responsible for the fine-grained configuration and whether this extension will meet your business requirements.

5
Security Considerations for Developers

This chapter provides information for developers about how to create secure applications for Oracle Communications Order and Service Management (OSM), and how to extend OSM without compromising security.

Securely Communicating with External Systems

Securely communicating with an external system requires managing the following securely:

- the credentials required to access the system
- the communication between OSM and the external system

You store credentials securely by using the OSM secure credential storage feature, described in "Secure Credential Management."

For reliability, Oracle recommends that communication with external systems be over Java Messaging Service (JMS). OSM creates a JMS module, **oms_jms_module**, for this purpose, which is secured from unauthorized access. Only members of the following Oracle WebLogic Server groups are allowed access to resources created in this module:

- OMS_client
- OMS_ws_api
- OMS_xml_api
- OSM_automation
- Cartridge_Management_WebService

See the information about installed components in *OSM System Administrator's Guide* for additional information about the OSM WebLogic Server groups. Oracle recommends that any other JMS modules with which OSM interacts be similarly configured. Ensure that the associated persistent store is properly secured, as described in "WebLogic Server Security."

Security Callback

OSM allows developers to add additional authorization and auditing to the default order data access model. For information on where and how this feature may be leveraged, see the information about using OSM security callback in *OSM Developer's Guide*.

Hiding Sensitive Data in the Web Client

You can ensure that the OSM web clients obscure OSM solution data by identifying that the data is secret at design time. A data node declared as secret in Oracle Communications Service Catalog and Design - Design Studio is rendered as a password field in the OSM web clients. For more information, see Design Studio Modeling OSM Processes Help.

Web Service Security

Access to the OSM Web Services is restricted to members of the **OMS_ws_api** WebLogic Server group. Access to specific operations, such as CreateOrder, CancelOrder, and UpdateOrder, are further restricted through OSM role and order life cycle policy permissions. For information about OSM roles and order life cycle policies, see Design Studio Modeling OSM Processes Help.

A

Secure Deployment Checklist

The following checklist provides guidelines to help you secure Oracle Communications Order and Service Management (OSM) and its components.

- Install only the components you need.
- Enforce strong password management.
- Restrict and control user privileges.
- Restrict network access by doing the following:
  - Use firewalls.
  - Never leave an unnecessary opening in a firewall.
  - Monitor who accesses your systems.
  - Encrypt network traffic.
  - Install the operating system in a secure location that is difficult for a hacker to access.
- Apply all security patches and workarounds.
- Encrypt sensitive information.
- Contact Oracle support if you discover a vulnerability in any Oracle product.