Contents

Preface .................................................................................................................................................................. v
    Audience ........................................................................................................................................................ v
    Scope of the Guide ........................................................................................................................................ v
    Related Resources ........................................................................................................................................ v
    Conventions ................................................................................................................................................ vi

1 Identity Concepts in Oracle Cloud
    Oracle Cloud Overview ........................................................................................................................... 1-1
    Oracle Cloud Services Overview ............................................................................................................ 1-1
    Identity Overview .................................................................................................................................... 1-2
    Identity Domain Overview .................................................................................................................... 1-2
    Role of the Identity Domain .................................................................................................................... 1-3

2 Identity Use Cases in Oracle Cloud
    Customer Buys an Oracle Cloud Service (Onboarding) ................................................................. 2-1
    Administrator Sets Up Users and Roles (Delegated Administration) ............................................. 2-1
    Customer Extends the Oracle Cloud Service (Role Lifecycle Management) ............................ 2-2
    Administrator Manages Customer Accounts and Passwords (Identity Federation) .................. 2-2
    Service Authenticates User Credentials (Identity Propagation) .................................................. 2-2

3 Identity Features in Oracle Cloud
    Vanity URLs .............................................................................................................................................. 3-1
    User and Role Management .................................................................................................................. 3-1
        Cloud Directory and User Schema ................................................................................................. 3-2
        User Onboarding ............................................................................................................................. 3-2
        Users and Roles ............................................................................................................................... 3-2
        User Self-Service .............................................................................................................................. 3-4
        Service-Specific Predefined Roles and Custom Roles ............................................................ 3-4
    Login Management ............................................................................................................................... 3-4
    Oracle Single Sign-On ........................................................................................................................... 3-5
        About SSO ..................................................................................................................................... 3-5
        User Login with SSO ...................................................................................................................... 3-7
SAML Identity Provider Requirements ................................................................. 3-7
User Synchronization Requirements for SSO Configuration ................................. 3-7
Automated and Manual Configuration ..................................................................... 3-8
Testing Your SSO Configuration .............................................................................. 3-9
Troubleshooting Your SSO Configuration ................................................................. 3-9
Authorization .............................................................................................................. 3-9
About OAuth ............................................................................................................... 3-10
OAuth Client Definitions ............................................................................................ 3-10
Supported OAuth Workflows ...................................................................................... 3-11

4 Introduction to OAuth in Oracle Cloud

Overview of OAuth in Oracle Cloud ........................................................................ 4-1
Use of OAuth in Oracle Cloud .................................................................................... 4-2
OAuth Client Types and Digital Signatures ............................................................... 4-3
Overview of OAuth Roles ......................................................................................... 4-3
OAuth Endpoints ....................................................................................................... 4-4
How Does OAuth Work in Oracle Cloud? ................................................................. 4-4
Supported OAuth Workflows .................................................................................... 4-5
Different Types of OAuth Tokens ............................................................................... 4-6
Supported Grant Types ............................................................................................. 4-7
Resource Owner Password Credentials Grant ........................................................ 4-7
Client Credentials Grant ............................................................................................ 4-7
Client Assertion ......................................................................................................... 4-8
User Assertion Grant ................................................................................................ 4-10
User Assertion .......................................................................................................... 4-10
Preface

Oracle® Cloud Understanding Identity Concepts explains the concepts of identity that are applicable in Oracle Cloud. This guide provides an overview of Oracle Cloud, cloud services, identity, identity domain, and the role of the identity domain. Five use cases are provided that are associated with Oracle Cloud identity. The guide also includes identity features that are supported by Oracle Cloud. These features include user and role management, login management, Oracle Single Sign-On, and authorization.

Topics:
• Audience
• Scope of the Guide
• Related Resources
• Conventions

Audience

Oracle® Cloud Understanding Identity Concepts is intended for Oracle Cloud service, identity domain, and account administrators (that is the customer buying Oracle Cloud services) who want to learn about identity concepts that are applicable in Oracle Cloud. These concepts are used to provision and secure access to Oracle Cloud services.

Scope of the Guide

The concepts covered in the guide include:
• Cloud identity concepts, including Oracle Cloud services, identity, and identity domain.
• Identity use cases associated with onboarding (initial setup), delegated administration, role lifecycle management, identity federation, and identity propagation of Oracle Cloud services.
• Identity features supported by Oracle Cloud, including user and role management, login management, Oracle Single Sign-On, and authorization.

Related Resources
For additional documentation related to your Oracle Cloud service, visit the Oracle Cloud website at:

http://cloud.oracle.com

Open the Support menu at the top of the page and select Documentation to access the Oracle Cloud Documentation home page. Search or browse the library for documentation specific to your application, infrastructure, or platform cloud service.

Conventions

The following text conventions are used in this guide:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>boldface</td>
<td>Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.</td>
</tr>
<tr>
<td>italic</td>
<td>Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.</td>
</tr>
<tr>
<td>monospace</td>
<td>Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.</td>
</tr>
</tbody>
</table>
Identity Concepts in Oracle Cloud

This chapter describes at a high level the interaction of Oracle Cloud and Oracle Cloud services with identity and identity domain features. You learn how the identity domain is used to manage features of Oracle Cloud.

Topics:

• Oracle Cloud Overview
• Oracle Cloud Services Overview
• Identity Overview
• Identity Domain Overview
• Role of the Identity Domain

Assumptions
Before reading this chapter, you should be familiar with the following terms:

• Service
• Account
• Service account

For more information about these terms, see Getting Started with Oracle Cloud.

Oracle Cloud Overview
Oracle Cloud is an enterprise cloud for business that offers self-service business applications delivered on an integrated development and deployment platform, with tools to extend and create new services rapidly. In addition, Oracle Cloud includes built-in identity management functionality.

Oracle Cloud Services Overview
Oracle Cloud offers many cloud services. A cloud service is a software offering in Oracle Cloud.

Classification of Different Cloud Services
Application services are classified in two categories:

• Software as a Service (SaaS): Provides a software licensing and delivery model in which software is licensed on a subscription basis and is centrally hosted.

• Data as a Service (DaaS): Provides data on demand to a user regardless of geographic or organizational separation of the provider and consumer.
Platform services are also classified into two categories:

- **Platform as a Service (PaaS):** Provides a platform that allows customers to develop, run, and manage applications without the complexity of building and maintaining the infrastructure typically associated with developing and deploying an application.

- **Infrastructure as a Service (IaaS):** Provides access to computing resources (that is, virtualized hardware and computing infrastructure) in Oracle Cloud across a public connection.

For a comprehensive list of the available Oracle Cloud services, go to [https://www.oracle.com/cloud](https://www.oracle.com/cloud) and from the Oracle Cloud menu, select that category of services that interests you. From the page that opens, you can find links to detailed information about each service.

Oracle Cloud enables secure integration between its different services, customer applications, and cloud services from other suppliers. These services can work together in an integrated cloud environment. For example, Oracle Sales Cloud service can be embedded within your own application running on Oracle Java Cloud Service-SaaS Extension. Customer Relationship Management can be extended with a custom application. You can tie together an Oracle Cloud service with functionality from other sites, such as Salesforce. You can use an Oracle Cloud service as the infrastructure for building your own applications.

**Identity Overview**

Identity is a core feature that Oracle customers rely on to provide secure access to Software as a Service (SaaS) and Platform as a Service (PaaS) services. The Oracle Cloud feature that brings users, services, and applications together in a secure manner is Shared Identity Management.

**Identity Domain Overview**

An identity domain is a construct for managing users and roles, integration standards, external identities, secure application integration through Oracle Single Sign-On (SSO) configuration and OAuth administration. OAuth is an authorization protocol — or in other words, a set of rules — that allows a third-party website or application to access a user's data without the user needing to share login credentials. In short, an identity domain controls the authentication and authorization of the users who can sign in to a service in Oracle Cloud, and what features they can access in relation to the service.

An Oracle Cloud service account is a unique customer account that can have multiple cloud services of different service types. For example, you could have three different cloud services, such as Java Cloud Service, Database Cloud Service, and Infrastructure as a Service (IaaS) as part of a single Oracle Cloud service account.

Every Oracle Cloud service belongs to an identity domain. Multiple services can be associated with a single identity domain to share user definitions and authentication. Users in an identity domain can be granted different levels of access to each service associated with the domain.

When a customer first signs up for an Oracle Cloud service account, the following tasks are performed:

- Oracle Cloud creates an identity domain specific to that customer.
- As part of the service account setup process, Oracle sends an identifier (the identity domain string) in the onboarding email.
The customer creates an account and specifies an account name.

Oracle creates the identity domain using the customer-specified account name.

When the customer log in to an Oracle Cloud service through the service account, the identity domain controls the user authentication (all users are authenticated before accessing an application in Oracle Cloud) and controls what features of the service they can access.

Note: The term Tenant is a synonym for identity domain. Oracle Cloud is a multitenant system, and each customer is a tenant in that system, much like the tenants of a building. So, an identity domain represents a slice of SIM, provisioned for a cloud tenant.

Role of the Identity Domain

The identity domain is a construct for managing certain features of Oracle Cloud. Many features of Oracle Cloud are managed within and between domains.

Users and Roles

Users and roles exist within the identity domain. Users with administrative roles manage local cloud identities and their rights. Access management within the domain depends upon users and roles.

You must manage two types of users in an identity domain:

- **Standard users**: These users add user accounts, import a batch of user accounts, assign roles to users, modify user accounts, reset passwords, and remove user accounts.

- **SFTP users**: These secure FTP (SFTP) users set passwords for their accounts, which are used to sign in to an SFTP server to perform FTP operations related to your Oracle Cloud service.

You must manage two types of roles in an identity domain:

- **Predefined roles**: View a list of all the predefined roles created by Oracle Cloud and link to a list of users assigned to the role that you select.

- **Custom roles**: View, add, and remove roles that you created for customized access to your Oracle Cloud services.

Upon activation of a service account, Oracle Cloud automatically assigns the following roles to a customer:

- **Account administrator**: The account administrator role is at the service instance level. It gives a user several responsibilities to manage one or more Oracle Cloud services. As account administrator, you’re responsible for managing an Oracle Cloud account through the cloud user Interface (UI) and you have business oversight responsibilities over service instances across one or more identity domains. You can nominate service administrators and identity domain administrators for services that you buy. You can view metrics for individual service instances.

Note: An account administrator doesn’t have to be a user in SIM.
• **Identity domain administrator:** As an identity domain administrator, you manage your own users and their roles. Your view is limited to the users and roles in the identity domains that you’ve been assigned to manage. You see all the roles at the domain and service levels. An identity domain administrator is a *super administrator* for an identity domain and for all the services within the domain. An identity domain administrator can delegate other identity domain administrators as well as manage roles assigned to service administrators. As an identity administrator you perform administrative responsibilities for the whole identity domain.

• **Service administrator:** As a service administrator, your view is limited to the users and roles for the services that you’re assigned to manage. You see the roles only at the service level. In addition, you’re limited to mostly search, view, and read-only functions. For example, you can’t create roles or user accounts, but you can assign an existing role to an existing user account. A service administrator is a *super administrator* for a given service instance. As a service administrator, you can assign more service administrators to roles as well as manage other roles associated with the service. However, you can’t create users or roles.

• **Customer service representative administrator:** As a customer service representatives administrator, you have administrative responsibilities for operations that perform in deployed cloud services. You’re the equivalent of an identity domain administrator for all of a customer’s identity domain.

**Relationship Between Identity Domains, and Administrative and Delegated Roles**

The following architectural illustration shows the relationship between identity domains, and administrative and delegated roles.

In this illustration, the account administrator (Acme Buyer) is a user defined in the Oracle.com SSO, and is identified within the Oracle Store — global single instance. This user represents a buyer for a customer, and is responsible for managing an Oracle Public Cloud account through the cloud UI. In addition, the account administrator has business oversight responsibilities over service instances across one or more identity domains.
For this example, Acme Buyer is responsible for the Oracle Public Cloud identity domain. Within this identity domain, the Acme Buyer nominated both an identity domain administrator (ID domain administrator) and service administrators (Identity Admin, JCS-SX Admin, Schema Cloud Service Admin, and BI Cloud Service Admin) to manage services that the account administrator buys.

These services include:

- Role, user, and security management capabilities, which are associated with identity management (IDM) roles.
- Java administrator capabilities, which correspond with Oracle Java Cloud Service roles.
- User, developer, and administrator capabilities, which are linked to roles associated with schema service roles.
- Hierarchical role operations associated with Oracle Business Intelligence (BI) Cloud Service.

Because the ID domain administrator is the super administrator both for the OPC-IDM identity domain and for the services within the domain, this user can delegate service administrators (for this example, Identity Admin, JCS-SX Admin, Schema Cloud Service Admin, and BI Cloud Service Admin) and manage associated roles (including IDM, Oracle Java Cloud Service, Schema Cloud Service, and Oracle Business Intelligence (BI) Cloud Service roles).

The customer service representative administrator (CSR Admin) has administrative responsibilities for deployed cloud services, and has identity domain administrator privileges for all identity domains. For this example, the CSR Admin is a member of the OPC operations and support staff role. Because of this membership, the user has administrative privileges over three identity domains: OPC-IDM, CSR, and Acme2.

**Self-Service Profiles**

Within the identity domain, you can view your user information, change your password, and change your password challenge questions.

**Contacts**

Within the identity domain, you can add, modify, or delete contacts who would receive service notification email about planned maintenance, service outages, and so on. The **Contacts** tab is available only if the user is a service administrator for any services. Identity domain administrators (who aren’t service administrators) won’t be able to perform this functionality.

**Client and API Communication Using OAuth 2.0**

Oracle Cloud uses the open standard OAuth 2.0 authorization protocol to communicate with internal and external services.

The Oracle Cloud identity infrastructure provides an OAuth token service, which grants secure access to the Representational State Transfer (REST) endpoints of cloud services by other cloud services and by other clients. For example, a client application deployed to Oracle Java Cloud Service—SaaS Extension might need to access the APIs of a SaaS application, such as a human capital management (HCM) application. OAuth provides a way to allow access by known clients, while protecting the APIs against unauthorized access.
User Integration Using Single Sign-On with SAML 2.0

Oracle Cloud uses the open standard Security Markup Language (SAML), version 2.0 to implement SSO between internal and external users. The OAuth service in Oracle Cloud provides service-to-service authorization, and service-to-service identity propagation.

SSO lets you provide access to one domain to users who have been authenticated to a different domain, or to an on-premises identity store. For example, you might want your users to log in to Oracle Cloud by using a local directory, such as Microsoft Active Directory Federation Services (ADFS) or Oracle Unified Directory. The SSO service is built upon the federation capability of the identity infrastructure. SSO uses the industry standard SAML, version 2.0 to exchange information between an identity provider and a service provider.

External Identities and Single Sign-On

An identity domain may have a split user population, where user identities are:

- **Non-federated:** Stored on one local identity management system. These user identities are associated only with Oracle Cloud. As a result, they’re known only to Oracle Cloud.

- **Federated:** Stored across multiple distinct identity management systems, and are coming from different domains.

Federated users receive their identities from their administrator. These identities may be known to Oracle Cloud through a Lightweight Directory Access Protocol (LDAP) export to SIM. However, these users can’t log in to Oracle Cloud directly. Instead, they’re redirected to their federated site for login.

Then, SSO gets a SAML assertion from the identity provider. The SAML assertion contains user information from the identity provider. SSO parses the SAML assertion and asserts the identity to the identity domain.
Identity Use Cases in Oracle Cloud

This chapter describes five use cases associated with identity features in Oracle Cloud.

Topics:

- Customer Buys an Oracle Cloud Service (Onboarding)
- Administrator Sets up Users and Roles (Delegated Administration)
- Customer Extends the Oracle Cloud Service (Role Lifecycle Management)
- Administrator Manages Customer Accounts and Passwords (Identity Federation)
- Service Authenticates User Credentials (Identity Propagation)

Customer Buys an Oracle Cloud Service (Onboarding)

A customer (Acme) buys its first Oracle Cloud service. The customer's buyer (Bob) procures an Oracle Planning and Budgeting Cloud Service instance from Oracle Cloud. Because of this purchase, Bob becomes the account administrator.

As the account administrator, Bob nominates Charles to two roles:

- Identity domain administrator for Acme
- Service administrator for the Oracle Planning and Budgeting Cloud Service instance

An identity domain for Acme is created in Shared Identity Management (SIM). Also, the Oracle Planning and Budgeting Cloud Service instance is provisioned and associated with Acme.

Oracle creates a user account for Charles, and allocates the identity domain administrator and service administrator roles to the account. Charles receives a Welcome to Oracle Cloud notification and a temporary password. He uses this password to log in to the Acme identity domain, set his security questions, and reset his password. Now, he is ready to perform administrative responsibilities as both an identity domain administrator for Acme and service administrator for the Oracle Planning and Budgeting Cloud Service instance.

Administrator Sets Up Users and Roles (Delegated Administration)

As the identity domain administrator for Acme, Charles sets up users and roles.

First, he sets up Acme's user population in the Acme identity domain. These users are provisioned into the domain, and are notified, using email, about their new accounts for the domain. They can use the Oracle Identity Console to change their passwords and answer security questions related to self-service tasks they are to perform.
Also, because Charles is an identity domain administrator, he can use the Oracle Identity Console to delegate the service administrator role to other users.

Customer Extends the Oracle Cloud Service (Role Lifecycle Management)

Bob, the account administrator for Acme, procures an Oracle Java Cloud Service instance from Oracle Cloud to interface with and extend Acme’s Oracle Planning and Budgeting Cloud Service instance. As a result of this purchase, the new Oracle Java Cloud Service instance is provisioned and associated with the Acme identity domain.

Bob then nominates John to be the service administrator for the Oracle Java Cloud Service instance. Oracle creates a user account for John, and allocates the service administrator role to the account.

As service administrator, John can deploy a Java application associated with the Oracle Java Cloud Service instance. He requests that Charles, the identity domain administrator for the domain, creates roles relevant to the Java application. Charles uses the Oracle Identity Console to construct these additional roles in SIM, and to notify John of the role creation. John then assigns these Java-application-related roles to Oracle Planning and Budgeting Cloud Service users in Acme. These users can now communicate between the Oracle Planning and Budgeting Cloud Service instance and the Java application, using Oracle Single Sign-On.

Administrator Manages Customer Accounts and Passwords (Identity Federation)

With the purchase of Oracle Cloud services, Acme now has two environments: an on-premises environment and an Oracle Cloud environment. Because of this, Acme users have to manage accounts and passwords in two places: in their enterprise (on-premises) and within Oracle Cloud. In addition, whenever these users have problems associated with their accounts being locked or their passwords expiring in Oracle Cloud, they contact Charles, the identity domain administrator for Acme.

To rectify this situation, Charles implements identity federation between Acme’s on-premises and Oracle Cloud environments. Identity federation is a capability of Oracle Cloud services that allows managed cloud-based web applications to participate in cross-domain federated SSO and to create circles of trust that span across multiple sites. This lets users log in once and access integrated applications across the managed cloud and other federated sites, without having to log in to each application individually.

By implementing identity federation for his company, Charles gives Acme’s users access to Oracle Cloud based on their corporate SSO credentials. Now, these users have a single password to remember for both their enterprise and their Oracle Cloud applications.

Service Authenticates User Credentials (Identity Propagation)

During identity propagation, an Oracle Cloud service, such as the Oracle Mobile Cloud Service, challenges a user and authenticates the user's credentials.

Bob, the account administrator, procures Oracle Mobile Cloud Service from Oracle Cloud. This service calls another service (for example, Oracle Fusion Applications) by issuing a Representational State Transfer (REST) API call on behalf of a user. For this example, Oracle Mobile Cloud Service doesn’t know the user's credentials. However, Charles, the identity domain administrator, has identified Oracle Mobile Cloud Service as trusted. A trusted client is registered by the cloud infrastructure or by an identity domain administrator as being trusted to assert a user identity on behalf of
the user. As a result, Oracle Mobile Cloud Service can assert a user's identity for the user.

After the user's identity is validated, a token is established. This token acts as a pointer to a resource and provides information needed to gain access to the resource. A token is used in place of a password to prove that the user is who he or she claims to be. The asserted identity is then passed into a back-end system (for example, the OAuth resource server).

The OAuth token service acts as the central trust manager between the client (in this case, Oracle Mobile Cloud Service) and the back-end system (for example, the OAuth resource server). Because Oracle Mobile Cloud Service is designated as trusted, it propagates the user's identity by generating a signed JavaScript Object Notation (JSON) web token assertion and sends the assertion as part of a request to obtain an OAuth access token.

The OAuth authentication server verifies that the client is trusted, maps the user assertion's creator to the OAuth client, and validates the user assertion using the client's public key. A public key is a value provided by some designated authority as an encryption key that, combined with a private key derived from the public key, can be used to effectively encrypt messages and digital signatures. The result is service-to-service identity propagation. After validation, the OAuth service provides the OAuth client with the access token, which the client then presents to the OAuth resource server.

Because the identity has already been established, the back-end system (OAuth resource server) trusts that it's a valid user identity, and can use it as required.

---

**Note:**

For more information about trusted clients, see [OAuth Client Types and Digital Signatures](#).
Identity Features in Oracle Cloud

Oracle Cloud supports various identity features. These features are vanity URLs, user and role management, login management, Oracle single sign-on (SSO), and authorization.

Topics:
- Vanity URLs
- User and Role Management
- Login Management
- Oracle Single Sign-On
- Authorization

Vanity URLs

A vanity URL is a unique, customized web address that’s branded for marketing purposes and exists to help users remember and find a specific page of your web site. Customers may use their own vanity URL prefix instead of the usual Oracle Cloud URL, thereby customizing the user login experience.

Your company representative can sign up for a vanity URL for one or more supported cloud services. Limitations of vanity URLs are:

- You must sign up at the time the Oracle Cloud service instance is allocated.
- You can have only one vanity URL per domain.

The features of vanity URLs are:

- Your onboarding email from Oracle Cloud contains the vanity URL prefix.
- You can tailor the user login experience so that the vanity URL resolves differently for some users than for others. For example, the URL can resolve to a different URL for local users than for users configured to be authenticated by an external identity provider.

User and Role Management

The ability to manage users and roles is a basic identity feature of Oracle Cloud.

Topics:
- Cloud Directory and User Schema
- User Onboarding
• Users and Roles
• User Self-Service
• Service-Specific Predefined Roles and Custom Roles

Cloud Directory and User Schema


Like most identity management systems, Oracle Cloud includes an identity store. The format and rules for defining users, groups and roles are determined by specific LDAP directory schemas.

A directory schema:
• Contains rules about the kinds of objects that you can store in the directory
• Contains rules for how directory servers and clients treat information during operations (such as a search)
• Helps to maintain the integrity and quality of the data stored in the directory
• Reduces duplication of data
• Provides a predictable way for directory-enabled applications to access and modify directory objects

User Onboarding

When you request an Oracle Cloud trial subscription account, Oracle processes your request and then sends an email with setup information. After receiving the email, you activate your account and go to the associated cloud service.

You start your trial subscription from the Oracle Cloud website. To activate your trial subscription to a service go to the My Account application to activate your service.

You can activate your service in two ways: from the email or from My Account.

When you request for a trial subscription, the system assigns you to the following roles automatically:
• Account administrator for the Oracle Cloud service
• Identity domain administrator (for the domain)
• Service administrator (for the service)

Note:
For more information about the account administrator, identity domain administrator, and service administrator roles, see Role of the Identity Domain.

Users and Roles

Users and roles exist within an identity domain. Users with administrative roles manage local cloud service identities and their rights.
Users in the Domain
An identity domain administrator manages users and roles. As an identity domain administrator, you can add users individually, or in a batch, from a comma-separated values (CSV) list, using the Users tab in the My Services page.

Roles in the Domain
Roles control access to applications, resources, and services. Users with administrative roles manage local identities and their rights, including self-service capabilities, within the domain.

Predefined Roles
Some roles already exist in the identity domain when your company purchases an Oracle Cloud service. The predefined roles are:

- **Identity domain administrator**: Performs identity administration for the whole domain and all of the services within it.
- **Service administrator**: Manages a specific Oracle Cloud service instance.
- **Account administrator**: Manages an Oracle Cloud account through the cloud UI.
- **Service-specific non-administrative roles**: Performs service-specific responsibilities. The responsibilities for this role depend on the type of Oracle Cloud service associated with the role. For example, Oracle Mobile Cloud Service has different service-specific non-administrative roles than Oracle Java Cloud Service.

Note:
For more information about the account administrator, service administrator, and identity domain administrator roles, see Role of the Identity Domain.

Delegation of Administrator Roles
Administrative roles are created by delegation:

- The buyer who purchases Oracle Cloud services on behalf of a company assigns the account administrator role to a user.
- The account administrator is responsible for managing the Oracle Cloud account through the cloud UI. The account administrator can nominate identity domain administrators and service administrators.
- Identity domain administrators can delegate other identity domain administrators and service administrators. Identity domain administrators manage service administrator users and roles.
- Service administrators can assign other service administrators and manage other roles within the service, but they can’t create users or roles.
- Customer service representative administrators have the equivalent of identity domain administrator privileges for all identity domains.
User Self-Service

Select the My Profile tab in the My Services application to enable non-administrative users (end users) to view their user information, change their passwords, and change their password challenge questions.

When non-administrative users log in to the My Services application, they see the My Home application that displays the consoles and the Identity Self-Service pages. Unlike administrators, they can’t view all the pages in the My Services application.

On first login to My Services, the system prompts users to change their temporary passwords and set their password challenge questions. They can change their password and password challenge questions any time they’re logged in to My Services. Users who forget their passwords can reset them, provided they remember their identity domain, user name, and answers to their challenge questions.

Service-Specific Predefined Roles and Custom Roles

Oracle Cloud supports service-specific predefined roles and custom roles.

Service-Specific Administrator Roles

Oracle Cloud automatically populates the My Services application with several administrative roles. The roles created depend on the type of Oracle Cloud service being provisioned. Oracle Cloud also lets administrators provision service-specific administrative roles. Oracle Cloud populates the My Services application with three administrative roles: identity domain administrator, service administrator, and account administrator. For more information about these roles, see Role of the Identity Domain.

Service-Specific Non-Administrative Roles

Oracle Cloud also populates the My Services application with several non-administrative roles. The roles created depend on the type of Oracle Cloud service being provisioned. Oracle Cloud also lets administrators provision service-specific non-administrative roles to users.

Custom Roles

Custom roles are used by application developers to secure applications. For example, with Java EE applications deployed to Oracle Java Cloud Service, the application roles specified in application deployment descriptors are mapped to the enterprise roles created in the identity management system. The mapping is based on matching fully qualified role names.

Only identity domain administrators can create and delete custom roles, and only in the identity domains that they’ve been assigned to administer.

Login Management

An identity domain may have a split user population, where user identities are either non-federated or federated.

Some of the user identities are:

- Non-federated: Stored on one local identity management system. These user identities are associated only with Oracle Cloud. As a result, they’re known only to Oracle Cloud.
• **Federated**: Stored across multiple distinct identity management systems, and are coming from different domains.

Oracle Cloud handles this situation appropriately, using the login page.

• If a user has a non-federated identity, the user can log in through the local Oracle Cloud account.

• Federated users receive their identities from their administrator. These identities may be known to Oracle Cloud through an LDAP export to Shared Identity Management (SIM). However, these users can’t log in to Oracle Cloud directly. Instead, they’re redirected to their federated site for login.

• At that time, the SSO service uses Security Assertion Markup Language (SAML) to exchange information between the identity provider and service provider. SAML is also required because federated user passwords aren’t known to Oracle Cloud.

**Oracle Single Sign-On**

SSO lets users log in to one domain and access another domain without logging in again.

**Topics:**

• About SSO
• User Login with SSO
• SAML Identity Provider Requirements
• User Synchronization Requirements for SSO Configuration
• Automated and Manual Configuration
• Testing Your SSO Configuration
• Troubleshooting Your SSO Configuration

**About SSO**

Oracle Cloud uses the SAML 2.0 protocol to integrate internal and external users. Oracle Cloud doesn’t support all features of this standard.

Oracle Cloud uses the SAML standard to enable secure, cross-domain communication between Oracle Cloud and other SAML-enabled Oracle systems, as well as a selected number of non-Oracle identity management systems located on-premises or in a different cloud.

• In cases where a user is accessing Oracle Cloud resources from the Oracle Cloud, to leverage SSO only the Oracle Access Manager component WebGate is required because all the resources the user wants to access are in the same domain.

• Federated web SSO leveraging SAML is required when users are attempting to access Oracle Cloud resources from a different Internet domain using Oracle’s or a third-party SAML-compliant identity management system such as Oracle Access Manager or Microsoft Active Directory Federation Services.

Enabling SSO for your Oracle Cloud users provides these advantages:
• It eliminates the need for two different passwords for two different accounts (on premises and cloud).

• Because there’s only a single password, users are more likely to use a strong password and keep it confidential.

• It reduces failed login attempts and may reduce the frequency of password resets, thereby improving the user’s experience.

• Enabling SSO centralizes authentication and authorization.

Oracle Cloud SSO also includes a failback mechanism. If SSO becomes inoperable, then administrators can log in to their identity domains directly, bypassing the SSO identity provider. This is necessary to resolve problems.

SAML bindings define how the SAML protocols map to the type of transport used. There are five main bindings: Simple Object Access Protocol (SOAP), Reverse SOAP (mainly used for rich clients, where most of the data processing occurs on the client side), HTTP redirect, HTTP POST, and HTTP artifact:

• The HTTP redirect binding defines how SAML requests and responses are transported using HTTP redirect messages (HTTP 302 status code responses).

• The HTTP POST binding defines how SAML protocol messages can be transported with the base64-encoded content of a form control within an HTML form.

Base64 is an encoding and decoding technique used to convert binary data to an ASCII string.

• The HTTP artifact binding defines how a reference or artifact to a SAML request or response is transported over HTTP. The artifact (a very small representation of a full-blown SAML assertion) can be embedded in a URL as a query string parameter, or it can be placed in a hidden form control.

In SIM, the HTTP redirect binding is used to deliver the SAML authentication request message to the identity provider and the HTTP POST binding is used to return the SAML response message containing the SAML assertion to the service provider (for example, SIM in this case).

The HTTP artifact binding is rarely used. Although, it’s much more secure than the HTTP POST binding. The HTTP artifact binding has a bigger overhead, because for example, a back channel communication has to take place between the identity provider and service provider, which requires additional pieces of infrastructure. This is one of the reasons why companies tend to avoid using SAML artifacts.

SAML also defines profiles. SAML profiles define how SAML assertions, SAML protocols, and SAML bindings are combined in particular use cases. For example, SIM uses the web browser SSO profile, which is an SSO mechanism for web browsers using the Authentication Request Protocol in combination with the HTTP redirect, POST, and Artifact bindings as described.

The SAML implementation in Oracle Cloud has the following limitations:

• It supports only POST profile.

• It supports only the email address or the user ID as the mapping attribute.

• It supports only SAML 2.0. Identity providers that use either SAML 1.1 or SAML 1.0 are not compatible.
User Login with SSO

When SSO is enabled, the identity provider performs authentication for Oracle Cloud services.

When you attempt to log in to an Oracle Cloud service by selecting Sign in with your company ID on the login page, you’re redirected to the identity provider, which performs the authentication. Here’s the sequence of events:

1. In the browser, you try to reach an Oracle Cloud service. On the login page, the user selects Sign in with your company ID.
2. The service provider, Oracle Cloud, generates a SAML authentication request, encodes it, and embeds it into a URL, along with state information and the encoded URL of Oracle Cloud.
3. Oracle Cloud sends a redirect message to your browser, using the SSO URL. This URL redirects your browser to the identity provider.
4. The identity provider decodes and extracts the information from the URL, and then prompts you for credentials and authenticates you.
5. The identity provider generates a SAML response containing your user name, digitally signed with the identity provider’s public and private keys.
6. The identity provider encodes the SAML response and state information, and returns it to the browser. The browser then sends this information to Oracle Cloud.
7. Oracle Cloud verifies the SAML response using the identity provider’s key, and then redirects the browser to your destination URL.
8. In the browser, you’re logged in to Oracle Cloud and redirected to the Oracle Cloud service that you logged in to in Step 1.

SAML Identity Provider Requirements

To work with the service provider, which in this case is Oracle Cloud, an identity provider must support SAML 2.0.

Oracle Cloud supports any SAML 2.0–compliant identity provider. The following identity providers have been certified with Oracle Cloud:

- Microsoft Active Directory Federation Services 2.0, 2.1, and 3.0
- Oracle Identity Federation 11gR1 and Oracle Access Manager and Identity Federation 11gR2
- Shibboleth 2.4.0
- Oracle SaaS applications

User Synchronization Requirements for SSO Configuration

Before configuring SSO, you must ensure that the user population has been synchronized between the identity provider LDAP directory and the service provider directory, with the attribute used to identify the user being the same in both directories for each user.
You can synchronize users manually or import a batch of users into Oracle Cloud from an external directory in the **My Services** application from the **Users** tab. To synchronize users manually, create users one at a time. To import a batch of users, create a list of comma-separated values that contain the first name, last name, email address, and login ID for each user.

The attribute used by SAML to identify the user must be the same in both the identity provider LDAP directory and the service provider directory.

- If you want to use the email address as that attribute, then each user must have a unique email address. For example, if the LDAP directory is Oracle Internet Directory, `mail` would be such an attribute.
- If you want to use the user identifier as that attribute, then each user must have a unique user ID. For example, if the LDAP directory is Oracle Internet Directory, `uid` would be such an attribute.

### Automated and Manual Configuration

**You can select automated or manual when configuring SSO, and when configuring the service provider and the identity provider.**

When you configure the service provider, you can select whether to import identity provider metadata or to enter identity provider metadata manually. Your choice depends on whether your identity provider can export metadata.

**Configuring the Service Provider Automatically**

If your identity provider can export metadata, then you can configure the service provider automatically:

- Import the identity provider metadata into Oracle Cloud.
- Complete the SSO Protocol, User Identifier, and contained in fields.
- The user identifier is the Oracle LDAP directory attribute that’s used to map the user information contained in the incoming SSO SAML assertion to an Oracle Cloud user. It’s either the user’s email address or the user ID.
- If the user identifier is the user’s email address, then the contained in field must show `NameID`.
- If the user identifier is the user ID, then the contained in field must show `SAML Attribute`; you must specify the name of the SAML attribute to use.

**Configuring the Service Provider Manually**

If your identity provider can’t export metadata, then you must enter metadata information manually:

- Provide the issuer ID and the SSO service URL of the service provider.
- Indicate whether the global logout should be enabled.
- Download your identity provider’s signing certificate and encryption certificate.

**Configuring the Identity Provider**

When you obtain data from the service provider to configure the identity provider, you have two choices, depending on the identity provider’s capabilities:
1. If your identity provider can import metadata, you can export the metadata from the service provider to import into the identity provider.

2. If your identity provider can’t import metadata, then you must copy and paste the provider ID and the SAML assertion consumer URL of the Oracle Cloud service into the identity provider configuration. You must also export the service provider signing and encryption certificates into the identity provider configuration.

**Testing Your SSO Configuration**

The SSO Configuration page lets you configure and test your SSO configuration. Before you enable SSO, test your SSO configuration. A successful test ensures that your company’s users can use their company credentials to log in to all applications, including Oracle Cloud applications.

After you configure the service and identity providers, you must test the configuration, as follows:

1. Click the **Test** button. This opens the Initiate Federation SSO page.

2. Clicking the button triggers a Federation SSO flow. You’re redirected to the identity provider and challenged for authentication.

3. Log in as an administrator. After the Federation SSO is performed, the result is displayed in the Test SSO page. If testing your SSO configuration fails, then troubleshoot the configuration.

**Troubleshooting Your SSO Configuration**

If testing your SSO configuration was not successful, follow the troubleshooting guidelines.

To troubleshoot your SSO configuration:

1. Review any changes made on the identity provider and Oracle Cloud before the SSO flow failed.

2. To capture an HTTP trace of the SSO attempt, use a web debugging and tracing tool such as Fiddler Web Debugging Tool.

3. To determine the point where the SSO flow terminated and which identity-related components are involved (for example, the identity provider, service provider, web tier, gateways, proxies, and firewalls), review the SSO Test results page.

4. To identify exceptions, review the SAML protocol messages and component logs.

5. Go to MyOracle Support to review known problems and find out if your problem has been seen before.

6. If you have performed troubleshooting Step 1 through Step 5, and you’re confident that the problem is because of Oracle Cloud, contact Oracle. Be ready to provide all your information, including the Fiddler trace, identity provider metadata, and identity provider logs.

**Authorization**

OAuth 2.0 is an authorization framework that enables an application or service to obtain limited access to a protected HTTP resource.
OAuth is used to secure access to services that are exposed by Oracle Cloud. This is done by providing access tokens to clients with the (sometimes implicit) approval of the owner of the resource.

Topics:
- About OAuth
- OAuth Client Definitions
- OAuth Flows

About OAuth

Oracle Cloud uses the open standard OAuth 2.0 to integrate services. Oracle Cloud doesn’t provide all the features of this standard.

The OAuth token service provided by the Oracle Cloud identity infrastructure provides secure access to the Representational State Transfer (REST) endpoints of cloud services by other cloud services and user applications.

OAuth 2.0 provides the following benefits:

- It increases security by eliminating the use of passwords in service-to-service REST interactions.
- It reduces the lifecycle costs by centralizing trust management between clients and servers. OAuth reduces the number of configuration steps to secure service-to-service communication.

The OAuth 2.0 implementation in Oracle Cloud has the following limitations:

- It supports only two-legged OAuth, which means that no user is present to give consent. Instead, clients hold onto credentials from the user.
- It supports only the user client, resource owner, and assertions workflow.
- Because of the lack of three-legged OAuth, clients using the resource owner credentials workflow can’t leverage browser-based SSO protocols. Instead, clients route a user’s credentials to the OAuth token service URL to obtain an access token on behalf of the user.
- Obtaining tokens from third-party OAuth token services isn’t supported.

OAuth Client Definitions

The OAuth client is the application requesting access to the resources stored on the resource server.

A client application uses OAuth to access a protected resource on behalf of a user (typically, the resource owner).

Client Identifier

A client identifier is a unique string representing the registration information provided by the client application. Before a client application can request access to resources on a resource server, the client application must first register with the OAuth token service associated with the resource server. The registration is typically a one-time task. After the client application is registered, the registration remains valid, unless the client application registration is revoked.
**Client Password**

Clients in possession of a client password may use the HTTP basic authentication scheme to authenticate with the authorization server. During client registration, the client application is assigned a client ID and a client password by the authorization server. The client ID and password are unique to the client application on that authorization server. If a client application registers with multiple authorization servers, each authorization server provides a unique client ID and password to the client application.

**Client Redirect URI**

After the client application is authorized successfully, the client application is redirected to the redirect uniform resource identifier (URI). During client registration, the client also registers a redirect URI. This redirect URI is used when a resource owner grants authorization to the client application. When a resource owner has successfully authorized the client application using the OAuth token service, the resource owner is redirected back to the client application, using the redirect URI.

**Supported OAuth Workflows**

The OAuth authorization workflow includes two different usage scenarios, two-legged OAuth workflow and three-legged OAuth workflow.

Oracle Cloud supports only the two-legged OAuth workflow.

**Two-Legged OAuth**

The two-legged OAuth workflow includes a client and a resource server. In OAuth two-legged authorization, consent from the resource owner is either assumed or not required. An authorization grant is a credential representing the resource owner’s authorization (to access its protected resources) used by the client to obtain an access token. The authorization grant is given to a client application by the resource owner, in cooperation with the OAuth token service associated with the resource server.

The following grant types are supported in Oracle Cloud:

- **Resource owner password credentials grant**: The resource owner’s password credentials (that is a user name and password) can be used directly as an authorization grant to obtain an access token.

- **Client credentials grant**: The client credentials (or other forms of client authentication) can be used as an authorization grant when the authorization scope is limited to the protected resources under the control of the client application, or to protected resources registered with the authorization server. Client credentials are used as an authorization grant typically when the client application is acting on its own behalf (the client is also the resource owner) or is requesting access to protected resources based on an authorization registered with the authorization server.

- **User assertion grant**: A user assertion is a user token that contains identity and security information about the user. A user assertion can be used as an authorization grant to authenticate the user. This user token can be used instead of a user name and password.

**Working of Two-Legged OAuth Flow**

The authorization server returns a request token to the OAuth client, which the client then uses to request an access token. Because the request token is preauthorized, the authorization server’s token service returns an access token to the client. In the case of
the two-legged OAuth workflow, the OAuth client becomes the resource owner. In OAuth two-legged authorization, the OAuth Client is preapproved to access resources. This arrangement fits a service-to-service model, especially, when the requesting service and the resource server are in a close partnership and the resource owner’s approval is either assumed or not required.
Introduction to OAuth in Oracle Cloud

OAuth 2.0 is an authorization framework that enables an application to obtain limited access to a protected HTTP resource. Using OAuth, the applications are called clients; they access protected resources by presenting an access token to the HTTP resource.

OAuth is used to secure access to services that are exposed by Oracle Cloud. This is done by supplying access tokens to clients with the (sometimes implicit) approval of the owner of the resource.

Topics:

• Overview of OAuth in Oracle Cloud
• Use of OAuth in Oracle Cloud
• OAuth Client Types and Digital Signatures
• Overview of OAuth Roles
• OAuth Endpoints
• How Does OAuth Work in Oracle Cloud?
• Supported OAuth Workflows
• Different Types of OAuth Tokens
• Supported Grant Types
• Resource Owner Password Credentials Grant
• Client Credentials Grant
• Client Assertion
• User Assertion Grant
• User Assertion

Overview of OAuth in Oracle Cloud

OAuth provides secure delegated access to services in Oracle Cloud.

Use Case:

You’re an administrator for your company. Your company purchased two Oracle Cloud services, which include Oracle Java Cloud Service-SaaS Extension (a Platform-as-a-Service cloud service) and Oracle RightNow Cloud Service (a Software-as-a-Service cloud service). One of your responsibilities is to ensure that these services are protected.
In addition to protecting the recently purchased Oracle Cloud services, you’re responsible for provisioning these services to your company’s employees.

Your company not only has Oracle Cloud services, but also on-premises applications and cloud services from other vendors. You’re responsible for ensuring that communication between these services and applications is done in a secure fashion.

Solution:

The previous illustration shows different services in Oracle Cloud such as Oracle Java Cloud Service-SaaS Extension (JCS), Oracle Documents Cloud Service (DCS), Oracle Process Cloud Service (PCS), Oracle Fusion Applications (FA), and two SaaS applications like Oracle RightNow Cloud Service and Oracle Taleo Cloud Service. These services in Oracle Cloud work together in an integrated cloud environment. The identity information is stored in the Cloud Identity Store. As the administrator, you’re responsible for providing secure access to these cloud services. The feature that brings users, services, and applications in the cloud together in a secure manner is Shared Identity Management (SIM). Specifically, use OAuth as an authorizing mechanism to provide secure access to all your services in Oracle Cloud.

Use of OAuth in Oracle Cloud

The OAuth standard is used to implement authorization from one cloud service to another. This standard is used to secure commercial services like for example, Oracle Sales Cloud (a Software-as-a-Service), Oracle Java Cloud Service (a Platform-as-a-Service) and Oracle Storage Cloud Service (an Infrastructure-as-a-Service) in Oracle Cloud.

Your company has purchased a few Oracle Cloud services. In addition, it has its own on-premises applications and also some mobile applications. These services are exposed through APIs. OAuth increases security by eliminating the use of passwords, when one of the mentioned services talks to the other service. Instead, OAuth provides an access token for this communication, which is valid for a limited time. An access token contains the security credentials for a login session and identifies the user and the user’s groups. For example, an on-premises application can obtain an access token (which is time-limited) to access an Oracle Cloud service. The on-premises application uses the access token to pass the user information securely to the Oracle Cloud service.

Representational State Transfer (REST) is a software architecture style consisting of guidelines and best practices for creating scalable web services. REST can also be used to expose cloud services. You can use the OAuth service to secure REST API calls in Oracle Cloud.

Benefits of Using OAuth

OAuth brings the following benefits to Oracle Cloud services:

- There is no need to store the resource owner’s credentials (user name and password) for future use, because these credentials are stored in the access token.
• The duration or access to any resource is restricted. The resource owner also can
grant or deny access to the resource.

• The OAuth service secures REST interactions between services, thus eliminating
the use of passwords in service-to-service communications.

• The OAuth service reduces lifecycle costs by centralizing trust management
between clients and servers. OAuth reduces the number of configuration steps to
secure service-to-service communication in environments with many services.

OAuth Client Types and Digital Signatures

Different OAuth client types provide different credentials to the token service.

OAuth clients are classified by their ability to authenticate securely with the
authorization server. A confidential OAuth client is an application that’s capable of
keeping a client password confidential. On the other hand, a public client is an
application that isn’t capable of keeping a client password confidential.

Trusted and Untrusted OAuth Clients

All OAuth clients registered in Oracle Cloud are confidential by default. However,
they can be trusted or untrusted. A trusted client is an OAuth client that is registered
by the Oracle Cloud infrastructure or by the identity domain administrator as being
trusted to assert a user identity on behalf of the user. All trusted clients are
confidential clients. Untrusted clients are mostly external applications and can be
created by the customer. An untrusted client can’t assert a user identity on the user’s
behalf.

Digital Signatures

You can associate a certificate with an OAuth client. This is mandatory for trusted
clients and optional for untrusted clients. For a trusted client, you must create a key
pair and a public key certificate signed by the Oracle Cloud certificate authority, and
then import the certificate as part of the client registration process.

Overview of OAuth Roles

OAuth 2.0 defines four roles for users and applications.

The OAuth roles are resource owner, resource server, client application, and
authorization server.

Resource Owner

This entity can grant access to a protected resource or a service. The resource owner is
a person (like an end user), an application that owns the service, or a security policy.
The resource owner is depicted in the illustration that follows as a person, which is
probably the most common situation.

Resource Server

This is the server hosting the protected resource or service. The resource server can
accept and respond to protected resource requests. In Oracle Cloud, a resource server
represents an application hosting cloud services.

Client Application

This is an application or service that can make protected resource requests on behalf of
the resource owner. The client application is the application requesting access to
resources stored on the resource server. The client application also obtains
authorization from the resource owner. The client application is also referred to as
OAuth client. In Oracle Cloud, an OAuth client represents an application making a
REST API call. Examples of OAuth clients are the Oracle Mobile Cloud Service and the
Oracle Java Cloud Service-SaaS Extension.

Authorization Server
This server supplies access tokens to the client after successfully authenticating the
resource owner and obtaining authorization. In Oracle Cloud, the OAuth service takes
this responsibility.

OAuth Roles in Oracle Cloud
As an administrator, you’re responsible for configuring and managing OAuth roles.
The resource owner in this scenario is the identity domain administrator. The server
hosting the cloud service is the resource server. The client application can be one of the
cloud services or an on-premises application or even a mobile application. The OAuth
service in Oracle Cloud is the authorization server. Your tasks include configuring
OAuth resources and clients, and managing them.

Note: To learn more about the configuration tasks for OAuth resources and
clients, see Managing OAuth Resources and Clients.

OAuth Endpoints
An endpoint is typically a URI on a web server, but OAuth endpoints are the URLs
that you use to make OAuth authentication requests.

From the different endpoints defined in OAuth 2.0, the token endpoint is used in
Oracle Cloud.

Token Endpoint
This is the endpoint on the authorization server where the client application requests
an access token. In Oracle Cloud, the OAuth service validates the client credentials
and supplies an access token to the client. You need to use the correct OAuth endpoint
when sending authentication requests from your application.

Using Token Endpoints in Oracle Cloud
A user in your organization is trying to access a resource (for example, Oracle Java
Cloud Service), using a client application (for example, a mobile application). Because
the cloud services are protected, the user is immediately redirected to the OAuth token
service, which prompts the user to provide credentials. The OAuth service
authenticates the user. The client application requests an access token from the OAuth
service by providing the credentials. On successful validation, the OAuth service
provides an access token to the OAuth client which redirects it to the user. This access
token is used to access the protected Oracle Java Cloud Service. The token represents
an authenticated user. The access token contains the information to verify the user
identity and resource access, so the user doesn’t have to be authenticated multiple
times.

How Does OAuth Work in Oracle Cloud?
OAuth is used to secure access to web services like Oracle Java Cloud Service-SaaS
Extension, Oracle Mobile Cloud Service, and Oracle Fusion Applications that are
exposed by Oracle Cloud.

A tenant is a subscribing customer that has a group of users who share common
access with specific privileges to a software instance. A tenant can have one or more
services hosted by Oracle Cloud that need to communicate with each other using REST API calls.

For example, your company ACME Corp., is a typical tenant in Oracle Cloud. As tenant you’ve subscribed to cloud services like Oracle Fusion Applications, Oracle Java Cloud Service-SaaS Extension, and Oracle Mobile Cloud Service. Oracle Cloud hosts these services, which communicate with each other using REST API calls. OAuth Service provides access tokens that OAuth clients use to make REST API calls to the services hosted by Oracle Cloud.

There are two types of REST API calls that can be made between services. They are:

- **Intra-Oracle Cloud**: These REST API calls come from one cloud service to another cloud service within Oracle Cloud. An example of an intra-Oracle Cloud REST API call is Oracle Java Cloud Service-SaaS Extensions calling Oracle Mobile Cloud Service.

- **Inter-Oracle Cloud**: These REST API calls come from an outside Oracle Cloud service to a cloud service within Oracle Cloud. An example of an inter-Oracle Cloud REST API call is a mobile application calling the Oracle Mobile Cloud Service.

In this scenario, an OAuth client can make two different types of REST API calls:

- **Simple REST API**: A client has a credential and wants to make a REST API call to an Oracle Cloud service. (For example a mobile application client making a REST API call to Oracle Mobile Cloud Service).

- **REST with identity propagation**: Identity propagation is the replication of authenticated identities. An Oracle Java Cloud Service-SaaS application wants to make a REST API call to another Oracle Cloud service like Oracle Mobile Cloud Service. Oracle Java Cloud Service-SaaS makes a REST call on-behalf of the user in the current security context. But the Oracle Java Cloud Service-SaaS application does not know the user credentials.

The OAuth service acts as the central trust manager. The OAuth clients are registered with the OAuth service. During the registration of an OAuth client, information on which APIs the OAuth client can access is also sent to the OAuth service. When the OAuth client makes a new REST API call to access a protected service, it receives an access token from the OAuth service. The OAuth client can use this token for subsequent access to the same service until it expires. If the token expires, the OAuth client again acquires an access token from the OAuth service. The OAuth service infrastructure validates the access token and allows the API call only if the token is valid (the signature validation succeeds and the tenant information matches).

**Supported OAuth Workflows**

Oracle Cloud supports two-legged OAuth workflow.

The type of process flow used to describe an OAuth request, typically refers to the number of parties involved. Two-legged OAuth workflow includes an OAuth client and a resource server. In OAuth two-legged authorization, consent from the resource owner is either assumed or not required.

**Two-Legged OAuth Workflow**

In two-legged OAuth workflow, the OAuth client is either the resource owner or the OAuth client is preapproved by the resource owner to access services. The OAuth client requests an access token using either the client credentials workflow or the resource owner password credentials workflow. The OAuth service validates the
client's (and if supplied, the resource owner's) credentials, and returns an access token to the OAuth client.

This is how a two-legged OAuth workflow works:

1. A user in your organization wants to use a mobile application to access Oracle Java Cloud Service in Oracle Cloud. The user places a request in a mobile application.

2. The mobile application is the OAuth client, and it places a request for an access token to the OAuth service.

3. After authenticating the client application, the OAuth service sends back the access token.

4. The OAuth client makes a request, and submits the access token to the resource server hosting Oracle Java Cloud Service.

5. The resource server sends back the request with the access token as the authorization to access the requested resource.

6. The OAuth client sends the response to the request back to the user.

**Different Types of OAuth Tokens**

A token is used to make security decisions to authorize a user and to store tamper-proof information about a system entity. An OAuth client uses three different tokens: client, user, and access.

**Types of OAuth Tokens in Oracle Cloud**

As an administrator, you’re responsible for securing the communication between different on-premises and cloud services. OAuth uses tokens to establish a secure communication between these services.

The three different types of tokens used in Oracle Cloud are:

1. A client token contains information that validates the client. The intent of the client token is to provide an alternative client authentication mechanism (that is, one that doesn’t send client credentials). To build an assertion for a client and generate a client token, see **Client Assertion**. An assertion is a package of information that allows identity and security information to be shared across Oracle Cloud.

2. A user token contains identity and security information about the user. You can use a user token to authenticate the user instead of a user name and password. To build an assertion for a user and generate a user token, see **User Assertion**.

3. An access token represents authorization for the client. Access tokens are credentials used to access protected OAuth resources. In Oracle Cloud, you can request an access token to access protected services in different ways. The way depends on the different grant types. A grant is a credential representing the resource owner’s authorization to access a protected resource. In the resource owner password credentials grant, the resource owner’s password credentials (user name and password) can be used directly as an authorization grant to obtain an access token. In the client credentials grant, the client authenticates with the OAuth service and requests an access token. In the user assertion grant, the user assertion is sent along with the client information. For more information about these three tokens, see **Securing Authorizations in Oracle Cloud**.
Supported Grant Types

An authorization grant is a credential representing the resource owner’s authorization to access its protected resource. The authorization grant is used by the OAuth client to obtain an access token.

The authorization grant is given to an OAuth client by the resource owner, in cooperation with the OAuth token service associated with the resource server. The following grant types are supported in Oracle Cloud:

- **Resource Owner Password Credentials Grant**
- **Client Credentials Grant**
- **User Assertion Grant**

**Resource Owner Password Credentials Grant**

The resource owner’s password credentials (that is, the user name and password) can be used by the OAuth client directly as an authorization grant to obtain an access token.

The resource owner password credentials grant type is suitable in cases where the resource owner has a trust relationship with the OAuth client.

This grant type requires that the OAuth client directly uses the resource owner’s credentials, but the credentials are used only for a single request and are exchanged for an access token. This grant type eliminates the need for the OAuth client to store the resource owner’s credentials for future use. This is done by exchanging the credentials for an access token.

**How the Resource Owner Password Credentials Grant Works**

This is how a resource owner’s password credentials grant works:

1. The resource owner provides its user name and password to the OAuth client.
2. The OAuth client requests an access token from the authorization server’s token endpoint. In the request, the client includes the credentials received from the resource owner.
3. The authorization server authenticates the OAuth client and validates the resource owner’s credentials. If the credentials are valid, then the authorization server supplies an access token.

**Client Credentials Grant**

The client credentials can be used as an authorization grant when the authorization scope is limited to the protected resources under the control of the OAuth client (that is, the client owns the resource), or to the protected resources previously arranged with the OAuth service.

Client credentials are used as an authorization grant typically when the OAuth client is acting on its own behalf (the OAuth client is also the resource owner), or is requesting access to the protected resources based on an authorization previously arranged with the OAuth service. Only confidential OAuth clients can use the client credentials grant type.

**How the Client Credentials Grant Works**

This is how the client credentials grant works:
1. The client requests an access token from the OAuth service by providing its credentials.
2. The OAuth service authenticates the client, and if valid, supplies an access token.

**Client Assertion**

A client assertion contains information that validates the client.

It provides an alternative client authentication mechanism (one that doesn’t send client secrets). Once you register your OAuth client, the OAuth service will provide client credentials in the form of a client identifier and a client secret. The client secret authenticates the identity of the client to the service API when the client requests to access.

**Signing the Client Assertion**

Client assertions received by the OAuth service must be signed by a signing algorithm. The signing algorithm must be **RS256:RSASSA-PKCS-v1_5** using **SHA-256** algorithm. You must use this algorithm when building client assertions to send to the OAuth service.

The OAuth token service already has the certificate of the OAuth client registered with it. This certificate is used to verify the client assertion. The client assertion is trusted with the help of the client certificate present in the client profile. This certificate is made available during client registration.

**Format of a Client Assertion**

The format of the client assertion includes the following:

- For trusted clients, the client assertion is signed by the client certificate. The algorithm used (**RS256**), type of token (**JWT**), thumbprint of the client public key (the thumbprint is the result of a hash function applied to the certificate itself) and the alias of the certificate are included in the header.

- **iss**: This identifies the principal that supplied the token. This is the client ID of the OAuth client application.

- **jti**: This is the unique ID of the request.

- **prn**: This identifies the principal that’s the subject of the token. This is the client ID of the OAuth client application.

- **exp**: This is the expiration time of the client token.

- **iat**: This is the issue time of the client token.

- **user.tenant.name**: This is the name of the identity domain.

- **aud**: This is the audience claim. An identity management OAuth-generated client assertion has an audience claim and its audience value is oauth.idm.oracle.com. This means that the self-signed client assertion can only be used by the OAuth server.

**Claims in a Client Assertion**

A client assertion has a header, standard claims, and custom claims, as described in the following table:

<table>
<thead>
<tr>
<th>Claim Name</th>
<th>Type</th>
<th>Description</th>
<th>Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>Claim Name</td>
<td>Type</td>
<td>Description</td>
<td>Sample</td>
</tr>
</tbody>
</table>

---

4-8 Oracle Cloud Understanding Identity Concepts
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>alg</td>
<td>Header</td>
<td>The algorithm used to sign the token.</td>
<td>RS256</td>
</tr>
<tr>
<td>typ</td>
<td>Header</td>
<td>The classification type of the token. The default value is JWT. This indicates that this is a JSON web token (JWT).</td>
<td>JWT</td>
</tr>
<tr>
<td>x5t</td>
<td>Header</td>
<td>The X.509 certificate thumbprint ((x5t)) header parameter provides a base64 url-encoded SHA-256 thumbprint of the Distinguished Encoding Rules (DER) encoding of an X.509 certificate that can be used to match a certificate.</td>
<td>_hVX9pXq7pUxkk5ry-8vK8qb8L8</td>
</tr>
<tr>
<td>kid</td>
<td>Header</td>
<td>The key ID ((kid)) header parameter is a hint indicating which specific key owned by the signer should be used to validate the signature. This allows signers to signal a change of the key to recipients explicitly. Omitting this parameter is equivalent to setting it to an empty string. The interpretation of the contents of the kid parameter is unspecified.</td>
<td>oauth_psrtenantx3.cert</td>
</tr>
<tr>
<td>sub</td>
<td>Standard Claim</td>
<td>The subject ((sub)) claim identifies the principal that is the subject of the JSON web token.</td>
<td>4457b326-fe88-4851-baad-b9488895e808</td>
</tr>
<tr>
<td>prn</td>
<td>Standard Claim</td>
<td>The principal ((prn)) claim identifies the principal that is the subject of the JSON web token.</td>
<td>4457b326-fe88-4851-baad-b9488895e808</td>
</tr>
<tr>
<td>iss</td>
<td>Standard Claim</td>
<td>The issuer ((iss)) claim identifies the principal that issued the JSON web token.</td>
<td>987e4dbe19c94be0aa47ff3ca8c62385</td>
</tr>
<tr>
<td>iat</td>
<td>Standard Claim</td>
<td>The issued at ((iat)) claim identifies the time at which the JSON web token was supplied.</td>
<td>1429128747000</td>
</tr>
<tr>
<td>exp</td>
<td>Standard Claim</td>
<td>The expiration time ((exp)) claim identifies the expiration time on or after which the JSON web token must not be accepted for processing.</td>
<td>1429128747000</td>
</tr>
<tr>
<td>aud</td>
<td>Standard Claim</td>
<td>The audience ((aud)) claim identifies the recipients for which the JSON web token is intended.</td>
<td>(a list of audiences)</td>
</tr>
<tr>
<td>jti</td>
<td>Standard Claim</td>
<td>The JSON web token ID ((jti)) claim provides a unique identifier for the JSON web token.</td>
<td>0565e04e-3823-404f-b950-e970ea17f41f</td>
</tr>
<tr>
<td>oracle.oauth.svc_p_n</td>
<td>Custom Claim</td>
<td>IDM OAuth service profile name.</td>
<td>oauth_psrtenantx3ServiceProfile</td>
</tr>
<tr>
<td>oracle.oauth.prn.id_type</td>
<td>Custom Claim</td>
<td>Principal ID type. For client assertion, the value is always ClientID.</td>
<td>ClientID</td>
</tr>
<tr>
<td>oracle.oauth.sub.id_type</td>
<td>Custom Claim</td>
<td>Subject ID type. For client assertion, the value is always ClientID.</td>
<td>ClientID</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------</td>
<td>------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>oracle.oauth.id_d_id</td>
<td>Custom Claim</td>
<td>IDM OAuth server domain ID.</td>
<td>20625897169639935</td>
</tr>
<tr>
<td>user.tenant.name</td>
<td>Custom Claim</td>
<td>User tenancy for the OAuth token generated by the IDM OAuth server.</td>
<td>oauth_psrtenantx3</td>
</tr>
</tbody>
</table>

**User Assertion Grant**

A user assertion is a user token that contains identity information about the user. The user assertion can be used directly as an authorization grant to obtain an access token.

This workflow has significantly different security properties than the other OAuth workflows. The primary difference is that the user’s credentials are never accessible to the application.

**How the User Assertion Grant Works**

The user assertion grant is more secure than the resource owner password credentials grant or the client credentials grant because the user’s credentials are never exposed.

This is how the user assertion grant works:

1. The client requests an access token by providing a user assertion. The client details are provided either as an authentication header in the request or as a client assertion.
2. The OAuth service authenticates the client, and, if valid, supplies an access token.

**User Assertion**

A user assertion is a user token that contains identity and security information about the user. Use the user assertion to authenticate the user instead of providing a user name and a password.

**Signing the User Assertion**

User assertions received by the OAuth service must be signed by a signing algorithm. The signing algorithm must be RS256:RSASSA-PKCS-v1_5 using the SHA-256 algorithm. Use this algorithm when building assertions to send to the OAuth service.

The certificate of the OAuth client is already registered with the authorization server. This certificate is used to sign the user assertion. The clients sign the user assertion, using their own private keys. The OAuth service uses the registered clients’ public key to validate the client and user assertions.

**Format of a User Assertion**

The format of the user assertion includes the following:

- For trusted clients, the OAuth client application signs the user assertion.
- **iss**: This identifies the principal that provided the token. This is the client ID of the OAuth client application.
- **jti**: This is the unique ID of the request.
- **prn**: This is the user name of the resource owner.
- **exp**: This is the expiration time of the user token.
- **iat**: This is the issue time of the user token.
- **user.tenant.name**: This is the name of the identity domain.
- **aud**: This is the audience claim. The IDM OAuth-generated user assertion has an audience claim and the audience values is `oauth.idm.oracle.com`. This means that the self-signed user assertion is meant only to be used by the OAuth server.

## Claims in the User Assertion

A user assertion has a header, and standard and custom claims. These items are covered in the following table:

<table>
<thead>
<tr>
<th>Claim Name</th>
<th>Type</th>
<th>Description</th>
<th>Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>alg</td>
<td>Header</td>
<td>The algorithm used to sign the token.</td>
<td>RS256</td>
</tr>
<tr>
<td>typ</td>
<td>Header</td>
<td>The classification type of the token. The default value is JWT. This indicates that this is a JSON web token (JWT).</td>
<td>JWT</td>
</tr>
<tr>
<td>x5t</td>
<td>Header</td>
<td>The X.509 certificate thumbprint (x5t) header parameter provides a base64url-encoded SHA-256 thumbprint of the DER encoding of an X.509 certificate that can be used to match a certificate.</td>
<td>_hVX9pXq7pUxk5ry-8vK8qb8L8</td>
</tr>
<tr>
<td>kid</td>
<td>Header</td>
<td>The key ID (kid) header parameter is a hint indicating which specific key owned by the signer should be used to validate the signature. This allows signers to signal a change of the key to recipients explicitly. Omitting this parameter is equivalent to setting it to an empty string. The interpretation of the contents of the kid parameter is unspecified.</td>
<td>oauth_psrtenantx3.cert</td>
</tr>
<tr>
<td>sub</td>
<td>Standard Claim</td>
<td>The subject (sub) claim identifies the principal that’s the subject of the JWT.</td>
<td><a href="mailto:MyAdmin@oracle1.com">MyAdmin@oracle1.com</a></td>
</tr>
<tr>
<td>prn</td>
<td>Standard Claim</td>
<td>The principal (prn) claim identifies the principal that is the subject of the JWT.</td>
<td><a href="mailto:MyAdmin@oracle1.com">MyAdmin@oracle1.com</a></td>
</tr>
<tr>
<td>iss</td>
<td>Standard Claim</td>
<td>The issuer (iss) claim identifies the principal that supplied the JWT.</td>
<td>oauth_psrtenantx3</td>
</tr>
<tr>
<td>iat</td>
<td>Standard Claim</td>
<td>The issued at (iat) claim identifies the time at which the JWT was supplied.</td>
<td>1429128747000</td>
</tr>
<tr>
<td>exp</td>
<td>Standard Claim</td>
<td>The expiration time (exp) claim identifies the expiration time on or after which the JWT must not be accepted for processing.</td>
<td>1429128747000</td>
</tr>
<tr>
<td>aud</td>
<td>Standard Claim</td>
<td>The audience (aud) claim identifies the recipients for which the JWT is intended.</td>
<td>(a list of audiences)</td>
</tr>
<tr>
<td>jti</td>
<td>Standard Claim</td>
<td>The JWT ID (jti) claim provides a unique identifier for the JWT.</td>
<td>0565e04e-3823-404f-b950-e970eal7f41f</td>
</tr>
</tbody>
</table>
### User Assertion

<table>
<thead>
<tr>
<th>Custom Claim</th>
<th>IDM OAUTH service profile name.</th>
<th>oauth_psrtenantx3ServiceProfile</th>
</tr>
</thead>
<tbody>
<tr>
<td>oracle.oauth.svc_p_n</td>
<td>Custom Claim</td>
<td>IDM OAUTH service profile name.</td>
</tr>
<tr>
<td>oracle.oauth.prn.id_type</td>
<td>Custom Claim</td>
<td>Principal ID type. For user assertion, the value is always LDAP_UID.</td>
</tr>
<tr>
<td>oracle.oauth.sub.id_type</td>
<td>Custom Claim</td>
<td>Subject ID type. For user assertion, the value is always LDAP_UID.</td>
</tr>
<tr>
<td>oracle.oauth.id_d_id</td>
<td>Custom Claim</td>
<td>IDM OAUTH server domain ID.</td>
</tr>
<tr>
<td>oracle.oauth.client_origin_id</td>
<td>Custom Claim</td>
<td>Subject ID for client used when user assertion is generated.</td>
</tr>
<tr>
<td>user.tenant.name</td>
<td>Custom Claim</td>
<td>User tenancy for the OAuth token generated by IDM OAUTH server.</td>
</tr>
</tbody>
</table>