Oracle APEX + Microsoft Entra ID (Azure AD) Login

OpenID Connect (OIDC) Single Sign-On on Autonomous Database

OIDC login via Entra direct, or brokered through an OCI Identity Domain — Version 5 (tested)

Scope. This guide describes how to enable “Log in with Microsoft” for an Oracle APEX application using Microsoft Entra ID (formerly Azure AD) via OpenID Connect (OIDC) Social Sign-In. It covers the Entra app registration, the APEX web credential and authentication scheme, an **Application Access Control** model — app open to any authenticated user, with privileged pages and actions role-gated, and no local APEX user account — and the recurring client-secret rotation procedure. It also documents an alternative path that brokers authentication through an OCI Identity Domain (federated to Azure via SAML). The settings reflect a working, tested configuration.

Tested findings baked into this version — (1) Use preferred_username as the Username Attribute, not email: Entra users without a mailbox have no email claim, and that produces “Null username passed to login procedure” for those users. preferred_username (the UPN) is present for all users. (2) No APEX user account is required: with Social Sign-In the IdP authenticates the user; APEX needs no local account. Local accounts created via APEX_UTIL.CREATE_USER are unnecessary and were confirmed removable without breaking login.

Why OIDC and not SAML — On a managed Autonomous Database, ORDS is Oracle-managed and you cannot set the ORDS parameter security.externalSessionTrustedOrigins. SAML's callback is a cross-origin POST that ORDS blocks with HTTP 403 / ORDS-13002, and per Oracle's documentation SAML Sign-In on Autonomous Database is supported only with a customer-managed ORDS. OIDC uses a redirect-based authorization-code flow that does not trigger that cross-origin block, so it works on the default managed ORDS. Choose SAML only if it is a hard organizational requirement, in which case a customer-managed ORDS is needed.

Contents

(If the contents above are blank, right-click and choose “Update Field”.)

1. Prerequisites and Information to Collect

Before starting, gather the following. Replace the example values with your own throughout this guide.

Item	Value / Example
APEX instance host	`<host>.adb.<region>.oraclecloudapps.com`
APEX workspace	`demo`
APEX application ID / alias	`login-demo`
Entra tenant (directory) ID	`<tenant-guid>`
Roles required	An Application Access Control role to assign by default, e.g. `READER`
Access needed	Entra admin (App registrations) + APEX Workspace admin

Callback URL — APEX uses a single OAuth/OIDC callback endpoint of the form https://<host>/ords/apex_authentication.callback. The exact value is whatever APEX sends in the request; if Entra returns a redirect_uri mismatch (AADSTS50011), copy the URI shown in the Microsoft error page and register that verbatim. See Section 8 (Troubleshooting).

2. Microsoft Entra ID — App Registration

Note: OIDC uses an **App Registration** — this is a different Entra object than a SAML "Enterprise Application." If you previously set up SAML, do not reuse that object; create a new App Registration.

2.1 Create the registration

1. In the **Entra admin center**, go to **App registrations** → **New registration**.
2. **Name**: e.g. `APEX OIDC Login`.
3. **Supported account types**: single tenant (your organization only) is typical.
4. **Redirect URI**: platform **Web**, value:

```
https://<host>.adb.<region>.oraclecloudapps.com/ords/apex_authentication.callback
```

5. Click **Register**.

2.2 Capture the identifiers

From the registration Overview page, copy:

- **Application (client) ID** — used as the Client ID in APEX. (Not the Object ID, not the Directory/tenant ID.)
- **Directory (tenant) ID** — used to build the discovery URL.

2.3 Create a client secret

1. Go to **Certificates & secrets** → **New client secret**.
2. Set a description and an expiry (e.g. 6, 12, or 24 months). **Record the expiry date** — see Section 7 (Rotation).
3. Copy the secret **Value** immediately — it is shown only once. (Do not copy the Secret ID.)

2.4 Scopes and claims

1. Under **API permissions**, ensure delegated Microsoft Graph permissions `openid`, `profile`, `email` are present; grant admin consent if required.
2. (Optional) Under **Token configuration**, add the optional `email` claim so the email is reliably returned.

2.5 Discovery (well-known) endpoint

APEX needs only this single URL; it reads all endpoints (authorize, token, userinfo, JWKS) from it automatically:

```
https://login.microsoftonline.com/<tenant-guid>/v2.0/.well-known/openid-configuration
```

Open it in a browser to confirm it returns JSON. Useful fields you should see: `authorization_endpoint`, `token_endpoint`, `userinfo_endpoint` (`https://graph.microsoft.com/oidc/userinfo`), and `claims_supported` including `email` and `preferred_username`.

3. APEX — Network Prerequisites

Social Sign-In makes outbound web-service calls from the database to Microsoft, so the schema must be permitted to reach the Microsoft endpoints and the workspace must allow sufficient web-service requests.

- **Network ACL:** grant the APEX/parsing schema connect privilege to `login.microsoftonline.com` and `graph.microsoft.com` (using `DBMS_NETWORK_ACL_ADMIN`). On Autonomous Database, appending to the ACL is typically done as ADMIN.
- **Web-service request limit:** in the INTERNAL workspace → Manage Instance → Security → Workspace Isolation, raise **Maximum Web Service Requests** to an adequate value.

Confidence — The exact ACL principal (schema name) and the precise DBMS_NETWORK_ACL_ADMIN call vary by APEX version and by whether you run as ADMIN on Autonomous Database. Verify against current Oracle documentation for your release before running ACL changes. If login fails with an ORA-24247 / network access error, the ACL is the cause.

4. APEX — Create the Web Credential

In **App Builder → Workspace Utilities → Web Credentials** (or Shared Components → Web Credentials), click **Create** and set:

Field	Value
Name	`Azure OIDC`
Authentication Type	`OAuth2 Client Credentials Flow`
Client ID or Username	Application (client) ID from §2.2
Client Secret or Password	Secret Value from §2.3
Valid for URLs (optional)	`https://login.microsoftonline.com`

Save. The secret is stored encrypted; updating the “Valid for URLs” later requires re-entering the secret.

5. APEX — Create the Social Sign-In Scheme

Open the **Login Demo** application → **Shared Components → Authentication Schemes → Create → Based on a pre-configured scheme from the gallery**, and choose Scheme Type **Social Sign-In**.

Field	Value
Name	`Azure AD OIDC`
Scheme Type	`Social Sign-In`
Credential Store	Azure OIDC (from §4)
Authentication Provider	`OpenID Connect Provider`
Discovery URL	`.../v2.0/.well-known/openid-configuration`
Scope	`openid email profile`
Username Attribute	`preferred_username` (UPN — present for all users; use this, not `email`)
Convert Username To Upper Case	Yes (if APEX users/roles are stored uppercase)
Verify Attributes	Yes

Username attribute — Set Username to #preferred_username#. Do NOT use #sub# (an opaque Entra object GUID — unreadable and won't match ACL grants) and do NOT use #email# as the sole source (users without a mailbox return null → “Null username passed to login procedure”). preferred_username is the UPN and is populated for every Entra user, including those without email. Note it is the UPN, so guest/external users may appear as user_domain.com#EXT#@tenant.onmicrosoft.com; ensure ACL role grants use the same value the token returns.

5.1 Activate and test safely

Lockout safety: making a scheme Current routes all app login through Entra. If misconfigured, you cannot fall back to the old login from the runtime app. Follow this order:

1. Keep your **App Builder session open** in the current browser — this is your revert path.
2. Make `Azure AD OIDC` the **Current** scheme for the app.
3. In a separate **incognito window**, open the app home URL:

```
https://<host>.adb.<region>.oraclecloudapps.com/ords/r/demo/login-demo/home
```

4. Expected: redirect to Microsoft sign-in → authenticate → first-time consent → redirect back, logged in.
5. If login fails, return to the open App Builder session and set the auth scheme back to `Oracle APEX Accounts`.

5B. Alternative IdP — OCI Identity Domain (brokered to Azure via SAML)

Instead of APEX talking OIDC directly to Entra, APEX can talk OIDC to an **OCI Identity Domain**, which in turn federates user authentication to Azure/Entra via **SAML**. The user still authenticates against Azure, but APEX only ever speaks OIDC to OCI. This is a **brokered / hub** identity model and is a tested, working alternative to the direct-Entra path in Sections 2–5.

Why broker through OCI — (1) One integration for APEX: APEX trusts only the OCI domain; you can federate Azure (and other IdPs) behind it without changing APEX. (2) The secret APEX holds is the OCI confidential-app secret, not the Entra secret — OCI Identity Domain secrets are generally not on Entra's mandatory 24-month expiry clock, which can reduce or remove the rotation burden (VERIFY in your domain — see note below). (3) If the OCI domain already fronts your Fusion pod, APEX and Fusion can share one identity source.

5B.1 Architecture

- **APEX → OIDC → OCI Identity Domain** (Authorization Code flow; the part configured in APEX).
- **OCI Identity Domain → SAML → Azure/Entra** (the domain delegates authentication; the user signs in at Azure).

APEX never speaks SAML and never talks to Azure directly in this path. The SAML federation between OCI and Azure is configured in the OCI domain (Security → Identity Providers), independent of APEX.

5B.2 Create the OCI confidential application

1. OCI Console → **Identity & Security → Domains** → select the domain → **Integrated applications → Add application → Confidential Application**.
2. Configure this application as a client now; grant type **Authorization Code**; Redirect URL = your APEX callback:

```
https://<host>.adb.<region>.oraclecloudapps.com/ords/apex_authentication.callback
```

3. **Activate** the application and copy the **Client ID** and **Client Secret**.
4. Note the domain **discovery URL**:

```
https://<domain-id>.identity.oraclecloud.com/.well-known/openid-configuration
```

Check secret expiry here — When you open the confidential application's client secret in the OCI Console, note whether it shows an expiry date. OCI Identity Domain secrets are generally long-lived and not forced to expire on Entra's schedule, but this is environment/policy dependent — confirm in your console rather than assuming. Whatever you find determines whether Section 7 (rotation) applies to this path.

5B.3 APEX Web Credential and scheme

Field	Value
Web Credential name	`OCI OIDC`

Field	Value
Client ID	the OCI confidential-app Client ID
Client Secret + Verify	the OCI client secret (both fields)
OAuth Scope (credential)	leave blank — scope is set on the scheme
Scheme Type	`Social Sign-In`
Authentication Provider	`OpenID Connect Provider`
Discovery URL	the OCI domain `.well-known/openid-configuration`
Scope	`profile email` (do NOT include `openid` — see note)
Username Attribute	`#sub#` (on an IDCS-style domain this is usually the username)

Duplicate-scope error (tested) — IDCS-style OCI domains advertise only a minimal base claim set (aud, exp, iat, iss, jti, sub) — no preferred_username/email guaranteed in discovery — so #sub# is the safe Username Attribute. Also: APEX adds openid to the scope automatically. If you ALSO put openid in the Scope field, the request sends “openid openid profile email” and the domain rejects it with invalid_request / “duplicate values.” Set Scope to “profile email” and let APEX add openid. This was hit and resolved during testing.

5B.4 Username resolution — verify after first login

Because the user is authenticated via Azure-over-SAML and surfaced to APEX through OCI, confirm what `#sub#` actually resolves to on first login (a readable username, an email-style value, or a domain-local id). Whatever it is becomes the key for any application-role grants (Section 6), so verify it matches how your roles are keyed before granting elevated access.

5B.5 Choosing between the two paths

Path	Trade-off
Direct Entra (§2–5)	APEX → OIDC → Entra. Fewer hops. Entra client secret on a forced expiry clock; rotation required (§7).
Brokered OCI (§5B)	APEX → OIDC → OCI → SAML → Entra. One APEX integration; can federate more IdPs behind OCI; secret APEX holds is the OCI secret (verify expiry). Slightly more moving parts (the OCI–Azure SAML trust).

6. Authorization — App Access and Role-Gated Pages

Authentication (who the user is) is handled entirely by Entra/OIDC. Authorization (what they may do) is handled by Application Access Control. These are independent. There are two layers, and you choose how open the app is at each.

No APEX user account is required — Do not create local APEX accounts (APEX_UTIL.CREATE_USER / Administration → Users) for OIDC users. With Social Sign-In the user is authenticated by the IdP; APEX does not need a local account. Verified by deleting the local account and confirming login still works.

6.1 Layer 1 — who can open the app at all

In **Shared Components → Application Access Control → Configure Access Control**, the setting **"Any authenticated user may access this application"** controls app-level entry:

- **Yes** — any successfully authenticated user can open the app, **without** being listed in the access control list. No per-user role grant is needed just to get in. (This removes the "Access denied by Application security check" wall.)
- **No** — only users explicitly added to the access control list (with a role) may open the app.

For broad access (e.g. all employees in the IdP may use the app), set this to **Yes**. Fine-grained control is then applied per page/action in Layer 2.

6.2 Layer 2 — role-gated pages and actions

Even with app entry open to all authenticated users, you can restrict specific **pages, regions, buttons, items, or processes** by attaching an **Authorization Scheme** tied to an **Application Role**. Only users granted that role see or run the gated component; everyone else gets the default (e.g. read-only) experience.

Setup: define roles in **Application Access Control → Roles** (e.g. `ADMINISTRATOR`, `CONTRIBUTOR`, `READER`); APEX auto-creates a matching Authorization Scheme per role; set that scheme on the page/component you want to gate; grant the role only to the users who need it.

6.3 Recommended model

Aspect	Setting
App entry	"Any authenticated user may access" = **Yes**
Default experience	Read-only / general pages available to all authenticated users
Privileged pages/actions	Gated by an Authorization Scheme tied to an application role
Role grants	Assigned only to the few users who need elevated access
Local APEX accounts	None — identity comes from the IdP

Why this over auto-provisioning — Setting app entry to "any authenticated user" plus role-gating privileged components avoids granting a role to every user on login. Most users need no role at all (they get the default experience); only elevated users are granted a role. This is simpler to operate than per-login role provisioning and keeps the access list small and meaningful.

7. Client-Secret Rotation

Which path this applies to — This section is primarily for the DIRECT ENTRA path (§2–5), whose client secret is on Entra's forced expiry clock. For the BROKERED OCI path (§5B), the secret APEX holds is the OCI confidential-app secret, which is generally long-lived — confirm its expiry in the OCI Console (§5B.2). If your OCI secret has no forced expiry, the manual-deadline rotation below is replaced by occasional policy-driven rotation only.

Entra client secrets **expire**. When the secret lapses, every login fails (token request returns `invalid_client` / AADSTS7000215 or AADSTS700016). Rotate **before** the recorded expiry. Entra allows two valid secrets on one registration at once, which enables zero-downtime rotation.

7.1 Rotation procedure (zero downtime)

1. **Create a new secret.** Entra → your App Registration → Certificates & secrets → New client secret. Set a new expiry and copy the new **Value** immediately.
2. **Update the APEX web credential.** App Builder → Workspace Utilities → Web Credentials → `Azure OIDC` → paste the new secret into **Client Secret** (and Verify), then Apply Changes. The Client ID is unchanged.
3. **Test.** In an incognito window, sign in to the app to confirm the new secret works.
4. **Delete the old secret.** Once verified, remove the previous secret in Entra so a leaked old value cannot be used.
5. **Record the new expiry** and set a reminder ahead of it.

7.2 Recommended cadence and reminders

Aspect	Recommendation
Secret lifetime	12 or 24 months (Entra max is 24 months)
Rotate by	At least 2–4 weeks before expiry
Track where	Calendar reminder + secret-expiry note in your config record
Outage symptom	All logins fail at the token step; APEX log shows `invalid_client` / AADSTS7000215
Owner	Document a named owner so rotation isn't missed when staff change

Eliminate secrets entirely (optional) — To avoid rotation altogether, switch the App Registration to a certificate credential (client assertion) instead of a shared secret, or use a managed identity where supported. These remove the periodic secret-expiry outage but add certificate lifecycle management. Validate support against your APEX release before adopting.

8. Troubleshooting

Symptom	Cause / Fix
HTTP 403 / ORDS-13002	Cross-origin block. Indicates a SAML callback on managed ORDS. Use OIDC (this guide), or move to customer-managed ORDS for SAML.
AADSTS50011 redirect_uri mismatch	The redirect URI APEX sent is not registered. Copy the exact URI from the Microsoft error page into the App Registration → Authentication → Redirect URIs.
invalid_client / AADSTS7000215	Wrong or expired client secret. Rotate per Section 7; confirm the secret Value (not Secret ID) is stored.
Access denied by Application security check	Authentication OK, authorization failing. Either set “Any authenticated user may access” = Yes (§6.1), or grant the user a role.
Null username passed to login procedure	The username claim resolved to null — typically #email# on a user without a mailbox. Set Username Attribute to #preferred_username# (§5).
Username is a GUID / blank	Username Attribute set to sub or wrong claim. Use preferred_username (§5).
ORA-24247 network access denied	Missing network ACL to the Microsoft endpoints. See Section 3.

9. Configuration Record (fill in for your environment)

Item	Value
Tenant (directory) ID	_____
Application (client) ID	_____
Discovery URL	_____
Redirect URI registered	_____
Web credential name	Azure OIDC
Auth scheme name	Azure AD OIDC
App access model	Any authenticated user = Yes; privileged pages role-gated
Secret created on	_____ Expires: _____
Rotation owner	_____

End of document.