8

Authenticating and Authorizing Microsoft Azure Users for Oracle Databases

An Oracle database can be configured for Microsoft Azure users of Microsoft Entra ID (previously called Microsoft Azure AD) to connect using single sign-on authentication.

Note:

Microsoft recently changed the name of Microsoft Azure AD to Microsoft Entra ID. This name change is used in the current Oracle Database documentation. Earlier Oracle Database releases use the name Azure AD.

- Introduction to Oracle Database Integration with Microsoft Entra ID
  Before you begin configuring Microsoft Entra AD to access an Oracle database, you must understand the overall process.
- Configuring the Oracle Database for Microsoft Entra ID Integration
  The Microsoft Entra ID integration with the Oracle Database instance requires the database to be registered with Entra ID.
- Mapping Oracle Database Schemas and Roles
  Azure users will be mapped to one database schema and optionally to one or more database roles.
- Configuring Entra ID Client Connections to the Oracle Database
  You can configure client connections to connect with the registered database.
- Configuring Microsoft Entra ID Proxy Authentication
  Proxy authentication allows an Azure user to proxy to a database schema for tasks such as application maintenance.
- Configuring Microsoft Power BI Single-Sign On
  Users have an option of a simpler configuration if only Power BI users will connect to the Oracle Database.
- Troubleshooting Microsoft Entra ID Connections
  You can use trace files to diagnose problems with Microsoft Entra ID connections. You also can easily remedy `ORA-12599` and `ORA-03114` errors.

8.1 Introduction to Oracle Database Integration with Microsoft Entra ID

Before you begin configuring Microsoft Entra AD to access an Oracle database, you must understand the overall process.

- About Integrating Oracle Database with Microsoft Entra ID
  Oracle Database and Microsoft Entra ID can be configured to allow users and applications to connect to the database using their Entra ID credentials.

- Architecture of Oracle Database Integration with Microsoft Entra ID
  Microsoft Azure Active Directory access tokens follow the OAuth 2.0 standard with extensions.
- Azure Users Mapping to an Oracle Database Schema and Roles
  Microsoft Azure users must be mapped to an Oracle Database schema and have the necessary privileges (through roles) before being able to authenticate to the Oracle Database instance.
- Use Cases for Connecting to an Oracle Database Using Entra ID
  Oracle Database supports several use cases for connecting to the database.
- General Process of Authenticating Microsoft Entra ID Identities with Oracle Database
  The Oracle Database administrator and the Microsoft Entra ID administrator play roles to allow Azure users to connect to the database using Entra ID `OAuth2` access tokens.

8.1.1 About Integrating Oracle Database with Microsoft Entra ID

Oracle Database and Microsoft Entra ID can be configured to allow users and applications to connect to the database using their Entra ID credentials.

Azure users and applications can log in with Entra ID Single Sign On (SSO) credentials to access the database. This is done with an Entra ID `OAuth2` access token that the user or application first requests from Entra ID. This `OAuth2` access token contains the user identity and database access information and is then sent to the database. Refer to Refer to the Microsoft article Passwordless authentication options for Azure Active Directory for information about configuring multi-factor and passwordless authentication.

You can perform this integration in the following Oracle Database environments:

- On-premises Oracle Database release 19.18 and later, excluding 21c
- All Oracle Database server platforms: Linux, Windows, AIX, Solaris, and HPUX
- Oracle Autonomous Database Serverless
- Oracle Autonomous Database on Dedicated Exadata Infrastructure
- Oracle Autonomous Database on Exadata Cloud@Customer
- Oracle Exadata Database Service on Dedicated Infrastructure
- Oracle Exadata Database Service on Cloud@Customer
- Oracle Base Database Service

The instructions for configuring Entra ID use the term "Oracle Database" to encompass these environments.

This type of integration enables the Azure user to access an Oracle Database instance. Azure users and applications can log in with Entra ID Single Sign On (SSO) credentials to get an Entra ID `OAuth2` access token to send to the database.

The Entra ID administrator creates and registers Oracle Database with Entra ID. Within Entra ID, this is called an app registration, which is short for application registration. This is the digital information that Entra ID must know about the software that is using Entra ID. The Entra ID administrator also creates application (app) roles for the database app registration in Entra ID. App roles connect Azure users, groups, and applications to database schemas and roles. The Entra ID administrator assigns Azure users, groups, or applications to the app roles. These app roles are mapped to a database global schema or a global role or to both a schema and a role. An Azure user, group, or application that is assigned to an app role will be mapped to a database global schema, global role, or to both a schema and a role. An Oracle global schema can also be mapped exclusively to an Azure user. An Azure guest user (non-organization user) or an Entra ID service principal (application) can only be mapped to a database global schema through an Entra ID app role. An Oracle global role can only be mapped from an Azure app role and cannot be mapped from an Azure user.

Tools and applications that are updated to support Entra ID tokens can authenticate users directly with Entra ID and pass the database access token to the Oracle Database instance. You can configure existing database tools such as SQL*Plus to use an Entra ID token from a file location. In these cases, Entra ID tokens can be retrieved using tools like Microsoft PowerShell or Azure CLI and put into a file location. An Entra ID `OAuth2` database access tokens are issued with an expiration time. The Oracle Database client driver will ensure that the token is in a valid format and that it has not expired before passing it to the database. The token is scoped for the database, which means that there is information in the token about the database where the token will be used. The app roles the Entra ID principal was assigned to in the database Entra ID app registration are included as part of the access token. The directory location for the Entra ID token should only have enough permission for the user to write the token file to the location and the database client to retrieve these files (for example, just read and write by the user). Because the token allows access to the database, it should be protected within the file system.

Azure users can request a token from Entra ID using a number of methods to open an Azure login window to enter their Entra ID credentials.

Oracle Database accepts tokens representing the following Entra ID principals:

- Azure user, who is registered user in the Entra ID tenancy
- Guest user, who is registered as a guest user in the Entra ID tenancy
- Service, which is the registered application connecting to the database as itself with the client credential flow (connection pool use case)

Oracle Database supports the following Entra ID authentication flows:

- Interactive flow (also called authorization code flow) using Proof Key for Code Exchange (PKCE), most commonly used for human users (not applications) to authenticate to Entra ID in a client environment with a browser
- Client credentials, which are for database applications that connect as themselves (and not the end-user)
- On-Behalf-Of (OBO), where an application requests an access token on behalf of a logged-in user to send to the database
- Resource owner password credential (ROPC), which is not recommended for production use, but can be used in test environments where a pop-up browser user authentication would be difficult to incorporate. ROPC needs the Entra ID user name and password credential to be part of the token request call.

Note:

The DBaaS integration with Microsoft Entra ID does not support users with administrative privileges (`SYSDBA`, `SYSOPER`, `SYSBACKUP`, `SYSDG`, `SYSKM`, and `SYSRAC`).

Related Topics

- Oracle Autonomous Database Serverless
- Oracle Autonomous Database on Dedicated Exadata Infrastructure
- Oracle Autonomous Database on Exadata Cloud@Customer

- Oracle Exadata Database Service on Dedicated Infrastructure
- Oracle Exadata Database Service on Cloud@Customer
- Oracle Base Database Service

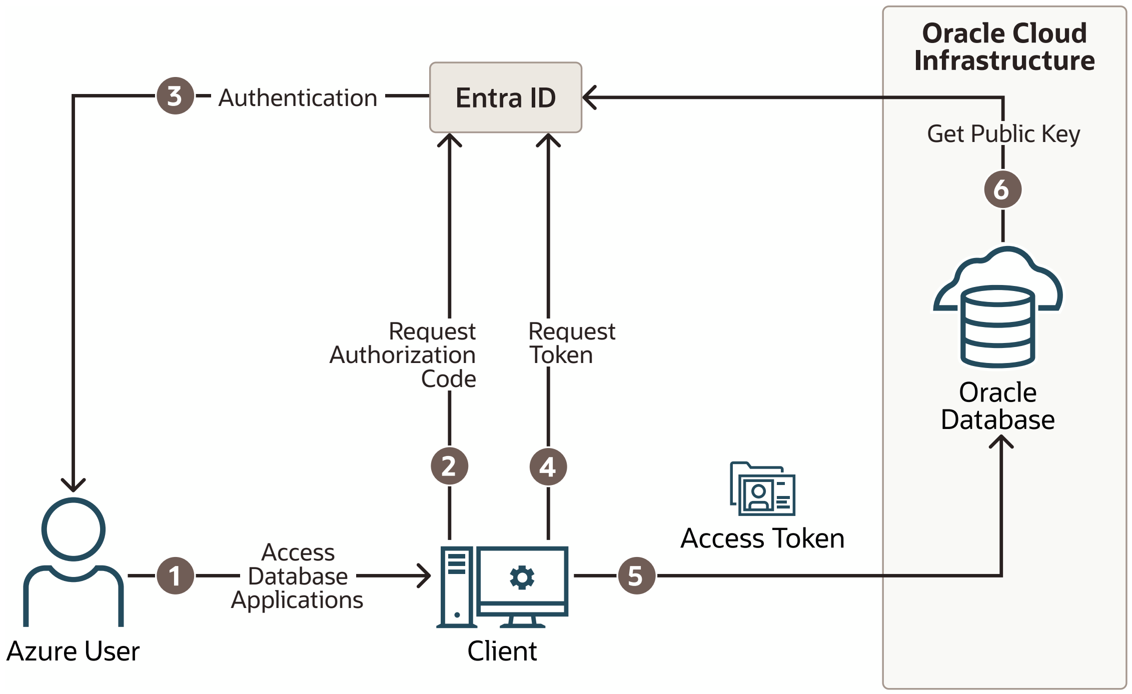
8.1.2 Architecture of Oracle Database Integration with Microsoft Entra ID

Microsoft Azure Active Directory access tokens follow the OAuth 2.0 standard with extensions.

The Entra ID access token will be needed before you access the database from the database client (for example, with SQLPlus or SQLcl). The Oracle clients (for example, OCI, JDBC, and ODP) can be configured to pick up an Entra ID token from a file location or the token can be passed to the client through the database client API. An Azure user can use a script (examples available from Microsoft) to retrieve a token and put it into a file location for the database client to retrieve. Applications can use the Azure SDK to get an access token and pass the token through the database client API. Command-line tools such as Microsoft PowerShell or the Azure command-line interface can be used to retrieve the Entra ID token if the application cannot directly get the token.

The following diagram is a generalized flow diagram for OAuth 2.0 standard, using the `OAuth2` token. See Authentication flow support in MSAL in the Microsoft Entra ID documentation for more details about each supported flow.

Figure 8-1 Azure User Accessing the Database with the Interactive Authorization Code Flow



The authorization code flow is an OAuth2 standard and is described in detail as part of the standards. There are two steps in the flow. The first step authenticates the user and retrieves the authorization code. The second step uses the authorization code to get the database access token.

1. The Azure user requests access to the resource, the Oracle Database instance.

2. The database client or application requests an authorization code from Entra ID.
3. Entra ID authenticates the Azure user and returns the authorization code.
4. The helper tool or application uses the authorization code with Entra ID to exchange it for the `OAuth2` token.
5. The database client sends the `OAuth2` access token to the Oracle database. The token includes the database app roles the user was assigned to in the Entra ID app registration for the database.
6. The Oracle Database instance uses the Entra ID public key to verify that the access token was created by Entra ID.

Both the database client and the database server must be registered with the **app registrations** feature in the Azure Active Directory section of the Azure portal. The database client must be registered with Entra ID app registration. Permission must also be granted to allow the database client to get an access token for the database.

8.1.3 Azure Users Mapping to an Oracle Database Schema and Roles

Microsoft Azure users must be mapped to an Oracle Database schema and have the necessary privileges (through roles) before being able to authenticate to the Oracle Database instance.

In Microsoft Azure, an Entra ID administrator can assign users, groups, and applications to the database app roles.

Exclusively mapping an Entra ID user to a database schema requires the database administrator to create and manage a database schema for the lifecycle of the user (joining, moving, leaving). The database administrator must create the schema when the user joins the organization. The database administrator must also modify the privileges and roles that are granted to the database schema to align them with the tasks the Azure user is assigned to. When the Azure user leaves the organization, the database administrator must drop the database schema so that an unused account is not left on the database. Using the database app roles enables the Entra ID administrator to control access and roles by assigning users to app roles that are mapped to global schemas and global roles. This way, user access to the database is managed by Entra ID administrators and database administrators do not need to create, manage, and drop schemas for every user.

An Azure user can be mapped to a database schema (user) either exclusively or through an app role.

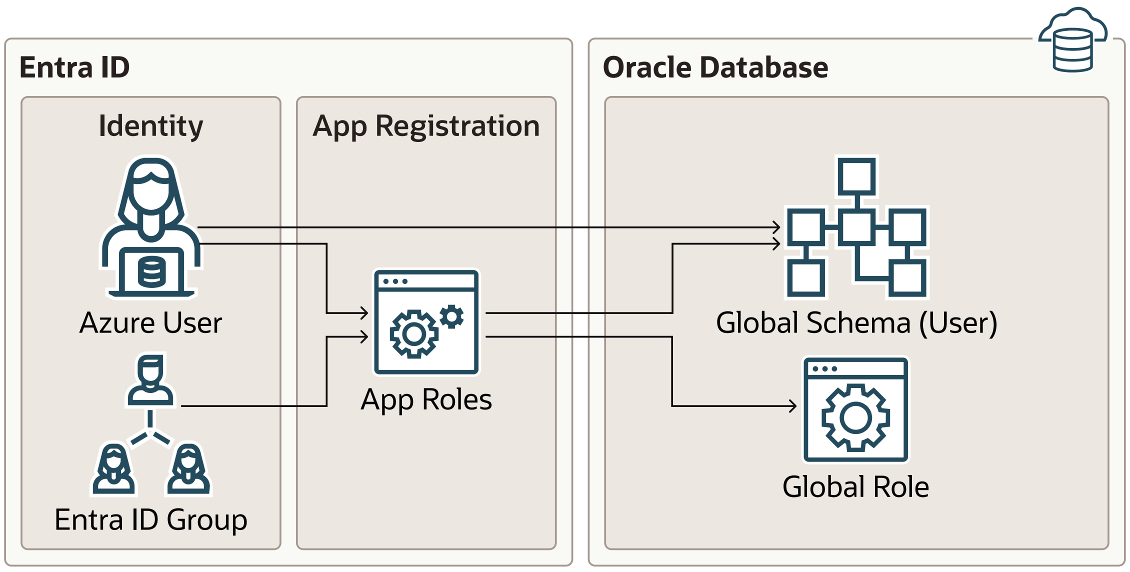
- **Creating an exclusive mapping between an Azure user and an Oracle Database schema.** In this type of mapping, the database schema must be created for the Azure user. Database privileges and roles that are needed by the Azure user must be granted to the database schema. The database schema not only must be created when the Azure user is authorized to the database, but the granted privileges and roles must be modified as the Entra ID roles and tasks change. Finally, the database schema must be dropped when the Azure user leaves the organization.
- **Creating a shared mapping between an Entra ID app role and an Oracle Database schema.** This type of mapping, which is more common than exclusive mappings, is for Azure users who have been assigned directly to the app role or is a member of an Entra ID group that is assigned to the app role. The app role is mapped to an Oracle Database schema (shared schema mapping). Shared schema mapping allows multiple Azure users to share the same Oracle Database schema so a new database schema is not required to be created every time a new user joins the organization. This operational efficiency allows database administrators to focus on database application maintenance, performance, and

tuning tasks instead of configuring new users, updating privileges and roles, and removing accounts.

In addition to database roles and privileges being granted directly to the mapped global schema, additional roles and privileges can be granted through mapped global roles. Different Azure users mapped to the same shared global schema may need different privileges and roles. Azure app roles can be mapped to Oracle Database global roles. Azure users who are assigned to the app role or are a member of an Entra ID group that is assigned to the app role will be granted the Oracle Database global role when they access the database.

The following diagram illustrates the different types of assignments and mappings that are available.

Figure 8-2 Assignments and Mappings Between Entra ID and Oracle Database



These mappings are as follows:

- An Azure user can be mapped directly to an Oracle Database global schema (user).
- An Azure user, Entra ID group, or application is assigned to an app role, which is then mapped to either an Oracle Database global schema (user) or a global role.

8.1.4 Use Cases for Connecting to an Oracle Database Using Entra ID

Oracle Database supports several use cases for connecting to the database.

- **OAuth2 authorization code flow:** This is the most common flow for human users. The client directs the Azure user to Entra ID to get the authorization code. This code is used to get the database access token. See the Microsoft Azure article Microsoft identity platform and OAuth 2.0 authorization code flow.
- **Resource owner password credentials (ROPC):** This flow is not recommended for production servers. It is useful for test software that cannot work with a pop-up authentication window. It is used in non-graphic user interface environments when a pop-up window cannot be used to authenticate a user.
- **Client credentials:** This flow is used for applications to connect with the database. The application must register with Entra ID app registration and needs a client ID and client

password. These client credentials must be used to get the database access token from Entra ID when the application connects to the database. The application can pass the token through the file system or through the database client API.

- **On-behalf-of (OBO) token:** An Azure application requests an OBO token for a logged in user. The OBO token will also be an access token for the database with the Azure user identity and assigned app roles for the database. This enables the Azure user to log in to the database as the user and not the application. Only an application can request an OBO token for its Azure user and pass it to the database client through the API.

8.1.5 General Process of Authenticating Microsoft Entra ID Identities with Oracle Database

The Oracle Database administrator and the Microsoft Entra ID administrator play roles to allow Azure users to connect to the database using Entra ID `OAuth2` access tokens.

The general process is as follows:

1. The Oracle Database administrator ensures that the Oracle Database environment meets the requirements for the Microsoft Entra ID integration. See Oracle Database Requirements for the Microsoft Entra ID Integration.
2. The Entra ID administrator creates an Entra ID app registration for the database and the Oracle Database administrator enables the database to use Entra ID tokens for database access.
   As part of the app registration process, the Entra ID administrator creates Azure app roles to be used for the mappings between the Azure users, groups, and applications to the Oracle Database schemas and roles.
3. The Oracle Database administrator creates and maps global schemas to either an Azure user (exclusive schema mapping) or to an Azure app role (shared schema mapping). The Azure user or application must be mapped to one schema.
4. Optionally, the Oracle administrator creates and maps global Oracle Database roles to Azure app roles.
5. The Azure end user who wants to connect with the Oracle Database instance registers the client application as an Entra ID client (similar to how the Oracle database is registered). The Entra ID client will have a client identification and a client secret, unless the application client is public. If the application client is public, then only the application client identification is necessary.
6. The Azure user (who can be a database administrator) connects using an utility such as PowerShell or the Azure command-line interface to retrieve the `OAuth2` database access token and store it in a local file directory. An application can also request an Entra ID `OAuth2` access token directly from Entra ID and pass it through a database client API. Refer to the following Oracle Database client documentation for information about passing Entra ID `OAuth2` tokens:
   - JDBC-thin clients: *Oracle Database JDBC Developer’s Guide*
   - Oracle Call Interface (OCI): *Oracle Call Interface Developer's Guide*
   - Oracle Data Provider for .NET (ODP): *Oracle Data Provider for .NET Developer's Guide*Connecting to Oracle Database
7. Once connected to the Oracle Database instance, the Azure end user performs database operations as needed.

8.2 Configuring the Oracle Database for Microsoft Entra ID Integration

The Microsoft Entra ID integration with the Oracle Database instance requires the database to be registered with Entra ID.

- Oracle Database Requirements for the Microsoft Entra ID Integration
  Before you can configure an Oracle Database instance with Microsoft Entra ID, you must ensure that your environment meets special requirements.
- Registering the Oracle Database Instance with a Microsoft Entra ID Tenancy
  A user with Entra ID administrator privileges uses Microsoft Entra ID to register the Oracle Database instance with the Microsoft Entra ID tenancy.
- Enabling Microsoft Entra ID v2 Access Tokens
  Oracle Database supports integration with the v1 and v2 Azure AD `OAuth2` access token.
- Managing App Roles in Microsoft Entra ID
  In Entra ID, you can create and manage app roles that will be assigned to Azure users and groups and also be mapped to Oracle Database global schemas and roles.
- Enabling Entra ID External Authentication for Oracle Database
  You need to enable Microsoft Entra ID external authentication with Oracle Database.
- Disabling Entra ID External Authentication for Oracle Database
  To disable Entra ID External authentication for an Oracle Database instance, you must use the `ALTER SYSTEM` statement.

8.2.1 Oracle Database Requirements for the Microsoft Entra ID Integration

Before you can configure an Oracle Database instance with Microsoft Entra ID, you must ensure that your environment meets special requirements.

For an on-premises, non-cloud Oracle database, follow the steps in this document. If your Oracle database is in one of the following DBaaS platforms, then refer to the platform documentation for additional requirements.

- *Using Oracle Autonomous Database Serverless*
- *Using Oracle Autonomous Database on Dedicated Exadata Infrastructure*
- Use Azure Active Directory Authentication with Base Database Service
- Use Azure Active Directory Authentication with Exadata Database on Dedicated Infrastructure

Note the following:

- The Oracle Database server must be able to request the Entra ID public key. Depending on the enterprise network connectivity setup, you may need to configure a proxy setting.
- Users and applications that need to request an Entra ID token must also be able to have network connectivity to Entra ID. You may need to configure a proxy setting for the connection.
- You must configure Transport Layer Security (TLS) between the Oracle Database client and the Oracle Database server so that the token can be transported securely. This TLS connection can be either one-way or mutual.

- You can create the TLS server certificate to be self-signed or be signed by a well known certificate authority. The advantage of using a certificate that is signed by a well known Certificate Authority (CA) is that the database client can use the system default certificate store to validate the Oracle Database server certificate instead of having to create and maintain a local wallet with the root certificate. Note that this applies to Linux and Windows clients only.

Related Topics

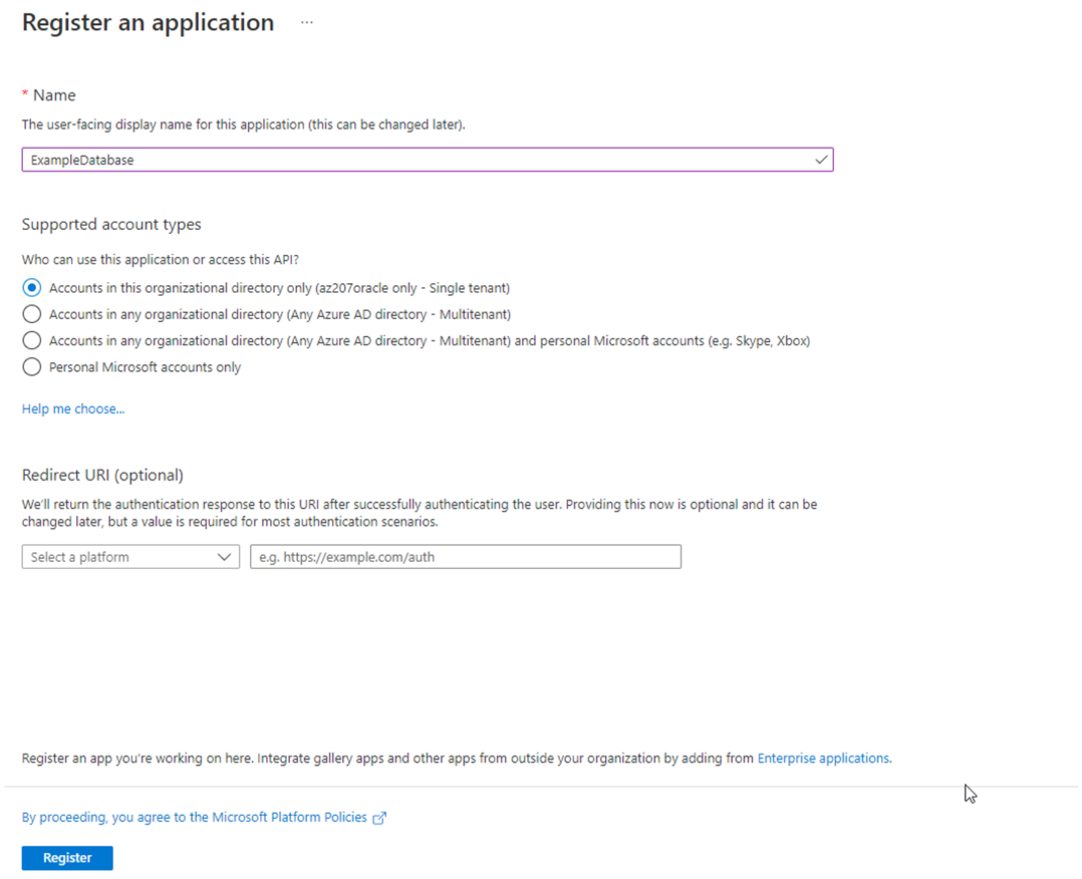
- Configuring TLS Using a Public Certificate Authority Root of Trust for the Database Server Certificate
  Before you can configure TLS without using client wallets, you must first create the server wallet and ensure that the database and listener are properly configured.

8.2.2 Registering the Oracle Database Instance with a Microsoft Entra ID Tenancy

A user with Entra ID administrator privileges uses Microsoft Entra ID to register the Oracle Database instance with the Microsoft Entra ID tenancy.

1. Log in to the Azure portal as an administrator who has Microsoft Entra ID privileges to register applications.
2. In the Azure Active directory admin center page, from the left navigation bar, select **Azure Active Directory**.
3. In the MS - App registrations page, select **App registrations** from the left navigation bar.
4. Select **New registration**.

The Register an application window appears.


Register an application ···

* Name

The user-facing display name for this application (this can be changed later).

ExampleDatabase

Supported account types

Who can use this application or access this API?

- [✓ Accounts in this organizational directory only (az207oracle only - Single tenant)]
- Accounts in any organizational directory (Any Azure AD directory - Multitenant)
- Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- Personal Microsoft accounts only

Help me choose...

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Select a platform | e.g. https://example.com/auth

Register an app you're working on here. Integrate gallery apps and other apps from outside your organization by adding from Enterprise applications.

By proceeding, you agree to the Microsoft Platform Policies

5. In the Register an application page, enter the following Oracle Database instance registration information:
   - In the **Name** field, enter a name for the Oracle Database instance connection (for example, `Example Database`).
   - Under Supported account types, select the account type that matches your use case.
     - **Accounts in this organizational directory only (*tenant_name* only - Single tenant)**
     - **Accounts in any organizational directory (Any Entra ID directory - Multitenant)**
     - **Accounts in any organizational directory (Any Entra ID directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)**
     - **Personal Microsoft accounts only**
6. Bypass the Redirect URI (Optional) settings. You do not need to create a redirect URI because Entra ID does not need one for the database server.
7. Click **Register**.

   After you click **Register**, Entra ID displays the app registration's Overview pane, which will show the Application (client) ID under Essentials. This value is a unique identifier for the application in the Microsoft identity platform. Note the term Application refers to the Oracle Database instance.
8. Register a scope for the database app registration.

A scope is a permission to access the database. Each database will need a scope so that clients can establish a trust with the database by requesting permission to use the database scope. This allows the database client to get access tokens for the database.

a. In the left navigation bar, select **Expose an API**.

b. Under Set the App ID URI, in the **Application ID URI** field, enter the app ID URI for the database connection using the following format, and then click **Save**:

   ```
   your_tenancy_url/application_(client)_id
   ```

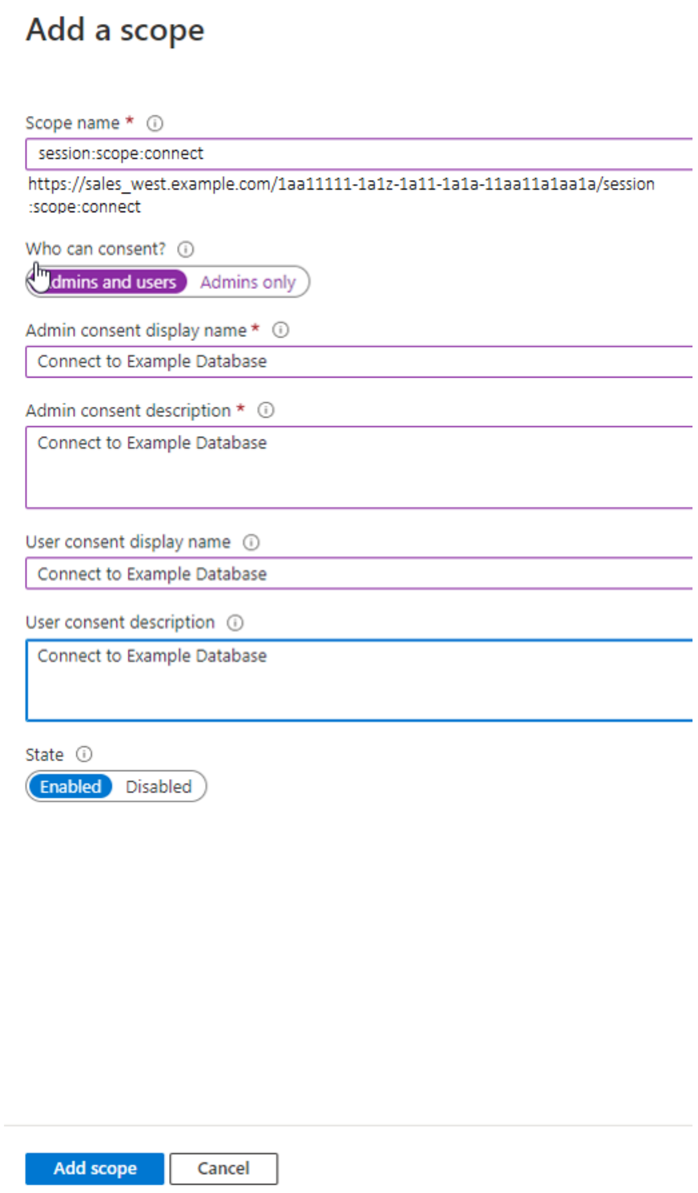
   In this specification:

   - *your_tenancy_url* must include `https` as the prefix and the fully qualified domain name of your Entra ID tenancy.
   - *application_(client)_id* is the ID that was generated when you registered the Oracle Database instance with Entra ID. It is displayed in the Overview pane of the app registration.

   For example:

   ```
   https://sales_west.example.com/1aa11111-1a1z-1a11-1a1a-11aa11a1aa1a
   ```

c. Select **Add a scope** and then enter the following settings:


Add a scope

Scope name *
session:scope:connect
https://sales_west.example.com/1aa11111-1a1z-1a11-1a1a-11aa11a1aa1a/session:scope:connect

Who can consent?
[✓ Admins and users] Admins only

Admin consent display name *
Connect to Example Database

Admin consent description *
Connect to Example Database

User consent display name
Connect to Example Database

User consent description
Connect to Example Database

State
[✓ Enabled] Disabled

- **Scope name** specifies a name for the scope. Enter the following name:

  ```
  session:scope:connect
  ```

  This name can be any text. However, a scope name must be provided. You will need to use this scope name later when you give consent to the database client application to access the database.
- **Who can consent** specifies the necessary permissions. Select **Admins and users**, or for higher restrictions, **Admins only**.
- **Admin consent** display name describes the scope's purpose (for example, `Connect to Oracle`), which only administrators can see.
- **Admin consent display name** describes the scope's purpose (for example, `Connect to Example Database`), which only administrators can see.

- **User consent display name** is a short description of the purpose of the scope (for example, `Connect to Example Database`), which users can see if you specify **Admins and users** in **Who can consent**.
- **User consent description** is a more detailed description of the purpose of the scope (for example, `Connect to Example Database`), which users can see if you specify **Admins and users** in **Who can consent**.
- **State** enables or disables the connection. Select **Enabled**.

After you complete these steps, you are ready to add one or more Azure app roles, and then perform the mappings of Oracle schemas and roles.

Related Topics

- Quickstart: Register an application with the Microsoft identity platform

8.2.3 Enabling Microsoft Entra ID v2 Access Tokens

Oracle Database supports integration with the v1 and v2 Azure AD `OAuth2` access token.

Oracle Database supports the Entra ID v2 token as well as the default v1 token. However, to use the Entra ID v2 token, you must perform some additional steps to ensure it works with the Oracle Database. You can use this token with applications that are registered in the Azure portal using the **App registrations** experience.

When you use the Azure AD v2 `OAuth2` access token, the credential flow continues to work as it did before without any changes. However, the `upn:` claim must be added when you use v2 tokens with the interactive flow.

1. Check the version of the Entra ID access token that you are using.
2. Log in to the Microsoft Entra ID portal.
3. Search for and select **Entra ID**.
4. Under **Manage**, select **App registrations**.
5. Choose the application for which you want to configure optional claims based on your scenario and desired outcome.
6. Under **Manage**, select **Token configuration**.
7. Click **Add optional claim** and select **upn**.

When you use v2 tokens, the `aud:` claim only reflects the APP ID value. You do not need to set the `https:domain` prefix to the APP ID URI when v2 tokens are being used. This simplifies the configuration for the database because the default APP ID URI can be used.

Related Topics

- Checking the Entra ID Access Token Version
  You can check the version of the Entra ID access token that your site uses by using the JSON Web Tokens web site.

8.2.4 Managing App Roles in Microsoft Entra ID

In Entra ID, you can create and manage app roles that will be assigned to Azure users and groups and also be mapped to Oracle Database global schemas and roles.

- Creating a Microsoft Entra ID App Role
  Azure users, groups, and applications that need to connect to the database will be assigned to the database app roles.
- Assigning Users and Groups to the Microsoft Entra ID App Role
  Before Microsoft Azure users can have access to the Oracle database, they must first be assigned to the app roles that will be mapped to Oracle Database schema users or roles.
- Assigning an Application to an App Role
  An application that must connect to the database using the client credential flow must to be assigned to an app role.

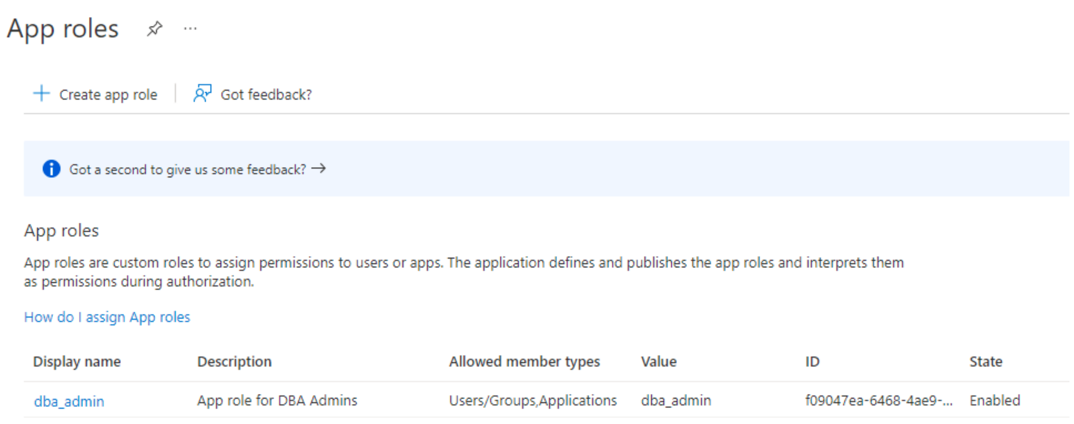
8.2.4.1 Creating a Microsoft Entra ID App Role

Azure users, groups, and applications that need to connect to the database will be assigned to the database app roles.

See the Microsoft Azure article Create and assign a custom role in Azure Active Directory for detailed steps on how to create an app role. The following steps describe how to create the app role for use with an Oracle database.

1. Log in to Entra ID as an administrator who has privileges for creating app roles.
2. Access the Oracle Database app registration that you created.
   a. Use the **Directory + subscription** filter to locate the Entra ID tenant that contains the Oracle Database app registration.
   b. Select **Azure Active Directory**.
   c. Under **Manage**, select **App registrations**, and then select the Oracle Database instance that you registered earlier.
3. Under **Manage**, select **App roles**.
4. In the App roles page, select **Create app role**.
5. In the Create app role page, enter the following information:
   - **Display name** is the displayed name of the role (for example, `HR App Schema`). You can include spaces in this name.
   - **Value** is the actual name of the role (for example, `HR_APP`). Ensure that this setting matches exactly the string that is referenced in the database mapping to a schema or role. Do not include spaces in this name.
   - **Description** provides a description of the purpose of this role.
   - **Do you want to enable this app role?** enables you to activate the role.
6. Click **Apply**.

   The app role appears in the App roles pane.


App roles
Create app role
Got feedback?
Got a second to give us some feedback?
How do I assign App roles

App roles

App roles are custom roles to assign permissions to users or apps. The application defines and publishes the app roles and interprets them as permissions during authorization.

Display name	Description	Allowed member types	Value	ID	State
dba_admin	App role for DBA Admins	Users/Groups,Applications	dba_admin	f09047ea-6468-4ae9-...	Enabled

8.2.4.2 Assigning Users and Groups to the Microsoft Entra ID App Role

Before Microsoft Azure users can have access to the Oracle database, they must first be assigned to the app roles that will be mapped to Oracle Database schema users or roles.

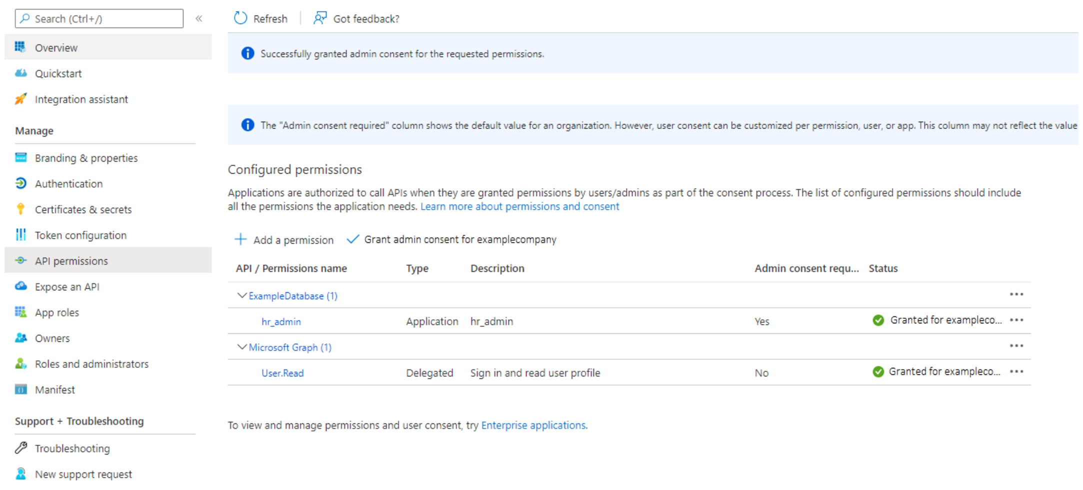
See the Microsoft Azure article Add app roles to your application and receive them in the token for detailed steps assigning users and groups to an app role. The following steps explain how to do this for an Oracle database.

1. Log in to Entra ID as an administrator who has privileges for assigning Azure users and Entra ID groups to app roles.
2. In enterprise applications, find the name of the Oracle Database app registration that you created. This is automatically created when you create an app registration.
   a. Use the **Directory + subscription** filter to locate the Azure Active Directory tenant that contains the Oracle connection.
   b. Select **Azure Active Directory**.
   c. Under **Manage**, select **Enterprise applications**, and then select the Oracle Database app registration name that you registered earlier.
3. Under Getting Started, select **Assign users and groups**.
4. Select **Add user/group**.
5. In the Add assignment window, select **Users and groups** to display a list of users and security groups.
6. From this list, select the users and groups that you want to assign to the app role, and then click **Select**.
7. In the Add assignment window, select **Select a role** to display a list of the app roles that you have created.
8. Select the app role and then select **Select**.
9. Click **Assign**.

8.2.4.3 Assigning an Application to an App Role

An application that must connect to the database using the client credential flow must to be assigned to an app role.

1. Log in to Entra ID as an administrator who has privileges for assigning Azure users and Entra ID groups to app roles.
2. Access the app registration for the application.
3. Under Manage, select **API permissions**.
4. In the Configured permissions area, select **+ Add a permission**.
5. In the Request API permission pane, select the **My APIs** tab.
6. Select the Oracle Database app that you want to give permission for this application to access. Then select the **Application permissions** option.
7. Select the database app roles to assign to the application and then click the **Add Permission** box at the bottom of the screen to assign the app roles and close the dialog box. Ensure that the app roles that you just assigned appear under Configured permissions.



Successfully granted admin consent for the requested permissions.

The "Admin consent required" column shows the default value for an organization. However, user consent can be customized per permission, user, or app. This column may not reflect the value

Configured permissions

Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs. Learn more about permissions and consent

API / Permissions name	Type	Description	Admin consent requ...	Status
ExampleDatabase (1)				
hr_admin	Application	hr_admin	Yes	✓ Granted for exampleco...
Microsoft Graph (1)				
User.Read	Delegated	Sign in and read user profile	No	✓ Granted for exampleco...

To view and manage permissions and user consent, try Enterprise applications.

8. Select **Grant admin consent for *tenancy*** to grant consent for the tenancy users, then select **Yes** in the confirmation dialog box.

Related Topics

- Configure the admin consent workflow

8.2.5 Enabling Entra ID External Authentication for Oracle Database

You need to enable Microsoft Entra ID external authentication with Oracle Database.

For additional information about Entra ID authentication for your platform, see the documentation links below.

1. Log in to the Oracle Database instance as a user who has been granted the `ALTER SYSTEM` system privilege.
2. Set the `IDENTITY_PROVIDER_TYPE` parameter as follows:

```
ALTER SYSTEM SET IDENTITY_PROVIDER_TYPE=AZURE_AD SCOPE=BOTH;
```

3. Ensure that you set the `IDENTITY_PROVIDER_TYPE` parameter correctly.

```
SELECT NAME, VALUE FROM V$PARAMETER WHERE NAME='identity_provider_type';
```

The following output should appear:

```
NAME                    VALUE
----------------------  -------
identity_provider_type  AZURE_AD
```

4. Set the `IDENTITY_PROVIDER_CONFIG` parameter by using the following syntax:

   ```
   ALTER SYSTEM SET IDENTITY_PROVIDER_CONFIG =
   '{
      "application_id_uri": string , // from registered app, to be mapped in 
   jwt "aud" claim;
                                    // Domain qualified to support cross 
   tenancy resource access
      "tenant_id": string, // from tenant config
      "app_id": string // from registered resource app
   }' SCOPE=BOTH;
   ```

   For example:

   ```
   ALTER SYSTEM SET IDENTITY_PROVIDER_CONFIG =
   '{
     "application_id_uri" : "https://www.example.com/11aa1a11-
   aaaa-1111-1111-1111aa11111",
     "tenant_id" : "111a1111-a11a-111a-1a1a-1111111111a",
     "app_id" : "11aa1a11-aaaa-1111-1111-1111aa11111"
   }' SCOPE=BOTH;
   ```

See the following platform-specific documentation for information about enabling Oracle Database for Entra ID external authentication, in addition to the information detailed in this document for on-premises (non-cloud) Oracle databases.

- *Using Oracle Autonomous Database Serverless*
- *Oracle Autonomous Database on Dedicated Exadata Infrastructure*

8.2.6 Disabling Entra ID External Authentication for Oracle Database

To disable Entra ID External authentication for an Oracle Database instance, you must use the `ALTER SYSTEM` statement.

In addition to Oracle Database, this procedure can be used for Oracle Autonomous Database on Dedicated Exadata Infrastructure and Oracle Exadata Cloud Service (Oracle ExaCS). If you want to disable Entra ID external authentication with these products, see their product documentation.
To disable Entra ID from Oracle Autonomous Database Serverless, see *Using Oracle Autonomous Database Serverless*. The following procedure applies to all other platforms:

1. Log in to the Oracle Database instance as a user who has been granted the `ALTER SYSTEM` system privilege.
2. Set the identity provider parameters as follows:

   ```
   ALTER SYSTEM RESET IDENTITY_PROVIDER_CONFIG SCOPE=BOTH;
   ALTER SYSTEM RESET IDENTITY_PROVIDER_TYPE SCOPE=BOTH;
   ```

8.3 Mapping Oracle Database Schemas and Roles

Azure users will be mapped to one database schema and optionally to one or more database roles.

- Exclusively Mapping an Oracle Database Schema to a Microsoft Azure User
  You can exclusively map an Oracle Database schema to a Microsoft Azure user.
- Mapping a Shared Oracle Schema to an App Role
  In this mapping, an Oracle schema is mapped to an app role. Therefore, anyone who has that app role would get the same shared schema.
- Mapping an Oracle Database Global Role to an App Role
  Oracle Database global roles that are mapped to Entra ID app roles give Azure users and applications additional privileges and roles above those that they have been granted through their login schemas.

8.3.1 Exclusively Mapping an Oracle Database Schema to a Microsoft Azure User

You can exclusively map an Oracle Database schema to a Microsoft Azure user.

1. Log in to the Oracle Database instance as a user who has been granted the `CREATE USER` or `ALTER USER` system privilege.
2. Run the `CREATE USER` or `ALTER USER` statement with the `IDENTIFIED GLOBALLY AS` clause specifying the Azure user name.

   For example, to create a new database schema user named `peter_fitch` and map this user to an existing Azure user named `peter.fitch@example.com`:

   ```
   CREATE USER peter_fitch IDENTIFIED GLOBALLY AS
   'AZURE_USER=peter.fitch@example.com';
   ```
3. Grant the `CREATE SESSION` privilege to the user.

   ```
   GRANT CREATE SESSION TO peter_fitch;
   ```

8.3.2 Mapping a Shared Oracle Schema to an App Role

In this mapping, an Oracle schema is mapped to an app role. Therefore, anyone who has that app role would get the same shared schema.

1. Log in to the Oracle Database instance as a user who has the `CREATE USER` or `ALTER USER` system privilege.
2. Run the `CREATE USER` or `ALTER USER` statement with the `IDENTIFIED GLOBALLY AS` clause specifying the Azure application role name.

   For example, to create a new database global user account (schema) named `dba_azure` and map it to an existing Entra ID application role named `AZURE_DBA`:

   ```
   CREATE USER dba_azure IDENTIFIED GLOBALLY AS 'AZURE_ROLE=AZURE_DBA';
   ```

8.3.3 Mapping an Oracle Database Global Role to an App Role

Oracle Database global roles that are mapped to Entra ID app roles give Azure users and applications additional privileges and roles above those that they have been granted through their login schemas.

1. Log in to the Oracle Database instance as a user who has been granted the `CREATE ROLE` or `ALTER ROLE` system privilege
2. Run the `CREATE ROLE` or `ALTER ROLE` statement with the `IDENTIFIED GLOBALLY AS` clause specifying the name of the Entra ID application role.

   For example, to create a new database global role named `widget_sales_role` and map it to an existing Entra ID application role named `WidgetManagerGroup`:

   ```
   CREATE ROLE widget_sales_role IDENTIFIED GLOBALLY AS
   'AZURE_ROLE=WidgetManagerGroup';
   ```

8.4 Configuring Entra ID Client Connections to the Oracle Database

You can configure client connections to connect with the registered database.

- About Configuring Client Connections to Entra ID
  There are three different ways for an Oracle Database client to use an Entra ID `OAuth2` token to send to the database for access.
- Operational Flow for SQL*Plus Client Connection to Oracle Database Using Microsoft Entra ID OAuth2 Token
  The connection between the Azure user, Entra ID, and an Oracle database relies on the passing of the `OAuth2` token throughout these three components.
- Supported Client Drivers for Entra ID Connections
  Oracle Database supports several types of client drivers for Entra ID connections.
- Registering a Client with Entra ID Application Registration
  This type of registration is similar to registering Oracle Database with Entra ID app registration.
- Configuration of Clients to Work with Microsoft Entra ID Tokens
  Depending on the Oracle Database client, you can configure the client to either directly request the token from Entra ID or retrieve it from a file location.
- Examples of Retrieving Entra ID OAuth2 Tokens Outside an Oracle Database Client
  These examples show different ways that you can retrieve Entra ID `OAuth2` token separately from the database client if you are not using the database client to retrieve the tokens directly.
- Creating a Network Proxy for the Database to Connect with the Internet
  This network proxy will enable the Oracle database to reach the Entra ID endpoint.
- Using Centralized Entra ID Services for Net Naming and Secrets
  You can use the Azure app configuration and Azure Vault to centrally store net names and secrets.

8.4.1 About Configuring Client Connections to Entra ID

There are three different ways for an Oracle Database client to use an Entra ID `OAuth2` token to send to the database for access.

- Connect to Entra ID endpoint directly and retrieve the token for the user (interactive flow).
- Retrieve the token from a file location (all supported Entra ID flows).
- Pass the token to the client by using the client API (all supported Entra ID flows).

Oracle Database supports several Entra ID flows for different use cases. You should review the details of each flow in the Microsoft documentation. Each database client can support different flows with different versions. Details of these types are available in the JDBC, ODP.NET, and other platform-specific client documentation for the supported Entra ID flows for the client. This section focuses on the use of the OCI and Instant Clients, which are also called thick clients.

The types of available flows are as follows:

- The interactive flow (also known as the OAuth2 authz flow) is the primary flow used by human actors. This flow requires an environment that can open a browser so that the user to enter their Entra ID credentials.
- The device code flow is supported by some clients, but not currently with the OCI and Instant Clients. This type of flow is also for human actors but for environments that cannot open a browser.
- The managed identity flow (supported by some clients, but not the OCI and Instant Clients) is for applications that run on Azure compute nodes and have access to the managed identity for the node.
- The client credential flow is designed for applications, especially if they are not running in an Azure environment.
- The Resource Owner Password Credential (ROPC) flow is not recommended for production use.

When a user must access the database as a human actor, Oracle recommends that you configure the interactive flow and configure the database client to retrieve the token directly from Entra ID. An application will need to use the client credential flow. Commonly, the application will use a script that is run periodically to retrieve a token from Entra ID and place it into a file location for the database client to use. If the application can be modified to integrate with the Entra ID SDK, then it can alternatively use the SDK to retrieve the token and pass it to the client using the client API.

You should choose the client connection method that works best with your use case. This guide provides examples of connecting SQL*Plus with different methods of getting an Entra ID `OAuth2` access token. All Oracle Database release 19c clients can accept a token that is passed as a file or through the client API. The JDBC-thin, Instant Client, and ODP.net drivers also accept the token through the database client API from an application. Tools such as PowerShell or Azure CLI can retrieve the Entra ID `OAuth2` access token for use by the client driver. To retrieve an Entra ID token, the client must be registered through the Entra ID app (application) registration process. Registering the client is similar to registering the Oracle Database server with Entra ID using the app registration. Both the database and client must be registered with Entra ID.

The database must be registered so the client can get permission to get an access token for the database. The client must be registered so that Entra ID can recognize a trusted client is asking for an access token.

See the following Microsoft Azure articles for more information about connecting clients to Entra ID:

- Quickstart: Configure a client application to access a web API
- Choose the right Azure command-line tool
- Get Entra ID tokens by using the Microsoft Authentication Library
- Install the Azure CLI on Linux

Related Topics

- *Oracle Database JDBC Developer's Guide*
- *Oracle Data Provider for .NET Developer's Guide*

8.4.2 Operational Flow for SQL*Plus Client Connection to Oracle Database Using Microsoft Entra ID OAuth2 Token

The connection between the Azure user, Entra ID, and an Oracle database relies on the passing of the `OAuth2` token throughout these three components.

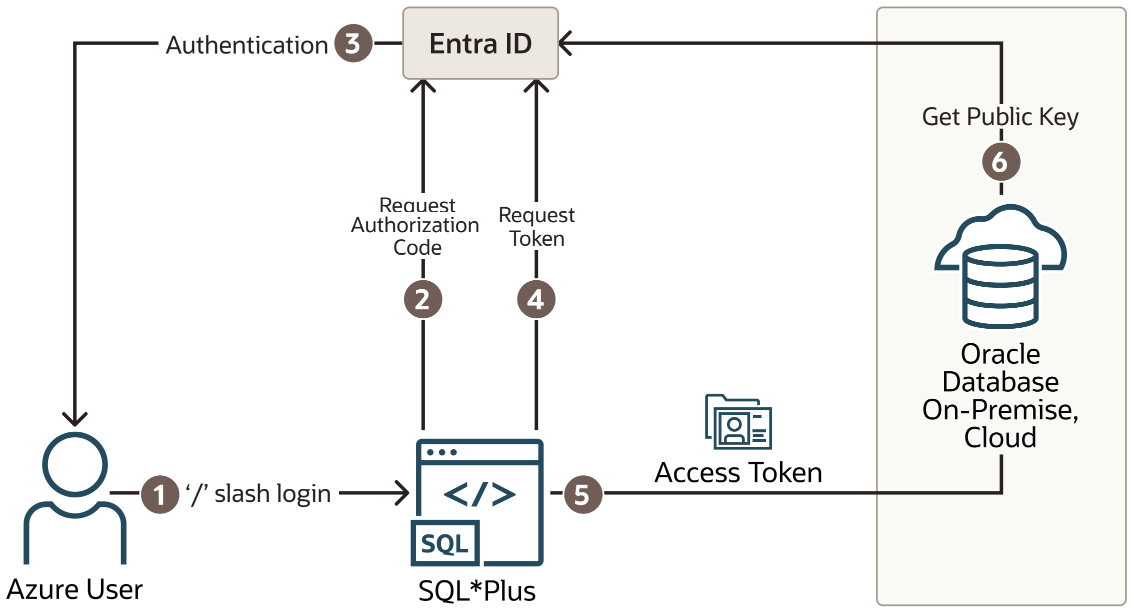
There are three ways for an Oracle Database client to send an Entra ID `OAuth2` token to an Oracle database.

- Through the Oracle Database client
- By specifying a file location
- Using the the Oracle Database client API

Using an Oracle Database Client to Send the Entra ID OAuth2 Token to the Oracle Database

The Oracle Database client can request an `OAuth2` token directly from the Entra ID endpoint. This method simplifies the required configuration. The following diagram shows the use of the interactive flow with a public client. The interactive flow is also called the `OAuth2` authorization flow. See the Microsoft identity platform and OAuth2.0 authorization code flow Microsoft article for detailed information about the authorization flow.

Figure 8-3   Entra ID OAuth2 Tokens Sent to the Oracle Database Using Client

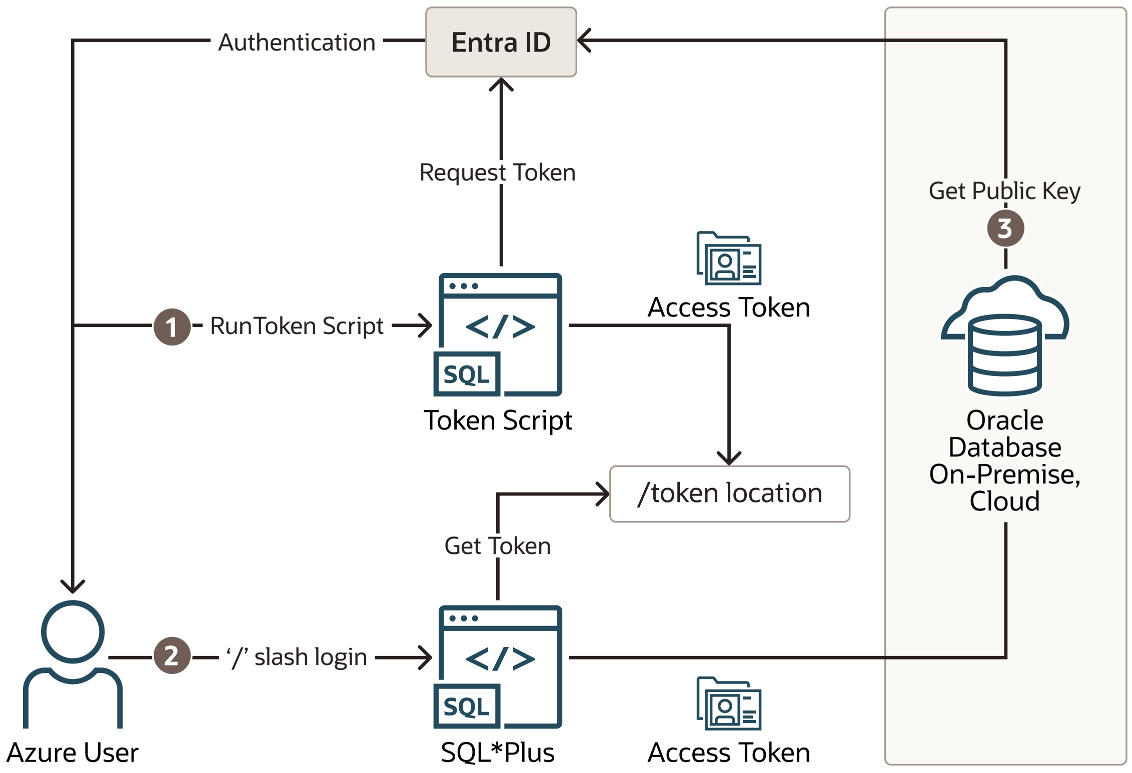


1. The user uses a `/` slash login to use the Azure SSO login. The connect string (or `sqlnet.ora`) includes all the parameters that are required for the Oracle Database client to get a token for the user.
2. The Oracle Database client connects with the Entra ID endpoint to request an authorization code.
3. If the user has not logged in with Entra ID, then a browser window opens and requests the user to enter their Azure SSO credentials.
4. The Oracle Database client requests an `OAuth2` access token using the authorization code.
5. When the Oracle Database client receives the `OAuth2` access token, it sends this token to the Oracle database.
6. The Oracle database verifies that the access token came from Entra ID (using the Entra ID public key) and then checks the token for additional claims. Next, the database finds the schema mapping (exclusive or shared) and creates the session. The database will also grant any global roles that the Azure user is also assigned to through an app role.

Specifying a File Location to Send the Entra ID OAuth2 Token to the Oracle Database

The following diagram illustrates how a file location can be used to send the Entra ID `OAuth2` token to an Oracle database.

Figure 8-4 Entra ID OAuth2 Tokens Sent to the Oracle Database Using File Location

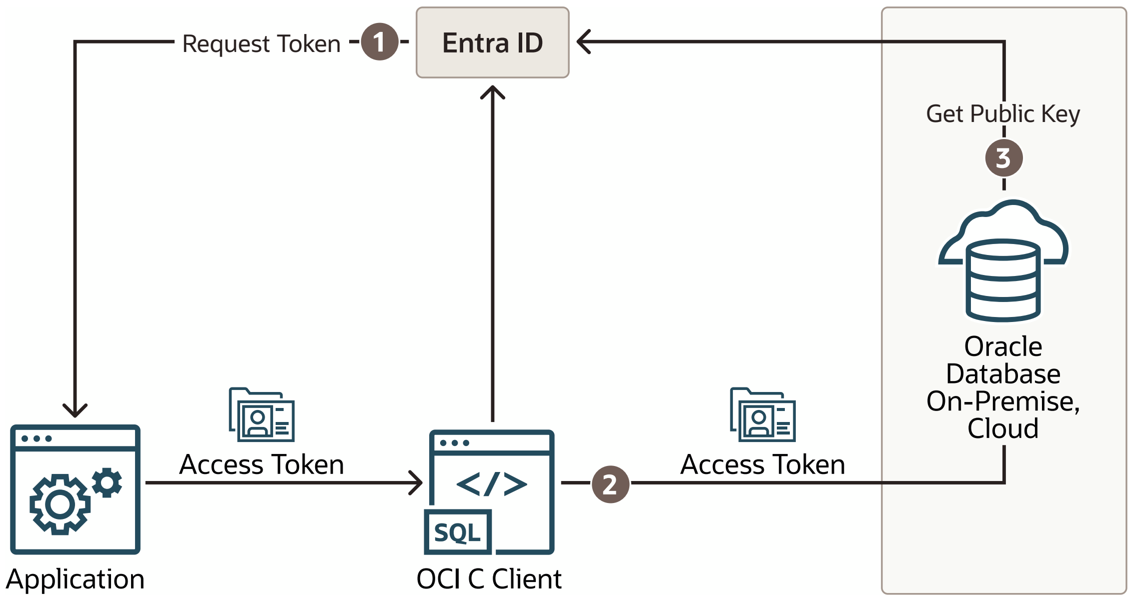


1. The Azure user requests an Entra ID access token for the database using a script and the returned token is written into a file called `token` at a file location. The Azure user may be requested to authenticate with Entra ID at this time.
2. The Azure user connects to the database using the `/` slash login. Either the `sqlnet.ora` or `tnsnames.ora` connection string tells the Oracle Instant Client that an Entra ID `OAuth2` token is needed and to retrieve it from a specified file location. The access token is then sent to the Oracle database.
3. The Oracle database verifies that the access token came from Entra ID (using the Entra ID public key) and then checks the token for additional claims. The database then finds the schema mapping (exclusive or shared) and creates the database session. The database will also grant any global roles that the Azure user is also assigned to through an app role.

Using the Oracle Database Client API to Send the Entra ID OAuth2 Token to the Oracle Database

The following diagram illustrates how the Oracle Database Client API can be used to send the Entra ID OAuth2 Token to the Oracle database.

Figure 8-5 Entra ID OAuth2 Tokens Sent to the Oracle Database Using the Client API



1. The application requests an Entra ID access token for the Oracle database using a script. The returned token is then sent to the database client using the client API. The token can represent the user (on-behalf-of token flow) or the application (client credential flow)
2. The Oracle Database client sends the access token to the Oracle database.
3. The Oracle database verifies that the access token came from Entra ID (using the Entra ID public key) and then checks the token for additional claims. The database finds the schema mapping (exclusive or shared) and creates the session. The database will also grant any global roles that the application or user is assigned to through an app role.

8.4.3 Supported Client Drivers for Entra ID Connections

Oracle Database supports several types of client drivers for Entra ID connections.

Oracle recommends that you use the latest quarterly patch for the supported versions because enhancements are added with the quarterly releases. In addition, some features will only exist in the Oracle Database 23ai version and will not be backported.

- **Thick clients (OCI C driver, Oracle Instant Client, Oracle Data Provider - Unmanaged (ODP.NET-Unmanaged), JDBC-thick, and others based on OCI C driver):** Oracle Database 19.16 (July 2022) and above, not supported with 21c, fully supported with Database 23ai
- **JDBC-thin:** Oracle Database 19.16 (July 2022), Oracle Database 21.8 (October 2022)
- **Oracle Data Provider (ODP.NET core, managed):** Oracle Database 19.16, Oracle Database 21.7
- **Python-thin:** 1.1.0+
- **Node.js-thin:** v6.3+

8.4.4 Registering a Client with Entra ID Application Registration

This type of registration is similar to registering Oracle Database with Entra ID app registration.

- Confidential and Public Client Registration
  You can register the database client with Entra ID as either confidential or public depending on your use case.
- Registering a Database Client App with Entra ID
  Creating the client app registration is similar to creating the Oracle Database instance with the Microsoft Entra ID tenancy.

8.4.4.1 Confidential and Public Client Registration

You can register the database client with Entra ID as either confidential or public depending on your use case.

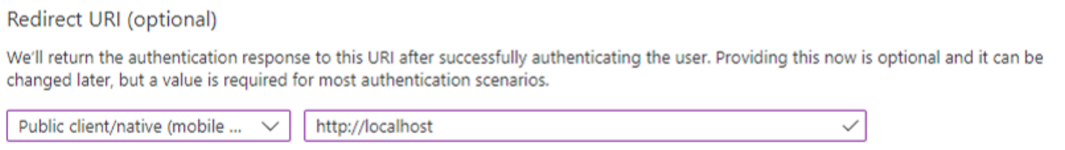
See the Microsoft Azure article Authentication flows and application scenarios for detailed information about authentication flows and application scenarios.

Registering a confidential client app requires that the client have a secret, in addition to the client ID. The confidential client app uses both the client ID and the secret when it makes Entra ID requests. However, in an enterprise, it is not practical for every SQL*Plus and SQLcl user to create a separate app registration with its own secret. In addition, a secret is no longer a secret when you start to share it within an organization. It is far better to just create a public client app. A public client app does not have a secret; it only has a client ID. All database tool users can use the public client ID when they connect to Entra ID to get an access token. The Azure user still needs to authenticate to Entra ID with their own user credential.

8.4.4.2 Registering a Database Client App with Entra ID

Creating the client app registration is similar to creating the Oracle Database instance with the Microsoft Entra ID tenancy.

1. Log in to the Azure portal as an administrator who has Microsoft Entra ID privileges to register applications.
2. In the Azure Active directory admin center page, from the left navigation bar, select **Microsoft Entra ID**.
3. In the MS - App registrations page, select **App registrations** from the left navigation bar.
4. Select **New registration**.
5. In the Register an application page, enter the following Oracle Database client registration information:
   - In the **Name** field, enter a name for the client app (for example, `DatabaseClientApplication`)..
   - Under Supported account types, select the account type that matches your use case.
     – **Accounts in this organizational directory only (*tenant_name* only - Single tenant)**
     – **Accounts in any organizational directory (Any Entra ID directory - Multitenant)**
     – **Accounts in any organizational directory (Any Entra ID directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)**
     – **Personal Microsoft accounts only**
6. Under Redirect URI (optional), configure the redirect URI for the client app.



Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Public client/native (mobile ... | http://localhost

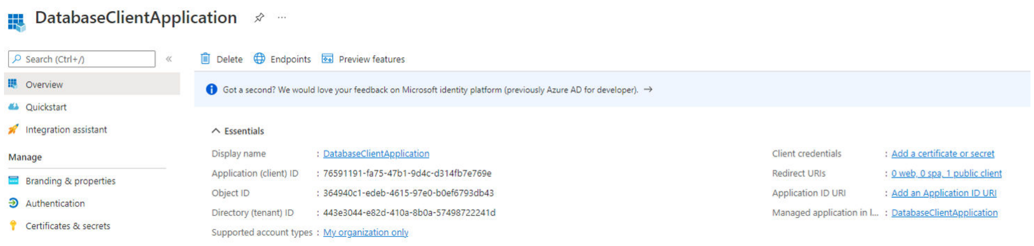
- Select **Public client/native (mobile & desktop)**, **Web**, or **Single-page application (SPA)**. Choose **Public client** if this client app will be used by multiple users such as database administrators who need to use SQL*Plus to access the Oracle Database instance.
- Add a redirect URI of `http://localhost`, unless you have another address to use. This redirect URI is needed for the authorization flow.

7. Click **Register**.

   At this stage, the database client has been registered with Entra ID. Next, you must add the new client to the list of authorized client apps for the Oracle Database instance.

8. To add the new client to this list of client apps, do the following:

   a. Make a note of the new client's Application (client) ID. This ID is in the Overview page for the app.



DatabaseClientApplication

Essentials

Display name	DatabaseClientApplication
Application (client) ID	76591191-fa75-47b1-9d4c-d314fb7e769e
Object ID	364940c1-edeb-4615-97e0-b0ef6793db43
Directory (tenant) ID	443e3044-e82d-410a-8b0a-57498722241d
Supported account types	My organization only
Client credentials	Add a certificate or secret
Redirect URIs	0 web, 0 spa, 1 public client
Application ID URI	Add an Application ID URI
Managed application in l...	DatabaseClientApplication

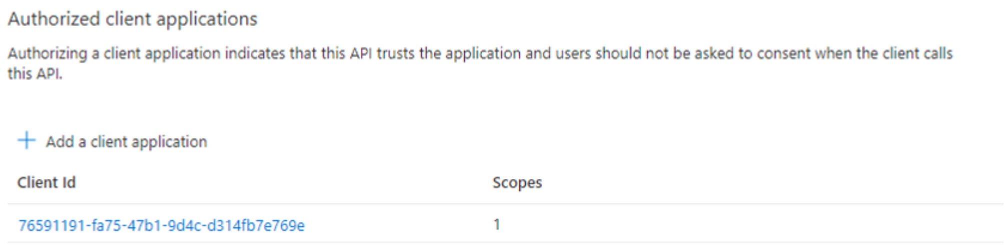
   b. On the App registrations page, open the app registration page for the database server by selecting it from the menu.

   c. On the left side, select **Expose an API**.

   d. Scroll down on the main page until you see **Authorized client applications**.

   e. Select **+** to add a client application.

   f. Copy the new client's Application (client) ID to the **Client Id** field.



Authorized client applications

Authorizing a client application indicates that this API trusts the application and users should not be asked to consent when the client calls this API.

+ Add a client application

Client Id	Scopes
76591191-fa75-47b1-9d4c-d314fb7e769e	1

   g. Click **Add application**.

Related Topics

- Quickstart: Register an application with the Microsoft identity platform

8.4.5 Configuration of Clients to Work with Microsoft Entra ID Tokens

Depending on the Oracle Database client, you can configure the client to either directly request the token from Entra ID or retrieve it from a file location.

- Configuring Clients to Work with Microsoft Entra ID Tokens
  There are different ways to configure your database client to work with Entra ID `OAuth2` access tokens.
- Enabling Clients to Directly Retrieve Entra ID Tokens
  You can set parameters to enable clients to directly retrieve Entra ID tokens on their own.
- Client Credential Flow
  The client credential flow allows on-premises applications and applications in non-Azure cloud environments to get an MS-EI OAuth2 token to connect to the Oracle Database.
- Enabling Clients to Retrieve Entra ID Tokens from a File Location
  If you choose to retrieve the Entra ID location from a file location when you use the `/` slash login, then you will need to configure your client.
- Using Azure App Configuration Store for Network Service Configuration Information
  You can store connect string and other network configuration information in Azure App Configuration Store.

8.4.5.1 Configuring Clients to Work with Microsoft Entra ID Tokens

There are different ways to configure your database client to work with Entra ID `OAuth2` access tokens.

Depending on your use case (flow), the database client can directly request the `OAuth2` token from the Entra ID endpoint. In other cases, a separate utility will need to be run to get the token and put it into a file location for use by the database client. An application can also use the Azure SDK to get a token and send it through the database client API. Refer to the database client specific documentation for using the client API and for client configuration information. Before you can request a token from Entra ID, you must perform the following configuration.

1. Ensure that you have an Azure user account.
2. Check with an Entra ID administrator or Oracle Database administrator for one of the following:
   - An application client ID that you can use to get Entra ID tokens. If you have Entra ID privileges to do so, then create your own client app registration, similar to registering the Oracle Database instance with an Entra ID tenancy.
   - You are mapped to a global schema in the database either directly or through an app role.
3. Ensure that you are using the latest release updates for the Oracle Database client releases 19c or 23ai and later.

   Entra ID integration is not supported with Oracle Database 21c.

A TLS connection is required between the database client and the database server to pass `OAuth2` tokens. You can use TLS (server authentication) or mTLS (client and server authentication). If your database client and platform support it, then you can simply use your system default certificate store when using TLS and not use a wallet. In addition to using TLS, you must specify either partial or full DN matching (`SSL_SERVER_DN_MATCH = ON`).

8.4.5.2 Enabling Clients to Directly Retrieve Entra ID Tokens

You can set parameters to enable clients to directly retrieve Entra ID tokens on their own.

Oracle Database clients differ by platform and version for what flows they support. The following table shows what each client can support.

Table 8-1 Parameters to Directly Retrieve Tokens

Database Clients	Passing Using Client API	Using File Location	Database Client Direct Support
Thick clients (OCI C driver, Instant Clients Along with platform specific drivers that use the thick client (for example, JDBC-thick, ODP.NET unmanaged, Python-thick)	Client versions 19.16+, not 21c, all 23ai Supported for all flows (interactive, client credential, OBO, ROPC)	Client versions 19.16+, not 21c, all 23ai Supported for all flows (interactive, client credential, OBO, ROPC)	Client version 23.4+ Interactive flow support only
JDBC-thin	Client versions 19.16+, 21.7+, all 23ai Supported for all flows (interactive, client credential, OBO, ROPC)	Client versions 19.16+, 21.7+, all 23ai Supported for all flows (interactive, client credential, OBO, ROPC)	Client version 23ai Supports the following flows (interactive, device code, client credential, managed identity, OBO, ROPC)
ODP.NET core, managed	Client versions 19.16+, 21.7+, all 23ai Supported for all flows (interactive, client credential, OBO, ROPC)	Client versions 19.16+, 21.7+, all 23ai Supported for all flows (interactive, client credential, OBO, ROPC)	Client version 23ai Supports the following flows (interactive, device code, client credential, managed identity, OBO, ROPC)
Python-thin	Not supported	Not supported	Not supported
Node.js	Not supported.	Not supported	Not supported

The connect string parameters are common across the database clients. Refer to each database client documentation (JDBC-thin, ODP.NET core, managed) for more specific information regarding this feature with those drivers. The following information is specifically for the OCI thick client/Instant client. However, the information about connect string parameters will remain consistent across the drivers.

To enable this feature in the client to get a token directly from Entra ID for a supported flow, you must set the following parameters in either the client's `sqlnet.ora` file or in a connect string. The connect string takes precedence over `sqlnet.ora`.

In order for the database client to retrieve the Entra ID OAuth2 token, the database client must be able to connect with the Entra ID endpoint. If you are working behind a firewall, you may need to set a proxy to reach the internet. See the Troubleshooting Microsoft Entra ID Connections section if you're not sure if you are able to connect to the internet.

Table 8-2 Parameters to Directly Retrieve Tokens

Parameter	Description
`TOKEN_AUTH`	Sets the token authentication. This parameter is mandatory when you are asking the database client to get the database token or pick it up from a file location. This parameter is not required when you are passing the token through the client API. Enter one of the following values: • `AZURE_INTERACTIVE` tells the driver that it must use the Entra `OAuth2` interactive (`OAuth2` authorization) flow to get an access token for the database. This configures the database client to get the token directly from Entra ID without having to use an external script. This is for human users who are logging into tools such as SQLcl and can also open a browser window in their environment to authenticate • `AZURE_DEVICE_CODE` signals the database driver to follow the device code flow for requesting an Entra ID access token. This is also for human users, when their environment cannot open a browser: a command line only environment. A device code and Entra ID login URL is written out to the standard output of the tool and the user logs into Entra ID on their cellphone or laptop, and then enters the device code. Users are authenticated through a separate channel and then allowed to continue access the database if the authentication is successful. • `AZURE_MANAGED_IDENTITY` enables the driver to authenticate as an identity that has been assigned to the host system. The host system must be a resource which is managed by Entra ID, such as a virtual machine. • `AZURE_SERVICE_PRINCIPAL` enables the driver to authenticate using a secret or certificate of the registered application.
`CLIENT_ID`	The unique application (client) ID assigned to your app by Entra ID when the app was registered. This app is your database client that will request to get an access token for the database for the user. This is not the client ID for the database server.
`AZURE_DB_APP_ID_URI`	The application ID URI is a URI that uniquely identifies the database in your Entra ID. You get this value from the overview screen of your database Entra ID app registration.
`TENANT_ID`	Specifies the Azure tenancy ID of the database.
`REDIRECT_URI`	Optional parameter for setting the port number for the HTTP server. This URL obtains the authorization code from the Entra authentication endpoint and determines which port to use to receive the authorization code. If `REDIRECT_URI` is not set, then the default is `http://localhost:8400`. If `8400` is already in use, then Oracle Database tries the next available number after `8400`, ranging from `8400` to `90000`. If you explicitly specify an unavailable port number, then the connection fails.

See *Oracle Database Net Services Reference* for specific information about each parameter. The following is an example of specifying use of interactive flow to get a token.

```
conn /@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=example.us-
phoenix-1.oraclecloud.com)(PORT=6010))
(SECURITY=(SSL_SERVER_DN_MATCH=YES)
(AZURE_DB_APP_ID_URI=https://oracledevelopment.onmicrosoft.com/
11111111-11a1-1a11-111a-a11a11111111)
(TENANT_ID=1a111aa1-a1a1-1a11-a1a1-a11aaaaa1111)
(CLIENT_ID=aa11a111-111a-1a11-1aa1-1aa1a1aa1111)
(TOKEN_AUTH=AZURE_INTERACTIVE))
(CONNECT_DATA=(SERVICE_NAME=cdb1_pdb3.regress.rdbms.dev.us.oracle.com)))
```

8.4.5.3 Client Credential Flow

The client credential flow allows on-premises applications and applications in non-Azure cloud environments to get an MS-EI OAuth2 token to connect to the Oracle Database.

The client credential flow is supported using the token file passing method and through the OCI-C client API since Oracle Database 19c (not Oracle Database 21c). The Oracle Database 23ai OCI-C client also supports getting the MS-EI OAuth2 token directly from the MS-EI endpoint without requiring a script to initially get the token. In order to get the token for the Client Credential flow, the client will need a client ID and a client secret from MS-EI when the application is registered using MS-EI app registration. This is different than when setting up a public client for DBAs to connect to the database using the interactive flow. A public client doesn't need a client secret since the human user will be signing into Azure using their credentials. In the client credential flow, the application must have a client secret to authenticate to MS-EI and get a token. Since the client secret is sensitive, we recommend using an Oracle Wallet to store the client ID and client secret.

There are a few differences between the client used for interactive flow (for human users) and the client credential flow (for applications). In the interactive flow, users and groups are mapped to the database app roles in MS-EI enterprise applications. With the client credential flow, the client application can only be mapped to a database app role directly.

Follow the below steps to configure client credential flow between the Oracle Database and Microsoft Entra ID.

Register the Oracle Database with Microsoft Entra ID

Follow the below Microsoft documentation to create an app registration for the application client:

1. Register an application
2. Expose scopes in web API registrations
3. Grant scopes permission to web API

Ensure that you:

- You are an owner of the database app
- Create an app role in the database app for the application
- Create a client secret for the application client app
- Create an API permission to connect to the database app and grant consent to it

Create an application role mapping in Oracle Database

In the previous step you created a new application role. You now have to create a schema mapping in the database and grant the appropriate roles and privileges to the schema for the new role.

1. Log in to the Oracle Database instance as a user who has the `CREATE USER` or `ALTER USER` system privilege.
2. Run the `CREATE USER` or `ALTER USER` statement with the `IDENTIFIED GLOBALLY AS` clause specifying the Azure application role name.
   For example, to create a new database global user account (schema) named `hr_app` and map it to an existing Entra ID application role named `hrapp`:

   ```
   CREATE USER hr_app IDENTIFIED GLOBALLY AS 'AZURE_ROLE=HRAPP';
   ```

Configure the Oracle Call Interface (OCI-C) client for client credential flow

You must define parameters for the OCI-C client to get an OAuth2 token for the application credential flow.

The following parameters can be defined either in the `sqlnet.ora` file or in the `tnsnames.ora` file. Parameters in the `tnsnames.ora` file will have precedence over the same parameter in `sqlnet.ora`.

Parameter	Value	Usage Notes
`TOKEN_AUTH`	`AZURE_SERVICE_PRINCIPAL`	This tells the OCI-C driver to follow the client credential flow
`TENANT_ID`	The tenancy ID for the application app registration	This may or may not be the same tenancy for the database app registration
`AZURE_DB_APP_ID_URI`	This is from the database app registration	This was configured when creating the database app registration
`CLIENT_ID`	This is the client ID for the application app registration	This is not the client id from the database app registration
`AZURE_CREDENTIALS`	This is the location of the wallet holding the client secret	

Here is a sample connect string:

```
conn2=
    (DESCRIPTION=
        (ADDRESS=
            (PROTOCOL=tcps)
            (HOST=phoenix99201)
             (PORT=6679)
        )
        (SECURITY=
            (SSL_SERVER_CERT_DN="C=US,O=OracleCorporation,CN=sslserver3")
            (TOKEN_AUTH=AZURE_SERVICE_PRINCIPAL)
            (TENANT_ID=aaaaaaaa-bbbb-cccc-eeee)
            (AZURE_DB_APP_ID_URI=https://examplecorp.onmicrosoft.com/aaaa-
bbbb-cccc-dddd)
            (CLIENT_ID=aaaa-bbbb-cccc-dddd-eeee)
            (AZURE_CREDENTIALS=/scratch/secret)
        )
        (CONNECT_DATA=
            (SERVICE_NAME=database.examplecorp.com)
        )
    )
```

The first four parameter values can be in the connect string or `sqlnet.ora` file. But the client secret needs to be in the wallet with the location identified by `AZURE_CREDENTIALS`.

The client secret is paired with the client ID in the wallet. The database driver will look up the client secret in the wallet using the `CLIENT_ID` parameter. The client ID is a case sensitive parameter so the case for the client ID in the wallet must match the case of the client ID in the connect string or `sqlnet.ora`.

When you display the wallet content, you will find something similar to:

```
oracle.security.azure.credential.<client id> = <client secret>
```

The `CLIENT_ID` and `CLIENT_SECRET` is obfuscated/encrypted in the wallet and only user with right privilege can open/view the value.

Create the wallet for storing the client secret

Use `orapki` to create the wallet and store the client secret.

1. Create a wallet and set the wallet password:

   ```
   orapki wallet create -wallet . -auto_login_only
   ```

2. Create an entry with the client id and client secret:

   ```
   orapki secretstore create_entry -wallet . -alias
   oracle.security.azure.credential.<CLIENT_ID> -secret <CLIENT_SECRET>
   ```

   Note:

   The `CLIENT_ID` value is case sensitive and must match the case of the `CLIENT_ID` vale in the connect string or `sqlnet.ora` file.

- Display the entry:

  ```
  orapki wallet display -wallet .
  ```

- Show a specific entry:

  ```
  orapki secretstore view_entry -wallet . -alias
  oracle.security.azure.credential.<CLIENT_ID>
  ```

- Modify the entry:

  ```
  orapki secretstore modify_entry -wallet . -alias
  oracle.security.azure.credential.<CLIENT_ID> -secret <CLIENT_SECRET>
  ```

- Delete the entry:

  ```
  orapki secretstore delete_entry -wallet . -alias
  oracle.security.azure.credential.<CLIENT_ID>
  ```

Related Topics

- orapki Utility Commands Summary
  The `orapki` commands perform a variety of wallet, certificate revocation lists (CRL), and certificate management tasks.

8.4.5.4 Enabling Clients to Retrieve Entra ID Tokens from a File Location

If you choose to retrieve the Entra ID location from a file location when you use the `/` slash login, then you will need to configure your client.

You can configure the Entra ID file location in either the `sqlnet.ora` file or the `tnsnames.ora` file.

- On the client, set or check the following parameters in the `tnsnames.ora` connect string or in the `sqlnet.ora` file:
  - `SSL_SERVER_DN_MATCH`: Ensure that this parameter is set to `ON` so that DN matching is enabled.
  - `TOKEN_AUTH`: Set this parameter to `OAUTH`.
  - `TOKEN_LOCATION`: Set this parameter to the file location of the token. There is no default location for the token. If the token is named `token`, then you only need to specify the file directory (for example, `/test/oracle/aad-token`). If the token name is different from `token` (for example, `azure.token`), then you must include this name in the path (for example, `/test/oracle/aad-token/azure.token`).

The parameter values in the `tnsnames.ora` connect string take precedence over the `sqlnet.ora` settings for that connection. The following code is an example of a `tnsnames.ora` entry. In this case, `SSL_SERVER_DN_MATCH` is specified in `sqlnet.ora` and will not appear in the connect string:

```
(description=
  (retry_count=20)(retry_delay=3)
  (address=(protocol=tcps)(port=1522)
  (host=example.us-phoenix-1.oraclecloud.com))

(connect_data=(service_name=aaabbbccc_exampledb_high.example.oraclecloud.com))
  (security=(ssl_server_cert_dn="CN=example.uscom-east-1.oraclecloud.com,
     OU=Oracle BMCS US, O=Example Corporation,
     L=Redwood City, ST=California, C=US")
  (TOKEN_AUTH=OAUTH)(TOKEN_LOCATION="/oracle/tokens/aad-token"))
```

After the connect string is updated with these parameters, the Azure user can log in to the Oracle Database instance by first running the external utility to get the token and then running the following command to start SQL*Plus. You can include the connect descriptor itself or use the name of the descriptor from the `tnsnames.ora` file.

```
connect /@exampledb_high
```

The database client is already configured to get an Azure `OAuth2` token because `TOKEN_AUTH` has already been set, either through the connect string or the `sqlnet.ora` file. The database client gets the `OAuth2` token and then sends the token to the Oracle Database instance.

8.4.5.5 Using Azure App Configuration Store for Network Service Configuration Information

You can store connect string and other network configuration information in Azure App Configuration Store.

See Azure App Configuration Store in the *Oracle Database Net Services Administrator's Guide* for more information.

8.4.6 Examples of Retrieving Entra ID OAuth2 Tokens Outside an Oracle Database Client

These examples show different ways that you can retrieve Entra ID `OAuth2` token separately from the database client if you are not using the database client to retrieve the tokens directly.

- About Examples of Retrieving Microsoft Entra ID OAuth2 Tokens Outside of an Oracle Database Client
  Oracle Database clients have differing abilities in directly retrieving an Entra ID `OAuth2` token.
- Example: Requesting a Token Using a Python Script for the Interactive (Authorization) Flow
  The interactive (authorization) flow is the most common for human users to access the database.
- Example: Requesting a Token Using Azure CLI for the Interactive (Authorization) Flow
  This example shows how to use the Azure CLI to retrieve an access token and then write the token to a file.
- Requesting a Token Using the Azure CLI for the Client Credential Flow
  The client credential flow is used for applications that need to use an Entra ID `OAuth2` token to access the database.

8.4.6.1 About Examples of Retrieving Microsoft Entra ID OAuth2 Tokens Outside of an Oracle Database Client

Oracle Database clients have differing abilities in directly retrieving an Entra ID `OAuth2` token.

Review the specific client documentation for configuring the database client to retrieve tokens for different flows. The other two ways to work with Entra ID tokens are as follows:

- Passing tokens by using the client API
- Passing tokens through the file system

Review the client documentation on using the API. A utility or script is used to request a token from Entra ID and store it in a file location for the database client to pick up. Using a script or utility to request and store the token is outside Oracle Database. There are many examples available from Microsoft and others on the internet on how to get an Entra ID token. (Also search for Azure AD `OAuth2` token). The samples in this section are just some examples and not supported by Oracle.

8.4.6.2 Example: Requesting a Token Using a Python Script for the Interactive (Authorization) Flow

The interactive (authorization) flow is the most common for human users to access the database.

If the user has not already logged into their Azure account, they will be prompted with a web page to enter their Azure credentials. They will also need to complete any multi-factor authentication required by the organization before they retrieve the database `OAuth2` access token. This example with the Microsoft Authentication Library (MSAL) is in Python and can be run on a variety of platforms such as Windows PowerShell and Linux. Because the authorization flow requires two round trips to Azure AD, it is best handled using the MSAL. See the Microsoft article Get Entra ID tokens by using the Microsoft Authentication Library for how to use a python script with MSAL. These instructions are for the Databricks service, but the scope is changed to the database App ID URI and scope instead of the Databricks scope.

1. Bypass the steps to set up the client app registration, since you have already accomplished that step except make sure you add a Redirect URI (`http://localhost`) for your client app registration.
2. Go directly to **Get Entra ID tokens by using the MSAL Python library**.

   You will need the Directory (tenant) ID, Client ID for the public app client, and the database App ID URI and scope. You will see a code section for **scopes** with directions to not modify this variable. Because this python code was written for Databricks scope, you will need to change this scope variable to the scope of your database. For example:

   ```
   scopes = ['https://example.com/1111aa1a-a1aa-1a11-11aa-1a1a11aa1111/
   session:connect']
   ```

3. Modify the code to write the token to a file location.

   Use the following example code and append it to the print statements at the end. Note the extra lines to back up and restore the original `stdout`.

   ```
   stdout_backup = sys.stdout
   with open('token', 'w') as token_file:
       sys.stdout = token_file
       print(acquire_tokens_result['access_token'])

   sys.stdout = stdout_backup
   ```

8.4.6.3 Example: Requesting a Token Using Azure CLI for the Interactive (Authorization) Flow

This example shows how to use the Azure CLI to retrieve an access token and then write the token to a file.

See the Microsoft article Install the Azure CLI on Linux for information about installing the Azure CLI.

1. Log in to your Azure tenancy.

   ```
   $ az login
   ```

2. Get an access token and assign it to the token variable using the following syntax:

   ```
   token=$(az account get-access-token --resource=database_app_id_uri --query
   accessToken --output tsv)
   ```

   For example:

   ```
   token=$(az account get-access-token --resource=https://example.com/
   1111aa1a-a1aa-1a11-11aa-1a1a11aa1111 --query accessToken --output tsv)
   ```

   If you get an Azure CLI error saying that the client app ID does not have permission to access the database resource, then copy the client app ID from the error message and add it to the list of authorized client applications for the database resource. (Go to the database app registration in Entra ID, click **Expose an API** and then **Add a client application**).
3. Write the token to a file.

   ```
   $ echo "$token" >> token
   ```

8.4.6.4 Requesting a Token Using the Azure CLI for the Client Credential Flow

The client credential flow is used for applications that need to use an Entra ID `OAuth2` token to access the database.

Because applications are "headless" and do not have a user to authenticate interactively with the Azure portal, the interactive flow cannot be used with applications. The client credential flow is designed for applications. In these flows, the application app registration requires a client secret along with the client ID. These are used to retrieve the Entra ID `OAuth2` database access token.

After the script gets the token, this token will need to be written to a file (as shown in the examples in this section) so that the database client can access it. Microsoft provides several examples for a service principal to request a token. See then Microsoft article Get Microsoft Entra ID (formerly Azure Active Directory) tokens for service principals.

8.4.7 Creating a Network Proxy for the Database to Connect with the Internet

This network proxy will enable the Oracle database to reach the Entra ID endpoint.

- About Creating a Network Proxy for the Database to Connect with the Internet
  The Oracle database must connect to Entra ID endpoints and it may require network configuration and default trust store access.
- Testing the Accessibility of the Entra ID Endpoint
  You must ensure that your Oracle Database can access the Entra ID endpoint.
- Creating the Network Proxy for the Default Oracle Database Environment
  To create the network proxy, you must set environment variables and then restart the listener.
- Creating the Network Proxy for an Oracle Real Application Clusters Environment
  To create the network proxy, you must set an environment variable and then restart the database.

- Creating the Network Proxy in the Windows Registry Editor
  To create the network proxy in a Windows environment, you must update the Registry Editor (`regedit`).

8.4.7.1 About Creating a Network Proxy for the Database to Connect with the Internet

The Oracle database must connect to Entra ID endpoints and it may require network configuration and default trust store access.

You can configure the database when HTTP network proxy is in place in an enterprise, for a default Oracle Database environment and for an Oracle Real Applications Clusters environment. The database establishes a Transport Layer Security (TLS) link to Entra ID, so it also needs access to the default trust store on the database server. To enable this, ensure that the database server has access to the system default certificate store.

Related Topics

- Certificate Store Location for System Wallets
  System wallets are located in the certificate store location.

8.4.7.2 Testing the Accessibility of the Entra ID Endpoint

You must ensure that your Oracle Database can access the Entra ID endpoint.

If your database client is configured to get Microsoft Entra ID OAuth2 tokens, then the database client must be able to access the Entra ID endpoint. Run the following command to check if you have internet access:

```
curl  https://login.windows.net/common/discovery/keys
```

A status code of 200 indicates success.

Check with your IT help desk for the proxy information if you weren't successful running this command.

For an Oracle database to accept Entra ID `OAuth2` tokens, the database must request the public key from the Microsoft Entra ID endpoint.

- Run the following test to determine if the database can connect with the Microsoft Entra ID endpoint:

  ```
  SET SERVEROUTPUT ON SIZE 40000
  DECLARE
    req UTL_HTTP.REQ;
    resp UTL_HTTP.RESP;
  BEGIN
    UTL_HTTP.SET_WALLET(path => 'system:');
    req := UTL_HTTP.BEGIN_REQUEST('https://login.windows.net/common/
  discovery/keys');
    resp := UTL_HTTP.GET_RESPONSE(req);
    DBMS_OUTPUT.PUT_LINE('HTTP response status code: ' || resp.status_code);
    UTL_HTTP.END_RESPONSE(resp);
  END;
  /
  ```

  If this test is successful, then a `PL/SQL procedure successfully completed` message appears.

If the following messages appear, then it means that a database network access control list (ACL) policy blocked your test and you will need to temporarily set an access control list policy to allow you to test this:

```
ORA-29273: HTTP request failed
ORA-24247: network access denied by access control list (ACL)
```

1. Set the ACL as follows:

   ```
   BEGIN
   DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
     host => '*',
     ace  =>  xs$ace_type(privilege_list => xs$name_list('connect'),
                          principal_name => 'username_placeholder',
                          principal_type => xs_acl.ptype_db));
   END;
   /
   ```

   Replace *username_placeholder* with the user name of the database user who is running the test. For example:

   ```
   BEGIN
   DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
     host => '*',
     ace  =>  xs$ace_type(privilege_list => xs$name_list('connect'),
                          principal_name => 'DBA_DEBRA',
                          principal_type => xs_acl.ptype_db));
   END;
   /
   ```

2. Try running the test again.
3. Remove the ACL, because you now no longer need it. For example, assuming your user name is `dba_debra`:

   ```
   BEGIN
   DBMS_NETWORK_ACL_ADMIN.REMOVE_HOST_ACE(
     host => '*',
     ace  =>  xs$ace_type(privilege_list => xs$name_list('connect'),
                          principal_name => 'DBA_DEBRA',
                          principal_type => xs_acl.ptype_db));
   END;
   /
   ```

If the database cannot connect with the Microsoft Entra ID endpoint, even after you set the ACL policy, you will most likely need to set the `HTTP_PROXY` package for your database. Review the topics listed in Related Topics, depending if you are using a default Oracle Database environment or an Oracle Real Application Clusters RAC environment. Your network administrator should be able to tell you what the correct `HTTP_PROXY` setting should be.

Related Topics

- Creating the Network Proxy for the Default Oracle Database Environment
  To create the network proxy, you must set environment variables and then restart the listener.

- Creating the Network Proxy for an Oracle Real Application Clusters Environment
  To create the network proxy, you must set an environment variable and then restart the database.

8.4.7.3 Creating the Network Proxy for the Default Oracle Database Environment

To create the network proxy, you must set environment variables and then restart the listener.

You do not need to restart the database.

1. In the server where the Oracle database is installed, set the `http_proxy` environment variable.

   For example:

   ```
   export http_proxy=http://www-proxy-example.com:80/
   ```

2. Restart the listener.

   ```
   lsnrctl stop
   lsnrctl start
   ```

> **Note:**
>
> The `http_proxy` environment variable must be set. If the `https_proxy` environment variable is set, but not the `http_proxy` variable, then set the `http_proxy` environment variable to the same value set for the `https_proxy` environment variable.

8.4.7.4 Creating the Network Proxy for an Oracle Real Application Clusters Environment

To create the network proxy, you must set an environment variable and then restart the database.

1. In the server where the Oracle database is installed, set the `http_proxy` environment variable.

   Use this syntax to set the network proxies. the proxy command that you enter must have `http://` preceding the proxy name and must have the port number at the end of the proxy:

   ```
   http_proxy=http://....:80/
   ```

   For example:

   ```
   srvctl setenv database -db db_name -env "http_proxy=http://www-
   proxy.example.com:80/"
   ```

2. Stop the database.

   ```
   $srvctl stop database -db db_name
   ```

3. Display the environment variable values to ensure that they are correctly set.

   ```
   $ srvctl getenv database -db db_name
   ```

   Output similar to the following should appear:

   ```
   db_name:
   http_proxy=http://www-proxy.example.com:80/
   https_proxy=http://www-proxy.example.com:80/
   ```

4. Restart the database.

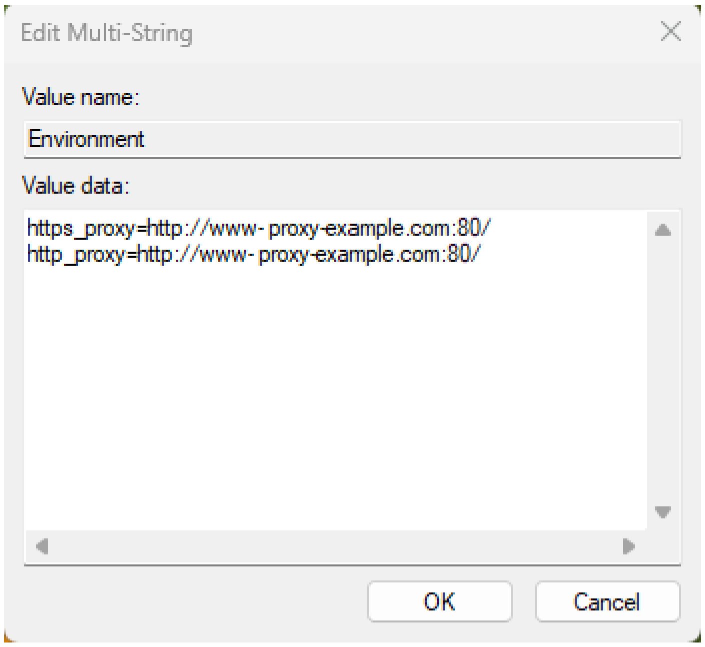
   ```
   $ srvctl start database -db db_name
   ```

8.4.7.5 Creating the Network Proxy in the Windows Registry Editor

To create the network proxy in a Windows environment, you must update the Registry Editor (`regedit`).

1. Start the Registry Editor (`regedit`).
2. Locate the `\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\OracleService`*`version`* key.
3. Select this key, and then in the right panel, locate **Environment**.
4. Edit **Environment** to add a new multi-string value to it.

   The following example uses the domain of `example.com`:



Edit Multi-String

Value name:

Environment

Value data:

```
https_proxy=http://www- proxy-example.com:80/
http_proxy=http://www- proxy-example.com:80/
```

5. Click **OK**.
6. Restart the database server.

For example:

```
net start Oracle_service_name
sqlplus "/as sysdba"
startup;
```

7. Open the PDBs.

```
ALTER PLUGGABLE DATABASE ALL OPEN;
```

8.4.8 Using Centralized Entra ID Services for Net Naming and Secrets

You can use the Azure app configuration and Azure Vault to centrally store net names and secrets.

This functionality is currently supported with the JDBC-thin and .NET-thin drivers.

See the following guides:

- *Oracle Database Net Services Administrator's Guide*
- *Oracle Database Net Services Reference*

8.5 Configuring Microsoft Entra ID Proxy Authentication

Proxy authentication allows an Azure user to proxy to a database schema for tasks such as application maintenance.

- About Configuring Microsoft Entra ID Proxy Authentication
  Azure users can connect to Oracle Autonomous Database by using proxy authentication.
- Configuring Proxy Authentication for the Azure User
  To configure proxy authentication for an Azure user, this user must already have a mapping to a global schema (exclusive or shared mapping). A separate database schema for the Azure user to proxy to must also be available.
- Validating the Azure User Proxy Authentication
  You can validate the Azure user proxy configuration for token authentication.

8.5.1 About Configuring Microsoft Entra ID Proxy Authentication

Azure users can connect to Oracle Autonomous Database by using proxy authentication.

Proxy authentication is typically used to authenticate the real user and then authorize them to use a database schema with the schema privileges and roles in order to manage an application. Alternatives such as sharing the application schema password are considered insecure and unable to audit which actual user performed an action.

A use case can be in an environment in which a named Azure user who is an application database administrator can authenticate by using their credentials and then proxy to a database schema user (for example, `hrapp`). This authentication enables the Entra ID administrator to use the `hrapp` privileges and roles as user `hrapp` in order to perform application maintenance, yet still use their Entra ID credentials for authentication. An application database administrator can sign in to the database and then proxy to an application schema to manage this schema.

8.5.2 Configuring Proxy Authentication for the Azure User

To configure proxy authentication for an Azure user, this user must already have a mapping to a global schema (exclusive or shared mapping). A separate database schema for the Azure user to proxy to must also be available.

After you ensure that you have this type of user, alter the database user to allow the Azure user to proxy to it.

1. Log in to the Autonomous Database instance as a user who has the `ALTER USER` system privileges.
2. Grant permission for the Azure user to proxy to the local database user account.

   An Azure user cannot be referenced in the command so the proxy must be created between the database global user (mapped to the Azure user) and the target database user.

   In the following example, `hrapp` is the database schema to proxy to, and `peterfitch_schema` is the database global user exclusively mapped to user `peterfitch`.

   ```
   ALTER USER hrapp GRANT CONNECT THROUGH peterfitch_schema;
   ```

At this stage, the Azure user can log in to the database instance using the proxy. For example:

```
CONNECT [hrapp]/@connect_string
```

8.5.3 Validating the Azure User Proxy Authentication

You can validate the Azure user proxy configuration for token authentication.

1. Log in to the Oracle Autonomous Database instance as a user who has the `CREATE USER` and `ALTER USER` system privileges.
2. Connect as the Azure user and run the `SHOW USER` and `SELECT SYS_CONTEXT` commands.

   For example, suppose you want to check the proxy authentication of the Azure user `peterfitch` when they proxy to database user `hrapp`:

   ```
   CONNECT [hrapp]/@connect_string
   SHOW USER;
   --The output should be USER is "HRAPP "
   SELECT SYS_CONTEXT('USERENV','AUTHENTICATION_METHOD') FROM DUAL;
   --The output should be "TOKEN_GLOBAL"
   SELECT SYS_CONTEXT('USERENV','PROXY_USER') FROM DUAL;
   --The output should be "PETERFITCH_SCHEMA"
   SELECT SYS_CONTEXT('USERENV','CURRENT_USER') FROM DUAL;
   --The output should be "HRAPP"
   ```

8.6 Configuring Microsoft Power BI Single-Sign On

Users have an option of a simpler configuration if only Power BI users will connect to the Oracle Database.

- About Configuring Microsoft Power BI Single-Sign On
  Users of the Microsoft Power BI data visualization tool frequently also use Microsoft Entra ID (MSEI). These users want to use their MSEI Single Sign-On (SSO) credentials to access their Oracle data sources seamlessly.
- Configuring the Oracle Database
  Configure the Oracle database to accept access tokens from Microsoft Power BI.
- Authorizing the User
  The Power BI Azure AD user must be authorized to the database.
- Connecting Power BI to Oracle Database using Microsoft Entra ID
  Once the database has been configured, you will need to configure Power BI Desktop or service.

8.6.1 About Configuring Microsoft Power BI Single-Sign On

Users of the Microsoft Power BI data visualization tool frequently also use Microsoft Entra ID (MSEI). These users want to use their MSEI Single Sign-On (SSO) credentials to access their Oracle data sources seamlessly.

Previously, Power BI users either had to access the Oracle Database using the database local username and password or had to migrate data from the Oracle Database to a different database if the security teams demanded centralized access management.

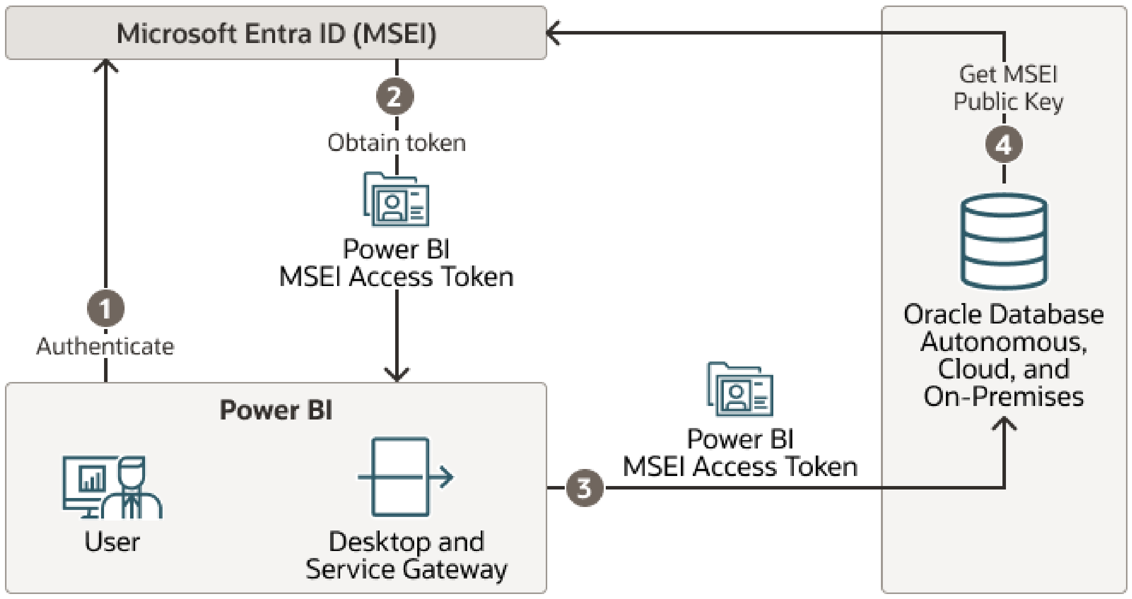
By using MSEI SSO to access Oracle data sources, security is improved since the users are centrally managed and Azure AD tokens are used instead of password credentials. Ease of use for DBAs is also improved since data can remain in the Oracle Database and not have to be migrated. Users also benefit since they can use their SSO to access their source database and not have to remember and continuously rotate their database password credentials.

Configuring Microsoft Power BI SSO is supported with:

- Oracle Database server 23ai (on-premises and cloud)
- Oracle Database server 19c (19.20 and above, on-premises and cloud)
- Any database client that supports MSEI tokens
  See Supported Client Drivers for Entra ID Connections for more information.

The following diagram illustrates how MSEI SSO can be used to access the Oracle database used as a source for Microsoft Power BI.

Figure 8-6 Microsoft Entra ID Access Tokens Sent to the Oracle Database For Power BI



1. Power BI user authenticates themselves with MSEI
2. Power BI gets the user’s access token for the database when a connection is opened to the database
3. Power BI sends the MSEI Power BI access token to the Oracle Database
4. The Oracle Database caches the MSEI public key to validate the MSEI Power BI token

Related Topics

- Power BI Desktop: Connect to Oracle Database
- Power BI Serve: Connect to Oracle Database
- Configure the Oracle Database for Power BI (video)
- Configure Power BI Desktop (video)
- Microsoft Power BI documentation

8.6.2 Configuring the Oracle Database

Configure the Oracle database to accept access tokens from Microsoft Power BI.

Prerequisites:

The Oracle database mush be registered with MSEI app registration.

1. Set the external identity provider as Microsoft Entra ID:

   ```
   ALTER SYSTEM SET IDENTITY_PROVIDER_TYPE=AZURE_AD SCOPE=BOTH;
   ```

2. Configure the external identify provider.

   ```
   ALTER SYSTEM SET identity_provider_config='{"application_id_uri":
   111-111-111, "tenant_id": "111-111-111", "app_id":"111-111-111"}';
   ```

Note:

The values `identity_provider_config` can be anything such as the "111-111-111" used in this example when working with Power BI access tokens

This configuration is specific for Power BI user SSO integration. Power BI user SSO integration is also supported with the full MSEI integration. The full MSEI integration allows both the Power BI user access as well as DBAs using SQLPlus and the MSEI interactive login and applications using client credential flow to access the database. The simpler Power BI SSO configuration described in this topic only allows Power BI users to access the database.

See Configuring the Oracle Database for Microsoft Entra ID Integration for more information about MSEI full integration.

8.6.3 Authorizing the User

The Power BI Azure AD user must be authorized to the database.

1. Log in to the Oracle database instance as a user who has the `CREATE USER` and `ALTER USER` system privileges.
2. Run the following command to create the Power BI Microsoft Entra ID user in the database:

   ```
   CREATE USER <first_last> IDENTIFIED GLOBALLY AS
   'AZURE_USER=<first.last@example.com>';GRANT CREATE SESSION TO <first_last>;
   ```

   All privileges and roles required by the user must be granted to the database schema/user. Power BI users cannot use a shared schema configuration; they can only use exclusive mapping to a schema.

8.6.4 Connecting Power BI to Oracle Database using Microsoft Entra ID

Once the database has been configured, you will need to configure Power BI Desktop or service.

Follow the instructions in this Oracle blog: Microsoft Power BI can now connect with the Oracle Database using Microsoft Entra ID SSO tokens.

8.7 Troubleshooting Microsoft Entra ID Connections

You can use trace files to diagnose problems with Microsoft Entra ID connections. You also can easily remedy `ORA-12599` and `ORA-03114` errors.

- Trace Files for Troubleshooting Oracle Database Client Connections with Entra ID
  You can use trace files to troubleshoot the Oracle Database integration with Entra ID.
- ORA-12599 and ORA-03114 Errors Caused When Trying to Access a Database Using a Token
  The `ORA-12599: TNS: cryptographic checksum mismatch` and `ORA-03114: not connected to ORACLE` errors indicate that the database to which you are trying to connect is protected by native network encryption.

- Checking the Entra ID Access Token Version
  You can check the version of the Entra ID access token that your site uses by using the JSON Web Tokens web site.

8.7.1 Trace Files for Troubleshooting Oracle Database Client Connections with Entra ID

You can use trace files to troubleshoot the Oracle Database integration with Entra ID.

- About Trace Files Used for Troubleshooting Connections
  You can generate two levels of trace files to troubleshoot Entra ID connections on client side.
- Setting Client Tracing for Token Authentication
  You can add `EVENT` settings to the client-side `sqlnet.ora` file to control client tracing.

8.7.1.1 About Trace Files Used for Troubleshooting Connections

You can generate two levels of trace files to troubleshoot Entra ID connections on client side.

The two levels of trace files that you can generate are as follows:

- Low level tracing prints traces in case of failures:
  - If TCPS is not set up for the Entra ID connection, then it prints a message that the protocol has to be TCPS.
  - If `SSL_SERVER_DN_MATCH` is not set to `TRUE`, then it prints a message that the value is `FALSE`.
  - If `TOKEN_LOCATION` has not been specified, then it prints a message that the token location does not exist.
  - If the token is not present at the specified `TOKEN_LOCATION`, then it prints a message.
  - If the application has passed in the token without setting `OCI_ATTR_TOKEN_ISBEARER` to true, it prints a message for the missing attribute.
  - If the application has set `OCI_ATTR_TOKEN_ISBEARER` to `TRUE` and not passed in the token, it prints a message for the missing attribute.
  - If the token has expired, then it prints a message.
  - If the token is a Microsoft Entra ID v2.0 token and it does not contain upn claim or roles claim, then it prints out a message that the needed claim is missing.
- High level tracing prints traces in case of failure as mentioned above. In addition, it prints traces in case of success, as follows:
  - It prints where `SSL_SERVER_DN_MATCH` is present, `tnsnames.ora` or `sqlnet.ora`. It also prints the value as `TRUE` if set to `TRUE`.
  - If both the token and `OCI_ATTR_TOKEN_ISBEARER=true` are set by the application, then it prints a message.
  - If `TOKEN_AUTH` has the correct value `OAUTH`, then it prints the value.
  - If the token is not expired, then it prints a message.
  - If the token is a Microsoft Entra ID v2.0 token and the upn claim or roles claim exist, then it prints out a message that the needed claim exists.

8.7.1.2 Setting Client Tracing for Token Authentication

You can add `EVENT` settings to the client-side `sqlnet.ora` file to control client tracing.

These `EVENT` settings can be used for both IAM and Entra ID connections with Oracle Database.

- Use either of the following methods:
  - Add the following settings to the client side `sqlnet.ora` file:
    - `EVENT_25701=14` for low level tracing
    - `EVENT_25701=15` for high level tracing
  - Set the environment variable `EVENT_25701`:
    - `EVENT_25701=14` for low level tracing
    - `EVENT_25701=15` for high level tracing

  Client trace files are created in the following locations:
  - **Linux:** `$ORACLE_HOME/log/diag/clients`
  - **Windows:** `%ORACLE_HOME%\log\diag\clients`

  You can use the `ADR_BASE` parameter in the client side `sqlnet.ora` to specify the directory in which tracing messages are stored. Ensure that the directory path is valid and has write permissions. Ensure that the `DIAG_ADR_ENABLED` parameter is not set to `FALSE`.

  An example of setting `ADR_BASE` is as follows:

  ```
  ADR_BASE=/oracle/oauth2/trace
  ```

8.7.2 ORA-12599 and ORA-03114 Errors Caused When Trying to Access a Database Using a Token

The `ORA-12599: TNS: cryptographic checksum mismatch` and `ORA-03114: not connected to ORACLE` errors indicate that the database to which you are trying to connect is protected by native network encryption.

When tokens are being used to access an Oracle database, a Transport Layer Security (TLS) connection must be established, not network native encryption. To remedy these errors, ensure that TLS is properly configured for your database. You should test the configuration with a local database user name and password and check the following `SYSCONTEXT USERENV` parameters:

- `NETWORK_PROTOCOL`
- `TLS_VERSION`

Related Topics

- Configuring PKI Certificate Authentication
  You can configure Oracle Database to use PKI certificates for end-user authentication.

8.7.3 Checking the Entra ID Access Token Version

You can check the version of the Entra ID access token that your site uses by using the JSON Web Tokens web site.

By default, Entra ID v1 access token, but your site may have chosen to use v2. Oracle Database supports v1 tokens and Autonomous Database Serverless supports v2 tokens, as well. If you want to use the v2 access tokens, then you can enable their use for the Oracle database. To find the version of the Entra ID access token that you are using, you can either check with your Entra ID administrator, or confirm the version from the JSON Web Tokens website, as follows.

1. Go to the JSON Web Tokens website.

   ```
   https://jwt.io/
   ```

2. Copy and paste the token string into the **Encoded** field.
3. Check the **Decoded** field, which displays information about the token string.

   Near or at the bottom of the field, you will see a claim entitled `ver`, which indicates either of the following versions:

   - `"ver": "1.0"`
   - `"ver": "2.0"`

Related Topics

- Enabling Microsoft Entra ID v2 Access Tokens
  Oracle Database supports integration with the v1 and v2 Azure AD `OAuth2` access token.