

SSO Implementation Guide: Azure AD (Microsoft Entra ID) + AWS Cognito + Next.js

1. Purpose

This document explains the complete Single Sign-On (SSO) implementation used in this project:

- Identity Provider (IdP): Microsoft Entra ID (Azure AD)
 - Federation Broker: AWS Cognito Hosted UI
 - Application: Next.js frontend
 - OAuth Flow: Authorization Code + PKCE
-

2. Final Architecture

1. User clicks **Sign in with Microsoft** in the app.
 2. App redirects to **AWS Cognito Hosted UI** `/oauth2/authorize`.
 3. Cognito forwards authentication to **Azure AD** (configured IdP name: `AzureAD`).
 4. Azure AD redirects back to Cognito at:
 - o `https://<cognito-domain>.auth.<region>.amazoncognito.com/oauth2/idpresponse`
 5. Cognito completes federation and redirects to app callback URL:
 - o `<app-callback-url>` (example: `https://<app-domain>/api/auth/callback`)
 6. App exchanges auth code at Cognito `/oauth2/token` and stores tokens.
-

3. Prerequisites

- AWS account with Cognito permissions
 - Azure/Entra tenant with App Registration permissions
 - Cognito User Pool domain configured
 - Next.js app environment variables configured
-

4. Azure AD Setup

4.1 Create App Registration

In Azure portal:

1. Go to **Microsoft Entra ID** → **App registrations** → **New registration**.
2. Name: `Philips-Sensei` (or tenant naming standard).
3. Supported account type: choose per org policy (typically single-tenant).
4. Create the application.

The screenshot shows the Azure portal's App registrations section. A search bar at the top left contains 'Task Amplify'. Below it, a navigation bar includes 'Overview' (which is selected), 'Quickstart', 'Integration assistant', 'Diagnose and solve problems', 'Manage' (expanded to show 'Branding & properties', 'Authentication (Preview)', 'Certificates & secrets', 'Token configuration', 'API permissions', 'Expose an API', 'App roles', 'Owners', 'Roles and administrators', and 'Manifest'), 'Endpoints', and 'Preview features'. The main content area is titled 'Essentials' and displays various application details: Display name ('Task Amplify'), Application (client) ID ('24ec5af7-b4e1-4b45-962c-3b3e5cd9c379'), Object ID ('e007087e-c7d9-4dad-82dc-b06908510c2e'), Directory (tenant) ID ('21f3742b-336b-4e6b-bc0b-8a0ef1a779fa'), Supported account types ('My organization only'), Client credentials ('0 certificate, 1 secret'), Redirect URIs ('1 web, 0 spa, 0 public client'), Application ID URI ('Add an Application ID URI'), Managed application in local directory ('Task Amplify'), and State ('Activated'). A note at the bottom states: 'Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure Active Directory Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications will need to be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph. [Learn more](#)'.

Get Started Documentation

Build your application with the Microsoft identity platform

4.2 Configure Redirect URI (Critical)

Add **Web** redirect URI:

- <https://<cognito-domain>.auth.<region>.amazoncognito.com/oauth2/idpresponse>

Why this matters:

- This URI is for **Azure** → **Cognito** federation callback.
- Do **not** use the app callback (</api/auth/callback>) here.

The screenshot shows the 'Authentication (Preview)' configuration page for the Task Amplify app. The left sidebar is identical to the previous screenshot, showing the 'Authentication (Preview)' option selected under 'Manage'. The main content area has a 'Welcome' message: 'Welcome to the new and improved experience for authentication. [To switch to the old experience, please click here.](#)' Below this, the 'Redirect URI configuration' tab is selected, showing a table of existing redirect URIs. One entry is visible: a 'Web' platform type with the URL 'https://ap-south-1caroawzdd.auth.ap-south-1.amazoncognito.com/oauth2/idpresponse'. Other columns in the table include 'Platform Type' (with sorting arrows), 'Redirect URI' (with an 'Edit' link), and three vertical ellipses for more options.

4.3 Generate Client Secret (for OIDC)

1. Go to **Certificates & secrets**.
2. Create a new client secret.
3. Copy and securely store the secret value.

Used where:

- Cognito OIDC IdP configuration requires Azure Client ID + Client Secret.

Home > App registrations > Task Amplify

Task Amplify | Certificates & secrets

Credentials enable confidential applications to identify themselves to the authentication service when receiving tokens at a web addressable location (using an HTTPS scheme). For a higher level of assurance, we recommend using a certificate (instead of a client secret) as a credential.

Certificates (0) Client secrets (1) Federated credentials (0)

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

Description	Expires	Value ⓘ	Secret ID
Task Amplify Secret	2/20/2028	C30*****	f2777774-4d14-4933-8b6a-c700dd823...

4.4 API Permissions

Ensure delegated OpenID permissions are granted:

- openid
- profile
- email

Home > App registrations > Task Amplify

Task Amplify | API permissions

Add a permission **Grant admin consent for Default Directory**

API / Permissions name	Type	Description	Admin consent req...	Status
Microsoft Graph (1)				...
User.Read	Delegated	Sign in and read user profile	No	...

Other permissions granted for Default Directory

These permissions have been granted for Default Directory but aren't in the configured permissions list. If your application requires these permissions, you should consider adding them to the configured permissions list. [Learn more](#)

API / Permissions name	Type	Description	Admin consent req...	Status
Microsoft Graph (3)				...
email	Delegated	View users' email address	No	Granted for Default Dire... ...
openid	Delegated	Sign users in	No	Granted for Default Dire... ...
profile	Delegated	View users' basic profile	No	Granted for Default Dire... ...

To view and manage consented permissions for individual apps, as well as your tenant's consent settings, try [Enterprise applications](#).

5. AWS Cognito Setup

5.1 User Pool + Domain

1. Create/use existing User Pool.
2. Configure Hosted UI domain.
3. Domain used in this project pattern:
 - o <https://<cognito-domain>.auth.<region>.amazoncognito.com>

The screenshot shows the 'Overview' section of a user pool named 'User pool - zl64qs'. The left sidebar includes sections for 'Overview', 'Applications' (App clients), 'User management' (Users, Groups), and 'Authentication' (Authentication methods, Sign-in, Sign-up, Social and external providers, Extensions). The main content area displays 'User pool information' such as the name, ID, ARN, token signing key URL, estimated number of users (3), and feature plan (Essentials). It also features 'Recommendations' for setting up an app and applying branding to login pages.

5.2 Add Azure AD as Identity Provider

In User Pool federation settings:

1. Add external IdP (OIDC).
2. Provider name: **AzureAD**.
3. Setup method: **Auto fill through issuer URL**.
4. Enter Azure OIDC Issuer URL in this format:
 - o <https://login.microsoftonline.com/<tenant-id>/v2.0>
5. Cognito auto-populates OIDC endpoints ([authorize](#), [token](#), [userInfo](#), [jwks_uri](#)) from Azure discovery metadata.
6. Enter Azure Client ID and Client Secret.
7. Enter openid profile email as your Authorized Scopes.
8. Save provider.

The screenshot shows the 'Edit identity provider: AzureAD' page in the Amazon Cognito console. The provider is configured as an OpenID Connect (OIDC) provider. Key settings include:

- Provider name:** AzureAD
- Client ID:** 24ec5a7-4e41-4b45-962c-3b5e5c9c579
- Client secret:** (Redacted)
- Authorized scopes:** openid profile email
- Identifiers - optional:** (Empty)
- Attribute request method:** GET (selected over POST)
- Issuer URL:** https://login.microsoftonline.com/21f5742b-336b-4e6b-bc0b-8a0ef1a779fa/v2.0

How to get <tenant-id>:

- Azure Portal → App Registrations → Your App → Overview → Tenant ID

Important clarification:

- The Issuer URL is **not generated by Cognito**.
- You provide the tenant-specific issuer URL from Azure.
- Cognito only auto-fills endpoint fields after you provide the Issuer URL.

Notes:

- If configured as SAML instead, secret is not used.
- This implementation assumes OIDC because the app uses OIDC scopes and **identity_provider=AzureAD**.

Social and custom providers

Federated identity provider sign-in (1) [Info](#)

[Delete](#) [Add identity provider](#) [View signing certificate](#)

Your app users can sign-in through external social identity providers like Facebook, Google, Amazon, or Apple, and through your on-prem directories via SAML or Open ID Connect.

⚠ Self-registration is enabled in this user pool, allowing anyone on the internet to sign up for a local user account. Self-registration isn't required for federated sign-in with third-party providers. As a security best practice, disable self-registration if your user pool is intended for enterprise OIDC and SAML federation. [Edit self-registration](#)

Identity provider	Identity provider type	Created time	Last updated time
AzureAD	OIDC	7 days ago	4 days ago

Identity provider: AzureAD [Info](#) [Delete](#)

Identity provider information

Provider type	OIDC	Identifiers	Token endpoint
Provider name	AzureAD	Attribute request method	Jwks_uri endpoint
Client ID	24ec5af7-b4e1-4b45-962c-3b3e5cd9c379	Issuer	Created time
Client secret	*****	https://login.microsoftonline.com/21f3742b-336b-4e6b-bc0b-8a0ef1a779fa/v2.0	February 20, 2026 at 20:51 GMT+5:30
<input checked="" type="checkbox"/> Show client secret		Authorization endpoint	Last updated time
		-	February 23, 2026 at 12:09 GMT+5:30

Attribute mapping (3) [Info](#) [Edit](#)

View, add, and edit attribute mappings between OpenID Connect and your user pool.

5.3 Configure App Client

In Cognito App Client settings:

1. Enable **Authorization code grant**.
2. Callback URL:
 - o <app-callback-url> (example: <https://<app-domain>/api/auth/callback>)
3. Sign-out URL:
 - o <app-logout-url> (example: <https://<app-domain>/login>)
4. Allowed scopes:
 - o [openid](#)
 - o [profile](#)
 - o [email](#)
5. Enable Identity providers: [AzureAD](#) (and Cognito native if needed by policy).

The screenshot shows the 'Managed login pages' configuration for an app client named 'task-amplify-client'. It includes fields for Allowed callback URLs (with a value of 'https://main.d1flytdzrouwf.amplifyapp.com/auth/callback'), Default redirect URL (with a value of 'https://main.d1flytdzrouwf.amplifyapp.com/login'), Allowed sign-out URLs (with a value of 'https://main.d1flytdzrouwf.amplifyapp.com/logout'), Identity providers (selected 'AzureAD'), OAuth 2.0 grant types (selected 'Authorization code grant'), and OpenID Connect scopes (selected 'Email', 'OpenID', and 'Profile'). The 'Save changes' button is at the bottom right.

6. Application Configuration

In `.env.local` (frontend):

```
NEXT_PUBLIC_COGNITO_REGION=<aws-region>
NEXT_PUBLIC_COGNITO_USER_POOL_ID=<user-pool-id>
NEXT_PUBLIC_COGNITO_CLIENT_ID=<app-client-id>
NEXT_PUBLIC_COGNITO_DOMAIN=<cognito-domain-prefix>
NEXT_PUBLIC_COGNITO_REDIRECT_URI=<app-callback-url>
NEXT_PUBLIC_COGNITO_LOGOUT_URI=<app-logout-url>
NEXT_PUBLIC_COGNITO_SCOPES=openid profile email
NEXT_PUBLIC_IDENTITY_PROVIDER=AzureAD
```

Behavior implemented:

- Login button triggers Hosted UI URL generation.
- URL includes:
 - `response_type=code`
 - `identity_provider=AzureAD`
 - `code_challenge_method=S256` (PKCE)
- Callback page exchanges code at Cognito `/oauth2/token`.
- Tokens are stored in browser storage for session continuity.

7. End-to-End Validation Steps

1. Open app login page.
2. Click **Sign in with Microsoft**.
3. Confirm browser goes to Cognito Hosted UI authorize endpoint.
4. Confirm redirect to Microsoft login.

5. Sign in with test account.
6. Confirm redirect path sequence:
 - Azure → `/oauth2/idpreresponse` (Cognito)
 - Cognito → `/api/auth/callback` (app)
7. Confirm app lands on `/home` with authenticated session.

Evidence to capture:

- Browser address bar at each redirect stage
 - Cognito Hosted UI network calls
 - Successful `/oauth2/token` response
-