Application-specific settings with app clients

On this page

Related resources

Amazon Cognito user pools API Reference (https://docs.aws.amazon.com/cognito-user-identity-pools/latest/APIReference/index.html)

AWS CLI commands for Amazon Cognito user pools (https://docs.aws.amazon.com/cli/latest/reference/cognito-idp/)

SDKs & Tools (https://aws.amazon.com/tools/)

Recommended tasks

How to

Configure Amazon Cognito to authorize REST APIs (https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-integrate-with-cognito.htm

Accessing resources after sign-in (accessing-resources.html)

M2M and scopes (cognito-user-pools-define-resource-servers.html)

Learn about

Understand Cognito user pools API capabilities (https://docs.aws.amazon.com/cognito-user-identity-pools/latest/APIReference/Welcome.html)

Recently added to this guide

Did this page help you?

Yes No

Provide feedback (https://docs.aws.amazon.com/forms/aws-doc-feedback?hidden_service_name=Cognito&topic_url=https://docs.aws.amazon.com/en_us/cognito/latest/developerguid pool-settings-client-apps.html)

A user pool app client is a configuration within a user pool that interacts with one mob web application that authenticates with Amazon Cognito. App clients can call authenti and unauthenticated API operations, and read or modify some or all of your users' attributes. Your app must identify itself to the app client in operations to register, sign and handle forgotten passwords. These API requests must include self-identification w an app client ID, and authorization with an optional client secret. You must secure any client IDs or secrets so that only authorized client apps can call these unauthenticated operations. Additionally, if you configure your app to sign authenticated API requests v AWS credentials, you must secure your credentials against user inspection.

You can create multiple apps for a user pool. An app client might be linked to the code platform of an app, or a separate tenant in your user pool. For example, you might cre an app for a server-side application and a different Android app. Each app has its own client ID.

You can apply settings for the following user pool features at the app client level:

1. Analytics (./cognito-user-pools-pinpoint-integration.html)
2. Managed login (./cognito-user-pools-managed-login.html) IdPs, grant types, callback U and customization
3. Resource servers and custom scopes (./cognito-user-pools-define-resource-servers.html
4. Threat protection (./cognito-user-pool-settings-threat-protection.html)
5. Attribute read and write permissions (./user-pool-settings-attributes.html#user-pool-settings-attribute-permissions-and-scopes)
6. Token expiration and revocation (./amazon-cognito-user-pools-using-tokens-with-ident providers.html)
7. Authentication flows (./authentication.html#amazon-cognito-user-pools-authentication-

App client types

When you create an app client in Amazon Cognito, you can pre-populate options basec the standard OAuth client types **public client** and **confidential client**. Configure a **confidential client** with a **client secret**. For more information about client types, see IE RFC 6749 #2.1 (https://datatracker.ietf.org/doc/html/rfc6749#section-2.1) .

Public client

A public client runs in a browser or on a mobile device. Because it does not have trusted server-side resources, it does not have a client secret.

Confidential client

A confidential client has server-side resources that can be trusted with a **client secre** for unauthenticated API operations. The app might run as a daemon or shell script c your backend server.

Client secret

A client secret, or client password, is a fixed string that your app must use in all API requests to the app client. Your app client must have a client secret to perform `client_credentials` grants. For more information, see IETF RFC 6749 #2.3.1 (https://datatracker.ietf.org/doc/html/rfc6749#section-2.3.1) .

You can't change secrets after you create an app. You can create a new app with a n secret if you want to rotate the secret. You can also delete an app to block access fr apps that use that app client ID.

> **Note**
>
> The Amazon Cognito console creates app clients with client secrets when you select the **Traditional web application** and **Machine-to-machine application** options for application type. Choose one of these options to generate a client secret, or create the client programmatically with CreateUserPoolClient (https://docs.aws.amazon.com/cognito-user-identity-pools/latest/APIReference/API_CreateUserPoolClient.html) and set `GenerateSecret to true`.

You can use a confidential client, and a client secret, with a public app. Use an Amazon CloudFront proxy to add a `SECRET_HASH` in transit. For more information, see Protect public clients for Amazon Cognito by using an Amazon CloudFront proxy (https://aws.amazon.com/blogs/security/protect-public-clients-for-amazon-cognito-by-using-a amazon-cloudfront-proxy/) on the AWS blog.

JSON web tokens

Amazon Cognito app clients can issue JSON web tokens (JWTs) of the following types.

Identity (ID) token

A verifiable statement that your user is authenticated from your user pool. OpenID Connect (OIDC) added the ID token specification (https://openid.net/specs/openid-connect-core-1_0.html#IDToken) to the access and refresh token standards defined by OAuth 2.0. The ID token contains identity information, like user attributes, that your app can use to create a user profile and provision resources. See Understanding the

identity (ID) token (./amazon-cognito-user-pools-using-the-id-token.html) for more information.

Access token

A verifiable statement of your user's access rights. The access token contains scopes (https://datatracker.ietf.org/doc/html/rfc6749#section-3.3) , a feature of OIDC and OAuth 2.0. Your app can present scopes to back-end resources and prove that your user po authorized a user or machine to access data from an API, or their own user data. An access token with *custom scopes*, often from an M2M client-credentials grant, authorizes access to a resource server. See Understanding the access token (./amazon cognito-user-pools-using-the-access-token.html) for more information.

Refresh token

An encrypted statement of initial authentication that your app can present to your user pool when your user's tokens expire. A refresh-token request returns new, unexpired access and ID tokens. See Refresh tokens (./amazon-cognito-user-pools-using the-refresh-token.html) for more information.

You can set the expiration of these tokens for each app client from the **App clients** me your user pool in the Amazon Cognito console (https://console.aws.amazon.com/cognito/v2/idp/user-pools) .

App client terms

The following terms are available properties of app clients in the Amazon Cognito con

Allowed callback URLs

A callback URL indicates where the user will be redirected after a successful sign-in. Choose at least one callback URL. The callback URL must:

- Be an absolute URI.
- Be pre-registered with a client.
- Not include a fragment component.

See OAuth 2.0 - redirection endpoint (https://tools.ietf.org/html/rfc6749#section-3.1.

Amazon Cognito requires `HTTPS` over `HTTP` except for `http://localhost` for testing purposes only.

App callback URLs such as `myapp://example` are also supported.

Allowed sign out URLs

A sign-out URL indicates where your user is to be redirected after signing out.

Attribute read and write permissions

Your user pool might have many customers, each with their own app client and IdPs You can configure your app client to have read and write access to only those user attributes that are relevant to the app. In cases like machine-to-machine (M2M) authorization, you can grant access to none of your user attributes.

Considerations for attribute read and write permissions configuration

- When you create an app client and don't customize attribute read and write permissions, Amazon Cognito grants read and write permissions to all user pool attributes.
- You can grant write access to immutable custom attributes (./user-pool-settings-attributes.html#user-pool-settings-custom-attributes.title) . Your app client can write values to immutable attributes when you create or sign up a user. After this, you can't write values to any immutable custom attributes for the user.
- App clients must have write access to required attributes in your user pool. The Amazon Cognito console automatically sets required attributes as writeable.
- You can't permit an app client to have write access to `email_verified` or `phone_number_verified`. A user pool administrator can modify these values user can only change the value of these attributes through attribute verification (./signing-up-users-in-your-app.html#allowing-users-to-sign-up-and-confirm-themselves.title) .

Authentication flows

The methods that your app client allows for sign-in. Your app can support authentication with username and password, email and SMS message OTPs, passkey authenticators, custom authentication with Lambda triggers, and token refresh. As a best security practice, use SRP authentication for username and password authentication in custom-built applications.

Custom scopes

A custom scope is one that you define for your own resource server in the **Resource Servers**. The format is *`resource-server-identifier/scope`*. See Scopes, M2M and APIs with resource servers (./cognito-user-pools-define-resource-servers.html) .

Default redirect URI

Replaces the `redirect_uri` parameter in authentication requests for users with third-party IdPs. Configure this app client setting with the `DefaultRedirectURI` parameter of a CreateUserPoolClient (https://docs.aws.amazon.com/cognito-user-identity-pools/latest/APIReference/API_CreateUserPoolClient.html) or UpdateUserPoolClient (https://docs.aws.amazon.com/cognito-user-identity-pools/latest/APIReference/API_UpdateUserPoolClient.html) API request. This URL must also be a member of the `CallbackURLs` for your app client. Amazon Cognito redirects authenticated sessions to this URL when:

1. Your app client has one identity provider (#app-client-terms-identity-provider) assigned and multiple callback URLs (#app-client-terms-callback-urls) defined. Your user pool redirects authentication requests to the authorization server (./authorization-endpoint.html) to the default redirect URI when they don't include `redirect_uri` parameter.
2. Your app client has one identity provider (#app-client-terms-identity-provider) assigned and one callback URLs (#app-client-terms-callback-urls) defined. In this scenario it's not necessary to define a default callback URL. Requests that don't include a `redirect_uri` parameter redirect to the one available callback URL.

Identity providers

You can choose some or all of your user pool external identity providers (IdPs) to authenticate your users. Your app client can also authenticate only local users in your

user pool. When you add an IdP to your app client, you can generate authorization links to the IdP and display it on your managed login sign-in page. You can assign multiple IdPs, but you must assign at least one. For more information on using external IdPs, see User pool sign-in with third party identity providers (./cognito-user-pools-identity-federation.html) .

OpenID Connect scopes

Choose one or more of the following `OAuth` scopes to specify the access privileges that can be requested for access tokens.

- The `openid` scope declares that you want to retrieve an ID token and a user's unique ID. It also requests all or some user attributes, depending on additional scopes in the request. Amazon Cognito doesn't return an ID token unless you request the `openid` scope. The `openid` scope authorizes structural ID token claims like expiration and key ID, and determines the user attributes that you receive in a response from the userInfo endpoint (./userinfo-endpoint.html) .
  - When `openid` is the only scope that you request, Amazon Cognito populat the ID token with all user attributes that the current app client can read. Th `userInfo` response to an access token with this scope alone returns all us attributes.
  - When you request `openid` with other scopes like `phone`, `email`, or `profile`, the ID token and `userInfo` return the user's unique ID and the attributes defined by the additional scopes.
- The `phone` scope grants access to the `phone_number` and `phone_number_verified` claims. This scope can only be requested with the `openid` scope.
- The `email` scope grants access to the `email` and `email_verified` claims. T scope can only be requested with the `openid` scope.
- The `aws.cognito.signin.user.admin` scope grants access to Amazon Cognito user pools API operations (./authentication-flows-public-server-side.html#us pools-API-operations) that require access tokens, such as UpdateUserAttributes (https://docs.aws.amazon.com/cognito-user-identity-pools/latest/APIReference/API_UpdateUserAttributes.html) and VerifyUserAttribute (https://docs.aws.amazon.com/cognito-user-identity-pools/latest/APIReference/API_VerifyUserAttribute.html) .
- The `profile` scope grants access to all user attributes that are readable by the client. This scope can only be requested with the `openid` scope.

For more information about scopes, see the list of standard OIDC scopes (http://openid.net/specs/openid-connect-core-1_0.html#ScopeClaims) .

OAuth grant types

An OAuth grant is a method of authentication that retrieves user-pool tokens. Amaz Cognito supports the following types of grants. To integrate these OAuth grants in your app, you must add a domain to your user pool.

Authorization code grant

The authorization code grant generates a code that your app can exchange for user pool tokens with the Token endpoint (./token-endpoint.html) . When you exchange an authorization code, your app receives ID, access, and refresh tokens. This OAuth flow

like the implicit grant, happens in your users' browsers. An authorization code grant the most secure grant that Amazon Cognito offers, because tokens aren't visible in your users' sessions. Instead, your app generates the request that returns tokens anc can cache them in protected storage. For more information, see *Authorization code* i IETF RFC 6749 #1.3.1 (https://datatracker.ietf.org/doc/html/rfc6749#section-1.3.1)

> **Note**
>
> As a best security practice in public-client apps, activate only the authorization-code grant OAuth flow, and implement Proof Key for Code Exchange (PKCE) to restrict token exchange. With PKCE, a client can only exchange an authorization code when they have provided the token endpoint with the same secret that was presented in the original authentication request. For more information on PKCE, see IETF RFC 7636 (https://datatracker.ietf.org/doc/html/rfc7636) .

Implicit grant

The implicit grant delivers an access and ID token, but not refresh token, to your use browser session directly from the Authorize endpoint (./authorization-endpoint.html) . implicit grant removes the requirement for a separate request to the token endpoin but isn't compatible with PKCE and doesn't return refresh tokens. This grant accommodates testing scenarios and app architecture that can't complete authorization-code grants. For more information, see *Implicit grant* in IETF RFC 674 #1.3.2 (https://datatracker.ietf.org/doc/html/rfc6749#section-1.3.2) . You can activate both the authorization-code grant and the implicit grant in an app client, and then u each grant as needed.

Client credentials grant

The client credentials grant is for machine-to-machine (M2M) communications. Authorization-code and implicit grants issue tokens to authenticated human users. Client credentials grant scope-based authorization from a non-interactive system to API. Your app can request client credentials directly from the token endpoint and receive an access token. For more information, see *Client Credentials* in IETF RFC 674 #1.3.4 (https://datatracker.ietf.org/doc/html/rfc6749#section-1.3.4) . You can only activate client-credentials grants in app clients that have a client secret and that dor support authorization-code or implicit grants.

> **Note**
>
> Because you don't invoke the client credentials flow as a user, this grant can only add *custom* scopes to access tokens. A custom scope is one that you define for your own resource server. Default scopes like `openid` and `profile` don't apply to nonhuman users.
>
> Because ID tokens are a validation of user attributes, they aren't relevant to M2M communication, and a client credentials grants doesn't issue them. See Scopes, M2M, and APIs with resource servers (./cognito-user-pools-define-resource-servers.html) .

Client credentials grants add costs to your AWS bill. For more information, see Amaz Cognito Pricing (https://aws.amazon.com/cognito/pricing) .

Creating an app client

AWS Management Console | AWS CLI | Amazon Cognito user pools API

To create an app client (console)

1. Go to the Amazon Cognito console (https://console.aws.amazon.com/cognito/home) . If prompted, enter your AWS credentials.
2. Choose **User Pools**.
3. Choose an existing user pool from the list, or create a user pool. Both options prompt you to configure an app client with application-specific settings.
4. Choose an **Application type** that reflects your application architecture.
5. **Name your application** with a friendly identifier.
6. Enter a **Return URL**.
7. Choose **Create app client**. You can change advanced options after you create your app client.
8. Amazon Cognito returns you to app client details. To access example code for your application, select a platform from the **Quick setup guide** tab.

Updating a user pool app client (AWS CLI and AWS API)

At the AWS CLI, enter the following command:

```
aws cognito-idp update-user-pool-client --user-pool-id
"MyUserPoolID" --client-id "MyAppClientID" --allowed-o-auth-
flows-user-pool-client --allowed-o-auth-flows "code"
"implicit" --allowed-o-auth-scopes "openid" --callback-urls "
["https://example.com"]" --supported-identity-providers "
["MySAMLIdP", "LoginWithAmazon"]"
```

If the command is successful, the AWS CLI returns a confirmation:

```
{
    "UserPoolClient": {
        "ClientId": "MyClientID",
        "SupportedIdentityProviders": [
            "LoginWithAmazon",
```

```
            "MySAMLIdP"
        ],
        "CallbackURLs": [
            "https://example.com"
        ],
        "AllowedOAuthScopes": [
            "openid"
        ],
        "ClientName": "Example",
        "AllowedOAuthFlows": [
            "implicit",
            "code"
        ],
        "RefreshTokenValidity": 30,
        "AuthSessionValidity": 3,
        "CreationDate": 1524628110.29,
        "AllowedOAuthFlowsUserPoolClient": true,
        "UserPoolId": "MyUserPoolID",
        "LastModifiedDate": 1530055177.553
    }
}
```

See the AWS CLI command reference for more information: update-user-pool-client (https://docs.aws.amazon.com/cli/latest/reference/cognito-idp/update-user-pool-client.html) .

AWS API: UpdateUserPoolClient (https://docs.aws.amazon.com/cognito-user-identity-pools/latest/APIReference/API_UpdateUserPoolClient.html)

Getting information about a user pool app client (AWS CLI and AWS API)

```
aws cognito-idp describe-user-pool-client --user-pool-id
MyUserPoolID --client-id MyClientID
```

See the AWS CLI command reference for more information: describe-user-pool-client (https://docs.aws.amazon.com/cli/latest/reference/cognito-idp/describe-user-pool-client.html)

AWS API: DescribeUserPoolClient (https://docs.aws.amazon.com/cognito-user-identity-pools/latest/APIReference/API_DescribeUserPoolClient.html)

Listing all app client information in a user pool (AW CLI and AWS API)

```
aws cognito-idp list-user-pool-clients --user-pool-id
"MyUserPoolID" --max-results 3
```

See the AWS CLI command reference for more information: list-user-pool-clients (https://docs.aws.amazon.com/cli/latest/reference/cognito-idp/list-user-pool-clients.html) .

AWS API: ListUserPoolClients (https://docs.aws.amazon.com/cognito-user-identity-pools/latest/APIReference/API_ListUserPoolClients.html)

Deleting a user pool app client (AWS CLI and AWS A

```
aws cognito-idp delete-user-pool-client --user-pool-id
"MyUserPoolID" --client-id "MyAppClientID"
```

See the AWS CLI command reference for more information: delete-user-pool-client (https://docs.aws.amazon.com/cli/latest/reference/cognito-idp/delete-user-pool-client.html)

AWS API: DeleteUserPoolClient (https://docs.aws.amazon.com/cognito-user-identity-pools/latest/APIReference/API_DeleteUserPoolClient.html)

View related pages *Abstracts generated by AI*

1

Cognito › developerguide
Scopes, M2M, and APIs with resource servers...
Amazon Cognito enables authorizing API requests, managing machine identities, issuing access tokens, configuring resource servers, defining custom scopes, requesting OIDC scopes, and authorizing user attributes.
October 27, 2025

Cognito › developerguide
The redirect and authorization endpoi...
Amazon Cognito authorization server handles user authentication, authorization code grants, implicit grants, re-authentication, silent authentication, user profile scopes, and authorization server errors.
October 26, 2025

Cognito › developerguide
Authorization model for API and SDK...
Amazon Cognito API manages user authentication flows, implements server-side and client-side authentication options, integrates external identity providers, authorizes API requests with tokens, and manag user pools resources.
October 27, 2025

Discover highly rated pages *Abstracts generated by AI*

1

Cognito › developerguide
What is Amazon Cognito?...
Amazon Cognito authenticates users, authorizes AWS resource access, issues temporary AWS credentials, integrates with identity providers, manages user pools and identity pools, configures role-based access control.

Cognito › developerguide
Understanding user pool JSON web toke...
Authenticating users with tokens, storing tokens securely, customizing access and ID tokens, understanding ID token claims, understanding access token claims, refreshing tokens for new access, revoking tokens to end sessions.

Cognito › developerguide
Common Amazon Cognito scenarios...
Amazon Cognito enabl user authentication, access to back-end resources, AWS services via API Gateway, Lambd identity pools, third-pa IdPs, and AppSync resources.

October 26, 2025

October 26, 2025

October 26, 2025