
Amazon Cognito

Developer Guide



Amazon Cognito: Developer Guide

Copyright © Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is Amazon Cognito?	1
Features of Amazon Cognito	2
Getting started with Amazon Cognito	2
Regional availability	3
Pricing for Amazon Cognito	3
Using the Amazon Cognito console	3
Getting started with Amazon Cognito	6
Get an AWS account and your root user credentials	6
Creating an IAM user	7
Signing in as an IAM user	8
Creating IAM user access keys	8
Common Amazon Cognito scenarios	10
Authenticate with a user pool	10
Access your server-side resources	10
Access resources with API Gateway and Lambda	11
Access AWS services with a user pool and an identity pool	12
Authenticate with a third party and access AWS services with an identity pool	12
Access AWS AppSync resources with Amazon Cognito	13
Tutorials	14
Creating a user pool	14
Related Resources	15
Creating an identity pool	15
Related resources	15
Cleaning up your AWS resources	15
Integrating with apps	17
Amazon Cognito authentication with the AWS Amplify framework	17
Authentication with amazon-cognito-identity-js	17
Authentication with AWS SDKs	18
Multi-tenant application best practices	19
User-pool-based multi-tenancy	19
App-client-based multi-tenancy	20
Group-based multi-tenancy	20
Custom-attribute-based multi-tenancy	20
Multi-tenancy security recommendations	20
Amazon Cognito user pools	22
Getting started with user pools	23
Prerequisite: Sign up for an AWS account	23
Step 1. Create a user pool	23
Step 2. Add an app client and set up the hosted UI	24
Step 3. Add social sign-in to a user pool (optional)	27
Step 4. Add sign-in with a SAML identity provider to a user pool (optional)	34
Next steps	36
Updating a user pool	37
Updating a user pool with the Amazon Cognito API or AWS CLI	37
Using the hosted UI	38
Setting up the hosted UI with AWS Amplify	39
Setting up the hosted UI with the Amazon Cognito console	39
Configuring an app client	42
Configuring a domain	47
Customizing the built-in webpages	53
Defining resource servers	57
Adding sign-in through a third party	60
How federated sign-in works in Amazon Cognito user pools	60
Adding social identity providers	62

Adding SAML providers	68
Adding OIDC providers	79
Specifying attribute mappings	86
Linking federated users to an existing user profile	90
Using Lambda triggers	92
Important considerations	93
Adding a user pool trigger	95
User pool Lambda trigger event	95
User pool Lambda trigger common parameters	96
Lambda trigger sources	97
Pre sign-up Lambda trigger	98
Post confirmation Lambda trigger	105
Pre authentication Lambda trigger	108
Post authentication Lambda trigger	110
Challenge Lambda triggers	114
Pre token generation Lambda trigger	123
Migrate user Lambda trigger	127
Custom message Lambda trigger	131
Custom sender Lambda triggers	136
Using Amazon Pinpoint analytics	145
Find Amazon Cognito and Amazon Pinpoint Region mappings	145
Managing users	147
Signing up and confirming user accounts	148
Creating users as administrator	157
Adding groups to a user pool	162
Managing and searching for users	165
Recovering user accounts	169
Importing users into a user pool	170
Email settings	181
Default email functionality	181
Amazon SES email configuration	181
Configuring the email account	183
SMS message settings	187
Setting up SMS messaging for the first time in Amazon Cognito user pools	187
Using tokens	191
Using the ID token	191
Using the access token	194
Using the refresh token	195
Revoking tokens	197
Verifying a JSON web token	198
Accessing resources after sign-in	201
Accessing server-side resources	10
Accessing resources with API Gateway and Lambda	202
Accessing AWS resources using an identity pool	203
User pools console reference	205
User pool name	206
Users and groups	206
Attributes	206
Password requirements	214
Admin create user policy	215
Email or phone verification	215
Message customizations	217
Tags	222
Devices	222
App clients	223
Triggers	226
Review settings	226

Analytics	226
App client settings	227
Domain name	228
UI customization	229
Resource servers	230
Identity providers	231
Attribute mapping	237
Managing error responses	239
Amazon Cognito identity pools	242
Getting started with identity pools	242
Sign up for an AWS account	243
Create an identity pool in Amazon Cognito	243
Install the Mobile or JavaScript SDK	243
Integrate the identity providers	244
Get credentials	244
Using identity pools	244
User IAM roles	245
Authenticated and unauthenticated identities	245
Enable or disable unauthenticated identities	245
Change the role associated with an identity type	245
Enable or edit authentication providers	246
Delete an identity pool	246
Delete an identity from an identity pool	247
Managing datasets	247
Bulk publish data	248
Enable push synchronization	248
Set up Amazon Cognito Streams	248
Set up Amazon Cognito Events	248
Identity pools concepts	248
Identity pools authentication flow	249
IAM roles	254
Role trust and permissions	259
Using attributes for access control	260
Using attributes for access control with Amazon Cognito identity pools	261
Using attributes for access control policy example	261
Disable attributes for access control	263
Default provider mappings	263
Role-based access control	264
Creating roles for role mapping	264
Granting pass role permission	265
Using tokens to assign roles to users	265
Using rule-based mapping to assign roles to users	266
Token claims to use in rule-based mapping	267
Best practices for role-based access control	268
Getting credentials	268
Android	269
iOS - Objective-C	270
iOS - Swift	271
JavaScript	272
Unity	273
Xamarin	273
Accessing AWS services	274
Android	274
iOS - Objective-C	275
iOS - Swift	275
JavaScript	275
Unity	275

Xamarin	275
Identity pools external identity providers	276
Facebook	276
Login with Amazon	281
Google	284
Sign in with Apple	291
Open ID Connect providers	295
SAML identity providers	297
Developer authenticated identities	298
Understanding the authentication flow	299
Define a developer provider name and associate it with an identity pool	299
Implement an identity provider	299
Updating the logins map (Android and iOS only)	305
Getting a token (server side)	305
Connect to an existing social identity	306
Supporting transition between providers	307
Switching identities	309
Android	309
iOS - objective-C	310
iOS - swift	310
JavaScript	310
Unity	311
Xamarin	311
Amazon Cognito Sync	312
Getting started with Amazon Cognito Sync	312
Sign up for an AWS account	312
Set up an identity pool in Amazon Cognito	313
Store and sync data	313
Synchronizing data	313
Initializing the Amazon Cognito Sync client	313
Understanding datasets	315
Reading and writing data in datasets	316
Synchronizing local data with the sync store	318
Handling callbacks	320
Android	320
iOS - Objective-C	322
iOS - Swift	324
JavaScript	326
Unity	328
Xamarin	330
Push sync	332
Create an Amazon Simple Notification Service (Amazon SNS) app	332
Enable push sync in the Amazon Cognito console	332
Use push sync in your app: Android	333
Use push sync in your app: iOS - Objective-C	334
Use push sync in your app: iOS - Swift	336
Amazon Cognito Streams	338
Amazon Cognito Events	340
Security	344
Data protection	344
Data encryption	345
Identity and access management	345
Audience	346
Authenticating with identities	346
Managing access using policies	348
How Amazon Cognito works with IAM	349
Identity-based policy examples	355

Troubleshooting	358
Using service-linked roles	360
Authentication	362
Logging and monitoring	369
Tracking quotas and usage in CloudWatch and Service Quotas	369
Metrics for Amazon Cognito user pools	370
Dimensions for Amazon Cognito user pools	374
Use the Service Quotas console to track metrics	375
Use the CloudWatch console to track metrics	375
Create a CloudWatch alarm for a quota	376
Logging Amazon Cognito API calls with AWS CloudTrail	376
Analyzing Amazon Cognito CloudTrail events with Amazon CloudWatch Logs Insights	378
Compliance validation	379
Resilience	380
Regional data considerations	380
Infrastructure security	380
Configuration and vulnerability analysis	381
Security best practices	381
Adding MFA	381
Adding advanced security	387
Case sensitivity	398
AWS managed policies	398
Policy updates	399
Tagging resources	401
Supported resources	401
Tag restrictions	401
Managing tags with the console	402
AWS CLI examples	402
Assigning tags	402
Viewing tags	403
Removing tags	404
Applying tags when you create resources	404
API actions	405
API actions for user pool tags	405
API actions for identity pool tags	405
Quotas	406
Operation quotas	406
Quota categorization	406
Amazon Cognito user pools API operations with special request rate handling	406
Monthly active users	407
Amazon Cognito user pools API operation categories and request rate quotas	407
Track quota usage	412
Identify quota requirements	413
Optimize quotas	413
Requesting a quota increase	414
Amazon Cognito identity pools (federated identities) API operation request rate quotas	414
Resource quotas	415
API references	419
User pool endpoints reference	419
Authorize endpoint	420
Token endpoint	425
UserInfo endpoint	429
Login endpoint	430
Logout endpoint	432
Revoke endpoint	433
User pools API reference	435
Identity pools API reference	435

Cognito sync API reference	435
Document history	436

What is Amazon Cognito?

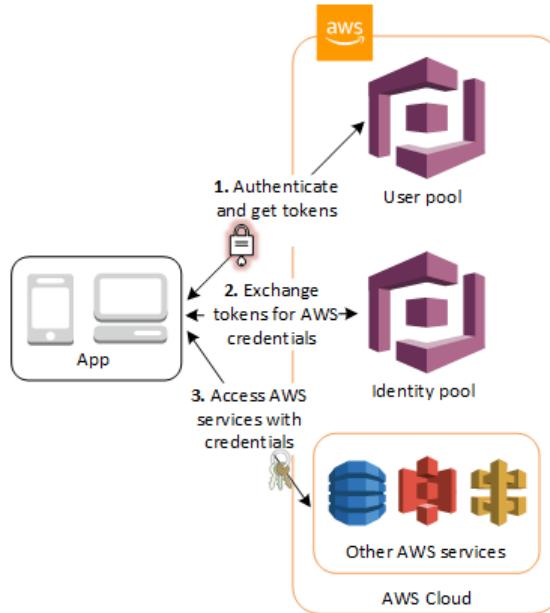
Amazon Cognito provides authentication, authorization, and user management for your web and mobile apps. Your users can sign in directly with a user name and password, or through a third party such as Facebook, Amazon, Google or Apple.

The two main components of Amazon Cognito are user pools and identity pools. User pools are user directories that provide sign-up and sign-in options for your app users. Identity pools enable you to grant your users access to other AWS services. You can use identity pools and user pools separately or together.

An Amazon Cognito user pool and identity pool used together

See the diagram for a common Amazon Cognito scenario. Here the goal is to authenticate your user, and then grant your user access to another AWS service.

1. In the first step your app user signs in through a user pool and receives user pool tokens after a successful authentication.
2. Next, your app exchanges the user pool tokens for AWS credentials through an identity pool.
3. Finally, your app user can then use those AWS credentials to access other AWS services such as Amazon S3 or DynamoDB.



For more examples using identity pools and user pools, see [Common Amazon Cognito scenarios \(p. 10\)](#).

Amazon Cognito is compliant with SOC 1-3, PCI DSS, ISO 27001, and is HIPAA-BAA eligible. For more information, see [AWS services in scope](#). See also [Regional data considerations \(p. 380\)](#).

Topics

- [Features of Amazon Cognito \(p. 2\)](#)
- [Getting started with Amazon Cognito \(p. 2\)](#)

- [Regional availability \(p. 3\)](#)
- [Pricing for Amazon Cognito \(p. 3\)](#)
- [Using the Amazon Cognito console \(p. 3\)](#)

Features of Amazon Cognito

User pools

A user pool is a user directory in Amazon Cognito. With a user pool, your users can sign in to your web or mobile app through Amazon Cognito, or federate through a third-party identity provider (IdP). Whether your users sign in directly or through a third party, all members of the user pool have a directory profile that you can access through an SDK.

User pools provide:

- Sign-up and sign-in services.
- A built-in, customizable web UI to sign in users.
- Social sign-in with Facebook, Google, Login with Amazon, and Sign in with Apple, and through SAML and OIDC identity providers from your user pool.
- User directory management and user profiles.
- Security features such as multi-factor authentication (MFA), checks for compromised credentials, account takeover protection, and phone and email verification.
- Customized workflows and user migration through AWS Lambda triggers.

For more information about user pools, see [Getting started with user pools \(p. 23\)](#) and the [Amazon Cognito user pools API reference](#).

Identity pools

With an identity pool, your users can obtain temporary AWS credentials to access AWS services, such as Amazon S3 and DynamoDB. Identity pools support anonymous guest users, as well as the following identity providers that you can use to authenticate users for identity pools:

- Amazon Cognito user pools
- Social sign-in with Facebook, Google, Login with Amazon, and Sign in with Apple
- OpenID Connect (OIDC) providers
- SAML identity providers
- Developer authenticated identities

To save user profile information, your identity pool needs to be integrated with a user pool.

For more information about identity pools, see [Getting started with Amazon Cognito identity pools \(federated identities\) \(p. 242\)](#) and the [Amazon Cognito identity pools API reference](#).

Getting started with Amazon Cognito

For a guide to top tasks and where to start, see [Getting started with Amazon Cognito \(p. 6\)](#).

For videos, articles, documentation, and sample apps, see [Amazon Cognito developer resources](#).

To use Amazon Cognito, you need an AWS account. For more information, see [Using the Amazon Cognito console \(p. 3\)](#).

Regional availability

Amazon Cognito is available in multiple AWS Regions worldwide. In each Region, Amazon Cognito is distributed across multiple Availability Zones. These Availability Zones are physically isolated from each other, but are united by private, low-latency, high-throughput, and highly redundant network connections. These Availability Zones enable AWS to provide services, including Amazon Cognito, with very high levels of availability and redundancy, while also minimizing latency.

For a list of all the Regions where Amazon Cognito is currently available, see [AWS regions and endpoints](#) in the *Amazon Web Services General Reference*. To learn more about the number of Availability Zones that are available in each Region, see [AWS global infrastructure](#).

Pricing for Amazon Cognito

For information about Amazon Cognito pricing, see [Amazon Cognito pricing](#).

Using the Amazon Cognito console

You can use the [Amazon Cognito console](#) to create and manage user pools and identity pools.

This guide provides step-by-step walkthroughs for common Amazon Cognito user pool tasks in both the original Amazon Cognito console (referred to as the *original console*), and the November 2021 console update (the *new console*). The new console update revises workflows for most actions in the Amazon Cognito user pools console. It does not change the Amazon Cognito API that underpins AWS SDKs and the AWS Command Line Interface.

New console experience highlights

- Logical grouping of features
- A user pool creation wizard
- Informative instructions that emphasize the user experience in your app
- Dynamic assistance with user pool configuration
 - Pop-out help text
 - Filtered views
 - App category settings and suggested presets
 - Warnings about irreversible configuration choices
- Updated information on resource limits and the limitations of user pool configuration

To get started with the new Amazon Cognito user pools console, select the link in the invitation banner displayed in your console. You can revert to the original console after previewing the new console experience.

Features retained in the original console

The following Amazon Cognito console workflows are not currently implemented in the new console. You will be redirected to the original console when you access these features in the new Amazon Cognito console.

- [Amazon Cognito identity pools \(federated identities\)](#)
- [Amazon Cognito Sync](#)
- [Importing Users into User Pools From a CSV File](#)

- [User Pool Analytics](#)

To use the Amazon Cognito console

In the remainder of this guide, you will find **Original console** and **New console** tabs where console-specific instructions are provided. Select the tab that corresponds to the console experience you have chosen.

Original console

1. To use Amazon Cognito, you need to [sign up for an AWS account](#).
2. Go to the [Amazon Cognito console](#). You might be prompted for your AWS credentials.
3. To create or edit a user pool, choose **Manage your User Pools**.

For more information, see [Getting started with user pools \(p. 23\)](#).

4. To create or edit an identity pool, choose **Manage Federated Identities**.

For more information, see [Getting started with Amazon Cognito identity pools \(federated identities\) \(p. 242\)](#).

The Amazon Cognito console is a part of the AWS Management Console, which provides information about your account and billing. For more information, see [Working with the AWS Management Console](#).



Add Sign-up and Sign-in

With Cognito User Pools, you can easily and securely add sign-up and sign-in functionality to your mobile and web apps with a fully-managed service that scales to support hundreds of millions of users.



Federate User Identities

With Cognito Federated Identities, your users can sign-in through social identity providers such as Facebook and Twitter, or through your own identity solution, and you can control access to AWS resources from your app.



Synchronize Data Across Devices

With Cognito Sync, your app can save user data, such as preferences and game state, and sync that data to make your users' experiences consistent across their devices and when they are disconnected.

Amazon Cognito documentation and support

[Getting Started Guide for Android](#) | [Getting Started Guide for iOS](#) | [API reference](#) | [Forums](#)

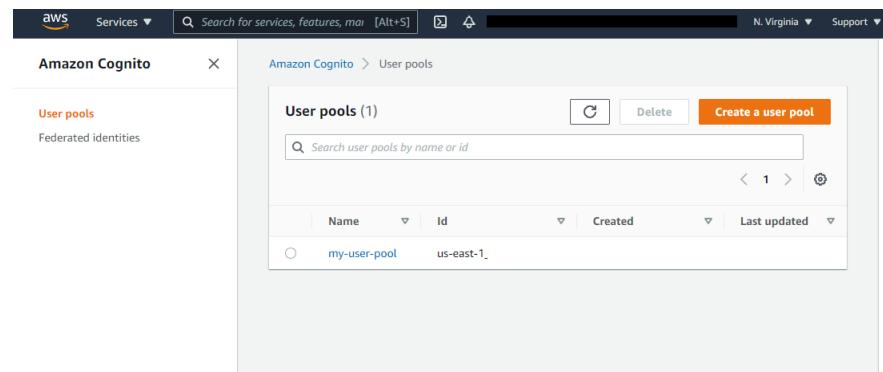


New console

1. To use Amazon Cognito, you need to [sign up for an AWS account](#).

2. Go to the [Amazon Cognito console](#). You might be prompted for your AWS credentials.
3. To create or edit a user pool, choose **User Pools** from the left navigation pane.
For more information, see [Getting started with user pools \(p. 23\)](#).
4. To create or edit an identity pool, choose **Federated identities**. You will be directed to the original console for Amazon Cognito identity pools.
For more information, see [Getting started with Amazon Cognito identity pools \(federated identities\) \(p. 242\)](#).

The Amazon Cognito console is a part of the AWS Management Console, which provides information about your account and billing. For more information, see [Working with the AWS Management Console](#).



Getting started with Amazon Cognito

This section describes the top Amazon Cognito tasks and where to start. For an overview of Amazon Cognito, see [What is Amazon Cognito? \(p. 1\)](#).

The two main components of Amazon Cognito are user pools and identity pools. User pools are user directories that provide sign-up and sign-in options for your web and mobile app users. Identity pools provide AWS credentials to grant your users access to other AWS services. You can use user pools and identity pools separately or together.

Top tasks and where to start
Add sign-up and sign-in with a user pool
<ol style="list-style-type: none">1. Create a user directory with a user pool.2. Add an app to enable the hosted UI.3. Add social sign-in to a user pool.4. Add sign-in through SAML-based identity providers (IdPs) to a user pool.5. Add sign-in through OpenID Connect (OIDC) IdPs to a user pool.6. Install a user pool SDK.7. Customize the built-in hosted web UI sign-in and sign-up pages.8. Configure user pool security features.9. Customize user pool workflows with Lambda triggers.10. Gather data and target campaigns with Amazon Pinpoint analytics.
Manage users in a user pool
<ul style="list-style-type: none">• Sign up and confirm user accounts.• Create user accounts as administrator.• Manage and search user accounts.• Add groups to a user pool.• Import users into a user pool.
Access resources
Common Amazon Cognito scenarios: <ul style="list-style-type: none">• Authenticate with a user pool.• Access backend resources through a user pool.• Access API Gateway and Lambda through a user pool.• Access AWS services with a user pool and an identity pool.• Access AWS services through a third party and an identity pool.• Access AWS AppSync resources through a user pool or an identity pool.

Get an AWS account and your root user credentials

To access AWS, you must sign up for an AWS account.

To sign up for an AWS account

1. Open <https://portal.aws.amazon.com/billing/signup>.
2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to <https://aws.amazon.com/> and choosing **My Account**.

Creating an IAM user

If your account already includes an IAM user with full AWS administrative permissions, you can skip this section.

When you first create an Amazon Web Services (AWS) account, you begin with a single sign-in identity. That identity has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user*. When you sign in, enter the email address and password that you used to create the account.

Important

We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the [best practice of using the root user only to create your first IAM user](#). Then securely lock away the root user credentials and use them to perform only a few account and service management tasks. To view the tasks that require you to sign in as the root user, see [Tasks that require root user credentials](#).

To create an administrator user for yourself and add the user to an administrators group (console)

1. Sign in to the [IAM console](#) as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

Note

We strongly recommend that you adhere to the best practice of using the **Administrator** IAM user that follows and securely lock away the root user credentials. Sign in as the root user only to perform a few [account and service management tasks](#).

2. In the navigation pane, choose **Users** and then choose **Add users**.
3. For **User name**, enter **Administrator**.
4. Select the check box next to **AWS Management Console access**. Then select **Custom password**, and then enter your new password in the text box.
5. (Optional) By default, AWS requires the new user to create a new password when first signing in. You can clear the check box next to **User must create a new password at next sign-in** to allow the new user to reset their password after they sign in.
6. Choose **Next: Permissions**.
7. Under **Set permissions**, choose **Add user to group**.
8. Choose **Create group**.
9. In the **Create group** dialog box, for **Group name** enter **Administrators**.
10. Choose **Filter policies**, and then select **AWS managed - job function** to filter the table contents.
11. In the policy list, select the check box for **AdministratorAccess**. Then choose **Create group**.

Note

You must activate IAM user and role access to Billing before you can use the `AdministratorAccess` permissions to access the AWS Billing and Cost Management console. To do this, follow the instructions in [step 1 of the tutorial about delegating access to the billing console](#).

12. Back in the list of groups, select the check box for your new group. Choose **Refresh** if necessary to see the group in the list.
13. Choose **Next: Tags**.
14. (Optional) Add metadata to the user by attaching tags as key-value pairs. For more information about using tags in IAM, see [Tagging IAM entities](#) in the *IAM User Guide*.
15. Choose **Next: Review** to see the list of group memberships to be added to the new user. When you are ready to proceed, choose **Create user**.

You can use this same process to create more groups and users and to give your users access to your AWS account resources. To learn about using policies that restrict user permissions to specific AWS resources, see [Access management](#) and [Example policies](#).

Signing in as an IAM user

Sign in to the [IAM console](#) by choosing **IAM user** and entering your AWS account ID or account alias. On the next page, enter your IAM user name and your password.

Note

For your convenience, the AWS sign-in page uses a browser cookie to remember your IAM user name and account information. If you previously signed in as a different user, choose the sign-in link beneath the button to return to the main sign-in page. From there, you can enter your AWS account ID or account alias to be redirected to the IAM user sign-in page for your account.

Creating IAM user access keys

Access keys consist of an access key ID and secret access key, which are used to sign programmatic requests that you make to AWS. If you don't have access keys, you can create them from the AWS Management Console. As a best practice, do not use the AWS account root user access keys for any task where it's not required. Instead, [create a new administrator IAM user](#) with access keys for yourself.

The only time that you can view or download the secret access key is when you create the keys. You cannot recover them later. However, you can create new access keys at any time. You must also have permissions to perform the required IAM actions. For more information, see [Permissions required to access IAM resources](#) in the *IAM User Guide*.

To create access keys for an IAM user

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Users**.
3. Choose the name of the user whose access keys you want to create, and then choose the **Security credentials** tab.
4. In the **Access keys** section, choose **Create access key**.
5. To view the new access key pair, choose **Show**. You will not have access to the secret access key again after this dialog box closes. Your credentials will look something like this:

- Access key ID: AKIAIOSFODNN7EXAMPLE
 - Secret access key: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
6. To download the key pair, choose **Download .csv file**. Store the keys in a secure location. You will not have access to the secret access key again after this dialog box closes.
- Keep the keys confidential in order to protect your AWS account and never email them. Do not share them outside your organization, even if an inquiry appears to come from AWS or Amazon.com. No one who legitimately represents Amazon will ever ask you for your secret key.
7. After you download the .csv file, choose **Close**. When you create an access key, the key pair is active by default, and you can use the pair right away.

Related topics

- [What is IAM?](#) in the *IAM User Guide*
- [AWS security credentials](#) in *AWS General Reference*

Common Amazon Cognito scenarios

This topic describes six common scenarios for using Amazon Cognito.

The two main components of Amazon Cognito are user pools and identity pools. User pools are user directories that provide sign-up and sign-in options for your web and mobile app users. Identity pools provide AWS credentials to grant your users access to other AWS services.

A user pool is a user directory in Amazon Cognito. Your app users can sign in either directly through a user pool, or federate through a third-party identity provider (IdP). The user pool manages the overhead of handling the tokens that are returned from social sign-in through Facebook, Google, Amazon, and Apple, and from OpenID Connect (OIDC) and SAML IdPs. Whether your users sign in directly or through a third party, all members of the user pool have a directory profile that you can access through an SDK.

With an identity pool, your users can obtain temporary AWS credentials to access AWS services, such as Amazon S3 and DynamoDB. Identity pools support anonymous guest users, as well as federation through third-party IdPs.

Topics

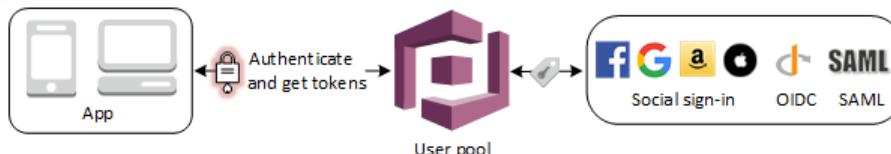
- [Authenticate with a user pool \(p. 10\)](#)
- [Access your server-side resources with a user pool \(p. 10\)](#)
- [Access resources with API Gateway and Lambda with a user pool \(p. 11\)](#)
- [Access AWS services with a user pool and an identity pool \(p. 12\)](#)
- [Authenticate with a third party and access AWS services with an identity pool \(p. 12\)](#)
- [Access AWS AppSync resources with Amazon Cognito \(p. 13\)](#)

Authenticate with a user pool

You can enable your users to authenticate with a user pool. Your app users can sign in either directly through a user pool, or federate through a third-party identity provider (IdP). The user pool manages the overhead of handling the tokens that are returned from social sign-in through Facebook, Google, Amazon, and Apple, and from OpenID Connect (OIDC) and SAML IdPs.

After a successful authentication, your web or mobile app will receive user pool tokens from Amazon Cognito. You can use those tokens to retrieve AWS credentials that allow your app to access other AWS services, or you might choose to use them to control access to your server-side resources, or to the Amazon API Gateway.

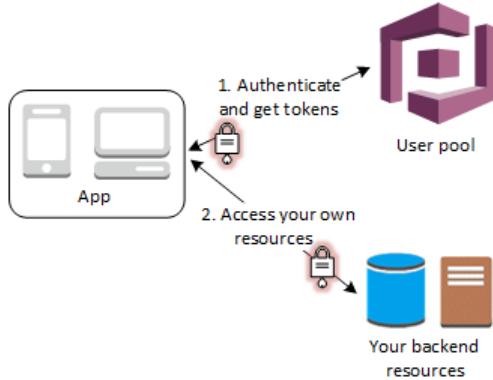
For more information, see [User pool authentication flow \(p. 364\)](#) and [Using tokens with user pools \(p. 191\)](#).



Access your server-side resources with a user pool

After a successful user pool sign-in, your web or mobile app will receive user pool tokens from Amazon Cognito. You can use those tokens to control access to your server-side resources. You can also create

user pool groups to manage permissions, and to represent different types of users. For more information on using groups to control access your resources see [Adding groups to a user pool \(p. 162\)](#).



Once you configure a domain for your user pool, Amazon Cognito provisions a hosted web UI that allows you to add sign-up and sign-in pages to your app. Using this OAuth 2.0 foundation you can create your own resource server to enable your users to access protected resources. For more information see [Defining resource servers for your user pool \(p. 57\)](#).

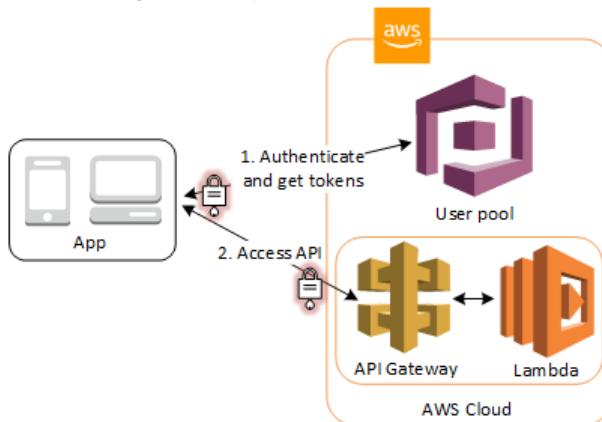
For more information about user pool authentication see [User pool authentication flow \(p. 364\)](#) and [Using tokens with user pools \(p. 191\)](#).

Access resources with API Gateway and Lambda with a user pool

You can enable your users to access your API through API Gateway. API Gateway validates the tokens from a successful user pool authentication, and uses them to grant your users access to resources including Lambda functions, or your own API.

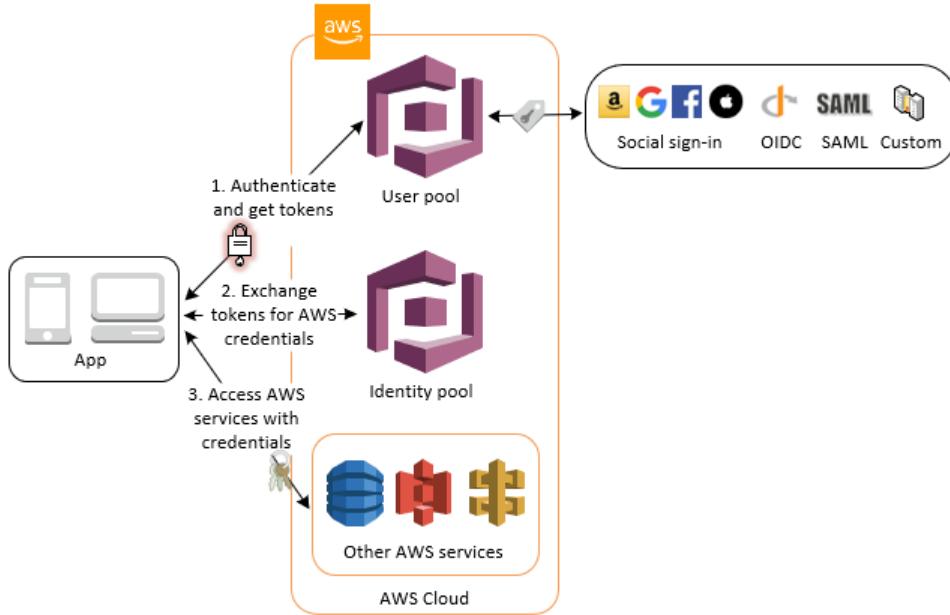
You can use groups in a user pool to control permissions with API Gateway by mapping group membership to IAM roles. The groups that a user is a member of are included in the ID token provided by a user pool when your app user signs in. For more information on user pool groups See [Adding groups to a user pool \(p. 162\)](#).

You can submit your user pool tokens with a request to API Gateway for verification by an Amazon Cognito authorizer Lambda function. For more information on API Gateway, see [Using API Gateway with Amazon Cognito user pools](#).



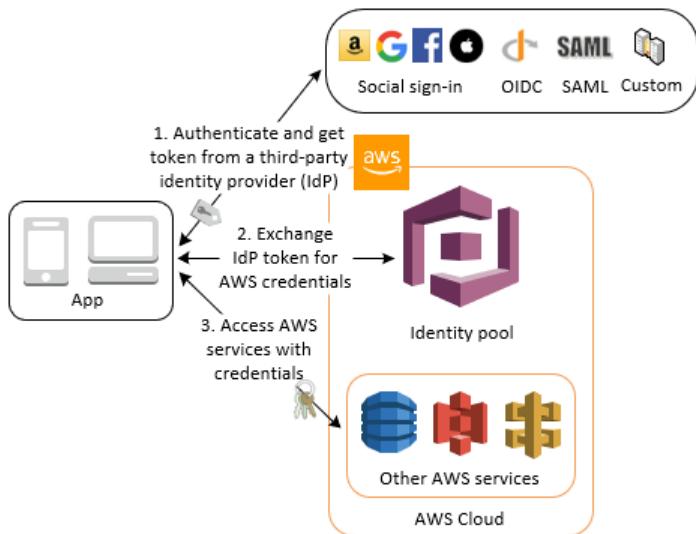
Access AWS services with a user pool and an identity pool

After a successful user pool authentication, your app will receive user pool tokens from Amazon Cognito. You can exchange them for temporary access to other AWS services with an identity pool. For more information, see [Accessing AWS services using an identity pool after sign-in \(p. 203\)](#) and [Getting started with Amazon Cognito identity pools \(federated identities\) \(p. 242\)](#).



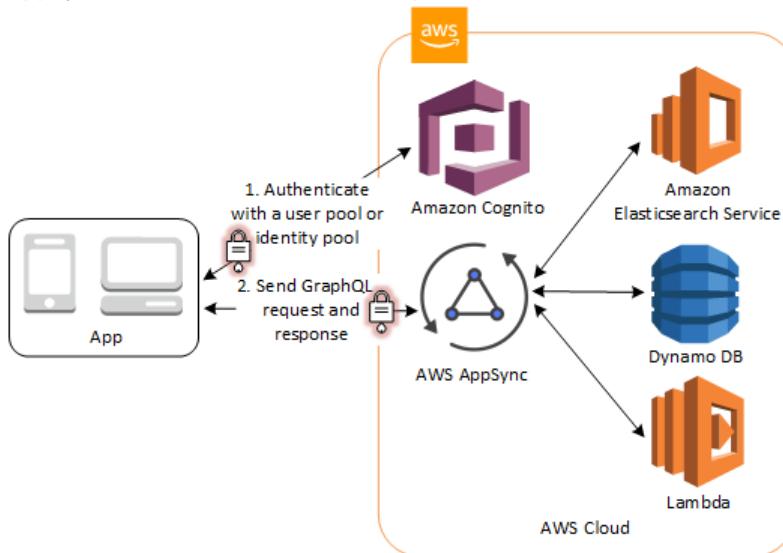
Authenticate with a third party and access AWS services with an identity pool

You can enable your users access to AWS services through an identity pool. An identity pool requires an IdP token from a user that's authenticated by a third-party identity provider (or nothing if it's an anonymous guest). In exchange, the identity pool grants temporary AWS credentials that you can use to access other AWS services. For more information, see [Getting started with Amazon Cognito identity pools \(federated identities\) \(p. 242\)](#).



Access AWS AppSync resources with Amazon Cognito

You can grant your users access to AWS AppSync resources with tokens from a successful Amazon Cognito authentication (from a user pool or an identity pool). For more information, see [Access AWS AppSync and data sources with user pools or federated identities](#).



Amazon Cognito tutorials

The two main components of Amazon Cognito are user pools and identity pools. User pools are user directories that provide sign-up and sign-in options for your web and mobile app users. Identity pools provide AWS credentials to grant your users access to other AWS services.

Topics

- [Tutorial: Creating a user pool \(p. 14\)](#)
- [Tutorial: Creating an identity pool \(p. 15\)](#)
- [Tutorial: Cleaning up your AWS resources \(p. 15\)](#)

Tutorial: Creating a user pool

With a user pool, your users can sign in to your web or mobile app through Amazon Cognito.

Original console

To create a user pool

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **Manage User Pools**.
3. Choose **Create a user pool**.
4. Enter a name for your user pool and choose **Review defaults** to save the name.
5. On the **Review** page, choose **Create pool**.

New console

To create a user pool

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **User Pools**.
3. In the top-right corner of the page, choose **Create a user pool** to start the user pool creation wizard.
4. In **Configure sign-in experience**, choose the federated providers that you will use with this user pool. For more information, see [Adding User Pool Sign-in Through a Third Party](#).
5. In **Configure security requirements**, choose your password policy, multi-factor authentication (MFA) requirements, and user account recovery options. For more information, see [Security in Amazon Cognito](#).
6. In **Configure sign-up experience**, determine how new users will verify their identities when signing up, and which attributes should be required or optional during the user sign-up flow. For more information, see [Managing users in user pools](#).
7. In **Configure message delivery**, configure integration with Amazon Simple Email Service and Amazon Simple Notification Service to send email and SMS messages to your users for sign-up, account confirmation, MFA, and account recovery. For more information, see [Email Settings for Amazon Cognito User Pools](#) and [SMS message settings for Amazon Cognito user pools](#).
8. In **Integrate your app**, name your user pool, configure the hosted UI, and create an app client. For more information, see [Add an App to Enable the Hosted Web UI](#)

9. Review your choices in the **Review and create** screen and modify any selections you wish to. When you are satisfied with your user pool configuration, select **Create user pool** to proceed.

Related Resources

For more information on user pools, see [Amazon Cognito user pools \(p. 22\)](#).

See also [User pool authentication flow \(p. 364\)](#) and [Using tokens with user pools \(p. 191\)](#).

Tutorial: Creating an identity pool

With an identity pool, your users can obtain temporary AWS credentials to access AWS services, such as Amazon S3 and DynamoDB.

Note

In the new Amazon Cognito console experience, you can manage identity pools from the **Federated identities** link on the left navigation bar.

To create an identity pool

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **Manage Identity Pools**.
3. Choose **Create new identity pool**.
4. Enter a name for your identity pool.
5. To enable unauthenticated identities, select **Enable access to unauthenticated identities** from the **Unauthenticated identities** collapsible section.
6. Choose **Create Pool**.
7. You will be prompted for access to your AWS resources.

Choose **Allow** to create the two default roles associated with your identity pool: one for unauthenticated users and one for authenticated users. These default roles provide your identity pool access to Amazon Cognito Sync. You can modify the roles associated with your identity pool in the IAM console.

8. Make a note of your identity pool Id number. You will use it to set up policies that will allow your app users to access other AWS services, such as Amazon Simple Storage Service or DynamoDB

Related resources

For more information on identity pools, see [Amazon Cognito identity pools \(federated identities\) \(p. 242\)](#).

For an example of using an identity pool with Amazon S3, see [Uploading Photos to Amazon S3 from a Browser](#).

Tutorial: Cleaning up your AWS resources

To delete an identity pool

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **Manage Identity Pools**.

3. Choose the name of the identity pool that you want to delete. The **Dashboard page** for your identity pool appears.
4. In the top-right corner of the Dashboard page, choose **Edit identity pool**. The **Edit identity pool** page appears.
5. Scroll down and choose **Delete identity pool** to expand it.
6. Choose **Delete identity pool**.
7. Choose **Delete pool**.

Original console

To delete a user pool

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **Manage User Pools**.
3. Choose the user pool you created in the previous step.
4. On the **Domain name** page under **App integration**, select **Delete domain**.
5. Choose **Delete domain** when prompted to confirm.
6. Go to the **General Settings** page.
7. Select **Delete pool** in the upper right corner of the page.
8. Enter **delete** and choose **Delete pool** when prompted to confirm.

New console

To delete a user pool

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. From the navigation pane, choose **User Pools**.
3. If you have not created a domain for your user pool, select the radio button next to a user pool and select **Delete**. Enter the name of the user pool to confirm, and stop here.
4. If you have created a domain for your user pool, select the user pool.
5. Navigate to the **App integration** tab for your user pool.
6. Next to **Domain**, choose **Actions** and select **Delete Cognito domain** or **Delete custom domain**.
7. Enter the domain name to confirm deletion.
8. Return to the **User pools** list and select the radio button next to your user pool. Select **Delete** and enter the name of the user pool to confirm.

Integrating Amazon Cognito with web and mobile apps

When new users discover your app, or when existing users return to it, their first task is to sign up or sign in. When you integrate Amazon Cognito with your client code, you connect your app to AWS resources that aid authentication and authorization workflows. For example, your app uses the Amazon Cognito API to create new users in your user pool, retrieve user pool tokens, and obtain temporary credentials from your identity pool. To integrate Amazon Cognito with your web or mobile app, use AWS SDKs and libraries.

Although Amazon Cognito offers visual tools such as AWS Management Console integration and the hosted UI, AWS has designed the service to work with your app code. You can only configure certain components of Amazon Cognito with the API or the AWS Command Line Interface. For example, you can only register a user for time-based one-time password (TOTP) multi-factor authentication (MFA) with a process that starts with [AssociateSoftwareToken](#). Before you use Amazon Cognito authentication and authorization, choose an app platform and prepare your code to integrate with the service.

Amazon Cognito authentication with the AWS Amplify framework

AWS Amplify provides services and libraries for web and mobile developers. With Amplify, you can build apps that integrate with backend environments that AWS services compose. To provision your backend environment, and to integrate AWS services with your client code, use the *Amplify framework*. The framework provides an interactive command line interface (CLI) that helps you configure AWS resources for features that are organized into categories. These categories include analytics, storage, and authentication, among many others. The framework also provides high-level SDKs and libraries for web and mobile platforms, including iOS, Android, and JavaScript. The JavaScript frameworks include React, React Native, Angular, Ionic, and Vue. Each of the SDKs and libraries includes authentication operations that you can use to implement the authentication workflows that Amazon Cognito drives.

To use the Amplify framework to add authentication to your app, see the Amplify authorization documentation for your platform:

- [Amplify authentication for JavaScript](#)
- [Amplify authentication for iOS](#)
- [Amplify authentication for Android](#)

Authentication with amazon-cognito-identity-js

The Amazon Cognito Identity SDK for JavaScript makes it possible for apps to sign up and authenticate users in Amazon Cognito user pools. Apps can also use this SDK to view, delete, and update user attributes in Amazon Cognito user pools. The *amazon-cognito-identity-js* package provides sample code that makes it possible for authenticated users to change their passwords. The package also provides sample code that can initiate and complete recovery of forgotten passwords for unauthenticated users.

[Amazon Cognito Identity SDK for JavaScript](#)

Authentication with AWS SDKs

If you must use a secure backend to build your own identity microservice that interacts with Amazon Cognito, you can use the Amazon Cognito user pools and Amazon Cognito federated identities API.

For details on each API operation, see the [Amazon Cognito user pools API Reference](#) and the [Amazon Cognito API Reference](#). These documents contain *see also* sections with resources for using a variety of SDKs in supported platforms.

Multi-tenant application best practices

You can use Amazon Cognito user pools to secure small multi-tenant applications where the number of tenants and expected volume match the related Amazon Cognito service quota.

Note

Amazon Cognito [Quotas](#) are applied per AWS account and Region. These quotas are shared across all tenants in your application. Review the Amazon Cognito service quotas and make sure that the quota meets the expected volume and the expected number of tenants in your application.

You have four ways to secure multi-tenant applications: user pools, application clients, groups, or custom attributes.

Topics

- [User-pool-based multi-tenancy \(p. 19\)](#)
- [App-client-based multi-tenancy \(p. 20\)](#)
- [Group-based multi-tenancy \(p. 20\)](#)
- [Custom-attribute-based multi-tenancy \(p. 20\)](#)
- [Multi-tenancy security recommendations \(p. 20\)](#)

User-pool-based multi-tenancy

With this design, you can create a user pool for each tenant in your application. This approach provides maximum isolation for each tenant. You can implement different configurations for each tenant. Tenant isolation by user pool gives you flexibility in user-to-tenant mapping. You can create multiple profiles for the same user. However, each user must sign up individually for each tenant they can access. Using this approach, you can set up hosted UI for each tenant independently and redirect users to their tenant-specific instance of your application. You can also use this approach to integrate with backend services like API Gateway.

You can use user-pool-based multi-tenancy in the following scenarios:

- Your application has different configurations for each tenant. For example, data residency requirements, password policy, and MFA configurations can be different for each tenant.
- Your application has complex user-to-tenant role mapping. For example, a single user could be a "Student" in tenant A, and the same user could also be a "Teacher" in tenant B.
- Your application uses the default Amazon Cognito hosted UI as the primary way for native users to authenticate. Native users are those that have been created in the user pool with user name and password.
- Your application has a silo multi-tenant application where each tenant gets a full instance of your application infrastructure for their usage.

Effort level

The development and operation effort to use this approach is high. You must build tenant onboarding and administration components into your application that uses Amazon Cognito API operations and automation tools. These components are necessary to create the required resources for each tenant.

You also must implement a tenant-matching user interface. In addition, you must add logic to your application so that users can sign up and sign in to their corresponding tenant's user pool.

App-client-based multi-tenancy

With application client-based multi-tenancy, you can map the same user to multiple tenants without the need to recreate a user's profile. You can create an application client for each tenant and make the tenant external IdP the only identity provider that this application client can use. For more information see, [Configuring a user pool app client](#).

The hosted UI sets a session cookie in the browser so that it recognizes a user who has already authenticated. When you authenticate users with *native accounts* in a user pool with multiple app clients, their session cookie authenticates them for all app clients in the same user pool. Native accounts are user accounts in the user pool directory that weren't created by federation with a third-party IdP. The session cookie is valid for one hour. You can't change the session cookie duration.

You can use app-client-based multi-tenancy in the following scenarios:

- Your application has the same configurations across all tenants. For example, data residency and password policy are the same across all tenants.
- Your application has a one-to-many mapping between the user and tenants. For example, a single user might have access to multiple tenants using the same profile.
- You have a federation-only multi-tenant application where tenants always use an external IdP to sign in to your application.
- You have a B2B multi-tenant application, and tenant backend services use a client-credentials grant to access your services. In this case, you can create an application client for each tenant and share the client-id and secret with the tenant backend service for machine-to-machine authentication.

Effort level

To use this approach, development effort is high. You must implement tenant-matching logic and a user interface to match a user to the application client for their tenant.

Group-based multi-tenancy

With group-based multi-tenancy, you can associate an Amazon Cognito [user pool group](#) with a tenant. That way, you can use additional functionality through role-based access control (RBAC). For more information see, [Role-based access control](#).

Custom-attribute-based multi-tenancy

With custom-attribute-based multi-tenancy, you can store tenant identification data like `tenant_id` as a [custom attribute](#) in a user's profile. You then handle all multi-tenancy logic in your application and backend services. With this approach, you can use a unified sign-up and sign-in experience for all users. To identify the user's tenant in your application, you can check this custom attribute.

Multi-tenancy security recommendations

To help make your application more secure, we recommend the following:

- Use only a verified email address to authorize user access to a tenant based on domain match. Do not trust email addresses and phone numbers unless your app verifies them, or the external IdP gives a proof of verification. For more details on setting these permissions, see [Attribute Permissions and Scopes](#).
- Use immutable or mutable attributes for the user profile attributes that identify tenants. Administrators must be able to change these attributes. Also, give app clients read-only access to the attributes.
- Use 1:1 mapping between external IdP and application client to prevent unauthorized cross-tenant access. A user who has been authenticated by an external IdP, and who has a valid Amazon Cognito session cookie, can access other tenant apps that trust the same IdP.
- When you implement tenant-matching and authorization logic in your application, restrict users so that they can't modify the criteria that authorize user access to the tenants. Also, if an external IdP is being used for federation, restrict tenant identity provider administrators so that they can't modify user access.

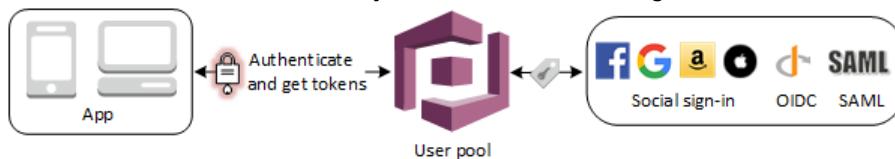
Amazon Cognito user pools

A user pool is a user directory in Amazon Cognito. With a user pool, your users can sign in to your web or mobile app through Amazon Cognito. Your users can also sign in through social identity providers like Google, Facebook, Amazon, or Apple, and through SAML identity providers. Whether your users sign in directly or through a third party, all members of the user pool have a directory profile that you can access through a Software Development Kit (SDK).

User pools provide:

- Sign-up and sign-in services.
- A built-in, customizable web UI to sign in users.
- Social sign-in with Facebook, Google, Login with Amazon, and Sign in with Apple, as well as sign-in with SAML identity providers from your user pool.
- User directory management and user profiles.
- Security features such as multi-factor authentication (MFA), checks for compromised credentials, account takeover protection, and phone and email verification.
- Customized workflows and user migration through AWS Lambda triggers.

After successfully authenticating a user, Amazon Cognito issues JSON web tokens (JWT) that you can use to secure and authorize access to your own APIs, or exchange for AWS credentials.



Amazon Cognito provides token handling through the Amazon Cognito user pools Identity SDKs for JavaScript, Android, and iOS. See [Getting started with user pools \(p. 23\)](#) and [Using tokens with user pools \(p. 191\)](#).

The two main components of Amazon Cognito are user pools and identity pools. Identity pools provide AWS credentials to grant your users access to other AWS services. To enable users in your user pool to access AWS resources, you can configure an identity pool to exchange user pool tokens for AWS credentials. For more information see [Accessing AWS services using an identity pool after sign-in \(p. 203\)](#) and [Getting started with Amazon Cognito identity pools \(federated identities\) \(p. 242\)](#).

Topics

- [Getting started with user pools \(p. 23\)](#)
- [Updating user pool configuration \(p. 37\)](#)
- [Using the Amazon Cognito hosted UI for sign-up and sign-in \(p. 38\)](#)
- [Adding user pool sign-in through a third party \(p. 60\)](#)
- [Customizing user pool workflows with Lambda triggers \(p. 92\)](#)
- [Using Amazon Pinpoint analytics with Amazon Cognito user pools \(p. 145\)](#)
- [Managing users in your user pool \(p. 147\)](#)
- [Email settings for Amazon Cognito user pools \(p. 181\)](#)
- [SMS message settings for Amazon Cognito user pools \(p. 187\)](#)
- [Using tokens with user pools \(p. 191\)](#)
- [Accessing resources after a successful user pool authentication \(p. 201\)](#)

- [User pools reference \(AWS Management Console\) \(p. 205\)](#)
- [Managing error responses \(p. 239\)](#)

Getting started with user pools

Follow these steps to set up and configure a user pool for the first time with the [Amazon Cognito console](#). Use this guide to begin to test the features of Amazon Cognito. The procedures suggest default settings for your new user pool to get you started. For more information, see [Getting started with Amazon Cognito \(p. 6\)](#).

Topics

- [Prerequisite: Sign up for an AWS account \(p. 23\)](#)
- [Step 1. Create a user pool \(p. 23\)](#)
- [Step 2. Add an app client and set up the hosted UI \(p. 24\)](#)
- [Step 3. Add social sign-in to a user pool \(optional\) \(p. 27\)](#)
- [Step 4. Add sign-in with a SAML identity provider to a user pool \(optional\) \(p. 34\)](#)
- [Next steps \(p. 36\)](#)

Prerequisite: Sign up for an AWS account

To use Amazon Cognito, you need an AWS account. If you don't already have one, use the following procedure to sign up:

To sign up for an AWS account

1. Open <https://portal.aws.amazon.com/billing/signup>.
2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

Next step

[Step 1. Create a user pool \(p. 23\)](#)

Step 1. Create a user pool

Using an Amazon Cognito user pool, you can create and maintain a user directory, and add sign-up and sign-in to your mobile app or web application.

Original console

To create a user pool

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **Manage User Pools**.
3. In the top-right corner of the page, choose **Create a user pool**.
4. Enter a name for your user pool, and choose **Review defaults** to save the name.
5. In the top-left corner of the page, choose **Attributes**, choose **Email address or phone number** and **Allow email addresses**, and then choose **Next step** to save.

Note

We recommend that you enable case insensitivity on the `username` attribute before you create your user pool. For example, when this option is selected, users will be able to sign in using either `username` or `Username`. Enabling this option also enables both `preferred_username` and `email` alias to be case insensitive, in addition to the `username` attribute. For more information, see [CreateUserPool](#) in the *Amazon Cognito user pools API Reference*.

6. In the left navigation menu, choose **Review**.
7. Review the user pool information and make any necessary changes. When the information is correct, choose **Create pool**.

New console

To create a user pool

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **User Pools**.
3. In the top-right corner of the page, choose **Create a user pool** to start the user pool creation wizard.
4. In **Configure sign-in experience**, choose the federated providers that you want to use with this user pool. For more information, see [Adding User Pool Sign-in Through a Third Party](#).

Note

The **Make user name case sensitive** option is turned off by default. We recommend that you do not activate this option. When the user name is not case sensitive, users can sign in with either `username` or `Username`. The **Make user name case sensitive** option also governs case sensitivity of the `preferred_username` and `email` aliases. When user name is case sensitive, you must take additional security precautions. For more information, see [User pool case sensitivity \(p. 398\)](#).

5. In **Configure security requirements**, choose your password policy, multi-factor authentication (MFA) requirements, and user account recovery options. For more information, see [Security in Amazon Cognito](#).
6. In **Configure sign-up experience**, determine how new users will verify their identities when signing up, and which attributes should be required or optional during the user sign-up flow. For more information, see [Managing users in user pools](#).
7. In **Configure message delivery**, configure integration with Amazon Simple Email Service (Amazon SES) and Amazon Simple Notification Service (Amazon SNS) to send email and SMS messages to your users for sign-up, account confirmation, MFA, and account recovery. For more information, see [Email Settings for Amazon Cognito User Pools](#) and [SMS message settings for Amazon Cognito user pools](#).
8. In **Integrate your app**, name your user pool, configure the hosted UI, and create an app client. For more information, see [Add an App to Enable the Hosted Web UI](#)
9. Review your choices in the **Review and create** screen and modify any selections you wish to. When you are satisfied with your user pool configuration, select **Create user pool** to proceed.

Next Step

[Step 2. Add an app client and set up the hosted UI \(p. 24\)](#)

Step 2. Add an app client and set up the hosted UI

After you create a user pool, you can create an app to use the built-in webpages for signing up and signing in your users.

Original console

To create an app in your user pool

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **Manage User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. On the navigation bar on the left-side of the page, choose **App clients** under **General settings**.
5. Choose **Add an app client**.
6. Enter a name for your app client.
7. For this exercise, clear the option **Generate client secret**. Using a client secret with client-side authentication, such as the JavaScript used in this exercise, is not secure and not recommended for a production app client. Client secrets should only be used by applications that have a server-side authentication component so that it can secure the client secret.
8. Choose **Create app client**.
9. Note the **App client ID**.
10. Choose **Return to pool details**.
11. Choose **App client settings** from the navigation bar on the left-side of the console page.
12. Select **Cognito User Pool** as one of the **Enabled Identity Providers**.

Note

To sign in with external identity providers (IdPs) such as Facebook, Amazon, Google, and Apple, as well as through OpenID Connect (OIDC) or SAML IdPs, first configure them as described next, and then return to the **App client settings** page to enable them.

13. Enter a callback URL for the Amazon Cognito authorization server to call after users are authenticated. For a web app, the URL should start with `https://`, such as `https://www.example.com`.

For an iOS or Android app, you can use a callback URL such as `myapp://`.
14. Enter a Sign-out URL.
15. Select **Authorization code grant** to return an authorization code that is then exchanged for user pool tokens. Because the tokens are never exposed directly to an end user, they are less likely to become compromised. However, a custom application is required on the backend to exchange the authorization code for user pool tokens. For security reasons, we recommend that you use the authorization code grant flow, together with [Proof Key for Code Exchange \(PKCE\)](#), for mobile apps.
16. Under **Allowed OAuth Flows**, select **Implicit grant** to have user pool JSON web tokens (JWT) returned to you from Amazon Cognito. You can use this flow when there's no backend available to exchange an authorization code for tokens. It's also helpful for debugging tokens.

Note

You can enable both the **Authorization code grant** and the **Implicit code grant**, and then use each grant as needed.

17. Unless you specifically want to exclude one, select the check boxes for all of the **Allowed OAuth scopes**.

Note

Select **Client credentials** only if your app needs to request access tokens on its own behalf, not on behalf of a user.

18. Choose **Save changes**.
19. On the **Domain name** page, type a domain prefix that's available.
20. Make a note of the complete domain address.

21. Choose **Save changes**.

New console

To create an app in your user pool

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
 2. Choose **User Pools**.
 3. Choose an existing user pool from the list, or [create a user pool](#). If you create a new user pool, you will be prompted to set up an app client and configure the hosted UI during the wizard.
 4. Choose the **App integration** tab for your user pool.
 5. Next to **Domain**, choose **Actions**, and then select either **Create custom domain** or **Create Cognito domain**. If you have already configured a user pool domain, choose **Delete Cognito domain** or **Delete custom domain** before creating your new custom domain.
 6. Enter an available domain prefix to use with a **Cognito domain**. For information on setting up a **Custom domain**, see [Using Your Own Domain for the Hosted UI](#).
 7. Choose **Create**.
 8. Navigate back to the **App integration** tab for the same user pool and locate **App clients**. Choose **Create an app client**.
 9. Choose an **Application type**. Some recommended settings will be provided based on your selection. An app that uses the hosted UI is a **Public client**.
 10. Enter an **App client name**.
 11. For this exercise, choose **Don't generate client secret**. The client secret is used by confidential apps that authenticate users from a centralized application. In this exercise, you will present a hosted UI sign-in page to your users and will not require a client secret.
 12. Choose the **Authentication flows** you will allow with your app. Ensure that **USER_SRP_AUTH** has been selected.
 13. Customize **token expiration**, **Advanced security configuration**, and **Attribute read and write permissions** as needed. For more information, see [Configuring App Client Settings](#).
 14. Add a **callback URL** for your app client. This is where you will be directed after hosted UI authentication. You do not need to add an **Allowed sign-out URL** until you are able to implement sign-out in your app.
- For an iOS or Android app, you can use a callback URL such as `myapp://`.
15. Select the **Identity providers** for the app client. At minimum, enable **Cognito user pool** as a provider.

Note

To sign in with external identity providers (IdPs) such as Facebook, Amazon, Google, and Apple, as well as through OpenID Connect (OIDC) or SAML IdPs, first configure them as shown in [Adding user pool sign-in through a third party](#), and then return to the **App client settings** page to enable them.

16. Choose **OAuth 2.0 Grant Types**. Select **Authorization code grant** to return an authorization code that is then exchanged for user pool tokens. Because the tokens are never exposed directly to an end user, they are less likely to become compromised. However, a custom application is required on the backend to exchange the authorization code for user pool tokens. For security reasons, we recommend that you use the authorization code grant flow, together with [Proof Key for Code Exchange \(PKCE\)](#), for mobile apps.

Select **Implicit grant** to have user pool JSON web tokens (JWT) returned to you from Amazon Cognito. You can use this flow when there's no backend available to exchange an authorization code for tokens. It's also helpful for debugging tokens.

Note

You can enable both the **Authorization code grant** and the **Implicit code grant**, and then use each grant as needed.

Select **Client credentials** only if your app needs to request access tokens on its own behalf, not on behalf of a user.

17. Unless you specifically want to exclude one, select all **OpenID Connect scopes**.
18. Select any **Custom scopes** you have configured. Custom scopes are typically used with confidential clients.
19. Choose **Create**.

To view your sign-in page

From your **App client page**, select **View hosted UI** to open a new browser tab to a sign-in page pre-populated with app client ID, scope, grant, and callback URL parameters.

You can view the hosted UI sign-in webpage manually with the following URL. Note the `response_type`. In this case, `response_type=code` for the authorization code grant.

```
https://your_domain/login?  
response_type=code&client_id=your_app_client_id&redirect_uri=your_callback_url
```

You can view the hosted UI sign-in webpage with the following URL for the implicit code grant where `response_type=token`. After a successful sign-in, Amazon Cognito returns user pool tokens to your web browser's address bar.

```
https://your_domain/login?  
response_type=token&client_id=your_app_client_id&redirect_uri=your_callback_url
```

You can find the JSON web token (JWT) identity token after the `#idtoken=` parameter in the response.

The following URL is a sample response from an implicit grant request. Your identity token string will be much longer.

```
https://www.example.com/  
#id_token=123456789tokens123456789&expires_in=3600&token_type=Bearer
```

Amazon Cognito user pools tokens are signed using an RS256 algorithm. You can decode and verify user pool tokens using AWS Lambda, see [Decode and verify Amazon Cognito JWT tokens](#) on the AWS GitHub website.

Your domain is shown on the **Domain name** page. Your app client ID and callback URL are shown on the **General settings** page. If the changes you made in the console do not appear immediately, wait a few minutes and then refresh your browser.

Next step

[Step 3. Add social sign-in to a user pool \(optional\) \(p. 27\)](#)

Step 3. Add social sign-in to a user pool (optional)

You can enable your app users to sign in through a social identity provider (IdP) such as Facebook, Google, Amazon, and Apple. Whether your users sign in directly or through a third party, all users have

a profile in the user pool. Skip this step if you don't want to add sign in through a social sign-in identity provider.

Step 1: Register with a social IdP

Before you create a social IdP with Amazon Cognito, you must register your application with the social IdP to receive a client ID and client secret.

To register an app with Facebook

1. Create a [developer account with Facebook](#).
2. [Sign in](#) with your Facebook credentials.
3. From the **My Apps** menu, choose **Create New App**.
4. Enter a name for your Facebook app and choose **Create App ID**.
5. On the left navigation bar, choose **Settings**, and then choose **Basic**.
6. Note the **App ID** and the **App Secret**. You will use them in the next section.
7. Choose **+ Add Platform** from the bottom of the page.
8. Choose **Website**.
9. Under **Website**, enter a sign-in URL for your app client endpoint into **Site URL**. Your sign-in URL should be in the following format:

```
https://your_user_pool_domain/login?  
response_type=code&client_id=your_app_client_id&redirect_uri=your_callback_url
```

10. Choose **Save changes**.
11. For **App Domains**, enter your user pool domain.

```
https://your_user_pool_domain
```

12. Choose **Save changes**.
13. From the navigation bar, choose **Products**, and then **Set up from Facebook Login**.
14. From the navigation bar, choose **Facebook Login** and then **Settings**.

Enter your redirect URL into **Valid OAuth Redirect URLs**. The redirect URL will consist of your user pool domain with the /oauth2/idpresponse endpoint.

```
https://your_user_pool_domain/oauth2/idpresponse
```

15. Choose **Save changes**.

To register an app with Amazon

1. Create a [developer account with Amazon](#).
2. [Sign in](#) with your Amazon credentials.
3. You need to create an Amazon security profile to receive the Amazon client ID and client secret.

Choose **Apps and Services** from navigation bar at the top of the page and then choose **Login with Amazon**.

4. Choose **Create a Security Profile**.
5. Enter a **Security Profile Name**, a **Security Profile Description**, and a **Consent Privacy Notice URL**.

6. Choose **Save**.
7. Choose **Client ID** and **Client Secret** to show the client ID and secret. You will use them in the next section.
8. Hover over the gear icon and choose **Web Settings**, and then choose **Edit**.
9. Enter your user pool domain into **Allowed Origins**.

```
https://<your-user-pool-domain>
```

10. Enter your user pool domain with the /oauth2/idpresponse endpoint into **Allowed Return URLs**.

```
https://<your-user-pool-domain>/oauth2/idpresponse
```

11. Choose **Save**.

To register an app with Google

For more information about OAuth 2.0 in the Google Cloud platform, see [Learn about authentication & authorization](#) in the Google Workspace for Developers documentation.

1. Create a [developer account with Google](#).
2. Sign in to the [Google Cloud Platform console](#).
3. From the top navigation bar, choose **Select a project**. If you already have a project in the Google platform, this menu displays your default project instead.
4. Select **NEW PROJECT**.
5. Enter a name for your product and then choose **CREATE**.
6. On the left navigation bar, choose **APIs and Services**, then **Oauth consent screen**.
7. Enter App information, an **App domain**, **Authorized domains**, and **Developer contact information**. Your **Authorized domains** must include amazoncognito.com and the root of your custom domain, for example example.com. Choose **SAVE AND CONTINUE**.
8. 1. Under **Scopes**, choose **Add or remove scopes**, and choose, at minimum, the following OAuth scopes.
 1. .../auth/userinfo.email
 2. .../auth/userinfo.profile
 3. openid
9. Under **Test users**, choose **Add users**. Enter your email address and any other authorized test users, then choose **SAVE AND CONTINUE**.
10. Expand the left navigation bar again, and choose **APIs and Services**, then **Credentials**.
11. Choose **CREATE CREDENTIALS**, then **OAuth client ID**.
12. Choose an **Application type** and give your client a **Name**.
13. Under **Authorized JavaScript origins**, choose **ADD URI**. Enter your user pool domain.

```
https://<your-user-pool-domain>
```

14. Under **Authorized redirect URIs**, choose **ADD URI**. Enter the path to the /oauth2/idpresponse endpoint of your user pool domain.

```
https://<your-user-pool-domain>/oauth2/idpresponse
```

15. Choose **CREATE**.

16. Securely store the values the Google displays under **Your client ID** and **Your client secret**. Provide these values to Amazon Cognito when you add a Google IdP.

To register an app with Apple

For more information about setting up Sign in with Apple, see [Configuring Your Environment for Sign in with Apple](#) in the Apple Developer documentation.

1. Create a [developer account with Apple](#).
2. [Sign in](#) with your Apple credentials.
3. On the left navigation bar, choose **Certificates, Identifiers & Profiles**.
4. On the left navigation bar, choose **Identifiers**.
5. On the **Identifiers** page, choose the + icon.
6. On the **Register a New Identifier** page, choose **App IDs**, and then choose **Continue**.
7. On the **Select a type** page, choose **App**, then choose **Continue**.
8. On the **Register an App ID** page, do the following:
 1. Under **Description**, enter a description.
 2. Under **App ID Prefix**, enter a **Bundle ID**. Make a note of the value under **App ID Prefix**. You will use this value after you choose Apple as your identity provider in [Step 2: Add a social IdP to your user pool \(p. 66\)](#).
 3. Under **Capabilities**, choose **Sign In with Apple**, and then choose **Edit**.
 4. On the **Sign in with Apple: App ID Configuration** page, choose to set up the app as either primary or grouped with other App IDs, and then choose **Save**.
 5. Choose **Continue**.
9. On the **Confirm your App ID** page, choose **Register**.
10. On the **Identifiers** page, choose the + icon.
11. On the **Register a New Identifier** page, choose **Services IDs**, and then choose **Continue**.
12. On the **Register a Services ID** page, do the following:
 1. Under **Description**, type a description.
 2. Under **Identifier**, type an identifier. Make a note of this Services ID as you will need this value after you choose Apple as your identity provider in [Step 2: Add a social IdP to your user pool \(p. 66\)](#).
 3. Choose **Continue**, then **Register**.
13. Choose the Services ID you just create from the **Identifiers** page.
 1. Select **Sign In with Apple**, and then choose **Configure**.
 2. On the **Web Authentication Configuration** page, select the app ID that you created earlier as the **Primary App ID**.
 3. Choose the + icon next to **Website URLs**.
 4. Under **Domains and subdomains**, enter your user pool domain without an `https://` prefix.

`<your-user-pool-domain>`

5. Under **Return URLs**, enter the path to the `/oauth2/idpresponse` endpoint of your user pool domain.

`https://<your-user-pool-domain>/oauth2/idpresponse`

6. Choose **Next**, and then **Done**. You don't need to verify the domain.

7. Choose **Continue**, and then choose **Save**.
14. On the left navigation bar, choose **Keys**.
15. On the **Keys** page, choose the + icon.
16. On the **Register a New Key** page, do the following:
 1. Under **Key Name**, enter a key name.
 2. Choose **Sign In with Apple**, and then choose **Configure**.
 3. On the **Configure Key** page and select the app ID that you created earlier as the **Primary App ID**. Choose **Save**.
 4. Choose **Continue**, and then choose **Register**.
17. On the **Download Your Key** page, choose **Download** to download the private key and note the **Key ID** shown, and then choose **Done**. You will need this private key and the **Key ID** value shown on this page after you choose Apple as your identity provider in [Step 2: Add a social IdP to your user pool \(p. 66\)](#).

Step 2: Add a social IdP to your user pool

In this section, you configure a social IdP in your user pool using the client ID and client secret from the previous section.

Original console

To configure a user pool social identity provider with the AWS Management Console

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **Manage User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. On the left navigation bar, choose **Identity providers**.
5. Choose a social identity provider: **Facebook**, **Google**, **Login with Amazon**, or **Sign in with Apple**.
6. Choose from the following steps, based on your choice of social identity provider:
 - **Google and Login with Amazon** — Enter the **app client ID** and **app client secret** generated in the previous section.
 - **Facebook** — Enter the **app client ID** and **app client secret** generated in the previous section, and then choose an API version (for example, version 2.12). We recommend choosing the latest possible version, as each Facebook API has a lifecycle and deprecation date. Facebook scopes and attributes can vary between API versions. We recommend testing your social identity log in with Facebook to ensure that federation works as intended.
 - **Sign in with Apple** — Enter the **Services ID**, **Team ID**, **Key ID**, and **private key** generated in the previous section.
7. Enter the names of the scopes that you want to authorize. Scopes define which user attributes (such as `name` and `email`) you want to access with your app. For Facebook, these should be separated by commas. For Google and Login with Amazon, they should be separated by spaces. For Sign in with Apple, select the check boxes for the scopes you want access to.

Social identity provider	Example scopes
Facebook	<code>public_profile, email</code>
Google	<code>profile email openid</code>

Social identity provider	Example scopes
Login with Amazon	profile postal_code
Sign in with Apple	email name

Your app user is asked to consent to providing these attributes to your app. For more information about their scopes, see the documentation from Google, Facebook, Login with Amazon, or Sign in with Apple.

With Sign in with Apple, the following are user scenarios where scopes might not be returned:

- An end user encounters failures after leaving Apple's sign in page (can be from Internal failures within Amazon Cognito or anything written by the developer)
 - The service ID identifier is used across user pools or other authentication services
 - A developer adds additional scopes after the user signs in. Users only retrieve new information when they authenticate and when they refresh their tokens.
 - A developer deletes the user and then the user signs in again without removing the app from their Apple ID profile
8. Choose **Enable** for the social identity provider that you are configuring.
 9. Choose **App client settings** from the navigation bar.
 10. Select your social identity provider as one of the **Enabled Identity Providers** for your user pool app.
 11. Enter your callback URL into **Callback URL(s)** for your user pool app. This is the URL of the page where your user will be redirected after a successful authentication.

`https://www.example.com`

12. Choose **Save changes**.
13. On the **Attribute mapping** tab, add mappings for at least the required attributes, typically `email`, as follows:
 - a. Select the check box to choose the Facebook, Google, or Amazon attribute name. You can also enter the names of additional attributes that are not listed in the Amazon Cognito console.
 - b. Choose the destination user pool attribute from the drop-down list.
 - c. Choose **Save changes**.
 - d. Choose **Go to summary**.

New console

To configure a user pool social identity provider with the AWS Management Console

1. Go to the [Amazon Cognito console](#). You might be prompted for your AWS credentials.
2. Choose **User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. Choose the **Sign-in experience** tab. Locate **Federated sign-in** and select **Add an identity provider**.
5. Choose a social identity provider: **Facebook**, **Google**, **Login with Amazon**, or **Sign in with Apple**.
6. Choose from the following steps, based on your choice of social identity provider:

- **Google and Login with Amazon** — Enter the **app client ID** and **app client secret** generated in the previous section.
 - **Facebook** — Enter the **app client ID** and **app client secret** generated in the previous section, and then choose an API version (for example, version 2.12). We recommend choosing the latest possible version, as each Facebook API has a lifecycle and deprecation date. Facebook scopes and attributes can vary between API versions. We recommend testing your social identity log in with Facebook to ensure that federation works as intended.
 - **Sign in with Apple** — Enter the **Services ID**, **Team ID**, **Key ID**, and **private key** generated in the previous section.
7. Enter the names of the **Authorized scopes** you want to use. Scopes define which user attributes (such as `name` and `email`) you want to access with your app. For Facebook, these should be separated by commas. For Google and Login with Amazon, they should be separated by spaces. For Sign in with Apple, select the check boxes for the scopes you want access to.

Social identity provider	Example scopes
Facebook	<code>public_profile, email</code>
Google	<code>profile email openid</code>
Login with Amazon	<code>profile postal_code</code>
Sign in with Apple	<code>email name</code>

Your app user is prompted to consent to providing these attributes to your app. For more information about social provider scopes, see the documentation from Google, Facebook, Login with Amazon, or Sign in with Apple.

With Sign in with Apple, the following are user scenarios where scopes might not be returned:

- An end user encounters failures after leaving Apple's sign in page (can be from Internal failures within Amazon Cognito or anything written by the developer)
 - The service ID identifier is used across user pools and/or other authentication services
 - A developer adds additional scopes after the user signs in. Users only retrieve new information when they authenticate and when they refresh their tokens.
 - A developer deletes the user and then the user signs in again without removing the app from their Apple ID profile
8. Map attributes from your identity provider to your user pool. For more information, see [Specifying Identity Provider Attribute Mappings for Your User Pool](#).
9. Choose **Create**.
10. From the **App client integration** tab, choose one of the **App clients** in the list and **Edit hosted UI settings**. Add the new social identity provider to the app client under **Identity providers**.
11. Choose **Save changes**.

Step 3: Test your social IdP configuration

You can create a login URL by using the elements from the previous two sections. Use it to test your social IdP configuration.

```
https://<your_user_pool_domain>/login?  
response_type=code&client_id=<your_client_id>&redirect_uri=https://www.example.com
```

You can find your domain on the user pool **Domain name** console page. The client_id is on the **App client settings** page. Use your callback URL for the **redirect_uri** parameter. This is the URL of the page where your user will be redirected after a successful authentication.

Note

Amazon Cognito cancels authentication requests that do not complete within 5 minutes, and redirects the user to the hosted UI. The page displays a `Something went wrong` error message.

Next step

[Step 4. Add sign-in with a SAML identity provider to a user pool \(optional\) \(p. 34\)](#)

Step 4. Add sign-in with a SAML identity provider to a user pool (optional)

You can enable your app users to sign in through a SAML identity provider (IdP). Whether your users sign in directly or through a third party, all users have a profile in the user pool. Skip this step if you don't want to add sign in through a SAML identity provider.

You need to update your SAML identity provider and configure your user pool. See the documentation for your SAML identity provider for information about how to add your user pool as a relying party or application for your SAML 2.0 identity provider.

You also need to provide an assertion consumer endpoint to your SAML identity provider. Configure this endpoint for SAML 2.0 POST binding in your SAML identity provider:

```
https://<yourDomainPrefix>.auth.<region>.amazoncognito.com/saml2/idpresponse
```

You can find your domain prefix and the region value for your user pool on the **Domain name** tab of the [Amazon Cognito console](#).

For some SAML identity providers, you also need to provide the SP urn / Audience URI / SP Entity ID, in the format:

```
urn:amazon:cognito:sp:<yourUserPoolID>
```

You can find your user pool ID on the **General settings** tab in the [Amazon Cognito console](#).

You should also configure your SAML identity provider to provide attribute values for any attributes that are required in your user pool. Typically, `email` is a required attribute for user pools. In that case, the SAML identity provider should provide an `email` value (claim) in the SAML assertion.

Amazon Cognito user pools support SAML 2.0 federation with post-binding endpoints. This eliminates the need for your app to retrieve or parse SAML assertion responses, because the user pool directly receives the SAML response from your identity provider through a user agent.

Original console

To configure a SAML 2.0 identity provider in your user pool

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **Manage User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).

4. On the left navigation bar, choose **Identity providers**.
5. Choose **SAML** to open the SAML dialog.
6. Under **Metadata document**, upload a metadata document from your SAML IdP. You can also enter a URL that points to the metadata document. For more information, see [Integrating third-party SAML identity providers with Amazon Cognito user pools \(p. 76\)](#).

Note

We recommend that you provide the endpoint URL if it is a public endpoint, rather than uploading a file, because this allows Amazon Cognito to refresh the metadata automatically. Typically metadata refresh happens every 6 hours or before the metadata expires, whichever is earlier.

7. Enter your **SAML Provider name**. For more information on SAML naming see [Choosing SAML identity provider names \(p. 71\)](#).
8. Enter any optional **SAML Identifiers** you want to use.
9. Select **Enable IdP sign out flow** if you want your user to be logged out from both Amazon Cognito and the SAML IdP when they log out from your user pool.

Enabling this flow sends a signed logout request to the SAML IdP when the **Logout-endpoint** is called.

Configure this endpoint for consuming logout responses from your IdP. This endpoint uses post binding.

```
https://<yourDomainPrefix>.auth.<region>.amazoncognito.com/saml2/logout
```

Note

If this option is selected and your SAML identity provider expects a signed log out request, you will also need to configure the signing certificate provided by Amazon Cognito with your SAML IdP.

The SAML IdP will process the signed logout request and log your user out of the Amazon Cognito session.

10. Choose **Create provider**.
 11. On the **Attribute mapping** tab, add mappings for at least the required attributes, typically **email**, as follows:
 - a. Type the SAML attribute name as it appears in the SAML assertion from your identity provider. Your identity provider might offer sample SAML assertions for reference. Some identity providers use simple names, such as **email**, while others use URL-formatted attribute names, such as the following example:
- ```
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
```
- b. Choose the destination user pool attribute from the drop-down list.
12. Choose **Save changes**.
13. Choose **Go to summary**.

New console

**To configure a SAML 2.0 identity provider in your user pool**

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).

4. Choose the **Sign-in experience** tab. Locate **Federated sign-in** and select **Add an identity provider**.
5. Choose a **SAML** social identity provider.
6. Enter **Identifiers** separated by commas. An identifier tells Amazon Cognito it should check the email address a user enters when they sign in, and then direct them to the provider that corresponds to their domain.
7. Choose **Add sign-out flow** if you want Amazon Cognito to send signed sign-out requests to your provider when a user logs out. You must configure your SAML 2.0 identity provider to send sign-out responses to the `https://<your Amazon Cognito domain>/saml2/logout` endpoint that is created when you configure the hosted UI. The `saml2/logout` endpoint uses POST binding.

**Note**

If this option is selected and your SAML identity provider expects a signed logout request, you will also need to configure the signing certificate provided by Amazon Cognito with your SAML IdP.

The SAML IdP will process the signed logout request and logout your user from the Amazon Cognito session.

8. Choose a **Metadata document source**. If your identity provider offers SAML metadata at a public URL, you can choose **Metadata document URL** and enter that public URL. Otherwise, choose **Upload metadata document** and select a metadata file you downloaded from your provider earlier.

**Note**

We recommend that you enter a metadata document URL if your provider has a public endpoint, rather than uploading a file; this allows Amazon Cognito to refresh the metadata automatically. Typically, metadata refresh happens every 6 hours or before the metadata expires, whichever is earlier.

9. Select **Map attributes between your SAML provider and your app** to map SAML provider attributes to the user profile in your user pool. Include your user pool required attributes in your attribute map.

For example, when you choose the **User pool attribute** `email`, enter the SAML attribute name as it appears in the SAML assertion from your identity provider. Your identity provider might offer sample SAML assertions for reference. Some identity providers use simple names, such as `email`, while others use URL-formatted attribute names, such as the following example:

```
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
```

10. Choose **Create**.

For more information, see [Adding SAML identity providers to a user pool \(p. 68\)](#).

## Next steps

Now that you've created a user pool, you can explore more of the features in Amazon Cognito

Dive deep into these user pool features:

- [Customizing the built-in sign-in and sign-up webpages \(p. 53\)](#)
- [Adding MFA to a user pool \(p. 381\)](#)
- [Adding advanced security to a user pool \(p. 387\)](#)
- [Customizing user pool workflows with Lambda triggers \(p. 92\)](#)
- [Using Amazon Pinpoint analytics with Amazon Cognito user pools \(p. 145\)](#)

For an overview of Amazon Cognito authentication and authorization cases, see [Common Amazon Cognito scenarios \(p. 10\)](#).

To access other AWS services after a successful user pool authentication, see [Accessing AWS services using an identity pool after sign-in \(p. 203\)](#).

In addition to the AWS Management Console and the user pool SDKs mentioned previously in this section, you can also interact with your user pool profiles by using the [AWS Command Line Interface](#).

## Updating user pool configuration

To change the settings of Amazon Cognito user pools in the AWS Management Console, navigate through the feature-based tabs in your user pool settings and update fields as described in other areas of this guide. You can't change some settings after you create a user pool. If you want to change the following settings, you must create a new user pool or app client.

### User pool name

The friendly name that you assigned to your user pool. To change the name of a user pool, create a new user pool.

### Amazon Cognito user pool sign-in options

The attributes that your users can pass as a user name when they sign in. When you create a user pool, you can choose to allow sign-in with user name, email address, phone number, or a preferred user name. To change user pool sign-in options, create a new user pool.

### Make user name case sensitive

When you create a user name that matches another user name except for the letter case, Amazon Cognito can treat them as either the same user or as unique users. For more information, see [User pool case sensitivity \(p. 398\)](#). To change case sensitivity, create a new user pool.

### Required attributes

The attributes that your users must provide values for when they sign up, or when you create them. For more information, see [User pool attributes \(p. 206\)](#). To change required attributes, create a new user pool.

### Client secret

When you create an app client, you can generate a client secret so that only trusted sources can make requests to your user pool. For more information, see [Configuring a user pool app client \(p. 223\)](#). To change a client secret, create a new app client in the same user pool.

### Custom attributes

Attributes with custom names. You can change the value of a user's custom attribute, but you can't delete a custom attribute from your user pool. For more information, see [User pool attributes \(p. 206\)](#). If you reach the maximum number of custom attributes and you want to modify the list, create a new user pool.

## Updating a user pool with the Amazon Cognito API or AWS CLI

You can change the configuration of an Amazon Cognito user pool with automation tools like the Amazon Cognito API or AWS Command Line Interface (AWS CLI). If you don't provide values for existing

parameters like `LambdaConfig`, Amazon Cognito sets them to default values. Plan accordingly when you want to automate changes to your user pool configuration. The following procedure updates your configuration with the [UpdateUserPool](#) API operation.

1. Capture the existing state of your user pool with [DescribeUserPool](#).
2. Format the output of [DescribeUserPool](#) to match the [request parameters](#) of [UpdateUserPool](#). Remove the following top-level fields and their child objects from the output JSON.
  - `Arn`
  - `CreationDate`
  - `CustomDomain`
    - Update this field with the [UpdateUserPoolDomain](#) API operation.
  - `Domain`
    - Update this field with the [UpdateUserPoolDomain](#) API operation.
  - `EmailConfigurationFailure`
  - `EstimatedNumberOfUsers`
  - `Id`
  - `LastModifiedDate`
  - `Name`
  - `SchemaAttributes`
  - `SmsConfigurationFailure`
  - `Status`
3. Confirm that the resulting JSON matches the [request parameters](#) of [UpdateUserPool](#).
4. Modify any parameters that you want to change in the resulting JSON.
5. Submit an [UpdateUserPool](#) API request with your modified JSON as the request input.

You can also use this modified [DescribeUserPool](#) output in the `--cli-input-json` parameter of `update-user-pool` in the AWS CLI.

## Using the Amazon Cognito hosted UI for sign-up and sign-in

The Amazon Cognito Hosted UI provides you an OAuth 2.0 compliant authorization server. It includes default implementation of end user flows such as registration and authentication. You can also customize user flows, such as the addition of Multi Factor Authentication (MFA), by changing your user pool configuration. Your application will redirect to the Hosted UI, which will handle your user flows. The user experience can be customized by providing brand-specific logos, as well as customizing the design of Hosted UI elements. The Amazon Cognito Hosted UI also allows you to add the ability for end users to sign in with social providers (Facebook, Amazon, Google, and Apple), as well as any OpenID Connect (OIDC)-compliant and SAML providers.

### Topics

- [Setting up the hosted UI with AWS Amplify \(p. 39\)](#)
- [Setting up the hosted UI with the Amazon Cognito console \(p. 39\)](#)
- [Configuring a user pool app client \(p. 42\)](#)
- [Configuring a user pool domain \(p. 47\)](#)
- [Customizing the built-in sign-in and sign-up webpages \(p. 53\)](#)

- [Defining resource servers for your user pool \(p. 57\)](#)

## Setting up the hosted UI with AWS Amplify

If you use AWS Amplify to add authentication to your web or mobile app, you can set up your hosted UI by using the command line interface (CLI) and libraries in the AWS Amplify framework. To add authentication to your app, you use the AWS Amplify CLI to add the `Auth` category to your project. Then, in your client code, you use the AWS Amplify libraries to authenticate users with your Amazon Cognito user pool.

You can display a pre-built hosted UI, or you can federate users through an OAuth 2.0 endpoint that redirects to a social sign-in provider, such as Facebook, Google, Amazon, or Apple. After a user successfully authenticates with the social provider, AWS Amplify creates a new user in your user pool if needed, and then provides the user's OIDC token to your app.

For more information, see the AWS Amplify framework documentation for your app platform:

- [AWS Amplify authentication for JavaScript](#).
- [AWS Amplify authentication for iOS](#).
- [AWS Amplify authentication for Android](#).

## Setting up the hosted UI with the Amazon Cognito console

Original console

### Create an app client

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **Manage User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. On the navigation bar on the left-side of the page, under **General settings**, choose **App clients**.
5. Choose **Add an app client**.
6. Enter a name for your app.
7. Unless required by your authorization flow, clear the option **Generate client secret**. The client secret is used by applications that have a server-side component that can secure the client secret.
8. (Optional) Change the token expiration settings.
9. Select **Auth Flows Configuration** options. For more information, see [User Pool Authentication Flow](#).
10. Choose a **Security configuration**. We recommend you select **Enabled**.
11. (Optional) Choose **Set attribute read and write permissions**. For more information, see [Attribute Permissions and Scopes](#).
12. Choose **Create app client**.
13. Note the **App client id**.
14. Choose **Return to pool details**.

### Configure the app

1. Choose **App client settings** from the navigation bar on the left side of the console page.

2. Select **Cognito User Pool** as one of the **Enabled Identity Providers**.

**Note**

To sign in with external identity providers (IdPs), such as Facebook, Amazon, Google, or Apple, as well as through OpenID Connect (OIDC) or SAML IdPs, first configure them as described in the following steps, and then return to the **App client settings** page to enable them.

3. Enter **Callback URL(s)**. A callback URL indicates where the user will be redirected after a successful sign-in.
4. Enter **Sign out URL(s)**. A sign-out URL indicates where your user will be redirected after signing out.
5. Select **Authorization code grant** to return an authorization code that is then exchanged for user pool tokens. Because the tokens are never exposed directly to an end user, they are less likely to be compromised. However, a custom application is required on the backend to exchange the authorization code for user pool tokens. For security reasons, we recommend that you use the authorization code grant flow, together with [Proof key for code Exchange \(PKCE\)](#), for mobile apps.
6. Select **Implicit grant** to have user pool JSON web tokens (JWT) returned to you from Amazon Cognito. You can use this flow when there's no backend available to exchange an authorization code for tokens. It's also helpful for debugging tokens.
7. You can enable both the **Authorization code grant** and the **Implicit code grant**, and then use each grant as needed.
8. Unless you specifically want to exclude one, select the check boxes for all of the **Allowed OAuth scopes**.
9. Select **Client credentials** only if your app needs to request access tokens on its own behalf, not on behalf of a user.
10. Choose **Save changes**.

### Configure a domain

1. Select **Choose domain name**.
2. On the **Domain name** page, type a domain prefix and check availability, or enter your own domain.
3. Make a note of the complete domain address.
4. Choose **Save changes**.

New console

### Create an app client

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. Select the **App integration** tab.
5. Under **App clients**, select **Create an app client**.
6. Select an **App type**: **Public client**, **Confidential client**, or **Other**. A **Public client** typically operates from your users' devices and uses unauthenticated and token-authenticated APIs. A **Confidential client** typically operates from an app on a central server that you trust with client secrets and API credentials, and uses authorization headers and AWS Identity and Access Management credentials to sign requests. If your use case is different from the preconfigured app client settings for a **Public client** or a **Confidential client**, select **Other**.

7. Enter an **App client name**.
8. Select the **Authentication flows** you want to allow in your app client.
9. (Optional) Configure token expiration.
  - a. Specify the **Refresh token expiration** for the app client. The default value is 30 days. You can change it to any value between 1 hour and 10 years.
  - b. Specify the **Access token expiration** for the app client. The default value is 1 hour. You can change it to any value between 5 minutes and 24 hours.
  - c. Specify the **ID token expiration** for the app client. The default value is 1 hour. You can change it to any value between 5 minutes and 24 hours.
10. Choose **Generate client secret** to have Amazon Cognito generate a client secret for you. Client secrets are typically associated with confidential clients.
11. Choose whether you will **Enable token revocation** for this app client. This will increase the size of tokens. For more information, see [Revoking Tokens](#).
12. Choose whether you will **Prevent error messages that reveal user existence** for this app client. Amazon Cognito will respond to sign-in requests for nonexistent users with a generic message stating that either the user name or password was incorrect.
13. (Optional) Configure **Attribute read and write permissions** for this app client. Your app client can have permission to read and write a limited subset of your user pool's attribute schema.
14. Choose **Create**.
15. Note the **Client id**. This will identify the app client in sign-up and sign-in requests.

#### **Important**

If you use the hosted UI and configure a token lifetime of less than an hour, your user will be able to use tokens based on their session cookie duration, which is currently fixed at one hour.

10. Choose **Generate client secret** to have Amazon Cognito generate a client secret for you. Client secrets are typically associated with confidential clients.
11. Choose whether you will **Enable token revocation** for this app client. This will increase the size of tokens. For more information, see [Revoking Tokens](#).
12. Choose whether you will **Prevent error messages that reveal user existence** for this app client. Amazon Cognito will respond to sign-in requests for nonexistent users with a generic message stating that either the user name or password was incorrect.
13. (Optional) Configure **Attribute read and write permissions** for this app client. Your app client can have permission to read and write a limited subset of your user pool's attribute schema.
14. Choose **Create**.
15. Note the **Client id**. This will identify the app client in sign-up and sign-in requests.

#### **Configure the app**

1. In the **App integration** tab, select your app client under **App clients**. Review your current **Hosted UI** information.
2. Add a **callback URL** under **Allowed callback URL(s)**. A callback URL is where the user is redirected to after a successful sign-in.
3. Add a **sign-out URL** under **Allowed sign-out URL(s)**. A sign-out URL is where your user is redirected to after signing out.
4. Add at least one of the listed options from the list of **Identity providers**.
5. Under **OAuth 2.0 grant types**, select **Authorization code grant** to return an authorization code that is then exchanged for user pool tokens. Because the tokens are never exposed directly to an end user, they are less likely to become compromised. However, a custom application is required on the backend to exchange the authorization code for user pool tokens. For security reasons, we recommend that you use the authorization code grant flow, together with [Proof key for code Exchange \(PKCE\)](#), for mobile apps.
6. Under **OAuth 2.0 grant types**, select **Implicit grant** to have user pool JSON web tokens (JWT) returned to you from Amazon Cognito. You can use this flow when there's no backend available to exchange an authorization code for tokens. It's also helpful for debugging tokens.
7. You can enable both the **Authorization code** and the **Implicit code** grants, and then use each grant as needed. If neither **Authorization code** or **Implicit code** grants are selected and your app client has a client secret, you can enable **Client credentials** grants. Select **Client credentials** only if your app needs to request access tokens on its own behalf and not on behalf of a user.
8. Select the **OpenID Connect scopes** that you want to authorize for this app client.
9. Choose **Save changes**.

## Configure a domain

1. Navigate to the **App integration** tab for your user pool.
2. Next to **Domain**, choose **Actions** and select **Create custom domain** or **Create Cognito domain**. If you have already configured a user pool domain, choose **Delete Cognito domain** or **Delete custom domain** before creating a new custom domain.
3. Enter an available domain prefix to use with a **Cognito domain**. For information on setting up a **Custom domain**, see [Using your own Domain for the hosted UI](#)
4. Choose **Create**.

### To view your sign-in page

You can view the hosted UI sign-in webpage with the following URL. Note the `response_type`. In this case, `response_type=code` for the authorization code grant.

```
https://<your_domain>/login?
response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>
```

You can view the hosted UI sign-in webpage with the following URL for the implicit code grant where `response_type=token`. After a successful sign-in, Amazon Cognito returns user pool tokens to your web browser's address bar.

```
https://<your_domain>/login?
response_type=token&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>
```

You can find the JSON web token (JWT) identity token after the `#idtoken=` parameter in the response.

Here's a sample response from an implicit grant request. Your identity token string will be much longer.

```
https://www.example.com/
#id_token=123456789tokens123456789&expires_in=3600&token_type=Bearer
```

If changes to your hosted UI pages do not immediately appear, wait a few minutes and then refresh the page. Amazon Cognito user pool tokens are signed using an RS256 algorithm. You can decode and verify user pool tokens using AWS Lambda, see [Decode and verify Amazon Cognito JWT tokens](#) on the AWS GitHub website.

Your domain is shown on the **Domain name** page. Your app client ID and callback URL are shown on the **App client settings** page.

#### Note

The Amazon Cognito hosted UI does not support the custom authentication flow.

## Configuring a user pool app client

After you create a user pool, you can configure an app client to use the built-in webpages for signing up and signing in your users. For terminology, see [App Client Settings Terminology \(p. 45\)](#).

### To configure an app client (AWS Management Console)

Original console

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.

2. Choose **Manage User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. On the navigation bar on the left-side of the page, choose **App clients** under **General settings**.
5. Choose **Add an app client**.
6. Enter your app name.
7. Unless required by your authorization flow, clear the option **Generate client secret**. The client secret is used by applications that have a server-side component that can secure the client secret.
8. (Optional) Change the token expiration settings.
9. Select **Auth Flows Configuration** options. For more information, see [User Pool Authentication Flow](#).
10. Choose a **Security configuration**. We recommend you select **Enabled**.
11. (Optional) Choose **Set attribute read and write permissions**.
12. Choose **Create app client**.
13. Note the **App client id**.
14. Choose **Return to pool details**.

#### New console

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. Choose the **App integration** tab.
5. Under **App clients**, select **Create an app client**.
6. Select an **App type**: **Public client**, **Confidential client**, or **Other**. A **Public client** typically operates from your users' devices and uses unauthenticated and token-authenticated APIs. A **Confidential client** typically operates from an app on a central server that you trust with client secrets and API credentials, and uses authorization headers and AWS Identity and Access Management credentials to sign requests. If your use case is different from the preconfigured app client settings for a **Public client** or a **Confidential client**, select **Other**.
7. Enter an **App client name**.
8. Select the **Authentication flows** you want to allow in your app client. For more information, see [User Pool Authentication Flow](#).
9. (Optional) configure token expiration.
  - a. Specify the **Refresh token expiration** for the app client. The default value is 30 days. You can change it to any value between 1 hour and 10 years.
  - b. Specify the **Access token expiration** for the app client. The default value is 1 hour. You can change it to any value between 5 minutes and 24 hours.
  - c. Specify the **ID token expiration** for the app client. The default value is 1 hour. You can change it to any value between 5 minutes and 24 hours.

**Important**

If you use the hosted UI and configure a token lifetime of less than an hour, your user will be able to use tokens based on their session cookie duration, which is currently fixed at one hour.

10. Choose **Generate client secret** to have Amazon Cognito generate a client secret for you. Client secrets are typically associated with confidential clients.
11. Choose whether you will **Enable token revocation** for this app client. This will increase the size of tokens that Amazon Cognito issues. For more information, see [Revoking Tokens](#).

12. Choose whether you will **Prevent error messages that reveal user existence** for this app client. Amazon Cognito will respond to sign-in requests for nonexistent users with a generic message stating that either the user name or password was incorrect.
13. (Optional) Configure **Attribute read and write permissions** for this app client. Your app client can have permission to read and write only a limited subset of your user pool's attribute schema. For more information, see [Attribute Permissions and Scopes](#).
14. Choose **Create**.
15. Note the **Client id**. This will identify the app client in sign-up and sign-in requests.

## To configure an app client (AWS CLI and AWS API)

You can use the AWS CLI to update, create, describe, and delete your user pool app client.

Replace "*MyUserPoolID*" and "*MyAppClientId*" with your user pool and app client ID values in these examples. Likewise, your parameter values might be different than those used in these examples.

### Note

Use JSON format for callback and logout URLs to prevent the CLI from treating them as remote parameter files:

```
--callback-urls "["https://example.com"]"
--logout-urls "["https://example.com"]"
```

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

At the AWS CLI, enter the following command:

```
aws cognito-identity 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,
 "CreationDate": 1524628110.29,
 "AllowedOAuthFlowsUserPoolClient": true,
 "UserPoolId": "MyUserPoolID",
 "LastModifiedDate": 1530055177.553
 }
}
```

See the AWS CLI command reference for more information: [update-user-pool-client](#).

AWS API: [UpdateUserPoolClient](#)

### Creating a user pool app client (AWS CLI and AWS API)

```
aws cognito-identity create-user-pool-client --user-pool-id MyUserPoolID --client-name myApp
```

See the AWS CLI command reference for more information: [create-user-pool-client](#)

AWS API: [CreateUserPoolClient](#)

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

```
aws cognito-identity describe-user-pool-client --user-pool-id MyUserPoolID --client-id MyClientId
```

See the AWS CLI command reference for more information: [describe-user-pool-client](#).

AWS API: [DescribeUserPoolClient](#)

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

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

See the AWS CLI command reference for more information: [list-user-pool-clients](#).

AWS API: [ListUserPoolClients](#)

### Deleting a user pool app client (AWS CLI and AWS API)

```
aws cognito-identity delete-user-pool-client --user-pool-id "MyUserPoolID" --client-id "MyAppClientId"
```

See the AWS CLI command reference for more information: [delete-user-pool-client](#)

AWS API: [DeleteUserPoolClient](#)

## App client settings terminology

The following terms and definitions can help you with configuring your app client.

### Enabled Identity Providers

You can choose your identity provider (IdP) to authenticate your users. This service can be performed by your user pool, or by a third party such as Facebook. Before you can use an IdP, you need to enable it. You can enable multiple IdPs, but you must enable at least one. For more information on using external IdPs see [Adding user pool sign-in through a third party \(p. 60\)](#).

### Callback URL(s)

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](#).

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

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

#### Sign out URL(s)

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

#### Allowed OAuth Flows

The **Authorization code grant** flow initiates a code grant flow, which provides an authorization code as the response. This code can be exchanged for access tokens with the [Token endpoint \(p. 425\)](#). Because the tokens are never exposed directly to an end user, they are less likely to become compromised. However, a custom application is required on the backend to exchange the authorization code for user pool tokens.

##### Note

For security reasons, we highly recommend that you use only the **Authorization code grant** flow, together with [PKCE](#), for mobile apps.

The **Implicit grant** flow allows the client to get the access token (and optionally the ID token, based on scopes) directly from the [Authorize endpoint \(p. 420\)](#). Choose this flow if your app can't initiate the **Authorization code grant** flow. For more information, see the [OAuth 2.0 specification](#).

You can enable both the **Authorization code grant** and the **Implicit code grant**, and then use each grant as needed.

The **Client credentials** flow is used in machine-to-machine communications. You can use this flow to request an access token to access your own resources. Choose this flow when your app is requesting the token on its own behalf and not on behalf of a user.

##### Note

Since the client credentials flow is not used on behalf of a user, only custom scopes can be used with this flow. A custom scope is one that you define for your own resource server. See [Defining resource servers for your user pool \(p. 57\)](#).

#### Allowed OAuth Scopes

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

- 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. This scope can only be requested with the `openid` scope.
- The `openid` scope returns all user attributes in the ID token that are readable by the client. The ID token is not returned if the `openid` scope is not requested by the client.
- The `aws.cognito.signin.user.admin` scope grants access to [Amazon Cognito user pool API operations](#) that require access tokens, such as `UpdateUserAttributes` and `VerifyUserAttribute`.
- 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.

#### Allowed 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/scoped`. See [Defining resource servers for your user pool \(p. 57\)](#).

For more information about OAuth scopes, see the list of [standard OIDC scopes](#).

## Configuring a user pool domain

After setting up an app client, you can configure the address of your sign-up and sign-in webpages. You can use an Amazon Cognito hosted domain and choose an available domain prefix, or you can use your own web address as a custom domain.

To add an app client and an Amazon Cognito hosted domain with the AWS Management Console, see [Adding an app to enable the hosted web UI](#).

**Note**

You can't use the text `aws`, `amazon`, or `cognito` in the domain prefix.

**Topics**

- [Using the Amazon Cognito domain for the hosted UI \(p. 47\)](#)
- [Using your own domain for the hosted UI \(p. 49\)](#)

## Using the Amazon Cognito domain for the hosted UI

After setting up an app client, you can configure the address for your sign-up and sign-in webpages. You can use the hosted Amazon Cognito domain with your own domain prefix.

To add an app client and an Amazon Cognito hosted domain with the AWS Management Console, see [Adding an app to enable the hosted web UI](#).

**Topics**

- [Prerequisites \(p. 47\)](#)
- [Step 1: Configure a hosted user pool domain \(p. 47\)](#)
- [Step 2: Verify your sign-in page \(p. 48\)](#)

### Prerequisites

Before you begin, you need:

- A user pool with an app client. For more information, see [Getting started with user pools \(p. 23\)](#).

### Step 1: Configure a hosted user pool domain

#### To configure a hosted user pool domain

You can use either the AWS Management Console or the AWS CLI or API to configure a user pool domain.

Original console

#### Configure a domain

1. Sign in to the [Amazon Cognito console](#).
2. In the navigation pane, choose **Manage your User Pools**, and choose the user pool you want to edit.
3. Choose the **Domain name** tab.
4. Type the domain prefix you want to use in the **Prefix domain name** box.

5. Choose **Check availability** to confirm that the domain prefix is available.
6. Choose **Save changes**.

New console

### Configure a domain

1. Navigate to the **App integration** tab for your user pool.
2. Next to **Domain**, choose **Actions** and select **Create custom domain** or **Create Cognito domain**. If you have already configured a user pool domain, choose **Delete Cognito domain** or **Delete custom domain** before creating your new custom domain.
3. Enter an available domain prefix to use with a **Cognito domain**. For information on setting up a **Custom domain**, see [Using your own Domain for the hosted UI](#)
4. Choose **Create**.

CLI/API

Use the following commands to create a domain prefix and assign it to your user pool.

### To configure a user pool domain

- AWS CLI: `aws cognito-identity create-user-pool-domain`

**Example:** `aws cognito-identity create-user-pool-domain --user-pool-id <user_pool_id> --domain <domain_name>`

- AWS API: [CreateUserPoolDomain](#)

### To get information about a domain

- AWS CLI: `aws cognito-identity describe-user-pool-domain`

**Example:** `aws cognito-identity describe-user-pool-domain --domain <domain_name>`

- AWS API: [DescribeUserPoolDomain](#)

### To delete a domain

- AWS CLI: `aws cognito-identity delete-user-pool-domain`

**Example:** `aws cognito-identity delete-user-pool-domain --domain <domain_name>`

- AWS API: [DeleteUserPoolDomain](#)

## Step 2: Verify your sign-in page

- Verify that the sign-in page is available from your Amazon Cognito hosted domain.

```
https://<your_domain>/login?
response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>
```

Your domain is shown on the **Domain name** page of the Amazon Cognito console. Your app client ID and callback URL are shown on the **App client settings** page.

## Using your own domain for the hosted UI

After setting up an app client, you can configure your user pool with a custom domain for the Amazon Cognito hosted UI and [auth API](#) endpoints. With a custom domain, you enable your users to sign in to your application by using your own web address.

### Topics

- [Adding a custom domain to a user pool \(p. 49\)](#)
- [Changing the SSL certificate for your custom domain \(p. 52\)](#)

### Adding a custom domain to a user pool

To add a custom domain to your user pool, you specify the domain name in the Amazon Cognito console, and you provide a certificate you manage with [AWS Certificate Manager \(ACM\)](#). After you add your domain, Amazon Cognito provides an alias target, which you add to your DNS configuration.

#### Prerequisites

Before you begin, you need:

- A user pool with an app client. For more information, see [Getting started with user pools \(p. 23\)](#).
- A web domain that you own. Its root must have a valid **A record** in DNS. For example, if your custom domain is `auth.example.com`, you must be able to resolve `example.com` to an IP address. For more information see [Domain Names](#).
- The ability to create a subdomain for your custom domain. We recommend using **auth** as the subdomain. For example: `auth.example.com`.

#### Note

You might need to obtain a new certificate for your custom domain's subdomain if you don't have a [wildcard certificate](#).

- A Secure Sockets Layer (SSL) certificate managed by ACM.

#### Note

You must change the AWS region to US East (N. Virginia) in the ACM console before you request or import a certificate.

- To set up a custom domain name or to update its certificate, you must have permission to update Amazon CloudFront distributions. You can do so by attaching the following IAM policy statement to an IAM user, group, or role in your AWS account:

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "AllowCloudFrontUpdateDistribution",
 "Effect": "Allow",
 "Action": [
 "cloudfront:updateDistribution"
],
 "Resource": [
 "*"
]
 }
]
}
```

See [Using Identity-Based Policies \(IAM Policies\) for CloudFront](#).

## Step 1: Enter your custom domain name

You can add your domain to your user pool by using the Amazon Cognito console or API.

Original console

### To add your domain to your user pool from the Amazon Cognito console:

1. Sign in to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **Manage User Pools**.
3. On the **Your User Pools** page, choose the user pool to which you want to add your domain.
4. In the navigation menu on the left, choose **Domain name**.
5. Under **Your own domain**, choose **Use your domain**.
6. For **Domain name**, enter your custom domain name. Your domain name can include only lowercase letters, numbers, and hyphens. Do not use a hyphen for the first or last character. Use periods to separate subdomain names.
7. For **AWS managed certificate**, choose the SSL certificate that you want to use for your domain. You can choose one of the certificates that you manage with ACM.

If you don't have a certificate that is available to choose, you can use ACM to provision one. For more information, see [Getting Started](#) in the *AWS Certificate Manager User Guide*.

8. Choose **Save changes**.

#### Note

Make sure to note the **Alias target** for your domain. Instead of an IP address or a domain name, the **Alias target** is an alias resource record set that points to an Amazon CloudFront distribution. You will use the **Alias target** when configuring DNS settings with Amazon Route 53 or a third-party DNS provider.

New console

### To add your domain to your user pool from the Amazon Cognito console:

1. Sign in to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **User pools**.
3. Choose the user pool to which you want to add your domain.
4. Choose the **App integration** tab.
5. Next to **Domain**, choose **Actions**, then choose **Create custom domain**.

#### Note

If you have already configured a user pool domain, choose **Delete Cognito domain** or **Delete custom domain** to delete the existing domain before creating your new custom domain.

6. For **Custom domain**, enter the URL of the domain you want to use with Amazon Cognito. Your domain name can include only lowercase letters, numbers, and hyphens. Do not use a hyphen for the first or last character. Use periods to separate subdomain names.
7. For **ACM certificate**, choose the SSL certificate that you want to use for your domain. Only ACM certificates in US East (N. Virginia) are eligible to use with an Amazon Cognito custom domain, regardless of the AWS Region of your user pool.

If you don't have an available certificate, you can use ACM to provision one in US East (N. Virginia). For more information, see [Getting Started](#) in the *AWS Certificate Manager User Guide*.

8. Choose **Create**.
9. Amazon Cognito returns you to the **App integration** tab. A message titled **Create an alias record in your domain's DNS** is displayed. Note down the **Domain** and **Alias target** displayed in the console. They will be used in the next step to direct traffic to your custom domain.

API

#### To add your domain to your user pool with the Amazon Cognito API:

- Use the [CreateUserPoolDomain](#) action.

#### Step 2: Add an alias target and subdomain

In this step, you set up an alias through your Domain Name Server (DNS) service provider that points back to the alias target from the previous step. If you are using Amazon Route 53 for DNS address resolution, choose the section **To add an alias target and subdomain using Route 53**.

#### To add an alias target and subdomain to your current DNS configuration

- If you aren't using Route 53 for DNS address resolution, then you must use your DNS service provider's configuration tools to add the alias target from the previous step to your domain's DNS record. Your DNS provider will also need to set up the subdomain for your custom domain.

#### To add an alias target and subdomain using Route 53

1. Sign in to the [Route 53 console](#). If prompted, enter your AWS credentials.
2. If you don't have a hosted zone in Route 53, you must create one using the following steps. Otherwise, continue to step 3.
  - a. Choose **Create Hosted Zone**.
  - b. Enter the parent domain, for example `auth.example.com`, of your custom domain, for example `myapp.auth.example.com`, from the **Domain Name** list.
  - c. {Optional} For **Comment**, enter a comment about the hosted zone.
  - d. Choose a hosted zone **Type of Public hosted zone** to allow public clients to resolve your custom domain. Choosing **Private hosted zone** is not supported.
  - e. Apply **Tags** as desired.
  - f. Choose **Create hosted zone**.

#### Note

You can also create a new hosted zone for your custom domain, and you can create a delegation set in the parent hosted zone that directs queries to the subdomain hosted zone. This method offers more flexibility and security with your hosted zones. For more information, see [Creating a subdomain for a domain hosted through Amazon Route 53](#).

3. On the **Hosted Zones** page, choose the name of your hosted zone.
4. Choose **Create record**. Add a DNS record for the parent domain and choose **Create records**. The following is an example record for the domain `auth.example.com`.

`auth.example.com. 60 IN A 198.51.100.1`

#### Note

Amazon Cognito verifies that there is a DNS record for the parent domain of your custom domain to protect against accidental hijacking of production domains. If you do not have a DNS record for the parent domain, Amazon Cognito will return an error when you attempt to set the custom domain.

5. Choose **Create record** again.
6. Enter a **Record name** that matches your custom domain, for example `myapp` to create a record for `myapp.auth.example.com`.
7. Enter the alias target name that you previously noted into **Alias Target**.
8. Enable the **Alias** option.
9. Choose to **Route traffic to an Alias to Cloudfront distribution**. Enter the **Alias target** provided by Amazon Cognito when you created your custom domain.
10. Choose **Create Records**.

**Note**

Your new records can take around 60 seconds to propagate to all Route 53 DNS servers. You can use the Route 53 [GetChange API](#) method to verify that your changes have propagated.

### Step 3: Verify your sign-in page

- Verify that the sign-in page is available from your custom domain.

Sign in with your custom domain and subdomain by entering this address into your browser. This is an example URL of a custom domain `example.com` with the subdomain `auth`:

```
https://myapp.auth.example.com/login?
response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>
```

## Changing the SSL certificate for your custom domain

When necessary, you can use Amazon Cognito to change the certificate that you applied to your custom domain.

Usually, this is unnecessary following routine certificate renewal with ACM. When you renew your existing certificate in ACM, the ARN for your certificate remains the same, and your custom domain uses the new certificate automatically.

However, if you replace your existing certificate with a new one, ACM gives the new certificate a new ARN. To apply the new certificate to your custom domain, you must provide this ARN to Amazon Cognito.

After you provide your new certificate, Amazon Cognito requires up to 1 hour to distribute it to your custom domain.

#### Before you begin

Before you can change your certificate in Amazon Cognito, you must add your certificate to ACM. For more information, see [Getting Started](#) in the *AWS Certificate Manager User Guide*. When you add your certificate to ACM, you must choose US East (N. Virginia) as the AWS Region.

You can change your certificate by using the Amazon Cognito console or API.

Original console

#### To renew a certificate from the Amazon Cognito console:

1. Sign in to the AWS Management Console and open the Amazon Cognito console at <https://console.aws.amazon.com/cognito/home>.
2. Choose **Manage User Pools**.
3. On the **Your User Pools** page, choose the user pool to which you want to add your domain.

4. On the navigation menu on the left, choose **Domain name**.
5. Under **Your own domain**, for **AWS managed certificate**, choose your new certificate.
6. Choose **Save changes**.

New console

**To renew a certificate from the Amazon Cognito console:**

1. Sign in to the AWS Management Console and open the Amazon Cognito console at <https://console.aws.amazon.com/cognito/home>.
2. Choose **User Pools**.
3. Choose the user pool for which you want to update the certificate.
4. Choose the **App integration** tab.
5. Choose **Actions, Edit ACM certificate**.
6. Select the new certificate you want to associate with your custom domain.
7. Choose **Save changes**.

API

**To renew a certificate (Amazon Cognito API)**

- Use the [UpdateUserPoolDomain](#) action.

## Customizing the built-in sign-in and sign-up webpages

You can use the AWS Management Console, or the AWS CLI or API, to specify customization settings for the built-in app UI experience. You can upload a custom logo image to be displayed in the app. You can also use cascading style sheets (CSS) to customize the look of the UI.

You can specify app UI customization settings for a single client (with a specific `clientId`) or for all clients (by setting the `clientId` to `ALL`). If you specify `ALL`, the default configuration will be used for every client that has no UI customization set previously. If you specify UI customization settings for a particular client, it will no longer fall back to the `ALL` configuration.

**Note**

To use this feature, your user pool must have a domain associated with it.

### Specifying a custom logo for the app

The maximum allowable size for a logo image file is 100 KB.

### Specifying CSS customizations for the app

You can customize the CSS for the hosted app pages, with the following restrictions:

- You can use any of the following CSS class names:
  - `background-customizable`
  - `banner-customizable`
  - `errorMessage-customizable`

- `idpButton-customizable`
- `idpButton-customizable:hover`
- `inputField-customizable`
- `inputField-customizable:focus`
- `label-customizable`
- `legalText-customizable`
- `logo-customizable`
- `submitButton-customizable`
- `submitButton-customizable:hover`
- `textDescription-customizable`
- Property values can contain HTML, except for the following values: `@import`, `@supports`, `@page`, or `@media` statements, or Javascript.

You can customize the following CSS properties.

#### Labels

- **font-weight** is a multiple of 100 from 100 to 900.

#### Input fields

- **width** is the width of the containing block as a percentage.
- **height** is the height of the input field in pixels (px).
- **color** is the text color. It can be any standard CSS color value.
- **background-color** is the background color of the input field. It can be any standard CSS color value.
- **border** is a standard CSS border value that specifies the width, transparency, and color of the border of your app window. Width can be any value from 1px to 100px. Transparency can be solid or none. Color can be any standard color value.

#### Text descriptions

- **padding-top** is the amount of padding above the text description.
- **padding-bottom** is the amount of padding below the text description.
- **display** can be `block` or `inline`.
- **font-size** is the font size for text descriptions.

#### Submit button

- **font-size** is the font size of the button text.
- **font-weight** is the font weight of the button text: `bold`, `italic`, or `normal`.
- **margin** is a string of 4 values indicating the top, right, bottom, and left margin sizes for the button.
- **font-size** is the font size for text descriptions.
- **width** is the width of the button text in percent of the containing block.
- **height** is the height of the button in pixels (px).
- **color** is the button text color. It can be any standard CSS color value.
- **background-color** is the background color of the button. It can be any standard color value.

#### Banner

- **padding** is a string of 4 values indicating the top, right, bottom, and left padding sizes for the banner.
- **background-color** is the banner's background color. It can be any standard CSS color value.

#### Submit button hover

- **color** is the foreground color of the button when you hover over it. It can be any standard CSS color value.
- **background-color** is the background color of the button when you hover over it. It can be any standard CSS color value.

#### Identity provider button hover

- **color** is the foreground color of the button when you hover over it. It can be any standard CSS color value.
- **background-color** is the background color of the button when you hover over it. It can be any standard CSS color value.

#### Password check not valid

- **color** is the text color of the "Password check not valid" message. It can be any standard CSS color value.

#### Background

- **background-color** is the background color of the app window. It can be any standard CSS color value.

#### Error messages

- **margin** is a string of 4 values indicating the top, right, bottom, and left margin sizes.
- **padding** is the padding size.
- **font-size** is the font size.
- **width** is the width of the error message as a percentage of the containing block.
- **background** is the background color of the error message. It can be any standard CSS color value.
- **border** is a string of 3 values specifying the width, transparency, and color of the border.
- **color** is the error message text color. It can be any standard CSS color value.
- **box-sizing** is used to indicate to the browser what the sizing properties (width and height) should include.

#### Identity provider buttons

- **height** is the height of the button in pixels (px).
- **width** is the width of the button text as a percentage of the containing block.
- **text-align** is the text alignment setting. It can be `left`, `right`, or `center`.
- **margin-bottom** is the bottom margin setting.
- **color** is the button text color. It can be any standard CSS color value.
- **background-color** is the background color of the button. It can be any standard CSS color value.
- **border-color** is the color of the button border. It can be any standard CSS color value.

#### Identity provider descriptions

- **padding-top** is the amount of padding above the description.
- **padding-bottom** is the amount of padding below the description.
- **display** can be `block` or `inline`.
- **font-size** is the font size for descriptions.

#### Legal text

- **color** is the text color. It can be any standard CSS color value.
- **font-size** is the font size.

#### Note

When you customize **Legal text**, you are customizing the message **We won't post to any of your accounts without asking first** that is displayed under social identity providers in the sign-in page.

### Logo

- **max-width** is the maximum width as a percentage of the containing block.
- **max-height** is the maximum height as a percentage of the containing block.

### Input field focus

- **border-color** is the color of the input field. It can be any standard CSS color value.
- **outline** is the border width of the input field, in pixels.

### Social button

- **height** is the height of the button in pixels (px).
- **text-align** is the text alignment setting. It can be `left`, `right`, or `center`.
- **width** is the width of the button text as a percentage of the containing block.
- **margin-bottom** is the bottom margin setting.

### Password check valid

- **color** is the text color of the "Password check valid" message. It can be any standard CSS color value.

## Specifying app UI customization settings for a user pool (AWS Management Console)

You can use the AWS Management Console to specify UI customization settings for your app.

### Original console

#### Note

You can view the hosted UI with your customizations by constructing the following URL, with the specifics for your user pool, and typing it into a browser: `https://<your_domain>/login?response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>`

You may have to wait up to one minute to refresh your browser before changes made in the console appear.

Your domain is shown on the **Domain name** tab. Your app client ID and callback URL are shown on the **General settings** tab.

### To specify app UI customization settings

1. Sign in to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. In the navigation pane, choose **Manage User Pools**, and choose the user pool you want to edit.
3. Choose the **UI customization** tab.
4. Under **App client to customize**, choose the app you want to customize from the dropdown menu of app clients that you previously created in the **App clients** tab.
5. To upload your own logo image file, choose **Choose a file** or drag a file into the **Logo (optional)** box.
6. Under **CSS customizations (optional)**, you can customize the appearance of the app by changing CSS properties from their default values.

### New console

#### Note

You can view the hosted UI with your customizations by constructing the following URL, with the specifics for your user pool, and typing it into a browser: `https://<your_domain>/login?`

`response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>`

You may have to wait up to one minute to refresh your browser before changes made in the console appear.

Your domain is shown on the **App integration** tab under **Domain**. Your app client ID and callback URL are shown under **App clients**.

### To specify app UI customization settings

1. Sign in to the [Amazon Cognito console](#).
2. In the navigation pane, choose **User Pools**, and choose the user pool you want to edit.
3. Choose the **App integration** tab.
4. To customize UI settings for all app clients, locate **Hosted UI customization** and select **Edit**.
5. To customize UI settings for one app client, locate **App clients** and select the app client you wish to modify, then locate **Hosted UI customization** and select **Edit**. To switch an app client from user pool default customization to client-specific customization, select **Use client-level settings**.
6. To upload your own logo image file, choose **Choose file** or **Replace current file**.
7. To customize hosted UI CSS, download **CSS template.css** and modify the template with the values you wish to customize. Only the keys that are included in the template can be used with the hosted UI. Added CSS keys will not be reflected in your UI. After you have customized the CSS file, choose **Choose file** or **Replace current file** to upload your custom CSS file.

## Specifying app UI customization settings for a user pool (AWS CLI and AWS API)

Use the following commands to specify app UI customization settings for your user pool.

### To get the UI customization settings for a user pool's built-in app UI

- AWS CLI: `aws cognito-idp get-ui-customization`
- AWS API: [GetUICustomization](#)

### To set the UI customization settings for a user pool's built-in app UI

- AWS CLI: `aws cognito-idp set-ui-customization --user-pool-id <your-user-pool-id> --client-id <your-app-client-id> --image-file <path-to-logo-image-file> --css ".label-customizable{ color: <color>; }"`
- AWS API: [SetUICustomization](#)

## Defining resource servers for your user pool

Once you configure a domain for your user pool, Amazon Cognito automatically provisions a hosted web UI with which you can add sign-up and sign-in pages to your app. For more information see [Step 2. Add an app client and set up the hosted UI \(p. 24\)](#).

A *resource server* is a server for access-protected resources. It handles authenticated requests from an app that has an access token. Typically the resource server provides a **CRUD** API for making these access requests. This API can be hosted in Amazon API Gateway or outside of AWS. The app passes the access token in the API call to the resource server. The app should treat the access token as opaque when it passes the token in the access request. The resource server inspects the access token to determine if access should be granted.

**Note**

Your resource server must verify the access token signature and expiration date before processing any claims inside the token. For more information about verifying and using user pool tokens, see [Integrating Amazon Cognito User Pools with API Gateway](#) on the [AWS Blog](#). Amazon API Gateway is a good option for inspecting access tokens and protecting your resources. For more about API Gateway Lambda authorizers, see [Use API Gateway Lambda authorizers](#).

A *scope* is a level of access that an app can request to a resource. For example, if you have a resource server for photos, it might define two scopes: one for read access to the photos and one for write/delete access. When the app makes an API call to request access and passes an access token, the token will have one or more scopes embedded inside of it.

**Overview**

Amazon Cognito allows app developers to create their own OAuth2.0 resource servers, and define custom scopes within them. Custom scopes can then be associated with a client, and the client can request those scopes in OAuth2.0 authorization code grant flow, implicit flow, and client credentials flow. Custom scopes are added in the `scope` claim in the access token. A client can use the access token against its resource server, which makes the authorization decision based on the scopes present in the token. For more information about access token scope, see [Using Tokens with User Pools \(p. 191\)](#).

**Note**

Your resource server must verify the access token signature and expiration date before processing any claims inside the token.

**Note**

An app client can only use the client credentials flow if the app client has a client secret.

**Managing the Resource Server and Custom Scopes**

When creating a resource server, you must provide a resource server name and a resource server identifier. For each scope you create in the resource server, you must provide the scope name and description.

Example:

- **Name:** A friendly name for the resource server, such as `Weather API` or `Photo API`.
- **Identifier:** A unique identifier for the resource server. This can be an HTTPS endpoint where your resource server is located. For example, `https://my-weather-api.example.com`
- **Scope Name:** The scope name. For example, `weather.read`
- **Scope Description:** A brief description the scope. For example, `Retrieve weather information`.

When a client app requests a custom scope in any of the OAuth2.0 flows, it must request the full identifier for the scope, which is `resourceServerIdentifier/scopedName`. For example, if the resource server identifier is `https://myphotosapi.example.com` and the scope name is `photos.read`, the client app must request `https://myphotosapi.example.com/photos.read` at runtime.

Deleting a scope from a resource server does not delete its association with all clients; instead, the scope is marked inactive. If a client app requests the deleted scope at runtime, the scope is ignored and is not included in the access token. If the scope is re-added later, then it is again included in the access token.

If a scope is removed from a client, the association between client and scope is deleted. If a client requests a disallowed scope at runtime, an error is returned and an access token isn't issued.

You can use the AWS Management Console, API, or CLI to define resource servers and scopes for your user pool.

## Defining a resource server for your user pool (AWS Management Console)

You can use the AWS Management Console to define a resource server for your user pool.

Original console

### To define a resource server

1. Sign in to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. In the navigation pane, choose **Manage User Pools**, and then choose the user pool you want to edit.
3. Choose the **Resource servers** tab.
4. Choose **Add a resource server**.
5. Enter the name of your resource server. For example, `Photo Server`.
6. Enter the identifier of your resource server. For example, `com.example.photos`.
7. Enter the names of the custom scopes for your resources, such as `read` and `write`.
8. For each scope name, enter a description, such as `view your photos` and `update your photos`.
9. Choose **Save changes**.

Each of the custom scopes that you define appears in the **App client settings** tab, under **OAuth2.0 Allowed Custom Scopes**; for example `com.example.photos/read`.

New console

### To define a resource server

1. Sign in to the [Amazon Cognito console](#).
2. In the navigation pane, choose **User Pools**, and choose the user pool you want to edit.
3. Choose the **App integration** tab and locate **Resource servers**.
4. Choose **Create a resource server**.
5. Enter a **Resource server name**. For example, `Photo Server`.
6. Enter a **Resource server identifier**. For example, `com.example.photos`.
7. Enter **Custom scopes** for your resources, such as `read` and `write`.
8. For each **Scope name**, enter a **Description**, such as `view your photos` and `update your photos`.
9. Choose **Create**.

Your custom scopes can be reviewed in the **App integration** tab under **Resource servers**, in the **Custom scopes** column. Custom scopes can be enabled for app clients from the **App integration** tab under **App clients**. Select an app client, locate **Hosted UI settings** and choose **Edit**. Add **Custom scopes** and choose **Save changes**.

## Defining a resource server for your user pool (AWS CLI and AWS API)

Use the following commands to specify resource server settings for your user pool.

### To create a resource server

- AWS CLI: `aws cognito-identity create-resource-server`
- AWS API: [CreateResourceServer](#)

### To get information about your resource server settings

- AWS CLI: `aws cognito-identity describe-resource-server`
- AWS API: [DescribeResourceServer](#)

### To list information about all resource servers for your user pool

- AWS CLI: `aws cognito-identity list-resource-servers`
- AWS API: [ListResourceServers](#)

### To delete a resource server

- AWS CLI: `aws cognito-identity delete-resource-server`
- AWS API: [DeleteResourceServer](#)

### To update the settings for a resource server

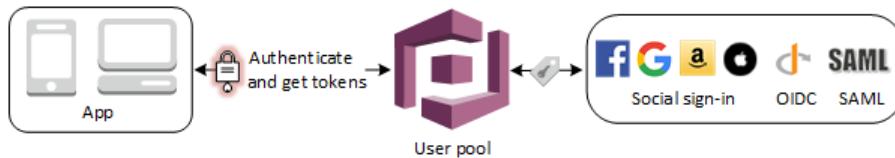
- AWS CLI: `aws cognito-identity update-resource-server`
- AWS API: [UpdateResourceServer](#)

## Adding user pool sign-in through a third party

Your app users can sign in either directly through a user pool, or federate through a third-party identity provider (IdP). The user pool manages the overhead of handling the tokens that are returned from social sign-in through Facebook, Google, Amazon, and Apple, and from OpenID Connect (OIDC) and SAML IdPs. With the built-in hosted web UI, Amazon Cognito provides token handling and management for authenticated users from all IdPs. This way, your backend systems can standardize on one set of user pool tokens.

## How federated sign-in works in Amazon Cognito user pools

Sign-in through a third party (federation) is available in Amazon Cognito user pools. This feature is independent of federation through Amazon Cognito identity pools (federated identities).



Amazon Cognito is a user directory and an OAuth 2.0 identity provider (IdP). When you sign in *native users* to the Amazon Cognito directory, your user pool is an IdP to your app. Native users are those who sign up or who your administrator creates directly in your user pool.

When you connect Amazon Cognito to social, SAML, or OpenID Connect (OIDC) IdPs, your user pool acts as a bridge between multiple service providers and your app. To your IdP, Amazon Cognito is a service provider (SP). Your IdPs pass an OIDC ID token or a SAML assertion to Amazon Cognito. Amazon Cognito reads the claims about your user in the token or assertion and maps those claims to a new user profile in your user pool directory.

Amazon Cognito then creates a user profile for your federated user in its own directory. Amazon Cognito adds attributes to your user based on the claims from your IdP and, in the case of OIDC and social identity providers, an IdP-operated public `userinfo` endpoint. Your user's attributes change in your user pool when a mapped IdP attribute changes. You can also add more attributes independent of those from the IdP.

After Amazon Cognito creates a profile for your federated user, it changes its function and presents itself as the IdP to your app, which is now the SP. Amazon Cognito is a combination OIDC and OAuth 2.0 IdP. It generates access tokens, ID tokens, and refresh tokens. For more information about tokens, see [Using tokens with user pools \(p. 191\)](#).

You must design an app that integrates with Amazon Cognito to authenticate and authorize your users, whether federated or native.

## The responsibilities of an app as an Amazon Cognito SP

### Verify and process the information in the tokens

In most scenarios, Amazon Cognito redirects your authenticated user to an app URL that it appends with an authorization code. Your app [exchanges the code](#) for access, ID, and refresh tokens. Then, it must [check the validity of the tokens](#) and serve information to your user based on the claims in the tokens.

### Respond to authentication events with Amazon Cognito API requests

Your app must integrate with the [Amazon Cognito user pools API](#) and the [authentication API endpoints](#). The authentication API signs your user in and out, and manages tokens. The user pools API has a variety of operations that manage your user pool, your users, and the security of your authentication environment. Your app must know what to do next when it receives a response from Amazon Cognito.

## Things to know about Amazon Cognito user pools third-party sign-in

- If you want your users to sign in with federated providers, you must choose a domain. This sets up the Amazon Cognito hosted UI and [auth API endpoints](#). For more information, see [Using your own domain for the hosted UI \(p. 49\)](#).
- You can't sign in federated users with API operations like `InitiateAuth` and `AdminInitiateAuth`. Federated users can only sign in with the [Login endpoint \(p. 430\)](#) or the [Authorize endpoint \(p. 420\)](#).
- You can use the [Authorize endpoint \(p. 420\)](#) to redirect your users transparently through the hosted UI to federated provider sign-in. For an example, see [Bookmark Amazon Cognito apps in an enterprise dashboard \(p. 78\)](#).
- When the hosted UI redirects a session to a federated IdP, Amazon Cognito includes the `user-agent` header `Amazon/Cognito` in the request.
- Amazon Cognito derives the `username` attribute for a federated user profile from a combination of a fixed identifier and the name of your IdP. To generate a user name that matches your custom requirements, create a mapping to the `preferred_username` attribute. For more information, see [Things to know about mappings \(p. 86\)](#).

Example: MyIDP\_bob@example.com

- Amazon Cognito records information about your federated user's identity to an attribute, and a claim in the ID token, called `identities`. This claim contains your user's provider and their unique ID

from the provider. You can't change the `identities` attribute in a user profile directly. For more information about how to link a federated user, see [Linking federated users to an existing user profile \(p. 90\)](#).

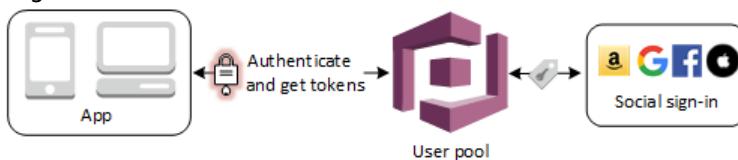
### Topics

- [Adding social identity providers to a user pool \(p. 62\)](#)
- [Adding SAML identity providers to a user pool \(p. 68\)](#)
- [Adding OIDC identity providers to a user pool \(p. 79\)](#)
- [Specifying identity provider attribute mappings for your user pool \(p. 86\)](#)
- [Linking federated users to an existing user profile \(p. 90\)](#)

## Adding social identity providers to a user pool

Your web and mobile app users can sign in through social identity providers (IdP) like Facebook, Google, Amazon, and Apple. With the built-in hosted web UI, Amazon Cognito provides token handling and management for all authenticated users, so your backend systems can standardize on one set of user pool tokens. You must enable the hosted UI to integrate with supported social identity providers. When Amazon Cognito builds your hosted UI, it creates OAuth 2.0 endpoints that Amazon Cognito and your OIDC and social IdPs use to exchange information. For more information, see the [Amazon Cognito user pools Auth API reference](#).

You can add a social IdP in the AWS Management Console, or you can use the AWS CLI or Amazon Cognito API.



### Note

Sign-in through a third party (federation) is available in Amazon Cognito user pools. This feature is independent of federation through Amazon Cognito identity pools (federated identities).

### Topics

- [Prerequisites \(p. 62\)](#)
- [Step 1: Register with a social IdP \(p. 62\)](#)
- [Step 2: Add a social IdP to your user pool \(p. 66\)](#)
- [Step 3: Test your social IdP configuration \(p. 68\)](#)

## Prerequisites

Before you begin, you need the following:

- A user pool with an app client and a user pool domain. For more information, see [Create a user pool](#).
- A social IdP.

## Step 1: Register with a social IdP

Before you create a social IdP with Amazon Cognito, you must register your application with the social IdP to receive a client ID and client secret.

### To register an app with Facebook

1. Create a [developer account with Facebook](#).
2. [Sign in](#) with your Facebook credentials.
3. From the **My Apps** menu, choose **Create New App**.
4. Enter a name for your Facebook app, and then choose **Create App ID**.
5. On the left navigation bar, choose **Settings**, and then **Basic**.
6. Note the **App ID** and the **App Secret**. You will use them in the next section.
7. Choose **+ Add Platform** from the bottom of the page.
8. Choose **Website**.
9. Under **Website**, enter the path to the sign-in page for your app into **Site URL**.

```
https://<your_user_pool_domain>/login?
response_type=code&client_id=<your_client_id>&redirect_uri=https://www.example.com
```

10. Choose **Save changes**.
  11. Enter the path to the root of your user pool domain into **App Domains**.
- ```
https://<your-user-pool-domain>
```
12. Choose **Save changes**.
 13. From the navigation bar choose **Add Product** and choose **Set up** for the **Facebook Login** product.
 14. From the navigation bar choose **Facebook Login** and then **Settings**.

Enter the path to the /oauth2/idpresponse endpoint for your user pool domain into **Valid OAuth Redirect URIs**.

```
https://<your-user-pool-domain>/oauth2/idpresponse
```

15. Choose **Save changes**.

To register an app with Amazon

1. Create a [developer account with Amazon](#).
2. [Sign in](#) with your Amazon credentials.
3. You need to create an Amazon security profile to receive the Amazon client ID and client secret.

Choose **Apps and Services** from navigation bar at the top of the page and then choose **Login with Amazon**.
4. Choose **Create a Security Profile**.
5. Enter a **Security Profile Name**, a **Security Profile Description**, and a **Consent Privacy Notice URL**.
6. Choose **Save**.
7. Choose **Client ID** and **Client Secret** to show the client ID and secret. You will use them in the next section.
8. Hover over the gear icon and choose **Web Settings**, and then choose **Edit**.
9. Enter your user pool domain into **Allowed Origins**.

```
https://<your-user-pool-domain>
```

10. Enter your user pool domain with the /oauth2/idpresponse endpoint into **Allowed Return URLs**.

`https://<your-user-pool-domain>/oauth2/idpresponse`

11. Choose **Save**.

To register an app with Google

For more information about OAuth 2.0 in the Google Cloud platform, see [Learn about authentication & authorization](#) in the Google Workspace for Developers documentation.

1. Create a [developer account with Google](#).
2. Sign in to the [Google Cloud Platform console](#).
3. From the top navigation bar, choose **Select a project**. If you already have a project in the Google platform, this menu displays your default project instead.
4. Select **NEW PROJECT**.
5. Enter a name for your product and then choose **CREATE**.
6. On the left navigation bar, choose **APIs and Services**, then **Oauth consent screen**.
7. Enter App information, an **App domain**, **Authorized domains**, and **Developer contact information**. Your **Authorized domains** must include `amazoncognito.com` and the root of your custom domain, for example `example.com`. Choose **SAVE AND CONTINUE**.
8. 1. Under **Scopes**, choose **Add or remove scopes**, and choose, at minimum, the following OAuth scopes.
 1. `.../auth/userinfo.email`
 2. `.../auth/userinfo.profile`
 3. `openid`
9. Under **Test users**, choose **Add users**. Enter your email address and any other authorized test users, then choose **SAVE AND CONTINUE**.
10. Expand the left navigation bar again, and choose **APIs and Services**, then **Credentials**.
11. Choose **CREATE CREDENTIALS**, then **OAuth client ID**.
12. Choose an **Application type** and give your client a **Name**.
13. Under **Authorized JavaScript origins**, choose **ADD URI**. Enter your user pool domain.

`https://<your-user-pool-domain>`

14. Under **Authorized redirect URIs**, choose **ADD URI**. Enter the path to the `/oauth2/idpresponse` endpoint of your user pool domain.

`https://<your-user-pool-domain>/oauth2/idpresponse`

15. Choose **CREATE**.
16. Securely store the values the Google displays under **Your client ID** and **Your client secret**. Provide these values to Amazon Cognito when you add a Google IdP.

To register an app with Apple

For more information about setting up Sign in with Apple, see [Configuring Your Environment for Sign in with Apple](#) in the Apple Developer documentation.

1. Create a [developer account with Apple](#).
2. [Sign in](#) with your Apple credentials.

3. On the left navigation bar, choose **Certificates, Identifiers & Profiles**.
4. On the left navigation bar, choose **Identifiers**.
5. On the **Identifiers** page, choose the + icon.
6. On the **Register a New Identifier** page, choose **App IDs**, and then choose **Continue**.
7. On the **Select a type** page, choose **App**, then choose **Continue**.
8. On the **Register an App ID** page, do the following:
 1. Under **Description**, enter a description.
 2. Under **App ID Prefix**, enter a **Bundle ID**. Make a note of the value under **App ID Prefix**. You will use this value after you choose Apple as your identity provider in [Step 2: Add a social IdP to your user pool \(p. 66\)](#).
 3. Under **Capabilities**, choose **Sign In with Apple**, and then choose **Edit**.
 4. On the **Sign in with Apple: App ID Configuration** page, choose to set up the app as either primary or grouped with other App IDs, and then choose **Save**.
 5. Choose **Continue**.
9. On the **Confirm your App ID** page, choose **Register**.
10. On the **Identifiers** page, choose the + icon.
11. On the **Register a New Identifier** page, choose **Services IDs**, and then choose **Continue**.
12. On the **Register a Services ID** page, do the following:
 1. Under **Description**, type a description.
 2. Under **Identifier**, type an identifier. Make a note of this Services ID as you will need this value after you choose Apple as your identity provider in [Step 2: Add a social IdP to your user pool \(p. 66\)](#).
 3. Choose **Continue**, then **Register**.
13. Choose the Services ID you just create from the **Identifiers** page.
 1. Select **Sign In with Apple**, and then choose **Configure**.
 2. On the **Web Authentication Configuration** page, select the app ID that you created earlier as the **Primary App ID**.
 3. Choose the + icon next to **Website URLs**.
 4. Under **Domains and subdomains**, enter your user pool domain without an `https://` prefix.

`<your-user-pool-domain>`
5. Under **Return URLs**, enter the path to the `/oauth2/idpresponse` endpoint of your user pool domain.

`https://<your-user-pool-domain>/oauth2/idpresponse`
6. Choose **Next**, and then **Done**. You don't need to verify the domain.
7. Choose **Continue**, and then choose **Save**.
14. On the left navigation bar, choose **Keys**.
15. On the **Keys** page, choose the + icon.
16. On the **Register a New Key** page, do the following:
 1. Under **Key Name**, enter a key name.
 2. Choose **Sign In with Apple**, and then choose **Configure**.
 3. On the **Configure Key** page and select the app ID that you created earlier as the **Primary App ID**. Choose **Save**.
 4. Choose **Continue**, and then choose **Register**.

17. On the **Download Your Key** page, choose **Download** to download the private key and note the **Key ID** shown, and then choose **Done**. You will need this private key and the **Key ID** value shown on this page after you choose Apple as your identity provider in [Step 2: Add a social IdP to your user pool \(p. 66\)](#).

Step 2: Add a social IdP to your user pool

Original console

To configure a user pool social IdP with the AWS Management Console

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **Manage User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. On the left navigation bar, choose **Identity providers**.
5. Choose a social IdP: **Facebook**, **Google**, **Login with Amazon**, or **Apple**.
6. Choose from the following steps, based on your choice of social IdP:
 - **Google and Login with Amazon** — Enter the **app client ID** and **app client secret** generated in the previous section.
 - **Facebook** — Enter the **app client ID** and **app client secret** generated in the previous section, and then choose an API version (for example, version 2.12). We recommend that you choose the latest possible version, as each Facebook API has a lifecycle and discontinuation date. Facebook scopes and attributes can vary between API versions. We recommend that you test your social identity log in with Facebook to make sure that federation works as you intend.
 - **Sign In with Apple** — Enter the **Services ID**, **Team ID**, **Key ID**, and **private key** generated in the previous section.
7. Enter the names of the scopes that you want to authorize. Scopes define which user attributes (such as `name` and `email`) you want to access with your app. For Facebook, these should be separated by commas. For Google and Login with Amazon, they should be separated by spaces. For Sign in with Apple, select the check boxes for the scopes you want access to.

Social identity provider	Example scopes
Facebook	<code>public_profile, email</code>
Google	<code>profile email openid</code>
Login with Amazon	<code>profile postal_code</code>
Sign in with Apple	<code>email name</code>

Your app user is asked to consent to providing these attributes to your app. For more information about their scopes, see the documentation from Google, Facebook, Login with Amazon, or Sign in with Apple.

With Sign in with Apple, the following are user scenarios where scopes might not be returned:

- An end user encounters failures after leaving Apple's sign in page (can be from Internal failures within Amazon Cognito or anything written by the developer)
- The service ID identifier is used across user pools and other authentication services
- A developer adds additional scopes after the end user has signed in before (no new information is retrieved)

- A developer deletes the user and then the user signs in again without removing the app from their Apple ID profile
8. Choose **Enable** for the social IdP that you are configuring.
 9. Choose **App client settings** from the navigation bar.
 10. Select your social IdP as one of the **Enabled Identity Providers** for your user pool app.
 11. Enter your callback URL into **Callback URL(s)** for your user pool app. This is the URL to which your user will be redirected after a successful authentication.

`https://www.example.com`

12. Choose **Save changes**.
13. On the **Attribute mapping** tab, add mappings for at least the required attributes, typically `email`, as follows:
 - a. Select the check box to choose the Facebook, Google, or Amazon attribute name. You can also type the names of additional attributes that are not listed in the Amazon Cognito console.
 - b. Choose the destination user pool attribute from the drop-down list.
 - c. Choose **Save changes**.
 - d. Choose **Go to summary**.

New console

To configure a user pool social IdP with the AWS Management Console

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. Choose the **Sign-in experience** tab. Locate **Federated sign-in** and then select **Add an identity provider**.
5. Choose a social IdP: **Facebook**, **Google**, **Login with Amazon**, or **Sign in with Apple**.
6. Choose from the following steps, based on your choice of social IdP:
 - **Google and Login with Amazon** — Enter the **app client ID** and **app client secret** generated in the previous section.
 - **Facebook** — Enter the **app client ID** and **app client secret** generated in the previous section, and then choose an API version (for example, version 2.12). We recommend that you choose the latest possible version, as each Facebook API has a lifecycle and discontinuation date. Facebook scopes and attributes can vary between API versions. We recommend that you test your social identity log in with Facebook to make sure that federation works as you intend.
 - **Sign In with Apple** — Enter the **Services ID**, **Team ID**, **Key ID**, and **private key** generated in the previous section.
7. Enter the names of the **Authorized scopes** you want to use. Scopes define which user attributes (such as `name` and `email`) you want to access with your app. For Facebook, these should be separated by commas. For Google and Login with Amazon, they should be separated by spaces. For Sign in with Apple, select the check boxes for the scopes you want access to.

Social identity provider	Example scopes
Facebook	<code>public_profile, email</code>
Google	<code>profile email openid</code>

Social identity provider	Example scopes
Login with Amazon	profile postal_code
Sign in with Apple	email name

Your app user is prompted to consent to providing these attributes to your app. For more information about social provider scopes, see the documentation from Google, Facebook, Login with Amazon, or Sign in with Apple.

With Sign in with Apple, the following are user scenarios where scopes might not be returned:

- An end user encounters failures after leaving Apple's sign in page (can be from Internal failures within Amazon Cognito or anything written by the developer)
 - The service ID identifier is used across user pools and/or other authentication services
 - A developer adds additional scopes after the end user has signed in before (no new information is retrieved)
 - A developer deletes the user and then the user signs in again without removing the app from their Apple ID profile
8. Map attributes from your IdP to your user pool. For more information, see [Specifying Identity Provider Attribute Mappings for Your User Pool](#).
 9. Choose **Create**.
 10. From the **App client integration** tab, choose one of the **App clients** in the list and **Edit hosted UI settings**. Add the new social IdP to the app client under **Identity providers**.
 11. Choose **Save changes**.

Step 3: Test your social IdP configuration

You can create a login URL by using the elements from the previous two sections. Use it to test your social IdP configuration.

```
https://<your_user_pool_domain>/login?  
response_type=code&client_id=<your_client_id>&redirect_uri=https://www.example.com
```

You can find your domain on the user pool **Domain name** console page. The client_id is on the **App client settings** page. Use your callback URL for the **redirect_uri** parameter. This is the URL of the page where your user will be redirected after a successful authentication.

Note

Amazon Cognito cancels authentication requests that do not complete within 5 minutes, and redirects the user to the hosted UI. The page displays a `Something went wrong` error message.

Adding SAML identity providers to a user pool

You can enable your web and mobile app users to sign in through a SAML identity provider (IdP) such as [Microsoft Active Directory Federation Services \(ADFS\)](#), or [Shibboleth](#). You must choose a SAML IdP which supports the [SAML 2.0 standard](#).

With the built-in hosted web UI, Amazon Cognito provides token handling and management for all authenticated users. This way, your backend systems can standardize on one set of user pool tokens. You can create and manage a SAML IdP in the AWS Management Console through the AWS CLI or with

Amazon Cognito API calls. To get started with the console see [Adding sign-in through SAML-based identity providers to a user pool with the AWS Management Console](#).



Note

Sign-in through a third party (federation) is available in Amazon Cognito user pools. This feature is independent of federation through Amazon Cognito identity pools (federated identities).

You need to update your SAML IdP and configure your user pool to support the provider. See the documentation for your SAML IdP for information about how to add your user pool as a relying party or application for your SAML 2.0 IdP.

Note

Amazon Cognito supports `relayState` values greater than 80 bytes. While SAML specifications state that the `relayState` value "Must not exceed 80 bytes in length", current industry practice often deviates from this behavior. As a consequence, rejecting `relayState` values greater than 80 bytes will break many standard SAML provider integrations.

You also need to provide an assertion consumer endpoint to your SAML IdP. Configure this endpoint for SAML 2.0 POST binding in your SAML IdP:

```
https://<yourDomainPrefix>.auth.<region>.amazoncognito.com/saml2/idpresponse
```

You can find your domain prefix and the region value for your user pool on the **Domain name** tab of the [Amazon Cognito console](#).

For some SAML IdPs, you also need to provide the SP urn / Audience URI / SP Entity ID, in the form:

```
urn:amazon:cognito:sp:<yourUserPoolID>
```

You can find your user pool ID on the **General settings** tab in the [Amazon Cognito console](#).

You should also configure your SAML IdP to provide attribute values for any attributes that are required in your user pool. For example, `email` is a typical required attribute for user pools. Thus, the SAML IdP should provide an `email` value (claim) in the SAML assertion.

Amazon Cognito user pools support SAML 2.0 federation with post-binding endpoints. This eliminates the need for your app to retrieve or parse SAML assertion responses, because the user pool directly receives the SAML response from your IdP through a user agent. Your user pool acts as a service provider (SP) on behalf of your application. Amazon Cognito supports SP-initiated single sign-on (SSO) as described in section 5.1.2 of the [SAML V2.0 Technical Overview](#).

Topics

- [SAML user pool IdP authentication flow \(p. 70\)](#)
- [Choosing SAML identity provider names \(p. 71\)](#)
- [Creating and managing a SAML identity provider for a user pool \(AWS Management Console\) \(p. 72\)](#)
- [Creating and managing a SAML identity provider for a user pool \(AWS CLI and AWS API\) \(p. 75\)](#)
- [Integrating third-party SAML identity providers with Amazon Cognito user pools \(p. 76\)](#)
- [SAML session initiation in Amazon Cognito user pools \(p. 78\)](#)

SAML user pool IdP authentication flow

You can integrate SAML-based IdPs directly from your user pool.

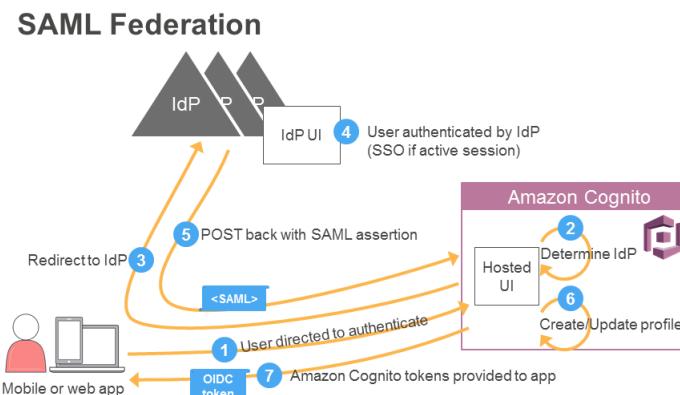
1. The app starts the sign-up and sign-in process by directing your user to the UI hosted by AWS. A mobile app can use web view to show the pages hosted by AWS.
 2. Typically, your user pool determines the IdP for your user from that user's email address.
- Alternatively, if your app gathered information before directing the user to your user pool, it can provide that information to Amazon Cognito through a query parameter.
3. Your user is redirected to the IdP.
 4. The IdP authenticates the user if necessary. If the IdP recognizes that the user has an active session, the IdP skips the authentication to provide a single sign-in (SSO) experience.
 5. The IdP POSTs the SAML assertion to the Amazon Cognito service.

Note

Amazon Cognito cancels authentication requests that do not complete within 5 minutes, and redirects the user to the hosted UI. The page displays a *Something went wrong* error message.

6. After verifying the SAML assertion and collecting the user attributes (claims) from the assertion, Amazon Cognito internally creates or updates the user's profile in the user pool. Amazon Cognito returns OIDC tokens to the app for the now signed-in user.

The following diagram shows the authentication flow for this process:



When a user authenticates, the user pool returns ID, access, and refresh tokens. The ID token is a standard OIDC token for identity management, while the access token is a standard [OAuth 2.0](#) token. The ID and access tokens expire after one hour. Your app can use a refresh token to get new tokens without having the user re-authenticate.

As a developer, you can choose the expiration time for refresh tokens, which changes how frequently users need to reauthenticate. If the user has authenticated through an external IdP as a federated user, your app uses the Amazon Cognito tokens with the refresh token to determine how long until the user reauthenticates, regardless of when the external IdP token expires. The user pool automatically uses the refresh token to get new ID and access tokens when they expire. If the refresh token has also expired, the server automatically initiates authentication through the pages in your app that AWS hosts.

Case sensitivity of SAML user names

When a federated user attempts to sign in, the SAML identity provider (IdP) passes a unique `NameId` from the IdP directory to Amazon Cognito in the user's SAML assertion. Amazon Cognito identifies a

SAML-federated user by their `NameId` claim. Regardless of the case sensitivity settings of your user pool, Amazon Cognito requires that a federated user from a SAML IdP pass a unique and case-sensitive `NameId` claim. If you map an attribute like `email` to `NameId`, and your user changes their email address, they can't sign in to your app.

Map `NameId` in your SAML assertions from an IdP attribute that has values that don't change.

For example, Carlos has a user profile in your case-insensitive user pool from an Active Directory Federation Services (ADFS) SAML assertion that passed a `NameId` value of `Carlos@example.com`. The next time Carlos attempts to sign in, your ADFS IdP passes a `NameId` value of `carlos@example.com`. Because `NameId` must be an exact case match, the sign-in doesn't succeed.

If your users can't log in after their `NameID` changes, delete their user profiles from your user pool. Amazon Cognito will create new user profiles the next time they sign in.

Choosing SAML identity provider names

Choose names for your SAML providers. The string can be up to 32 characters long and must match the following regular expression:

```
[ \p{L}\p{M}\p{S}\p{N}\p{P}\p{Z} ]+ | [ ^\p{Z} ][ \p{L}\p{M}\p{S}\p{N}\p{P} ][ ^\p{Z} ]+
```

You can also choose identifiers for your SAML providers. An identifier uniquely resolves to an identity provider (IdP) associated with your user pool. Typically, each identifier corresponds to an organization domain that the SAML IdP represents. For a multi-tenant app that multiple organizations share, you can use identifiers to redirect users to the correct IdP. Because the same organization can own multiple domains, you can provide multiple identifiers. To sign in your users with an identifier, direct their sessions to the [Authorize endpoint \(p. 420\)](#) for your app client with an `idp_identifier` parameter.

You can associate up to 50 identifiers with each SAML provider. Identifiers must be unique across the user pool.

You can create links to your Amazon Cognito user pool `/authorize` endpoint that silently redirect your user to the sign-in page for your IdP. Include an identifier as an `idp_identifier` parameter in a request to your user pool `/authorize` endpoint. Alternatively, you can populate the parameter `identity_provider` with the name of your IdP. Use the following URL format:

```
https://your_Amazon_Cognito_userpool_domain/authorize?  
response_type=code&identity_provider=your-SAML-IdP-name&client_id=your-client-  
id&redirect_uri=https://your_application_redirect_url
```

After a user signs in with your SAML IdP, they transparently submit the SAML response to your `saml2/idpresponse` endpoint. Amazon Cognito replies to the IdP response with a redirect to your app client callback URL. The user hasn't seen any interaction with your hosted UI.

Create IdP identifiers in a domain format to collect user email addresses and direct them to the sign-in page for the IdP that corresponds to their domain. For example, suppose that you built an app for employees of two different companies to use. The first company, AnyCompany A, owns `exampleA.com` and `exampleA.co.uk`. The second company, AnyCompany B, owns `exampleB.com`. For this example, you have set up two IdPs, one for each company, as follows:

- For IdP A, you define identifiers `exampleA.com` and `exampleA.co.uk`.
- For IdP B, you define identifier `exampleB.com`.

In your app, you can prompt users to enter their email addresses. Your app derives the users' domain from their email address and redirects them to the correct IdP by providing the domain in the

`idp_identifier` parameter in the call to the `/authorize` endpoint. For example, if a user enters `bob@exampleA.co.uk`, the user is redirected to IdP A.

The Amazon Cognito hosted UI can collect an email address for federated users and parse the domain. To do this, assign at least one identifier to each SAML IdP that you have assigned to your app client. If you have successfully assigned identifiers, your hosted UI sign-in page looks like the following image.

To use this strategy, you must use domains as your IdP identifiers. Amazon Cognito extracts the email domain from the email address that your user enters. Amazon Cognito then adds the email domain as an `IdPIdentifier` parameter in a request to the `/authorize` endpoint. Your user sees an IdP sign-in page after they choose the **Sign in** button shown in the previous image.

- If you assign only one SAML IdP to your app client, the hosted UI sign-in page displays a button to sign in with that IdP.
- If you assign an identifier to every SAML IdP that you activate for your app client, a box to enter an email address appears in the hosted UI sign-in page.
- If you have multiple IdPs and you do not assign an identifier to all of them, the hosted page will contain a list of IdPs.

If you have a custom UI, parse the domain name so that it matches the identifiers that you assigned when you added the IdP. For more information about IdP setup, see [Configuring identity providers for your user pool \(p. 231\)](#).

Creating and managing a SAML identity provider for a user pool (AWS Management Console)

You can use the AWS Management Console to create and delete SAML identity providers (IdPs).

Before you create a SAML IdP, you will need the SAML metadata document that you get from the third-party IdP. For instructions on how to get or generate the required SAML metadata document, see [Integrating third-party SAML identity providers with Amazon Cognito user pools \(p. 76\)](#).

Original console

To configure a SAML 2.0 IdP in your user pool

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **Manage User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. On the left navigation bar, choose **Identity providers**.
5. Choose **SAML** to open the SAML dialog.
6. Under **Metadata document**, upload a metadata document from your SAML IdP. You can also enter a URL that points to the metadata document. For more information, see [Integrating third-party SAML identity providers with Amazon Cognito user pools \(p. 76\)](#).

Note

We recommend that you provide the endpoint URL if it is a public endpoint, rather than uploading a file, because this allows Amazon Cognito to refresh the metadata automatically. Typically, metadata refresh happens every 6 hours or before the metadata expires, whichever is earlier.

7. Enter your **SAML Provider name**. For more information on SAML naming see [Choosing SAML identity provider names \(p. 71\)](#).
8. Enter any optional SAML **Identifiers** you want to use.

9. Select **Enable IdP sign out flow** if you want your user to be logged out from both Amazon Cognito and the SAML IdP when they log out from your user pool.

Enabling this flow sends a signed log out request to the SAML IdP when the [Logout-endpoint](#) is called.

Configure this endpoint for consuming log out responses from your IdP. This endpoint uses post binding.

```
https://<yourDomainPrefix>.auth.<region>.amazoncognito.com/saml2/logout
```

Note

If this option is selected and your SAML IdP expects a signed logout request, you will also need to configure the signing certificate provided by Amazon Cognito with your SAML IdP.

The SAML IdP will process the signed log out request and log your user out from the Amazon Cognito session.

10. Choose **Create provider**.

11. On the **Attribute mapping** tab, add mappings for at least the required attributes, typically `email`, as follows:

- a. Enter the SAML attribute name as it appears in the SAML assertion from your IdP. Your IdP might offer sample SAML assertions for reference. Some IdPs use simple names, such as `email`, while others use URL-formatted attribute names similar to:

```
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
```

- b. Choose the destination user pool attribute from the drop-down list.

12. Choose **Save changes**.

13. Choose **Go to summary**.

New console

To configure a SAML 2.0 IdP in your user pool

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. Choose the **Sign-in experience** tab. Locate **Federated sign-in** and choose **Add an identity provider**.
5. Choose a **SAML IdP**.
6. Enter **Identifiers** separated by commas. An identifier tells Amazon Cognito it should check the email address a user enters when they sign in, and then direct them to the provider that corresponds to their domain.
7. Choose **Add sign-out flow** if you want Amazon Cognito to send signed sign-out requests to your provider when a user logs out. You must configure your SAML 2.0 IdP to send sign-out responses to the `https://<your Amazon Cognito domain>/saml2/logout` endpoint that is created when you configure the hosted UI. The `saml2/logout` endpoint uses POST binding.

Note

If this option is selected and your SAML IdP expects a signed logout request, you will also need to configure the signing certificate provided by Amazon Cognito with your SAML IdP.

The SAML IdP will process the signed logout request and sign out your user from the Amazon Cognito session.

8. Choose a **Metadata document source**. If your IdP offers SAML metadata at a public URL, you can choose **Metadata document URL** and enter that public URL. Otherwise, choose **Upload metadata document** and select a metadata file you downloaded from your provider earlier.

Note

We recommend that you enter a metadata document URL if your provider has a public endpoint, rather than uploading a file; this allows Amazon Cognito to refresh the metadata automatically. Typically, metadata refresh happens every 6 hours or before the metadata expires, whichever is earlier.

9. Map attributes between your SAML provider and your app to map SAML provider attributes to the user profile in your user pool. Include your user pool required attributes in your attribute map.

For example, when you choose **User pool attribute** `email`, enter the SAML attribute name as it appears in the SAML assertion from your IdP. If your IdP offers sample SAML assertions, you can use these sample assertions to help you to find the name. Some IdPs use simple names, such as `email`, while others use names similar to this:

```
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
```

10. Choose **Create**.

Note

If you see `InvalidArgumentException` while creating a SAML IdP with an HTTPS metadata endpoint URL, for example, "Error retrieving metadata from <`metadata endpoint`>," make sure that the metadata endpoint has SSL correctly set up and that there is a valid SSL certificate associated with it.

To set up the SAML IdP to add a user pool as a relying party

- The user pools service provider URN is: `urn:amazon:cognito:sp:<user_pool_id>`. Amazon Cognito issues the `AuthnRequest` to SAML IdP to issue a SAML assertion with audience restriction to this URN. Your IdP uses the following POST binding endpoint for the IdP-to-SP response message: `https://<domain_prefix>.auth.<region>.amazoncognito.com/saml2/idpresponse`.
- Make sure your SAML IdP populates `NameID` and any required attributes for your user pool in the SAML assertion. `NameID` is used for uniquely identifying your SAML federated user in the user pool. Use persistent SAML Name ID format.

To set up the SAML IdP to add a signing certificate

- To get the certificate containing the public key which will be used by the IdP to verify the signed logout request, choose **Show signing certificate** under **Active SAML Providers** on the **SAML** dialog under **Identity providers** on the **Federation** console page.

You can delete any SAML provider you have set up in your user pool with the Amazon Cognito console.

Original console

To delete a SAML provider

1. Sign in to the [Amazon Cognito console](#). If prompted, enter your AWS credentials
2. In the navigation pane, choose **Manage your User Pools**, and choose the user pool you want to edit.

3. Choose **Identity providers** from the **Federation** console page.
4. Choose **SAML** to display the SAML identity providers.
5. Select the check box next to the provider to be deleted.
6. Choose **Delete provider**.

New console

To delete a SAML provider

1. Sign in to the [Amazon Cognito console](#).
2. In the navigation pane, choose **User Pools**, and choose the user pool you want to edit.
3. Choose the **Sign-in experience** tab and locate **Federated sign-in**.
4. Select the radio button next to the SAML IdPs you wish to delete.
5. When you are prompted to **Delete identity provider**, enter the name of the SAML provider to confirm deletion, and then choose **Delete**.

Creating and managing a SAML identity provider for a user pool (AWS CLI and AWS API)

Use the following commands to create and manage a SAML identity provider (IdP).

To create an IdP and upload a metadata document

- AWS CLI: `aws cognito-idp create-identity-provider`

Example with metadata file: `aws cognito-idp create-identity-provider --user-pool-id <user_pool_id> --provider-name=SAML_provider_1 --provider-type SAML --provider-details file:///details.json --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

Where `details.json` contains:

```
{  
    "MetadataFile": "<SAML metadata XML>"  
}
```

Note

If the `<SAML metadata XML>` contains any quotations ("), they must be escaped (\").

Example with metadata URL: `aws cognito-idp create-identity-provider --user-pool-id <user_pool_id> --provider-name=SAML_provider_1 --provider-type SAML --provider-details MetadataURL=<metadata_url> --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

- AWS API: [CreateIdentityProvider](#)

To upload a new metadata document for an IdP

- AWS CLI: `aws cognito-idp update-identity-provider`

Example with metadata file: `aws cognito-idp update-identity-provider --user-pool-id <user_pool_id> --provider-name=SAML_provider_1 --provider-details file:///details.json --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

Where details.json contains:

```
{  
    "MetadataFile": "<SAML metadata XML>"  
}
```

Note

If the `<SAML metadata XML>` contains any quotations ("), they must be escaped (\").

Example with metadata URL: `aws cognito-identity-provider --user-pool-id <user_pool_id> --provider-name=SAML_provider_1 --provider-details MetadataURL=<metadata_url> --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

- AWS API: [UpdateIdentityProvider](#)

To get information about a specific IdP

- AWS CLI: `aws cognito-identity-provider describe-identity-provider`

```
aws cognito-identity-provider --user-pool-id <user_pool_id> --  
provider-name=SAML_provider_1
```

- AWS API: [DescribeldentityProvider](#)

To list information about all IdPs

- AWS CLI: `aws cognito-identity-provider list-identity-providers`

Example: `aws cognito-identity-provider list-identity-providers --user-pool-id <user_pool_id> --max-results 3`

- AWS API: [ListIdentityProviders](#)

To delete an IdP

- AWS CLI: `aws cognito-identity-provider delete-identity-provider`

```
aws cognito-identity-provider --user-pool-id <user_pool_id> --  
provider-name=SAML_provider_1
```

- AWS API: [DeleteIdentityProvider](#)

Integrating third-party SAML identity providers with Amazon Cognito user pools

To configure third-party SAML 2.0 identity provider (IdP) solutions to work with federation for Amazon Cognito user pools, you must enter the following redirect URL:
`https://<yourDomainPrefix>.auth.<region>.amazoncognito.com/saml2/idpresponse`. You can find your domain prefix and the region value for your user pool on the **Domain name** console page of the [Amazon Cognito console](#).

Note

Any SAML IdPs that you created in a user pool during the public beta before August 10, 2017 have redirect URLs of

`https://<yourDomainPrefix>.auth.<region>.amazoncognito.com/login/`

redirect. If you have one of these SAML identity providers from the public beta in your user pool, you must either:

- Replace it with a new one that uses the new redirect URL.
- Update the configuration in your SAML IdP to accept both the old and new redirect URLs.

All SAML IdPs in Amazon Cognito switch to the new URLs. The old URL will stop working on October 31, 2017.

For some SAML IdPs, provide the `urn` / Audience URI / SP Entity ID, in the form `urn:amazon:cognito:sp:<yourUserPoolID>`. You can find your user pool ID on the **General settings** tab in the Amazon Cognito console.

You must also configure your SAML IdP to provide attributes values for any attributes required in your user pool. Typically, `email` is a required attribute for user pools, in which case the SAML IdP must provide an `email` value (claim) in the SAML assertion.

Note

Amazon Cognito will not accept an emoji that your IdP passes as an attribute value. You can Base64 encode the emoji to pass it as text, and then decode it in your app.

In the following example, the attribute claim will not be accepted:

```
<saml2:Attribute Name="Name" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
    <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="xsd:string">##</saml2:AttributeValue>
</saml2:Attribute>
```

In contrast to the preceding example, the following attribute claim will be accepted:

```
<saml2:Attribute Name="Name" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
    <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="xsd:string">8J+YkA==</saml2:AttributeValue>
</saml2:Attribute>
```

The following links help you configure third-party SAML 2.0 IdP solutions to work with federation for Amazon Cognito user pools.

Note

Amazon Cognito includes IdP support, so you only need to go to the following provider sites to get the SAML metadata document. You might see further instructions on the provider website about integrating with AWS, but you won't need those.

Solution	More information
Microsoft Active Directory Federation Services (AD FS)	You can download the SAML metadata document for your ADFS federation server from the following address: <a href="https://<yourservername>/FederationMetadata/2007-06/FederationMetadata.xml">https://<yourservername>/FederationMetadata/2007-06/FederationMetadata.xml .
Okta	Once you have configured your Amazon Cognito user pool as an application in Okta, you can find the metadata document in the Admin section of the Okta dashboard. Choose the application, select the Sign On section, and look under the

Solution	More information
	Settings for SAML. The URL should look like <a href="https://<app-domain>.oktapreview.com/app/<application-ID>/sso/saml/metadata">https://<app-domain>.oktapreview.com/app/<application-ID>/sso/saml/metadata .
Auth0	The metadata download document is obtained from the Auth0 dashboard. Choose Clients , and then choose Settings . Scroll down, choose Show Advanced Settings , and then look for your SAML Metadata URL . It should look like <a href="https://<your-domain-prefix>.auth0.com/samlp/metadata/<your-Auth0-client-ID>">https://<your-domain-prefix>.auth0.com/samlp/metadata/<your-Auth0-client-ID> .
Ping Identity	For PingFederate, you can find instructions for downloading a metadata XML file in Provide general SAML metadata by file .

SAML session initiation in Amazon Cognito user pools

Amazon Cognito supports service provider-initiated (SP-initiated) single sign-on (SSO). Section 5.1.2 of the [SAML V2.0 Technical Overview](#) describes SP-initiated SSO. Amazon Cognito is the identity provider (IdP) to your app. The app is the service provider (SP) that retrieves tokens for authenticated users. However, when you use a third-party IdP to authenticate users, Amazon Cognito is the SP. Amazon Cognito users who authenticate with a SAML IdP must always first make a request to Amazon Cognito and redirect to the IdP for authentication.

For some enterprise use cases, access to internal applications starts at a bookmark on a dashboard hosted by the enterprise IdP. When a user selects a bookmark, the IdP generates a SAML response and sends it to the SP to authenticate the user with the application.

Amazon Cognito doesn't support IdP-initiated SSO. Amazon Cognito can't verify that it has solicited the SAML response that it receives unless Amazon Cognito initiates authentication with a SAML request.

Bookmark Amazon Cognito apps in an enterprise dashboard

You can create bookmarks in your SAML or [OIDC](#) IdP dashboards that provide SSO access to web applications that authenticate with Amazon Cognito user pools. You can link to Amazon Cognito in a way that doesn't require users to sign in with the hosted UI. To do this, construct a bookmark link to the [Authorize endpoint \(p. 420\)](#) of your Amazon Cognito user pool from the following template:

```
https://<your_Amazon_Cognito_userpool_domain>/authorize?
response_type=code&identity_provider=<your-SAML-IdP-name>&client_id=<your-client-id>
&redirect_uri=<https://><your_application_redirect_url>
```

Note

You can also use an `idp_identifier` parameter instead of an `identity_provider` parameter in your request to the authorization endpoint. An IdP identifier is an alternative name you can configure when you create an identity provider in your user pool. See [Choosing SAML identity provider names \(p. 71\)](#).

When you use the appropriate parameters in your request to `/authorize`, Amazon Cognito silently begins the SP-initiated sign-in flow and redirects your user to sign in with your IdP.

The following diagram shows an authentication flow that emulates IdP-initiated SSO. Your users can authenticate with Amazon Cognito from a link in your company portal.

Upload or link a metadata document from your provider to add a SAML IdP in your user pool. Create an app client that uses your SAML IdP for sign-in, and has the URL for your app as an authorized callback URL. For more information about app clients, see [Configuring a user pool app client \(p. 223\)](#).

Before you try to use this solution to authenticate to your app, test SP-initiated sign-in to your app from your hosted UI. For more information about how to configure a SAML IdP in Amazon Cognito, see [Integrating third-party SAML identity providers with Amazon Cognito user pools \(p. 76\)](#).

After you meet these requirements, create a bookmark to your `/authorize` endpoint that includes either an `identity_provider` or an `idp_identifier` parameter. The preceding diagram illustrates the process that follows.

1. Your user signs in to the SSO IdP dashboard. Enterprise applications that the user is authorized to access populate this dashboard.
2. Your user chooses the link to the application that authenticates with Amazon Cognito. In many SSO portals, you can add a custom app link. Any feature that you can use to create a link to a public URL in your SSO portal will work.
3. Your custom app link in the SSO portal directs the user to the user pool [Authorize endpoint \(p. 420\)](#). The link includes parameters for `response_type`, `client_id`, `redirect_uri` and `identity_provider`. The `identity_provider` parameter is the name that you gave the IdP in your user pool. You can also use an `idp_identifier` parameter instead of the `identity_provider` parameter. A user accesses your hosted UI from a link that contains either a `idp_identifier` or `identity_provider` parameter. This user bypasses the sign-in page and navigates directly to authenticate with your IdP. For more information about naming SAML IdPs, see [Choosing SAML identity provider names \(p. 71\)](#).

Example URL:

```
https://your_Amazon_Cognito_userpool_domain/authorize?  
response_type=code&identity_provider=your-SAML-IdP-name&client_id=your-  
client-id&redirect_uri=https://your_application_redirect_url
```

4. Amazon Cognito redirects the user session to your IdP with a SAML request.
5. Your user might have received a session cookie from your IdP when they signed in to the dashboard. Your IdP uses this cookie to validate the user silently and redirect them to the Amazon Cognito `idpresponse` endpoint with a SAML response. If no active session exists, your IdP re-authenticates the user before it posts the SAML response.
6. Amazon Cognito validates the SAML response and creates or updates the user profile based on the SAML assertion.
7. Amazon Cognito redirects the user to your internal app with an authorization code. You configured your internal app URL as an authorized redirect URL for your app client.
8. Your app exchanges the authorization code for Amazon Cognito tokens.

Adding OIDC identity providers to a user pool

You can enable your users who already have accounts with [OpenID Connect \(OIDC\)](#) identity providers (IdPs) to skip the sign-up step and sign in to your application using an existing account. With the built-in hosted web UI, Amazon Cognito provides token handling and management for all authenticated users. This way, your backend systems can standardize on one set of user pool tokens.



Note

Sign-in through a third party (federation) is available in Amazon Cognito user pools. This feature is independent of federation through Amazon Cognito identity pools (federated identities).

You can add an OIDC IdP to your user pool in the AWS Management Console, through the AWS CLI, or with the user pool API method [CreateIdentityProvider](#).

Topics

- [Prerequisites \(p. 80\)](#)
- [Step 1: Register with an OIDC IdP \(p. 80\)](#)
- [Step 2: Add an OIDC IdP to your user pool \(p. 82\)](#)
- [Step 3: Test your OIDC IdP configuration \(p. 85\)](#)
- [OIDC user pool IdP authentication flow \(p. 85\)](#)

Prerequisites

Before you begin, you need the following:

- A user pool with an app client and a user pool domain. For more information, see [Create a user pool](#).
- An OIDC IdP with the following configuration:
 - Supports `client_secret_post` client authentication. Amazon Cognito doesn't check the `token_endpoint_auth_methods_supported` claim at the OIDC discovery endpoint for your IdP. Amazon Cognito doesn't support `client_secret_basic` client authentication. For more information on client authentication, see [Client Authentication](#) in the OpenID Connect documentation.
 - Only uses HTTPS for OIDC endpoints such as `openid_configuration`, `userInfo`, and `jwks_uri`.
 - Only uses TCP ports 80 and 443 for OIDC endpoints.

Step 1: Register with an OIDC IdP

Before you create an OIDC IdP with Amazon Cognito, you must register your application with the OIDC IdP to receive a client ID and client secret.

To register with an OIDC IdP

1. Create a developer account with the OIDC IdP.

Links to OIDC IdPs

OIDC IdP	How to Install	OIDC Discovery URL
Salesforce	Install a Salesforce identity provider	https://login.salesforce.com
Ping Identity	Install a Ping Identity identity provider	https://Your_Ping_domain_address:9031/idp/userinfo.openid For example: https://pf.company.com:9031/idp/userinfo.openid
Okta	Install an Okta identity provider	https://Your_Okta_subdomain.oktapreview.com

OIDC IdP	How to Install	OIDC Discovery URL
		or https://Your_Okta_subdomain.okta.com
Microsoft Azure Active Directory (Azure AD)	Install a Microsoft Azure AD identity provider	https://login.microsoftonline.com/{tenant}/v2.0
Google	Install a Google identity provider	https://accounts.google.com Note Amazon Cognito offers Google as an integrated social sign-in IdP. We recommend that you use the integrated IdP. See Adding social identity providers to a user pool (p. 62) .

2. Register your user pool domain URL with the /oauth2/idpresponse endpoint with your OIDC IdP. This ensures that the OIDC IdP later accepts it from Amazon Cognito when it authenticates users.

`https://<your-user-pool-domain>/oauth2/idpresponse`

3. Register your callback URL with your Amazon Cognito user pool. This is the URL of the page where Amazon Cognito redirects your user after a successful authentication.

`https://www.example.com`

4. Select your [scopes](#). The scope **openid** is required. The **email** scope is needed to grant access to the **email** and **email_verified** [claims](#).
5. The OIDC IdP provides you with a client ID and a client secret. You'll use them when you set up an OIDC IdP in your user pool.

Example: Use Salesforce as an OIDC IdP with your user pool

You use an OIDC IdP when you want to establish trust between an OIDC-compatible IdP such as Salesforce and your user pool.

1. [Create an account](#) on the Salesforce Developers website.
2. [Sign in](#) through your developer account that you set up in the previous step.
3. From your Salesforce page, do one of the following:
 - If you're using Lightning Experience, choose the setup gear icon, then choose **Setup Home**.
 - If you're using Salesforce Classic and you see **Setup** in the user interface header, choose it.
 - If you're using Salesforce Classic and you don't see **Setup** in the header, choose your name from the top navigation bar, and choose **Setup** from the drop-down list.
4. On the left navigation bar, choose **Company Settings**.
5. On the navigation bar, choose **Domain**, enter a domain, and choose **Create**.
6. On the left navigation bar, under **Platform Tools**, choose **Apps**.
7. Choose **App Manager**.
8. a. Choose **New connected app**.

- b. Complete the required fields.

Under **Start URL**, enter a URL at the /authorize endpoint for the user pool domain that signs in with your Salesforce IdP. When your users access your connected app, Salesforce directs them to this URL to complete sign-in. Then Salesforce redirects the users to the callback URL that you have associated with your app client.

```
https://<your_user_pool_domain>/authorize?  
response_type=code&client_id=<your_client_id>&redirect_uri=https://  
www.example.com&identity_provider=CorpSalesforce
```

- c. Enable **OAuth settings** and enter the URL of the /oauth2/idpresponse endpoint for your user pool domain in **Callback URL**. This is the URL where Salesforce issues the authorization code that Amazon Cognito exchanges for an OAuth token.

```
https://<your_user_pool_domain>/oauth2/idpresponse
```

9. Select your **scopes**. You must include the scope **openid**. To grant access to the **email** and **email_verified** claims, add the **email** scope. Separate scopes by spaces.

10. Choose **Create**.

In Salesforce, the client ID is called a **Consumer Key**, and the client secret is a **Consumer Secret**. Note your client ID and client secret. You will use them in the next section.

Step 2: Add an OIDC IdP to your user pool

In this section, you configure your user pool to process OIDC-based authentication requests from an OIDC IdP.

To add an OIDC IdP (Amazon Cognito console)

Original console

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. **Manage User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. On the left navigation bar, choose **Identity providers**.
5. Choose **OpenId Connect**.
6. Enter a unique name into **Provider name**.
7. Enter the OIDC IdP client ID from the previous section into **Client ID**.
8. Enter the client secret from the previous section into **Client secret**.
9. In the drop-down list, select the HTTP method (either GET or POST) used to fetch the details of the user from the **userinfo** endpoint into **Attributes request method**.
10. Enter the names of the scopes that you want to authorize. Scopes define which user attributes (such as `name` and `email`) that you want to access with your application. Scopes are separated by spaces, according to the [OAuth 2.0 specification](#).

Your app user is asked to consent to providing these attributes to your application.

11. Enter the URL of your IdP and choose **Run discovery**.

Note

The URL should start with `https://`, and shouldn't end with a slash `/`. Only port numbers 443 and 80 can be used with this URL. For example, Salesforce uses this URL: `https://login.salesforce.com`

- If **Run discovery** isn't successful, then you need to provide the **Authorization endpoint**, **Token endpoint**, **Userinfo endpoint**, and **Jwks uri** (the location of the [JSON Web Key](#)).

Note

If provider uses discovery for federated login, the discovery document must use HTTPS for the following values: `authorization_endpoint`, `token_endpoint`, `userinfo_endpoint`, and `jwks_uri`. Otherwise the login will fail.

12. Choose **Create provider**.
13. On the left navigation bar, choose **App client settings**.
14. Select the OIDC provider that you set up in the previous step as one of the **Enabled Identity Providers**.
15. Enter a callback URL for the Amazon Cognito authorization server to call after users are authenticated. This is the URL of the page where your user will be redirected after a successful authentication.

`https://www.example.com`

16. Under **Allowed OAuth Flows**, enable both the **Authorization code grant** and the **Implicit code grant**.

Unless you specifically want to exclude one, select the check boxes for all of the **Allowed OAuth scopes**.

17. Choose **Save changes**.
18. On the **Attribute mapping** tab on the left navigation bar, add mappings of OIDC claims to user pool attributes.
 - a. As a default, the OIDC claim **sub** is mapped to the user pool attribute **Username**. You can map other OIDC [claims](#) to user pool attributes. Enter the OIDC claim, and choose the corresponding user pool attribute from the drop-down list. For example, the claim **email** is often mapped to the user pool attribute **Email**.
 - b. In the drop-down list, choose the destination user pool attribute.
 - c. Choose **Save changes**.
 - d. Choose **Go to summary**.

New console

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **User Pools** from the navigation menu.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. Choose the **Sign-in experience** tab. Locate **Federated sign-in** and select **Add an identity provider**.
5. Choose an **OpenID Connect IdP**.
6. Enter a unique name into **Provider name**.
7. Enter the client ID you received from your provider into **Client ID**.
8. Enter the client secret you received from your provider into **Client secret**.
9. Enter **Authorized scopes** for this provider. Scopes define which groups of user attributes (such as `name` and `email`) that your application will request from your provider. Scopes must be separated by spaces, following the [OAuth 2.0 specification](#).

Your user will be asked to consent to providing these attributes to your application.

10. Choose an **Attribute request method** to provide Amazon Cognito with the HTTP method (either GET or POST) that it must use to fetch the details of the user from the **userInfo** endpoint operated by your provider.
 11. Choose a **Setup method** to retrieve OpenID Connect endpoints either by **Auto fill through issuer URL** or **Manual input**. Use **Auto fill through issuer URL** when your provider has a public `.well-known/openid-configuration` endpoint where Amazon Cognito can retrieve the URLs of the authorization, token, userInfo, and jwks_uri endpoints.
 12. Enter the issuer URL or authorization, token, userInfo, and jwks_uri endpoint URLs from your IdP.
- Note**
- The URL should start with `https://`, and shouldn't end with a slash `/`. Only port numbers 443 and 80 can be used with this URL. For example, Salesforce uses this URL: `https://login.salesforce.com`
- If you choose auto fill, the discovery document must use HTTPS for the following values: `authorization_endpoint`, `token_endpoint`, `userinfo_endpoint`, and `jwks_uri`. Otherwise the login will fail.
13. The OIDC claim **sub** is mapped to the user pool attribute **Username** by default. You can map other OIDC **claims** to user pool attributes. Enter the OIDC claim, and choose the corresponding user pool attribute from the drop-down list. For example, the claim **email** is often mapped to the user pool attribute **Email**.
 14. Map attributes from your IdP to your user pool. For more information, see [Specifying Identity Provider Attribute Mappings for Your User Pool](#).
 15. Choose **Create**.
 16. From the **App client integration** tab, choose one of the **App clients** in the list and **Edit hosted UI settings**. Add the new OIDC IdP to the app client under **Identity providers**.
 17. Choose **Save changes**.

To add an OIDC IdP (AWS CLI)

- See the parameter descriptions for the [CreateIdentityProvider](#) API method.

```
aws cognito-identity create-identity-provider
--user-pool-id string
--provider-name string
--provider-type OIDC
--provider-details map

--attribute-mapping string
--idp-identifiers (list)
--cli-input-json string
--generate-cli-skeleton string
```

Use this map of provider details:

```
{
  "client_id": "string",
  "client_secret": "string",
  "authorize_scopes": "string",
  "attributes_request_method": "string",
  "oidc_issuer": "string",
```

```
"authorize_url": "string",
"token_url": "string",
"attributes_url": "string",
"jwks_uri": "string"
}
```

Step 3: Test your OIDC IdP configuration

You can create the authorization URL by using the elements from the previous two sections, and using them to test your OIDC IdP configuration.

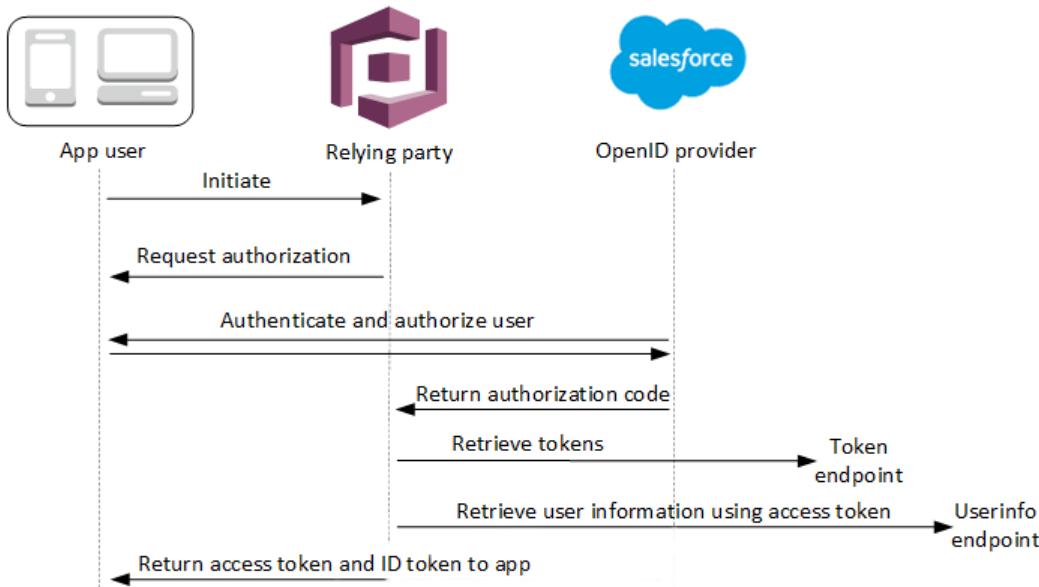
```
https://<your_user_pool_domain>/oauth2/authorize?
response_type=code&client_id=<your_client_id>&redirect_uri=https://www.example.com
```

You can find your domain on the user pool **Domain name** console page. The client_id is on the **General settings** page. Use your callback URL for the **redirect_uri** parameter. This is the URL of the page where your user will be redirected after a successful authentication.

OIDC user pool IdP authentication flow

When your user signs in to your application using an OIDC IdP, they pass through the following authentication flow.

1. Your user lands on the Amazon Cognito built-in sign-in page, and is offered the option to sign in through an OIDC IdP such as Salesforce.
2. Your user is redirected to the authorization endpoint of the OIDC IdP.
3. After your user is authenticated, the OIDC IdP redirects to Amazon Cognito with an authorization code.
4. Amazon Cognito exchanges the authorization code with the OIDC IdP for an access token.
5. Amazon Cognito creates or updates the user account in your user pool.
6. Amazon Cognito issues your application bearer tokens, which might include identity, access, and refresh tokens.



Note

Amazon Cognito cancels authentication requests that do not complete within 5 minutes, and redirects the user to the hosted UI. The page displays a `Something went wrong` error message.

OIDC is an identity layer on top of OAuth 2.0, which specifies JSON-formatted (JWT) identity tokens that are issued by IdPs to OIDC client apps (relying parties). See the documentation for your OIDC IdP for information about to add Amazon Cognito as an OIDC relying party.

When a user authenticates, the user pool returns ID, access, and refresh tokens. The ID token is a standard [OIDC](#) token for identity management, and the access token is a standard [OAuth 2.0](#) token.

Specifying identity provider attribute mappings for your user pool

You can use the AWS Management Console, or the AWS CLI or API, to specify attribute mappings for the identity provider (IdP) of your user pool.

Things to know about mappings

Before using mappings, review the following important details:

- When a federated user signs in to your application, a mapping must be present for each user pool attribute that your user pool requires. For example, if your user pool requires an `email` attribute for sign-up, map this attribute to its equivalent from the IdP.
- By default, mapped email addresses are unverified. You can't verify a mapped email address using a one-time code. Instead, map an attribute from your IdP to get the verification status. For example, Google and most OIDC providers include the `email_verified` attribute.
- For each mapped user pool attribute, the maximum value length (2048 characters) must be large enough for the value that Amazon Cognito obtains from the IdP. Otherwise, Amazon Cognito reports an error when users sign in to your application. If you map a custom attribute to an IdP token, set the length to 2048 characters.
- Amazon Cognito derives the `username` attribute in a federated user's profile from specific claims that your federated IdP passes, as follows. Amazon Cognito prepends this attribute value with the

name of your IdP. When you want your federated users to have an attribute that exactly matches an attribute in your third-party directory, map that attribute to an alternative Amazon Cognito attribute like `preferred_username`.

Identity Provider	username source attribute
Facebook	<code>id</code>
Google	<code>sub</code>
Login with Amazon	<code>user_id</code>
Sign in with Apple	<code>sub</code>
SAML providers	<code>NameID</code>
OpenID Connect (OIDC) providers	<code>sub</code>

- Amazon Cognito must be able to update your mapped user pool attributes when users sign in to your application. When a user signs in through an IdP, Amazon Cognito updates the mapped attributes with the latest information from the IdP. Amazon Cognito updates each mapped attribute, even if its current value already matches the latest information. If Amazon Cognito can't update the attribute, it reports an error. To ensure that Amazon Cognito can update the attributes, check the following requirements:
 - All of the user pool custom attributes that you map from your IdP must be *mutable*. You can update mutable custom attributes at any time. By contrast, you can only set a value for a user's *immutable* custom attribute when you first create the user profile. To create a mutable custom attribute in the Amazon Cognito console, activate the **Mutable** checkbox for the attribute you add when you select **Add custom attributes** in the **Sign-up experience** tab. Or, if you create your user pool by using the [CreateUserPool](#) API operation, you can set the `Mutable` parameter for each of these attributes to `true`.
 - In the app client settings for your application, the mapped attributes must be *writable*. You can set which attributes are writable in the **App clients** page in the Amazon Cognito console. Or, if you create the app client by using the [CreateUserPoolClient](#) API operation, you can add these attributes to the `WriteAttributes` array.
 - When IdP attributes contain multiple values, Amazon Cognito URL form encodes the values containing non-alphanumeric characters (excluding the '.', '-', '*', and '_' characters). You must decode the values before you use them in your application.

Specifying identity provider attribute mappings for your user pool (AWS Management Console)

You can use the AWS Management Console to specify attribute mappings for the IdP your user pool.

Note

Amazon Cognito will map incoming claims to user pool attributes only if the claims exist in the incoming token. If a previously mapped claim no longer exists in the incoming token, it won't be deleted or changed. If your application requires mapping of deleted claims, you can use the Pre-Authentication Lambda trigger to delete the custom attribute during authentication and allow these attributes to re-populate from the incoming token.

Original console

To specify a social IdP attribute mapping

1. Sign in to the [Amazon Cognito console](#). If prompted, enter your AWS credentials

2. In the navigation pane, choose **Manage your User Pools**, and choose the user pool you want to edit.
3. Choose the **Attribute mapping** tab.
4. Choose the **Facebook, Google, Amazon or Apple** tab.
5. For each attribute you need to map, complete the following steps:
 - a. Select the **Capture** check box.
 - b. For **User pool attribute**, in the drop-down list, choose the user pool attribute you want to map to the social IdP attribute.
 - c. If you need more attributes, choose **Add Facebook attribute** (or **Add Google attribute** or **Add Amazon attribute** or **Add Apple attribute**) and complete the following steps:
 - i. In the **Facebook attribute, Google attribute, Amazon attribute, or Apple attribute** field, enter the name of the attribute to be mapped.
 - ii. In the **User pool attribute** drop-down list, choose the user pool attribute to map to the social IdP attribute.
 - d. Choose **Save changes**.

To specify a SAML provider attribute mapping

1. Sign in to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. In the navigation pane, choose **Manage your User Pools**, and choose the user pool you want to edit.
3. Choose the **Attribute mapping** tab.
4. Choose the **SAML** tab.
5. Select the **Capture** box for all attributes for which you want to capture values. If you clear the **Capture** box for an attribute and save your changes, Amazon Cognito removes the mapping of that attribute.
6. Choose the IdP from the drop-down list.
7. For each attribute you need to map, complete the following steps:
 - a. Choose **Add SAML attribute**.
 - b. In the **SAML attribute** field, enter the name of the SAML attribute to be mapped.
 - c. In the **User pool attribute** field, choose the user pool attribute to map the SAML attribute to from the drop-down list.
8. Choose **Save changes**.

New console

To specify a social IdP attribute mapping

1. Sign in to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. In the navigation pane, choose **User Pools**, and choose the user pool you want to edit.
3. Choose the **Sign-in experience** tab and locate **Federated sign-in**.
4. Choose **Add an identity provider**, or choose the **Facebook, Google, Amazon or Apple** IdP you have configured. Locate **Attribute mapping** and choose **Edit**.

For more information about adding a social IdP, see [Adding social identity providers to a user pool \(p. 62\)](#).
5. For each attribute you need to map, complete the following steps:

- a. Select an attribute from the **User pool attribute** column. This is the attribute that is assigned to the user profile in your user pool. Custom attributes are listed after standard attributes.
 - b. Select an attribute from the **<provider> attribute** column. This will be the attribute passed from the provider directory. Known attributes from the social provider are provided in a drop-down list.
 - c. To map additional attributes between your IdP and Amazon Cognito, choose **Add another attribute**.
6. Choose **Save changes**.

To specify a SAML provider attribute mapping

1. Sign in to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
 2. In the navigation pane, choose **User Pools**, and choose the user pool you want to edit.
 3. Choose the **Sign-in experience** tab and locate **Federated sign-in**.
 4. Choose **Add an identity provider**, or choose the SAML IdP you have configured. Locate **Attribute mapping**, and choose **Edit**. For more information about adding a SAML IdP, see [Adding SAML identity providers to a user pool \(p. 68\)](#).
 5. For each attribute you need to map, complete the following steps:
 - a. Select an attribute from the **User pool attribute** column. This is the attribute that is assigned to the user profile in your user pool. Custom attributes are listed after standard attributes.
 - b. Select an attribute from the **SAML attribute** column. This will be the attribute passed from the provider directory.
- Your IdP might offer sample SAML assertions for reference. Some IdPs use simple names, such as `email`, while others use URL-formatted attribute names similar to:
- `http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`
- c. To map additional attributes between your IdP and Amazon Cognito, choose **Add another attribute**.
6. Choose **Save changes**.

Specifying identity provider attribute mappings for your user pool (AWS CLI and AWS API)

Use the following commands to specify IdP attribute mappings for your user pool.

To specify attribute mappings at provider creation time

- AWS CLI: `aws cognito-identity-provider create-identity-provider`

Example with metadata file: `aws cognito-identity-provider create-identity-provider --user-pool-id <user_pool_id> --provider-name=SAML_provider_1 --provider-type=SAML --provider-details file:///details.json --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

Where `details.json` contains:

```
{
```

```
    "MetadataFile": "<SAML metadata XML>"  
}
```

Note

If the `<SAML metadata XML>` contains any quotations ("), they must be escaped (\").

Example with metadata URL: `aws cognito-identity create-identity-provider --user-pool-id <user_pool_id> --provider-name=SAML_provider_1 --provider-type SAML --provider-details MetadataURL=<metadata_url> --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

- AWS API: [CreateIdentityProvider](#)

To specify attribute mappings for an existing IdP

- AWS CLI: `aws cognito-identity update-identity-provider`

Example: `aws cognito-identity update-identity-provider --user-pool-id <user_pool_id> --provider-name <provider_name> --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

- AWS API: [UpdateIdentityProvider](#)

To get information about attribute mapping for a specific IdP

- AWS CLI: `aws cognito-identity describe-identity-provider`

Example: `aws cognito-identity describe-identity-provider --user-pool-id <user_pool_id> --provider-name <provider_name>`

- AWS API: [DescribeIdentityProvider](#)

Linking federated users to an existing user profile

Often, the same user has a profile with multiple identity providers (IdPs) that you have connected to your user pool. Amazon Cognito can link each occurrence of a user to the same user profile in your directory. This way, one person who has multiple IdP users can have a consistent experience in your app. [AdminLinkProviderForUser](#) tells Amazon Cognito to recognize a user's unique ID in your federated directory as a user in the user pool. A user in your user pool counts as one monthly active user (MAU) for the purposes of [billing](#) when you have zero or more federated identities associated with the user profile.

Important

Because [AdminLinkProviderForUser](#) allows a user with an external federated identity to sign in as an existing user in the user pool, it is critical that it only be used with external IdPs and provider attributes that have been trusted by the application owner.

For example, if you're a managed service provider (MSP) with an app that you share with multiple customers. Each of the customers signs in to your app through Active Directory Federation Services (ADFS). Your IT administrator, Carlos, has an account in each of your customers' domains. You want Carlos to be recognized as an app administrator every time he signs in, regardless of the IdP.

Your ADFS IdPs present Carlos' email address `msp_carlos@example.com` in the `email` claim of the Carlos' SAML assertions to Amazon Cognito. You create a user in your user pool with the user name Carlos. The following AWS Command Line Interface (AWS CLI) commands link Carlos' identities from IdPs ADFS1, ADFS2, and ADFS3.

Note

You can link a user based on specific attribute claims. This ability is unique to SAML IdPs. For other provider types, you must link based on a fixed source attribute. For more information, see

[AdminLinkProviderForUser](#). You must set `ProviderAttributeName` to `Cognito_Subject` when you link an OIDC or social IdP to a user profile. `ProviderAttributeValue` must be the user's unique identifier with your IdP.

```
aws cognito-idp admin-link-provider-for-user \
--user-pool-id us-east-1_EXAMPLE \
--destination-user ProviderAttributeValue=Carlos,ProviderName=Cognito \
--source-user
  ProviderName=ADFS1,ProviderAttributeName=email,ProviderAttributeValue=msp_carlos@example.com

aws cognito-idp admin-link-provider-for-user \
--user-pool-id us-east-1_EXAMPLE \
--destination-user ProviderAttributeValue=Carlos,ProviderName=Cognito \
--source-user
  ProviderName=ADFS2,ProviderAttributeName=email,ProviderAttributeValue=msp_carlos@example.com

aws cognito-idp admin-link-provider-for-user \
--user-pool-id us-east-1_EXAMPLE \
--destination-user ProviderAttributeValue=Carlos,ProviderName=Cognito \
--source-user
  ProviderName=ADFS3,ProviderAttributeName=email,ProviderAttributeValue=msp_carlos@example.com
```

The user profile `Carlos` in your user pool now has the following `identities` attribute.

```
[{
  "userId": "msp_carlos@example.com",
  "providerName": "ADFS1",
  "providerType": "SAML",
  "issuer": "http://auth.example.com",
  "primary": false,
  "dateCreated": 1111111111111111
}, {
  "userId": "msp_carlos@example.com",
  "providerName": "ADFS2",
  "providerType": "SAML",
  "issuer": "http://auth2.example.com",
  "primary": false,
  "dateCreated": 1111111111111111
}, {
  "userId": "msp_carlos@example.com",
  "providerName": "ADFS3",
  "providerType": "SAML",
  "issuer": "http://auth3.example.com",
  "primary": false,
  "dateCreated": 1111111111111111
}]
```

Things to know about linking federated users

- You can link up to five federated users to each user profile.
- You can link federated users to either an existing federated user profile, or to a native user.
- You can't link providers to user profiles in the AWS Management Console.
- Your user's ID token contains all of their associated providers in the `identities` claim.

See [AdminLinkProviderForUser](#) for additional command syntax and examples in the AWS SDKs.

Customizing user pool workflows with Lambda triggers

You can create a Lambda function and then activate that function during user pool operations such as user sign-up, confirmation, and sign-in (authentication) with a Lambda trigger. You can add authentication challenges, migrate users, and customize verification messages.

The following table summarizes some of the ways you can use Lambda triggers to customize user pool operations:

User Pool Flow	Operation	Description
Custom Authentication Flow	Define Auth Challenge	Determines the next challenge in a custom auth flow
	Create Auth Challenge	Creates a challenge in a custom auth flow
	Verify Auth Challenge Response	Determines if a response is correct in a custom auth flow
Authentication Events	the section called “Pre authentication Lambda trigger” (p. 108)	Custom validation to accept or deny the sign-in request
	the section called “Post authentication Lambda trigger” (p. 110)	Logs events for custom analytics
	the section called “Pre token generation Lambda trigger” (p. 123)	Augments or suppresses token claims
Sign-Up	the section called “Pre sign-up Lambda trigger” (p. 98)	Performs custom validation that accepts or denies the sign-up request
	the section called “Post confirmation Lambda trigger” (p. 105)	Adds custom welcome messages or event logging for custom analytics
	the section called “Migrate user Lambda trigger” (p. 127)	Migrates a user from an existing user directory to user pools
Messages	the section called “Custom message Lambda trigger” (p. 131)	Performs advanced customization and localization of messages
Token Creation	the section called “Pre token generation Lambda trigger” (p. 123)	Adds or removes attributes in Id tokens
Email and SMS third-party providers	the section called “Custom sender Lambda triggers” (p. 136)	Uses a third-party provider to send SMS and email messages

Topics

- [Important considerations \(p. 93\)](#)
- [Adding a user pool Lambda trigger \(p. 95\)](#)
- [User pool Lambda trigger event \(p. 95\)](#)
- [User pool Lambda trigger common parameters \(p. 96\)](#)
- [User pool Lambda trigger sources \(p. 97\)](#)
- [Pre sign-up Lambda trigger \(p. 98\)](#)
- [Post confirmation Lambda trigger \(p. 105\)](#)
- [Pre authentication Lambda trigger \(p. 108\)](#)
- [Post authentication Lambda trigger \(p. 110\)](#)
- [Custom authentication challenge Lambda triggers \(p. 114\)](#)
- [Pre token generation Lambda trigger \(p. 123\)](#)
- [Migrate user Lambda trigger \(p. 127\)](#)
- [Custom message Lambda trigger \(p. 131\)](#)
- [Custom sender Lambda triggers \(p. 136\)](#)

Important considerations

Before you work with Lambda functions, consider the following:

- Except for Custom Sender Lambda triggers, Amazon Cognito invokes Lambda functions synchronously. When Amazon Cognito calls your Lambda function, it must respond within 5 seconds. If it does not, Amazon Cognito retries the call. After three unsuccessful attempts, the function times out. You can't change this five-second timeout value. For more information, see the [Lambda programming model](#).
- If you delete a Lambda trigger, you must update the corresponding trigger in the user pool. For example, if you delete the post authentication trigger, you must set the **Post authentication** trigger in the corresponding user pool to **none**.
- If your users use the Amazon Cognito hosted UI, they can see errors that Lambda triggers generate in query parameters that Amazon Cognito adds to the Callback URL, except for Custom Sender Lambda triggers. We recommend that Lambda triggers only generate errors that you want your users to see. Log any sensitive or debugging information in the Lambda trigger itself.
- When you create a Lambda trigger outside of the Amazon Cognito console, you must add permissions to the Lambda function. When you add permissions, Amazon Cognito can invoke the function on behalf of your user pool. You can [add permissions from the Lambda Console](#) or use the Lambda [AddPermission](#) operation.

Example Lambda Resource-Based Policy

The following example Lambda resource-based policy grants Amazon Cognito a limited ability to invoke a Lambda function. Amazon Cognito can only invoke the function when it does so on behalf of both the user pool in the `aws:SourceArn` condition and the account in the `aws:SourceAccount` condition.

```
{  
    "Version": "2012-10-17",  
    "Id": "default",  
    "Statement": [  
        {  
            "Sid": "lambda-allow-cognito",  
            "Effect": "Allow",  
            "Principal": {
```

```

        "Service": "cognito-idp.amazonaws.com"
    },
    "Action": "lambda:InvokeFunction",
    "Resource": "<your Lambda function ARN>",
    "Condition": {
        "StringEquals": {
            "AWS:SourceAccount": "<your account number>"
        },
        "ArnLike": {
            "AWS:SourceArn": "<your user pool ARN>"
        }
    }
}
]
}

```

Lambda triggers for federated users

You can use the following Lambda triggers to customize your user pool workflows for users who sign in with a federated provider.

Note

- Federated users must use the Amazon Cognito hosted UI to sign in.
- For some actions, federated users invoke more than one Lambda function.

Federated user trigger sources

Trigger	triggerSource value	Triggering event
Pre sign-up	PreSignUp_ExternalProvider	A federated user signs in from the Amazon Cognito hosted UI sign-in page for the first time.
Post confirmation	PostConfirmation_ConfirmSignIn	A federated user signs in from the Amazon Cognito hosted UI sign-in page for the first time.
Pre authentication	PreAuthentication_Authentication	A federated user signs in from the Amazon Cognito hosted UI sign-in page, except at first sign-in.
Post authentication	PostAuthentication_Authentication	A federated user successfully authenticates from the Amazon Cognito hosted UI sign-in page, except at first sign-in.
Pre token generation	TokenGeneration_HostedAuth	A federated user successfully authenticates from the Amazon Cognito hosted UI sign-in page and is about to be issued tokens, except at first sign-in.

Federated sign-in does not invoke any [Custom authentication challenge Lambda triggers \(p. 114\)](#), [Migrate user Lambda trigger \(p. 127\)](#), [Custom message Lambda trigger \(p. 131\)](#), or [Custom sender Lambda triggers \(p. 136\)](#) in your user pool.

Adding a user pool Lambda trigger

Original console

To add a user pool Lambda trigger with the console

1. Use the [Lambda console](#) to create a Lambda function . For more information on Lambda functions, see the [AWS Lambda Developer Guide](#).
2. Go to the [Amazon Cognito console](#). Then choose **Manage User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. In your user pool, choose the **Triggers** tab from the navigation bar.
5. Choose a Lambda trigger, such as **Pre sign-up** or **Pre authentication**. Then choose your Lambda function from the **Lambda function** drop-down list.
6. Choose **Save changes**.
7. You can use Amazon CloudWatch in the Lambda console to log your Lambda function. For more information, see [Accessing Amazon CloudWatch Logs for Lambda](#).

New console

To add a user pool Lambda trigger with the console

1. Use the [Lambda console](#) to create a Lambda function. For more information on Lambda functions, see the [AWS Lambda Developer Guide](#).
2. Go to the [Amazon Cognito console](#), and then choose **User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. Choose the **User pool properties** tab and locate **Lambda triggers**.
5. Choose **Add a Lambda trigger**.
6. Select a Lambda trigger **Category** based on the stage of authentication that you want to customize.
7. Select **Assign Lambda function** and select a function in the same AWS Region as your user pool.

Note

If your AWS Identity and Access Management (IAM) credentials have permission to update the Lambda function, Amazon Cognito adds a Lambda resource-based policy. With this policy, Amazon Cognito can invoke the function that you select. If the signed-in credentials do not have sufficient IAM permissions, you must update the resource-based policy separately. For more information, see [the section called "Important considerations" \(p. 93\)](#).

8. Choose **Save changes**.
9. You can use CloudWatch in the Lambda console to log your Lambda function . For more information, see [Accessing CloudWatch Logs for Lambda](#).

User pool Lambda trigger event

Amazon Cognito passes event information to your Lambda function. The Lambda function returns the same event object back to Amazon Cognito with any changes in the response. This event shows the Lambda trigger common parameters:

JSON

```
{
```

```
"version": "string",
"triggerSource": "string",
"region": AWSRegion,
"userPoolId": "string",
"userName": "string",
"callerContext":
{
    "awsSdkVersion": "string",
    "clientId": "string"
},
"request":
{
    "userAttributes": {
        "string": "string",
        ...
    }
},
"response": {}
```

User pool Lambda trigger common parameters

version

The version number of your Lambda function.

triggerSource

The name of the event that triggered the Lambda function. For a description of each triggerSource see [User pool Lambda trigger sources \(p. 97\)](#).

region

The AWS Region as an AWSRegion instance.

userPoolId

The user pool ID for the user pool.

userName

The user name of the current user.

callerContext

The caller context. This context consists of the following:

awsSdkVersion

The AWS SDK version number.

clientId

The user pool app client ID.

request

The request from the Amazon Cognito service. This request must include the following:

userAttributes

One or more pairs of user attribute names and values. Each pair takes the form "*name*": "*value*".

response

The response from your Lambda trigger. The return parameters in the response depend on the triggering event.

User pool Lambda trigger sources

This section describes each Amazon Cognito Lambda triggerSource parameter and its triggering event.

Sign-up, confirmation, and sign-in (authentication) triggers

Trigger	triggerSource value	Triggering event
Pre sign-up	PreSignUp_SignUp	Pre sign-up.
Pre sign-up	PreSignUp_AdminCreateUser	Pre sign-up when an admin creates a new user.
Pre sign-up	PreSignUp_ExternalProvider	Pre sign-up for external identity providers.
Post confirmation	PostConfirmation_ConfirmSignUp	Post sign-up confirmation.
Post confirmation	PostConfirmation_ConfirmForgotPassword	PostForgotPassword confirmation.
Pre authentication	PreAuthentication_Authenticate	Pre authentication.
Post authentication	PostAuthentication_Authenticate	Post authentication.

Custom authentication challenge triggers

Trigger	triggerSource value	Triggering event
Define auth challenge	DefineAuthChallenge_Authentication	DefineAuth Challenge.
Create auth challenge	CreateAuthChallenge_Authentication	CreateAuth Challenge.
Verify auth challenge	VerifyAuthChallengeResponse	VerifyAuthChallenge Response.

Pre token generation triggers

Trigger	triggerSource value	Triggering event
Pre token generation	TokenGeneration_HostedAuth	Amazon Cognito authenticates the user from your hosted UI sign-in page.
Pre token generation	TokenGeneration_Authentication	User authentication flows complete.
Pre token generation	TokenGeneration_NewPassword	Challenges the user. Amazon Cognito invokes this when the user must change a temporary password.
Pre token generation	TokenGeneration_AuthenticationDevice	End of the authentication of a user device.
Pre token generation	TokenGeneration_RefreshToken	User tries to refresh the identity and access tokens.

Migrate user triggers

Trigger	triggerSource value	Triggering event
User migration	UserMigration_Authentication	User migration at the time of sign-in.
User migration	UserMigration_ForgotPassword	User migration during the forgot-password flow.

Custom message triggers

Trigger	triggerSource value	Triggering event
Custom message	CustomMessage_SignUp	Custom message when a user signs up in your user pool.
Custom message	CustomMessage_AdminCreateUser	Custom message when you create a user as an administrator and Amazon Cognito sends them a temporary password.
Custom message	CustomMessage_ResendCode	Custom message when your existing user requests a new confirmation code.
Custom message	CustomMessage_ForgotPassword	Custom message when your user requests a password reset.
Custom message	CustomMessage_UpdateUserAttribute	Custom message when a user changes their email address or phone number and Amazon Cognito sends a verification code.
Custom message	CustomMessage_VerifyUserAttribute	Custom message when a user adds an email address or phone number and Amazon Cognito sends a verification code.
Custom message	CustomMessage_Authentication	Custom message when a user who has configured SMS MFA signs in.

Pre sign-up Lambda trigger

Shortly before Amazon Cognito signs up a new user, it activates the pre sign-up AWS Lambda function. As part of the sign-up process, you can use this function to perform custom validation and, based on the results of your validation, accept or deny the registration request.

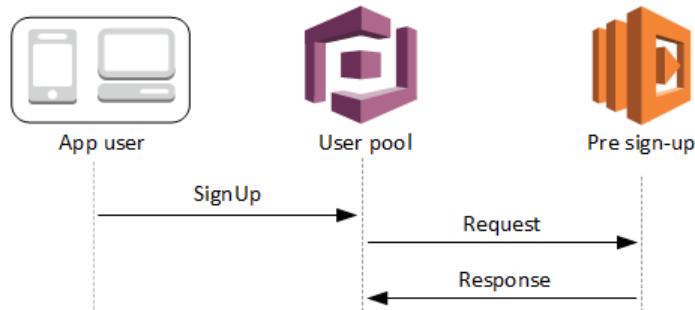
Topics

- [Pre sign-up Lambda flows \(p. 99\)](#)
- [Pre sign-up Lambda trigger parameters \(p. 99\)](#)
- [Sign-up tutorials \(p. 101\)](#)
- [Pre sign-up example: Auto-confirm users from a registered domain \(p. 101\)](#)

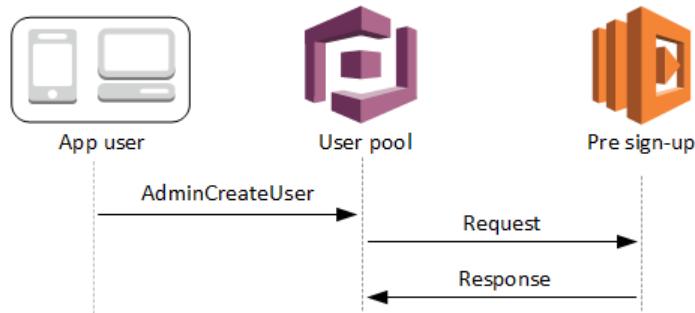
- Pre sign-up example: Auto-confirm and auto-verify all users (p. 103)
- Pre sign-up example: Deny sign-up if user name has fewer than five characters (p. 104)

Pre sign-up Lambda flows

Client sign-up flow



Server sign-up flow



The request includes validation data from the client. This data comes from the `ValidationData` values passed to the user pool `SignUp` and `AdminCreateUser` API methods.

Pre sign-up Lambda trigger parameters

These are the parameters that Amazon Cognito passes to this Lambda function along with the event information in the [common parameters](#).

JSON

```
{  
    "request": {  
        "userAttributes": {  
            "string": "string",  
            ...  
        },  
        "validationData": {  
            "string": "string",  
            ...  
        },  
        "clientMetadata": {  
            "string": "string",  
            ...  
        }  
    },  
}
```

```
    "response": {
        "autoConfirmUser": "boolean",
        "autoVerifyPhone": "boolean",
        "autoVerifyEmail": "boolean"
    }
}
```

Pre sign-up request parameters

userAttributes

One or more name-value pairs representing user attributes. The attribute names are the keys.

validationData

One or more name-value pairs containing the validation data in the request to register a user. The validation data is set and then passed from the client in the request to register a user. You can pass this data to your Lambda function by using the `ClientMetadata` parameter in the [InitiateAuth](#) and [AdminInitiateAuth](#) API actions.

clientMetadata

One or more key-value pairs that you can provide as custom input to the Lambda function that you specify for the pre sign-up trigger. You can pass this data to your Lambda function by using the `ClientMetadata` parameter in the following API actions: [AdminCreateUser](#), [AdminRespondToAuthChallenge](#), [ForgotPassword](#), and [SignUp](#).

Pre sign-up response parameters

In the response, you can set `autoConfirmUser` to `true` if you want to auto-confirm the user. You can set `autoVerifyEmail` to `true` to auto-verify the user's email. You can set `autoVerifyPhone` to `true` to auto-verify the user's phone number.

Note

Response parameters `autoVerifyPhone`, `autoVerifyEmail` and `autoConfirmUser` are ignored by Amazon Cognito when the Pre Sign-up lambda is triggered by the [AdminCreateUser](#) API.

autoConfirmUser

Set to `true` to auto-confirm the user, or `false` otherwise.

autoVerifyEmail

Set to `true` to set as verified the email address of a user who is signing up, or `false` otherwise. If `autoVerifyEmail` is set to `true`, the `email` attribute must have a valid, non-null value. Otherwise an error will occur and the user will not be able to complete sign-up.

If the `email` attribute is selected as an alias, an alias will be created for the user's email address when `autoVerifyEmail` is set. If an alias with that email address already exists, the alias will be moved to the new user and the previous user's email address will be marked as unverified. For more information, see [Aliases \(p. 209\)](#).

autoVerifyPhone

Set to `true` to set as verified the phone number of a user who is signing up, or `false` otherwise. If `autoVerifyPhone` is set to `true`, the `phone_number` attribute must have a valid, non-null value. Otherwise an error will occur and the user will not be able to complete sign-up.

If the `phone_number` attribute is selected as an alias, an alias will be created for the user's phone number when `autoVerifyPhone` is set. If an alias with that phone number already exists, the alias

will be moved to the new user and the previous user's phone number will be marked as unverified. For more information, see [Aliases \(p. 209\)](#).

Sign-up tutorials

The pre sign-up Lambda function is triggered before user sign-up. See these Amazon Cognito sign-up tutorials for JavaScript, Android, and iOS.

Platform	Tutorial
JavaScript Identity SDK	Sign up users with JavaScript
Android Identity SDK	Sign up users with Android
iOS Identity SDK	Sign up users with iOS

Pre sign-up example: Auto-confirm users from a registered domain

You can use the pre sign-up Lambda trigger to add custom logic that validates new users who sign up for your user pool. This is a sample JavaScript program that shows how to sign up a new user. It invokes a pre sign-up Lambda trigger as part of the authentication.

JavaScript

```
var attributeList = [];
var dataEmail = {
    Name : 'email',
    Value : '....' // your email here
};
var dataPhoneNumber = {
    Name : 'phone_number',
    Value : '....' // your phone number here with +country code and no delimiters in front
};

var dataEmailDomain = {
    Name: "custom:domain",
    Value: "example.com"
}
var attributeEmail =
new AmazonCognitoIdentity.CognitoUserAttribute(dataEmail);
var attributePhoneNumber =
new AmazonCognitoIdentity.CognitoUserAttribute(dataPhoneNumber);
var attributeEmailDomain =
new AmazonCognitoIdentity.CognitoUserAttribute(dataEmailDomain);

attributeList.push(attributeEmail);
attributeList.push(attributePhoneNumber);
attributeList.push(attributeEmailDomain);

var cognitoUser;
userPool.signUp('username', 'password', attributeList, null, function(err, result){
    if (err) {
        alert(err);
        return;
    }
    cognitoUser = result.user;
```

```
    console.log('user name is ' + cognitoUser.getUsername());
});
```

This is a sample Lambda trigger called just before sign-up with the user pool pre sign-up Lambda trigger. It uses a custom attribute **custom:domain** to automatically confirm new users from a particular email domain. Any new users not in the custom domain will be added to the user pool, but not automatically confirmed.

Node.js

```
exports.handler = (event, context, callback) => {
    // Set the user pool autoConfirmUser flag after validating the email domain
    event.response.autoConfirmUser = false;

    // Split the email address so we can compare domains
    var address = event.request.userAttributes.email.split("@");

    // This example uses a custom attribute "custom:domain"
    if (event.request.userAttributes.hasOwnProperty("custom:domain")) {
        if (event.request.userAttributes['custom:domain'] === address[1]) {
            event.response.autoConfirmUser = true;
        }
    }

    // Return to Amazon Cognito
    callback(null, event);
};
```

Python

```
def lambda_handler(event, context):
    # It sets the user pool autoConfirmUser flag after validating the email domain
    event['response']['autoConfirmUser'] = False

    # Split the email address so we can compare domains
    address = event['request']['userAttributes']['email'].split('@')

    # This example uses a custom attribute 'custom:domain'
    if 'custom:domain' in event['request']['userAttributes']:
        if event['request']['userAttributes']['custom:domain'] == address[1]:
            event['response']['autoConfirmUser'] = True

    # Return to Amazon Cognito
    return event
```

Amazon Cognito passes event information to your Lambda function. The function then returns the same event object to Amazon Cognito, with any changes in the response. In the Lambda console, you can set up a test event with data that is relevant to your Lambda trigger. The following is a test event for this code sample:

JSON

```
{
    "request": {
        "userAttributes": {
            "email": "testuser@example.com",
            "custom:domain": "example.com"
        }
    },
}
```

```
    "response": {}  
}
```

Pre sign-up example: Auto-confirm and auto-verify all users

This example confirms all users and sets the user's `email` and `phone_number` attributes to verified if the attribute is present. Also, if aliasing is enabled, aliases will be created for `phone_number` and `email` when auto-verify is set.

Note

If an alias with the same phone number already exists, the alias will be moved to the new user, and the previous user's `phone_number` will be marked as unverified. The same is true for email addresses. To prevent this from happening, you can use the user pools [ListUsers API](#) to see if there is an existing user who is already using the new user's phone number or email address as an alias.

Node.js

```
exports.handler = (event, context, callback) => {  
  
    // Confirm the user  
    event.response.autoConfirmUser = true;  
  
    // Set the email as verified if it is in the request  
    if (event.request.userAttributes.hasOwnProperty("email")) {  
        event.response.autoVerifyEmail = true;  
    }  
  
    // Set the phone number as verified if it is in the request  
    if (event.request.userAttributes.hasOwnProperty("phone_number")) {  
        event.response.autoVerifyPhone = true;  
    }  
  
    // Return to Amazon Cognito  
    callback(null, event);  
};
```

Python

```
def lambda_handler(event, context):  
    # Confirm the user  
    event['response']['autoConfirmUser'] = True  
  
    # Set the email as verified if it is in the request  
    if 'email' in event['request']['userAttributes']:  
        event['response']['autoVerifyEmail'] = True  
  
    # Set the phone number as verified if it is in the request  
    if 'phone_number' in event['request']['userAttributes']:  
        event['response']['autoVerifyPhone'] = True  
  
    # Return to Amazon Cognito  
    return event
```

Amazon Cognito passes event information to your Lambda function. The function then returns the same event object to Amazon Cognito, with any changes in the response. In the Lambda console, you can set up a test event with data that is relevant to your Lambda trigger. The following is a test event for this code sample:

JSON

```
{  
    "request": {  
        "userAttributes": {  
            "email": "user@example.com",  
            "phone_number": "+12065550100"  
        }  
    },  
    "response": {}  
}
```

Pre sign-up example: Deny sign-up if user name has fewer than five characters

This example checks the length of the user name in a sign-up request. The example returns an error if the user has entered a name less than five characters long.

Node.js

```
exports.handler = (event, context, callback) => {  
    // Impose a condition that the minimum length of the username is 5 is imposed on  
    // all user pools.  
    if (event.userName.length < 5) {  
        var error = new Error("Cannot register users with username less than the  
        minimum length of 5");  
        // Return error to Amazon Cognito  
        callback(error, event);  
    }  
    // Return to Amazon Cognito  
    callback(null, event);  
};
```

Python

```
def lambda_handler(event, context):  
    if len(event['userName']) < 5:  
        raise Exception("Cannot register users with username less than the minimum  
        length of 5")  
    # Return to Amazon Cognito  
    return event
```

Amazon Cognito passes event information to your Lambda function. The function then returns the same event object to Amazon Cognito, with any changes in the response. In the Lambda console, you can set up a test event with data that is relevant to your Lambda trigger. The following is a test event for this code sample:

JSON

```
{  
    "userName": "rroe",  
    "response": {}  
}
```

Post confirmation Lambda trigger

Amazon Cognito invokes this trigger after a new user is confirmed, allowing you to send custom messages or to add custom logic. For example, you could use this trigger to gather new user data.

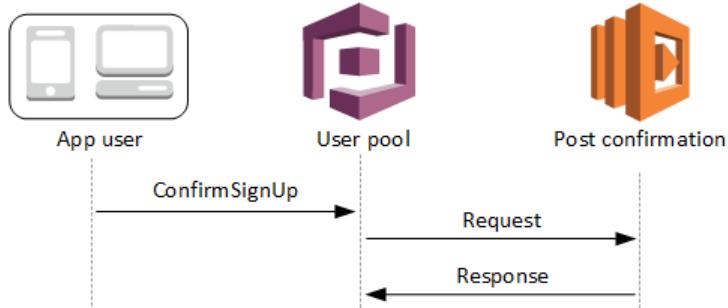
The request contains the current attributes for the confirmed user.

Topics

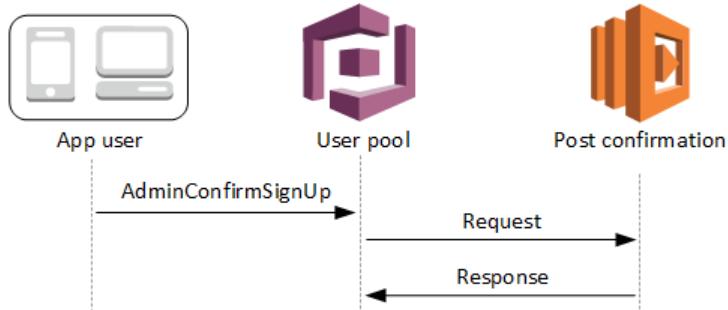
- [Post confirmation Lambda flows \(p. 105\)](#)
- [Post confirmation Lambda trigger parameters \(p. 106\)](#)
- [User confirmation tutorials \(p. 106\)](#)
- [Post confirmation example \(p. 106\)](#)

Post confirmation Lambda flows

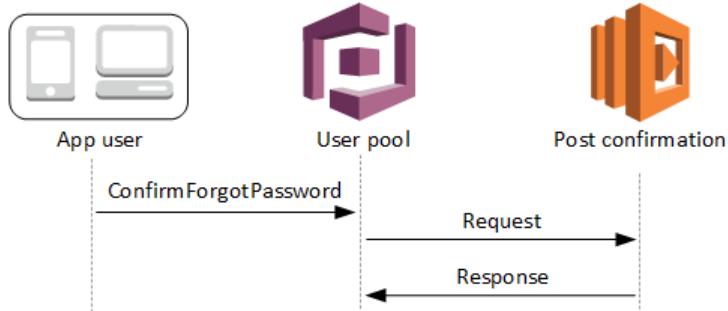
Client confirm sign-up flow



Server confirm sign-up flow



Confirm forgot password flow



Post confirmation Lambda trigger parameters

These are the parameters that Amazon Cognito passes to this Lambda function along with the event information in the [common parameters](#).

JSON

```
{  
    "request": {  
        "userAttributes": {  
            "string": "string",  
            . . .  
        },  
        "clientMetadata": {  
            "string": "string",  
            . . .  
        }  
    },  
    "response": {}  
}
```

Post confirmation request parameters

userAttributes

One or more key-value pairs representing user attributes.

clientMetadata

One or more key-value pairs that you can provide as custom input to the Lambda function that you specify for the post confirmation trigger. You can pass this data to your Lambda function by using the ClientMetadata parameter in the following API actions: [AdminConfirmSignUp](#), [ConfirmForgotPassword](#), [ConfirmSignUp](#), and [SignUp](#).

Post confirmation response parameters

No additional return information is expected in the response.

User confirmation tutorials

The post confirmation Lambda function is triggered just after Amazon Cognito confirms a new user. See these user confirmation tutorials for JavaScript, Android, and iOS.

Platform	Tutorial
JavaScript Identity SDK	Confirm users with JavaScript
Android Identity SDK	Confirm users with Android
iOS Identity SDK	Confirm users with iOS

Post confirmation example

This example Lambda function sends a confirmation email message to your user using Amazon SES. For more information see [Amazon Simple Email Service Developer Guide](#).

Node.js

```

var aws = require('aws-sdk');

var ses = new aws.SES();

exports.handler = (event, context, callback) => {
    console.log(event);

    if (event.request.userAttributes.email) {
        sendEmail(event.request.userAttributes.email, "Congratulations " +
event.userName + ", you have been confirmed: ", function(status) {

            // Return to Amazon Cognito
            callback(null, event);
        });
    } else {
        // Nothing to do, the user's email ID is unknown
        callback(null, event);
    }
};

function sendEmail(to, body, completedCallback) {
    var eParams = {
        Destination: {
            ToAddresses: [to]
        },
        Message: {
            Body: {
                Text: {
                    Data: body
                }
            },
            Subject: {
                Data: "Cognito Identity Provider registration completed"
            }
        },
        // Replace source_email with your SES validated email address
        Source: "<source_email>"
    };

    var email = ses.sendEmail(eParams, function(err, data){
        if (err) {
            console.log(err);
        } else {
            console.log("====EMAIL SENT====");
        }
        completedCallback('Email sent');
    });
    console.log("EMAIL CODE END");
};

```

Amazon Cognito passes event information to your Lambda function. The function then returns the same event object to Amazon Cognito, with any changes in the response. In the Lambda console, you can set up a test event with data that is relevant to your Lambda trigger. The following is a test event for this code sample:

JSON

```
{
}
```

```

    "request": {
        "userAttributes": {
            "email": "user@example.com",
            "email_verified": true
        }
    },
    "response": {}
}

```

Pre authentication Lambda trigger

Amazon Cognito invokes this trigger when a user attempts to sign in so that you can create custom validation that accepts or denies the authentication request.

Note

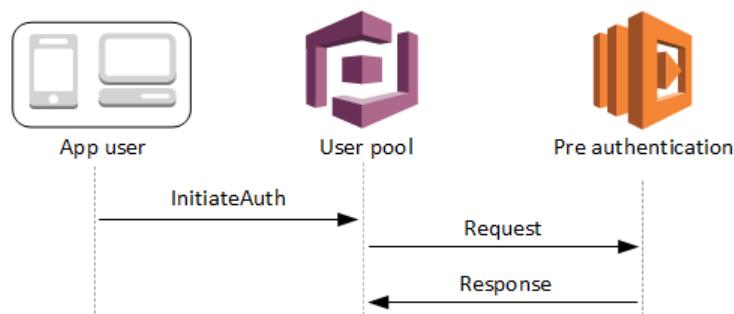
Triggers depend on the user existing in the user pool before Amazon Cognito activates the trigger.

Topics

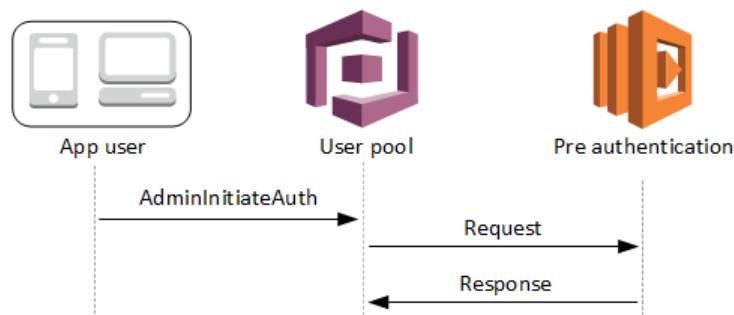
- [Pre authentication Lambda flows \(p. 108\)](#)
- [Pre authentication Lambda trigger parameters \(p. 109\)](#)
- [Authentication tutorials \(p. 109\)](#)
- [Pre authentication example \(p. 110\)](#)

Pre authentication Lambda flows

Client authentication flow



Server authentication flow



The request includes client validation data from the `ClientMetadata` values that your app passes to the user pool `InitiateAuth` and `AdminInitiateAuth` API operations.

For more information, see [User pool authentication flow \(p. 364\)](#).

Pre authentication Lambda trigger parameters

These are the parameters that Amazon Cognito passes to this Lambda function along with the event information in the [common parameters](#).

JSON

```
{  
    "request": {  
        "userAttributes": {  
            "string": "string",  
            ...  
        },  
        "validationData": {  
            "string": "string",  
            ...  
        },  
        "userNotFound": boolean  
    },  
    "response": {}  
}
```

Pre authentication request parameters

userAttributes

One or more name-value pairs that represent user attributes.

userNotFound

When you set `PreventUserExistenceErrors` to `ENABLED` for your user pool client, Amazon Cognito populates this Boolean.

validationData

One or more key-value pairs that contain the validation data in the user's sign-in request. To pass this data to your Lambda function, use the `ClientMetadata` parameter in the [InitiateAuth](#) and [AdminInitiateAuth](#) API actions.

Pre authentication response parameters

Amazon Cognito does not expect any additional return information in the response. Your function can return an error to reject the sign-in attempt, or use API operations to query and modify your resources.

Authentication tutorials

Amazon Cognito activates the pre-authentication Lambda function before Amazon Cognito signs in a new user. See these sign-in tutorials for JavaScript, Android, and iOS.

Platform	Tutorial
JavaScript Identity SDK	Sign in users with JavaScript
Android Identity SDK	Sign in users with Android

Platform	Tutorial
iOS Identity SDK	Sign in users with iOS

Pre authentication example

This sample function prevents users from a specific user pool app client from signing in to the user pool.

Node.js

```
exports.handler = (event, context, callback) => {
    if (event.callerContext.clientId === "user-pool-app-client-id-to-be-blocked") {
        var error = new Error("Cannot authenticate users from this user pool app
client");

        // Return error to Amazon Cognito
        callback(error, event);
    }

    // Return to Amazon Cognito
    callback(null, event);
};
```

Python

```
def lambda_handler(event, context):
    if event['callerContext']['clientId'] == "<user pool app client id to be blocked>":
        raise Exception("Cannot authenticate users from this user pool app client")

    # Return to Amazon Cognito
    return event
```

Amazon Cognito passes event information to your Lambda function. The function then returns the same event object to Amazon Cognito, with any changes in the response. In the Lambda console, you can set up a test event with data that is relevant to your Lambda trigger. The following is a test event for this code sample:

JSON

```
{  
    "callerContext": {  
        "clientId": "<user pool app client id to be blocked>"  
    },  
    "response": {}  
}
```

Post authentication Lambda trigger

Because Amazon Cognito invokes this trigger after signing in a user, you can add custom logic after Amazon Cognito authenticates the user.

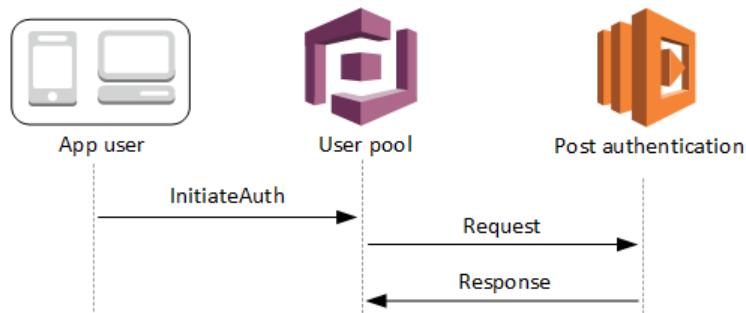
Topics

- [Post authentication Lambda flows \(p. 111\)](#)
- [Post authentication Lambda trigger parameters \(p. 111\)](#)

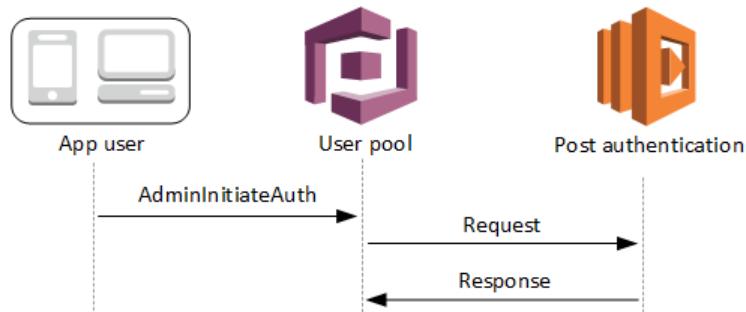
- [Authentication tutorials \(p. 112\)](#)
- [Post authentication example \(p. 112\)](#)

Post authentication Lambda flows

Client authentication flow



Server authentication flow



For more information, see [User pool authentication flow \(p. 364\)](#).

Post authentication Lambda trigger parameters

These are the parameters that Amazon Cognito passes to this Lambda function along with the event information in the [common parameters](#).

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            ...
        },
        "newDeviceUsed": boolean,
        "clientMetadata": {
            "string": "string",
            ...
        }
    },
    "response": {}
}
```

Post authentication request parameters

newDeviceUsed

This flag indicates if the user has signed in on a new device. Amazon Cognito only sets this flag if the remembered devices value of the user pool is `Always` or `User Opt-In`.

userAttributes

One or more name-value pairs representing user attributes.

clientMetadata

One or more key-value pairs that you can provide as custom input to the Lambda function that you specify for the post authentication trigger. To pass this data to your Lambda function, you can use the `ClientMetadata` parameter in the [AdminRespondToAuthChallenge](#) and [RespondToAuthChallenge](#) API actions. Amazon Cognito doesn't include data from the `ClientMetadata` parameter in [AdminInitiateAuth](#) and [InitiateAuth](#) API operations in the request that it passes to the post authentication function.

Post authentication response parameters

Amazon Cognito doesn't expect any additional return information in the response. Your function can use API operations to query and modify your resources, or record event metadata to an external system.

Authentication tutorials

Immediately after Amazon Cognito signs in a user, it activates the post authentication Lambda function. See these sign-in tutorials for JavaScript, Android, and iOS.

Platform	Tutorial
JavaScript Identity SDK	Sign in users with JavaScript
Android Identity SDK	Sign in users with Android
iOS Identity SDK	Sign in users with iOS

Post authentication example

This post authentication sample Lambda function sends data from a successful sign-in to CloudWatch Logs.

Node.js

```
exports.handler = (event, context, callback) => {

    // Send post authentication data to Cloudwatch logs
    console.log ("Authentication successful");
    console.log ("Trigger function =", event.triggerSource);
    console.log ("User pool =", event.userPoolId);
    console.log ("App client ID =", event.callerContext.clientId);
    console.log ("User ID =", event.userName);

    // Return to Amazon Cognito
    callback(null, event);
};
```

Python

```
from __future__ import print_function
def lambda_handler(event, context):

    # Send post authentication data to Cloudwatch logs
    print ("Authentication successful")
    print ("Trigger function =", event['triggerSource'])
    print ("User pool = ", event['userPoolId'])
    print ("App client ID = ", event['callerContext']['clientId'])
    print ("User ID = ", event['userName'])

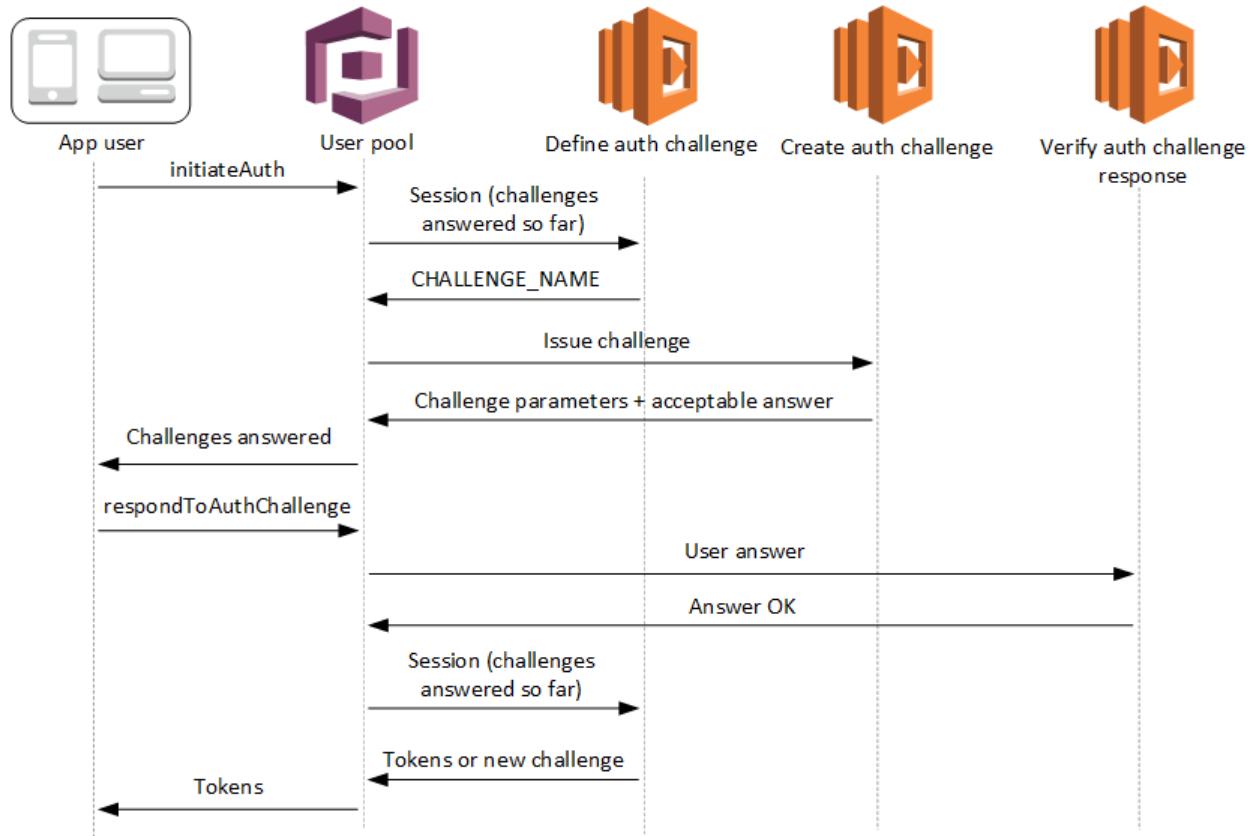
    # Return to Amazon Cognito
    return event
```

Amazon Cognito passes event information to your Lambda function. The function then returns the same event object to Amazon Cognito, with any changes in the response. In the Lambda console, you can set up a test event with data that is relevant to your Lambda trigger. The following is a test event for this code sample:

JSON

```
{
  "triggerSource": "testTrigger",
  "userPoolId": "testPool",
  "userName": "testName",
  "callerContext": {
    "clientId": "12345"
  },
  "response": {}
}
```

Custom authentication challenge Lambda triggers



These Lambda triggers issue and verify their own challenges as part of a user pool [custom authentication flow](#).

Define auth challenge

Amazon Cognito invokes this trigger to initiate the custom authentication flow.

Create auth challenge

Amazon Cognito invokes this trigger after **Define Auth Challenge** to create a custom challenge.

Verify auth challenge response

Amazon Cognito invokes this trigger to verify if the response from the end user for a custom challenge is valid or not.

You can incorporate new challenge types with these challenge Lambda triggers. For example, these challenge types might include CAPTCHAs or dynamic challenge questions.

You can generalize authentication into two common steps with the user pool `InitiateAuth` and `RespondToAuthChallenge` API methods.

In this flow, a user authenticates by answering successive challenges until authentication either fails or the user is issued tokens. These two API calls can be repeated to include different challenges.

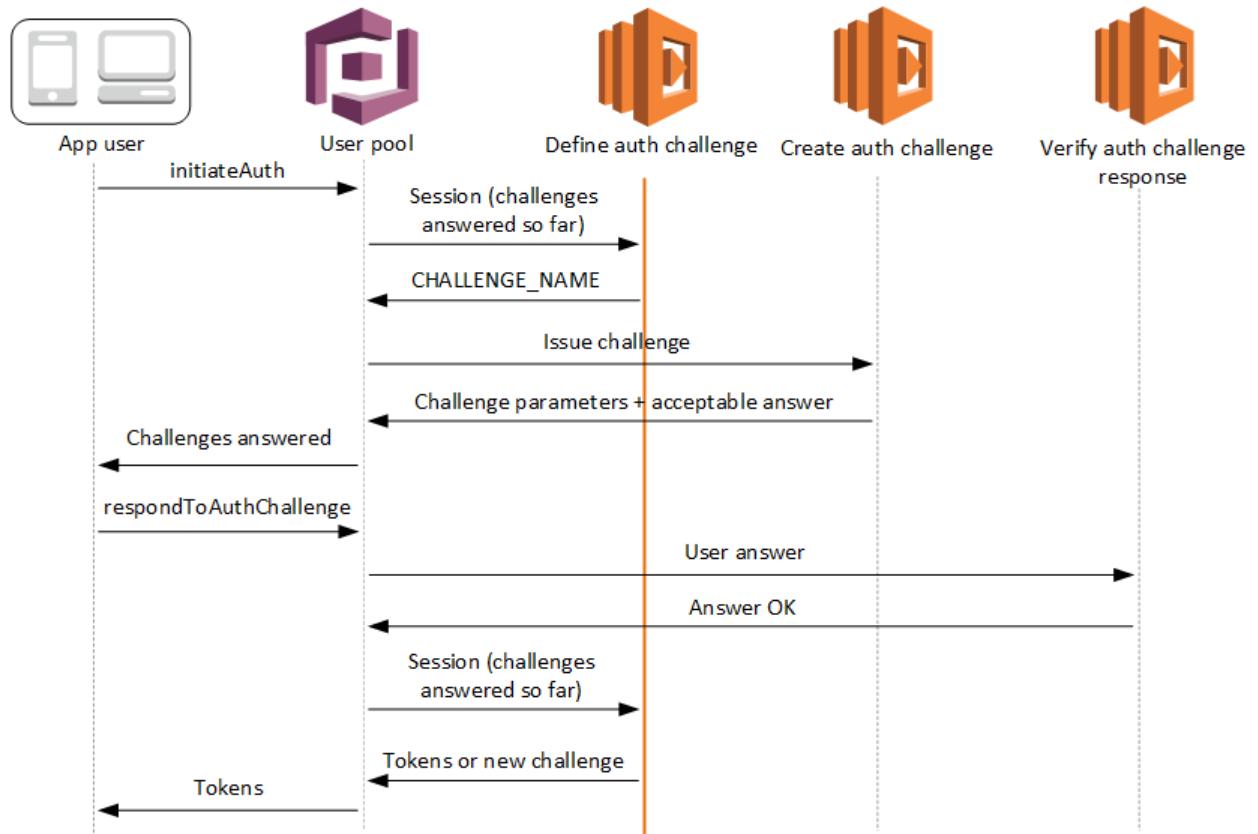
Note

The Amazon Cognito hosted UI does not support the custom authentication flow.

Topics

- [Define Auth challenge Lambda trigger \(p. 115\)](#)
- [Create Auth challenge Lambda trigger \(p. 118\)](#)
- [Verify Auth challenge response Lambda trigger \(p. 121\)](#)

Define Auth challenge Lambda trigger



Define auth challenge

Amazon Cognito invokes this trigger to initiate the [custom authentication flow](#).

The request for this Lambda trigger contains `session`. The `session` parameter is an array that contains all of the challenges that are presented to the user in the current authentication process. The request also includes the corresponding result. The `session` array stores challenge details (`ChallengeResult`) in chronological order. The challenge `session[0]` represents the first challenge that the user receives.

You can have Amazon Cognito verify user passwords before it issues your custom challenges. Here is an overview of the process:

1. Your app initiates sign-in by calling `InitiateAuth` or `AdminInitiateAuth` with the `AuthParameters` map. Parameters must include `CHALLENGE_NAME: SRP_A`, and values for `SRP_A` and `USERNAME`.
2. Amazon Cognito invokes your define auth challenge Lambda trigger with an initial session that contains `challengeName: SRP_A` and `challengeResult: true`.
3. After receiving those inputs, your Lambda function responds with `challengeName: PASSWORD_VERIFIER`, `issueTokens: false`, `failAuthentication: false`.

4. If the password verification succeeds, Amazon Cognito invokes your Lambda function again with a new session containing challengeName: PASSWORD_VERIFIER and challengeResult: true.
5. To initiate your custom challenges, your Lambda function responds with challengeName: CUSTOM_CHALLENGE, issueTokens: false, and failAuthentication: false. If you don't want to start your custom auth flow with password verification, you can initiate sign-in with the AuthParameters map including CHALLENGE_NAME: CUSTOM_CHALLENGE.
6. The challenge loop repeats until all challenges are answered.

Topics

- [Define Auth challenge Lambda trigger parameters \(p. 116\)](#)
- [Define Auth challenge example \(p. 117\)](#)

Define Auth challenge Lambda trigger parameters

These are the parameters that Amazon Cognito passes to this Lambda function along with the event information in the [common parameters](#).

JSON

```
{  
    "request": {  
        "userAttributes": {  
            "string": "string",  
            . . .  
        },  
        "session": [  
            ChallengeResult,  
            . . .  
        ],  
        "clientMetadata": {  
            "string": "string",  
            . . .  
        },  
        "userNotFound": boolean  
    },  
    "response": {  
        "challengeName": "string",  
        "issueTokens": boolean,  
        "failAuthentication": boolean  
    }  
}
```

Define Auth challenge request parameters

When Amazon Cognito invokes your Lambda function, Amazon Cognito provides the following parameters:

userAttributes

One or more name-value pairs that represent user attributes.

userNotFound

A Boolean that Amazon Cognito populates when PreventUserExistenceErrors is set to ENABLED for your user pool client. A value of true means that the user id (user name, email address, etc.) did not match any existing users. When PreventUserExistenceErrors is set to ENABLED, the service doesn't inform the app of nonexistent users. We recommend that your Lambda functions maintain the same user experience and account for latency. This way, the caller can't detect different behavior when the user exists or doesn't exist.

session

An array of `ChallengeResult` elements. Each contains the following elements:

challengeName

One of the following challenge types: `CUSTOM_CHALLENGE`, `SRP_A`, `PASSWORD_VERIFIER`, `SMS_MFA`, `DEVICE_SRP_AUTH`, `DEVICE_PASSWORD_VERIFIER`, or `ADMIN_NO_SRP_AUTH`.

Important

When your function determines if a user has successfully authenticated and you should issue them tokens, always check `challengeName` in your `DefineAuthChallenge` Lambda trigger to make sure it matches the expected value.

challengeResult

Set to `true` if the user successfully completed the challenge, or `false` otherwise.

challengeMetadata

Your name for the custom challenge. Used only if `challengeName` is `CUSTOM_CHALLENGE`.

clientMetadata

One or more key-value pairs that you can provide as custom input to the Lambda function that you specify for the define auth challenge trigger. To pass this data to your Lambda function, you can use the `ClientMetadata` parameter in the [AdminRespondToAuthChallenge](#) and [RespondToAuthChallenge](#) API operations. The request that invokes the define auth challenge function doesn't include data passed in the `ClientMetadata` parameter in [AdminInitiateAuth](#) and [InitiateAuth](#) API operations.

Define Auth challenge response parameters

In the response, you can return the next stage of the authentication process.

challengeName

A string that contains the name of the next challenge. If you want to present a new challenge to your user, specify the challenge name here.

issueTokens

If you determine that the user has completed the authentication challenges sufficiently, set to `true`. If the user has not met the challenges sufficiently, set to `false`.

failAuthentication

If you want to end the current authentication process, set to `true`. To continue the current authentication process, set to `false`.

Define Auth challenge example

This example defines a series of challenges for authentication and issues tokens only if the user has completed all of the challenges successfully.

Node.js

```
exports.handler = (event, context, callback) => {
    if (event.request.session.length == 1 && event.request.session[0].challengeName == 'SRP_A') {
        event.response.issueTokens = false;
        event.response.failAuthentication = false;
        event.response.challengeName = 'PASSWORD_VERIFIER';
```

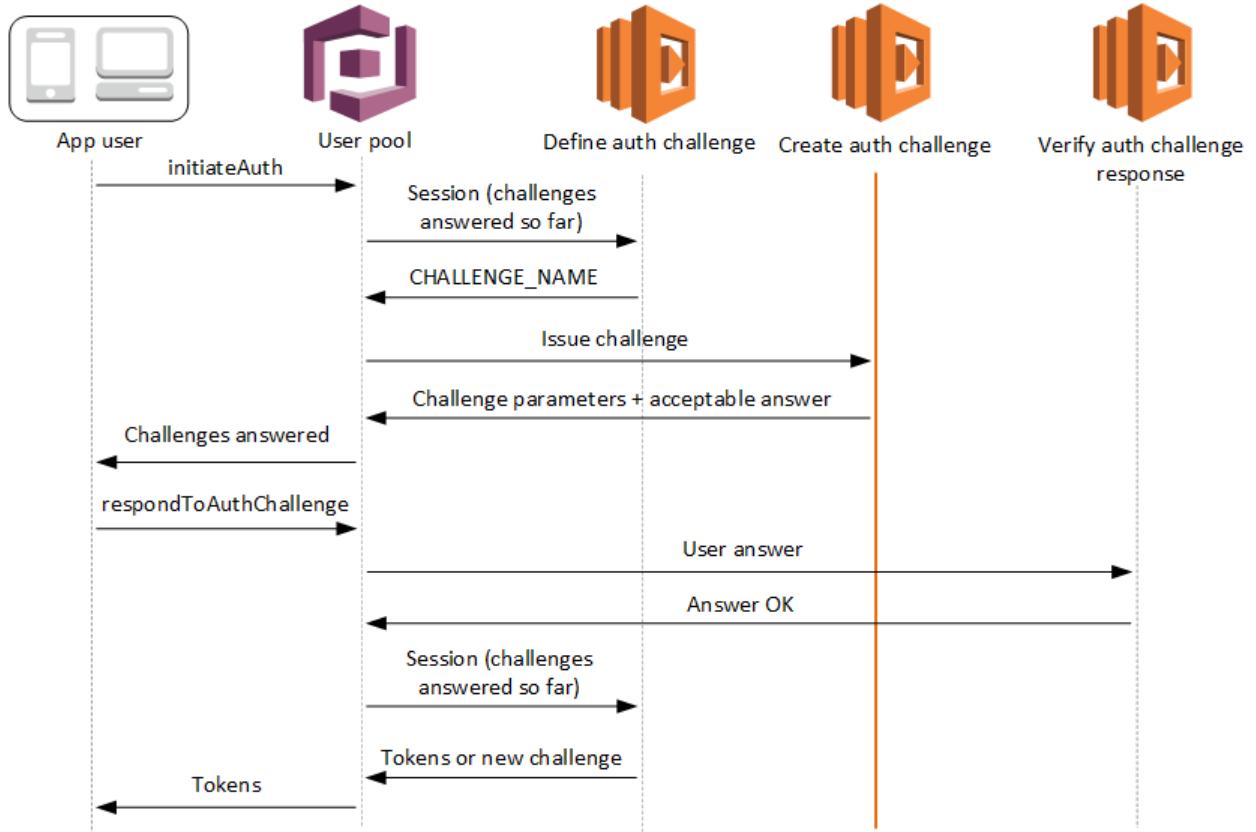
```

        } else if (event.request.session.length == 2 &&
event.request.session[1].challengeName == 'PASSWORD_VERIFIER' &&
event.request.session[1].challengeResult == true) {
    event.response.issueTokens = false;
    event.response.failAuthentication = false;
    event.response.challengeName = 'CUSTOM_CHALLENGE';
} else if (event.request.session.length == 3 &&
event.request.session[2].challengeName == 'CUSTOM_CHALLENGE' &&
event.request.session[2].challengeResult == true) {
    event.response.issueTokens = true;
    event.response.failAuthentication = false;
} else {
    event.response.issueTokens = false;
    event.response.failAuthentication = true;
}

// Return to Amazon Cognito
callback(null, event);
}

```

Create Auth challenge Lambda trigger



Create auth challenge

Amazon Cognito invokes this trigger after **Define Auth Challenge** if a custom challenge has been specified as part of the **Define Auth Challenge** trigger. It creates a [custom authentication flow](#).

This Lambda trigger is invoked to create a challenge to present to the user. The request for this Lambda trigger includes the `challengeName` and `session`. The `challengeName` is a string and is the name

of the next challenge to the user. The value of this attribute is set in the Define Auth Challenge Lambda trigger.

The challenge loop will repeat until all challenges are answered.

Topics

- [Create Auth challenge Lambda trigger parameters \(p. 119\)](#)
- [Create Auth challenge example \(p. 120\)](#)

Create Auth challenge Lambda trigger parameters

These are the parameters that Amazon Cognito passes to this Lambda function along with the event information in the [common parameters](#).

JSON

```
{  
    "request": {  
        "userAttributes": {  
            "string": "string",  
            ...  
        },  
        "challengeName": "string",  
        "session": [  
            ChallengeResult,  
            ...  
        ],  
        "clientMetadata": {  
            "string": "string",  
            ...  
        },  
        "userNotFound": boolean  
    },  
    "response": {  
        "publicChallengeParameters": {  
            "string": "string",  
            ...  
        },  
        "privateChallengeParameters": {  
            "string": "string",  
            ...  
        },  
        "challengeMetadata": "string"  
    }  
}
```

Create Auth challenge request parameters

userAttributes

One or more name-value pairs representing user attributes.

userNotFound

This boolean is populated when `PreventUserExistenceErrors` is set to `ENABLED` for your User Pool client.

challengeName

The name of the new challenge.

session

The session element is an array of ChallengeResult elements, each of which contains the following elements:

challengeName

The challenge type. One of: "CUSTOM_CHALLENGE", "PASSWORD_VERIFIER", "SMS_MFA", "DEVICE_SRP_AUTH", "DEVICE_PASSWORD_VERIFIER", or "ADMIN_NO_SRP_AUTH".

challengeResult

Set to `true` if the user successfully completed the challenge, or `false` otherwise.

challengeMetadata

Your name for the custom challenge. Used only if `challengeName` is "CUSTOM_CHALLENGE".

clientMetadata

One or more key-value pairs that you can provide as custom input to the Lambda function that you specify for the create auth challenge trigger. You can use the `ClientMetadata` parameter in the [AdminRespondToAuthChallenge](#) and [RespondToAuthChallenge](#) API actions to pass this data to your Lambda function. The request that invokes the create auth challenge function does not include data passed in the `ClientMetadata` parameter in [AdminInitiateAuth](#) and [InitiateAuth](#) API operations.

Create Auth challenge response parameters

publicChallengeParameters

One or more key-value pairs for the client app to use in the challenge to be presented to the user. This parameter should contain all of the necessary information to present the challenge to the user accurately.

privateChallengeParameters

This parameter is only used by the Verify Auth Challenge Response Lambda trigger. This parameter should contain all of the information that is required to validate the user's response to the challenge. In other words, the `publicChallengeParameters` parameter contains the question that is presented to the user and `privateChallengeParameters` contains the valid answers for the question.

challengeMetadata

Your name for the custom challenge, if this is a custom challenge.

Create Auth challenge example

A CAPTCHA is created as a challenge to the user. The URL for the CAPTCHA image is added to the public challenge parameters as "captchaUrl", and the expected answer is added to the private challenge parameters.

Node.js

```
exports.handler = (event, context, callback) => {
  if (event.request.challengeName == 'CUSTOM_CHALLENGE') {
    event.response.publicChallengeParameters = {};
    event.response.publicChallengeParameters.captchaUrl = 'url/123.jpg';
    event.response.privateChallengeParameters = {};
    event.response.privateChallengeParameters.answer = '5';
    event.response.challengeMetadata = 'CAPTCHA_CHALLENGE';
```

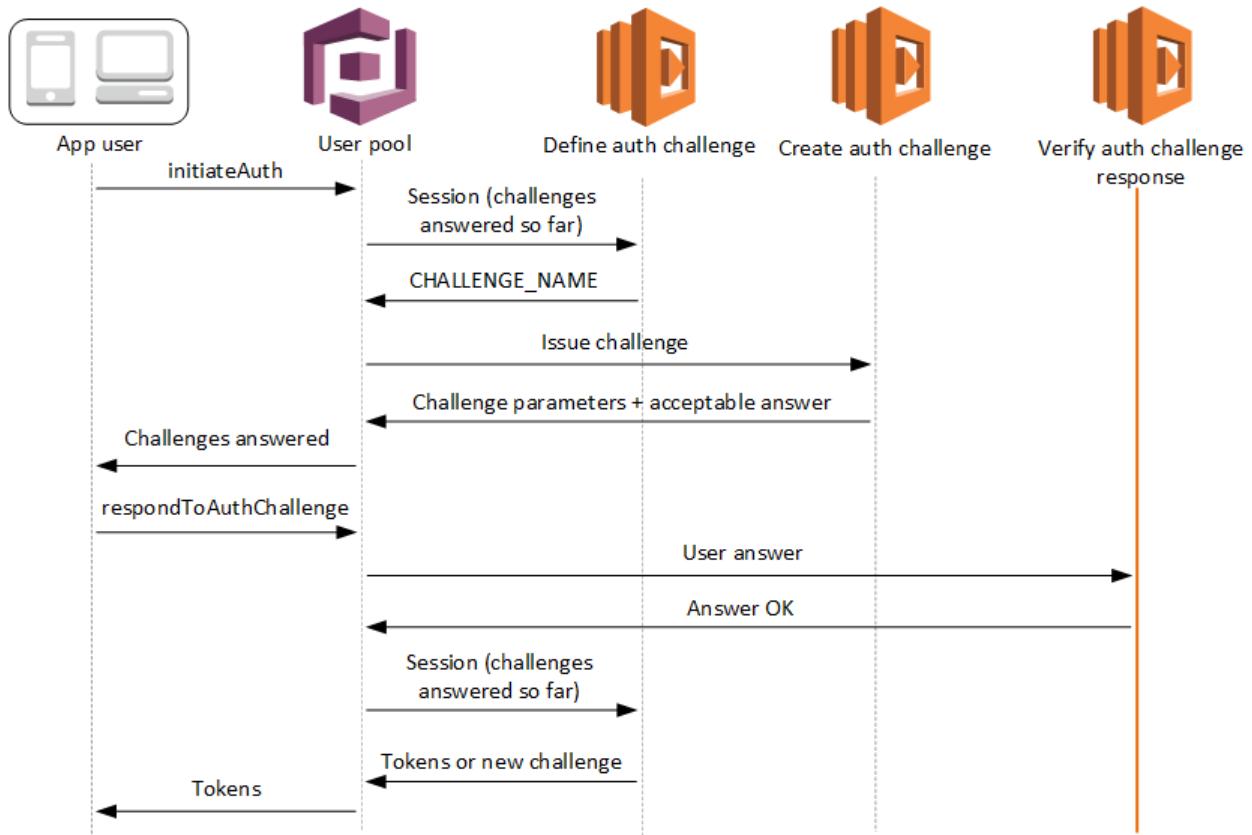
```

        }

        //Return to Amazon Cognito.
        callback(null, event);
    }
}

```

Verify Auth challenge response Lambda trigger



Verify auth challenge response

Amazon Cognito invokes this trigger to verify if the response from the user for a custom Auth Challenge is valid or not. It is part of a user pool [custom authentication flow](#).

The request for this trigger contains the `privateChallengeParameters` and `challengeAnswer` parameters. The Create Auth Challenge Lambda trigger returns `privateChallengeParameters` values, and contains the expected response from the user. The `challengeAnswer` parameter contains the user's response for the challenge.

The response contains the `answerCorrect` attribute. If the user successfully completes the challenge, Amazon Cognito sets the attribute value to `true`. If the user doesn't successfully complete the challenge, Amazon Cognito sets the value to `false`.

The challenge loop repeats until the users answers all challenges.

Topics

- [Verify Auth challenge Lambda trigger parameters \(p. 122\)](#)
- [Verify Auth challenge response example \(p. 123\)](#)

Verify Auth challenge Lambda trigger parameters

These are the parameters that Amazon Cognito passes to this Lambda function along with the event information in the [common parameters](#).

JSON

```
{  
    "request": {  
        "userAttributes": {  
            "string": "string",  
            ...  
        },  
        "privateChallengeParameters": {  
            "string": "string",  
            ...  
        },  
        "challengeAnswer": {  
            "string": "string",  
            ...  
        },  
        "clientMetadata": {  
            "string": "string",  
            ...  
        },  
        "userNotFound": boolean  
    },  
    "response": {  
        "answerCorrect": boolean  
    }  
}
```

Verify Auth challenge request parameters

userAttributes

This parameter contains one or more name-value pairs that represent user attributes.

userNotFound

When Amazon Cognito sets `PreventUserExistenceErrors` to `ENABLED` for your user pool client, Amazon Cognito populates this Boolean.

privateChallengeParameters

This parameter comes from the Create Auth Challenge trigger. To determine whether the user passed a challenge, Amazon Cognito compares the parameters against a user's `challengeAnswer`.

This parameter contains all of the information that is required to validate the user's response to the challenge. That information includes the question that Amazon Cognito presents to the user (`publicChallengeParameters`), and the valid answers for the question (`privateChallengeParameters`). Only the Verify Auth Challenge Response Lambda trigger uses this parameter.

challengeAnswer

This parameter value is the answer from the user's response to the challenge.

clientMetadata

This parameter contains one or more key-value pairs that you can provide as custom input to the Lambda function for the verify auth challenge trigger. To pass this data to your Lambda function, use the `ClientMetadata` parameter in the [AdminRespondToAuthChallenge](#) and

[RespondToAuthChallenge](#) API operations. Amazon Cognito doesn't include data from the ClientMetadata parameter in [AdminInitiateAuth](#) and [InitiateAuth](#) API operations in the request that it passes to the verify auth challenge function.

Verify Auth challenge response parameters

answerCorrect

If the user successfully completes the challenge, Amazon Cognito sets this parameter to `true`. If the user doesn't successfully complete the challenge, Amazon Cognito sets the parameter to `false`.

Verify Auth challenge response example

In this example, the Lambda function checks whether the user's response to a challenge matches the expected response. If the user's response matches the expected response, Amazon Cognito sets the `answerCorrect` parameter to `true`.

Node.js

```
exports.handler = (event, context, callback) => {
    if (event.request.privateChallengeParameters.answer ==
        event.request.challengeAnswer) {
        event.response.answerCorrect = true;
    } else {
        event.response.answerCorrect = false;
    }

    // Return to Amazon Cognito
    callback(null, event);
}
```

Pre token generation Lambda trigger

Because Amazon Cognito invokes this trigger before token generation, you can customize identity token claims.

You can use this AWS Lambda trigger to customize an identity token before Amazon Cognito generates it. You can use this trigger to add new claims, update claims, or suppress claims in the identity token. To use this feature, associate a Lambda function from the Amazon Cognito user pools console or update your user pool through the AWS Command Line Interface (AWS CLI).

You can't modify the following claims:

- acr
- amr
- aud
- at_hash
- auth_time
- azp
- cognito:username
- exp
- iat
- identities

- iss
- jti
- nbf
- nonce
- origin_jti
- sub
- token_use

Topics

- [Pre token generation Lambda trigger sources \(p. 124\)](#)
- [Pre token generation Lambda trigger parameters \(p. 124\)](#)
- [Pre token generation example: Add a new claim and suppress an existing claim \(p. 126\)](#)
- [Pre token generation example: Modify the user's group membership \(p. 127\)](#)

Pre token generation Lambda trigger sources

triggerSource value	Triggering event
TokenGeneration_HostedAuth	Called during authentication from the Amazon Cognito hosted UI sign-in page.
TokenGeneration_Authentication	Called after user authentication flows have completed.
TokenGeneration_NewPasswordChallenge	Called after the user is created by an admin. This flow is invoked when the user has to change a temporary password.
TokenGeneration_AuthenticateDevice	Called at the end of the authentication of a user device.
TokenGeneration_RefreshTokens	Called when a user tries to refresh the identity and access tokens.

Pre token generation Lambda trigger parameters

These are the parameters that Amazon Cognito passes to this Lambda function along with the event information in the [common parameters](#).

JSON

```
{
  "request": {
    "userAttributes": {"string": "string"},
    "groupConfiguration": [
      {
        "groupsToOverride": [
          "string",
          "string"
        ],
        "iamRolesToOverride": [
          "string",
          "string"
        ]
      }
    ]
  }
}
```

```

        ],
        "preferredRole": "string"
    }
],
"clientMetadata": {"string": "string"}
},
"response": {
    "claimsOverrideDetails": {
        "claimsToAddOrOverride": {"string": "string"},
        "claimsToSuppress": [
            "string",
            "string"
        ],
        "groupOverrideDetails": {
            "groupsToOverride": [
                "string",
                "string"
            ],
            "iamRolesToOverride": [
                "string",
                "string"
            ],
            "preferredRole": "string"
        }
    }
}
}

```

Pre token generation request parameters

groupConfiguration

The input object that contains the current group configuration. The object includes `groupsToOverride`, `iamRolesToOverride`, and `preferredRole`.

groupsToOverride

A list of the group names that correspond with the user who receives the identity token.

iamRolesToOverride

A list of the current AWS Identity and Access Management (IAM) roles that correspond with these groups.

preferredRole

A string that indicates the preferred IAM role.

clientMetadata

One or more key-value pairs that you can specify and provide as custom input to the Lambda function for the pre token generation trigger. To pass this data to your Lambda function, use the `ClientMetadata` parameter in the [AdminRespondToAuthChallenge](#) and [RespondToAuthChallenge](#) API operations. Amazon Cognito doesn't include data from the `ClientMetadata` parameter in [AdminInitiateAuth](#) and [InitiateAuth](#) API operations in the request that it passes to the pre token generation function.

Pre token generation response parameters

claimsToAddOrOverride

A map of one or more key-value pairs of claims to add or override. For group-related claims, use `groupOverrideDetails` instead.

claimsToSuppress

A list that contains claims that you want Amazon Cognito to suppress from the identity token.

Note

If your function both suppresses and replaces a claim value, then Amazon Cognito suppresses the claim.

groupOverrideDetails

The output object that contains the current group configuration. The object includes `groupsToOverride`, `iamRolesToOverride`, and `preferredRole`.

Your function replaces the `groupOverrideDetails` object with the object that you provide. If you provide an empty or null object in the response, then Amazon Cognito suppresses the groups. To keep the existing group configuration the same, copy the value of the `groupConfiguration` object of the request to the `groupOverrideDetails` object in the response. Then pass it back to the service.

Amazon Cognito ID and access tokens both contain the `cognito:groups` claim. Your `groupOverrideDetails` object replaces the `cognito:groups` claim in access tokens and ID tokens.

Pre token generation example: Add a new claim and suppress an existing claim

This example uses the Pre Token Generation Lambda function to add a new claim and suppresses an existing claim.

Node.js

```
exports.handler = (event, context, callback) => {
    event.response = {
        "claimsOverrideDetails": {
            "claimsToAddOrOverride": {
                "attribute_key2": "attribute_value2",
                "attribute_key": "attribute_value"
            },
            "claimsToSuppress": ["email"]
        }
    };

    // Return to Amazon Cognito
    callback(null, event);
};
```

Amazon Cognito passes event information to your Lambda function. The function then returns the same event object to Amazon Cognito, with any changes in the response. In the Lambda console, you can set up a test event with data that is relevant to your Lambda trigger. The following is a test event for this code sample: Because the code example doesn't process any request parameters, you can use a test event with an empty request. For more information about common request parameters, see [User pool Lambda trigger event \(p. 95\)](#).

JSON

```
{
    "request": {},
    "response": {}
}
```

Pre token generation example: Modify the user's group membership

This example uses the Pre Token Generation Lambda function to modify the user's group membership.

Node.js

```
exports.handler = (event, context, callback) => {
    event.response = {
        "claimsOverrideDetails": {
            "claimsToAddOrOverride": {
                "attribute_key2": "attribute_value2",
                "attribute_key": "attribute_value"
            },
            "claimsToSuppress": ["email"],
            "groupOverrideDetails": {
                "groupsToOverride": ["group-A", "group-B", "group-C"],
                "iamRolesToOverride": ["arn:aws:iam::XXXXXXXXXXXX:role/sns_callerA",
                    "arn:aws:iam::XXXXXXXXXXXX:role/sns_callerB", "arn:aws:iam::XXXXXXXXXXXX:role/
                    sns_callerC"],
                "preferredRole": "arn:aws:iam::XXXXXXXXXXXX:role/sns_caller"
            }
        }
    };
    // Return to Amazon Cognito
    callback(null, event);
};
```

Amazon Cognito passes event information to your Lambda function. The function then returns the same event object to Amazon Cognito, with any changes in the response. In the Lambda console, you can set up a test event with data that is relevant to your Lambda trigger. The following is a test event for this code sample:

JSON

```
{
    "request": {},
    "response": {}
}
```

Migrate user Lambda trigger

When a user doesn't exist in the user pool at sign-in with a password, or in the forgot-password flow, Amazon Cognito invokes this trigger. After the Lambda function returns successfully, Amazon Cognito creates the user in the user pool. For details on the authentication flow with the user migration Lambda trigger, see [Importing users into user pools with a user migration Lambda trigger \(p. 170\)](#).

To migrate users from your existing user directory into Amazon Cognito user pools at sign-in, or during the forgot-password flow, use this Lambda trigger.

Topics

- [Migrate user Lambda trigger sources \(p. 128\)](#)
- [Migrate user Lambda trigger parameters \(p. 128\)](#)
- [Example: Migrate a user with an existing password \(p. 130\)](#)

Migrate user Lambda trigger sources

triggerSource value	Triggering event
UserMigration_Authentication	User migration at sign-in.
UserMigration_ForgotPassword	User migration during forgot-password flow.

Migrate user Lambda trigger parameters

These are the parameters that Amazon Cognito passes to this Lambda function along with the event information in the [common parameters](#).

JSON

```
{  
    "userName": "string",  
    "request": {  
        "password": "string",  
        "validationData": {  
            "string": "string",  
            ...  
        },  
        "clientMetadata": {  
            "string": "string",  
            ...  
        }  
    },  
    "response": {  
        "userAttributes": {  
            "string": "string",  
            ...  
        },  
        "finalUserStatus": "string",  
        "messageAction": "string",  
        "desiredDeliveryMediums": [ "string", ... ],  
        "forceAliasCreation": boolean  
    }  
}
```

Migrate user request parameters

userName

The username that the user enters at sign-in.

password

The password that the user enters at sign-in. Amazon Cognito doesn't send this value in a request that's initiated by a forgot-password flow.

validationData

One or more key-value pairs that contain the validation data in the user's sign-in request. To pass this data to your Lambda function, you can use the ClientMetadata parameter in the [InitiateAuth](#) and [AdminInitiateAuth](#) API actions.

clientMetadata

One or more key-value pairs that you can provide as custom input to the Lambda function for the migrate user trigger. To pass this data to your Lambda function, you can use the `ClientMetadata` parameter in the [AdminRespondToAuthChallenge](#) and [ForgotPassword](#) API actions.

Migrate user response parameters

userAttributes

This field is required.

This field must contain one or more name-value pairs that Amazon Cognito stores in the user profile in your user pool and uses as user attributes. You can include both standard and custom user attributes. Custom attributes require the `custom:` prefix to distinguish them from standard attributes. For more information, see [Custom attributes](#).

Note

To reset their passwords in the forgot-password flow, a user must have either a verified email address or a verified phone number. Amazon Cognito sends a message containing a reset password code to the email address or phone number in the user attributes.

Attributes	Requirement
Any attributes marked as required when you created your user pool	If any required attributes are missing during the migration, Amazon Cognito uses default values.
username	<p>Required if you configured your user pool with alias attributes in addition to username for sign-in, and the user has entered an valid alias value as a username. This alias value can be an email address, preferred username, or phone number.</p> <p>If the request and the user pool meet the alias requirements, the response from your function must assign the <code>username</code> parameter that it received to an alias attribute, Also, the response must assign your own value to the <code>username</code> attribute. If your user pool doesn't meet the conditions required to map the received <code>username</code> to an alias, then the <code>username</code> parameter in the response must either exactly match the request, or be omitted.</p> <p>Note <code>username</code> must be unique in the user pool.</p>

finalUserStatus

You can set this parameter to `CONFIRMED` to auto-confirm your users so that they can sign in with their previous passwords. When you set a user to `CONFIRMED`, they do not need to take additional action before they can sign in. If you don't set this attribute to `CONFIRMED`, it's set to `RESET_REQUIRED`.

A `finalUserStatus` of `RESET_REQUIRED` means that the user must change their password immediately after migration at sign-in, and your client app must handle the `PasswordResetRequiredException` during the authentication flow.

Note

Amazon Cognito doesn't enforce the password strength policy that you configured for the user pool during migration using Lambda trigger. If the password doesn't meet the

password policy that you configured, Amazon Cognito still accepts the password so that it can continue to migrate the user. To enforce password strength policy and reject passwords that don't meet the policy, validate the password strength in your code. Then, if the password doesn't meet the policy, set finalUserStatus to RESET_REQUIRED.

messageAction

You can set this parameter to SUPPRESS to decline to send the welcome message that Amazon Cognito usually sends to new users. If your function doesn't return this parameter, Amazon Cognito sends the welcome message.

desiredDeliveryMediums

You can set this parameter to EMAIL to send the welcome message by email, or SMS to send the welcome message by SMS. If your function doesn't return this parameter, Amazon Cognito sends the welcome message by SMS.

forceAliasCreation

If you set this parameter to TRUE and the phone number or email address in the UserAttributes parameter already exists as an alias with a different user, the API call migrates the alias from the previous user to the newly created user. The previous user can no longer log in using that alias.

If you set this parameter to FALSE and the alias exists, Amazon Cognito doesn't migrate the user and returns an error to the client app.

If you don't return this parameter, Amazon Cognito assumes its value is "false".

Example: Migrate a user with an existing password

This example Lambda function migrates the user with an existing password and suppresses the welcome message from Amazon Cognito.

Node.js

```
exports.handler = (event, context, callback) => {

    var user;

    if ( event.triggerSource == "UserMigration_Authentication" ) {

        // authenticate the user with your existing user directory service
        user = authenticateUser(event.userName, event.request.password);
        if ( user ) {
            event.response.userAttributes = {
                "email": user.emailAddress,
                "email_verified": "true"
            };
            event.response.finalUserStatus = "CONFIRMED";
            event.response.messageAction = "SUPPRESS";
            context.succeed(event);
        }
        else {
            // Return error to Amazon Cognito
            callback("Bad password");
        }
    }
    else if ( event.triggerSource == "UserMigration_ForgotPassword" ) {

        // Lookup the user in your existing user directory service
        user = lookupUser(event.userName);
        if ( user ) {
            event.response.userAttributes = {

```

```

        "email": user.emailAddress,
        // required to enable password-reset code to be sent to user
        "email_verified": "true"
    };
    event.response.messageAction = "SUPPRESS";
    context.succeed(event);
}
else {
    // Return error to Amazon Cognito
    callback("Bad password");
}
else {
    // Return error to Amazon Cognito
    callback("Bad triggerSource " + event.triggerSource);
}
};

}

```

Custom message Lambda trigger

Amazon Cognito invokes this trigger before it sends an email or phone verification message or a multi-factor authentication (MFA) code. You can customize the message dynamically with your custom message trigger. You can edit static custom messages in the **Message customizations** tab of the original [Amazon Cognito](#) console.

The request includes `codeParameter`. This is a string that acts as a placeholder for the code that Amazon Cognito delivers to the user. Insert the `codeParameter` string into the message body where you want the verification code to appear. When Amazon Cognito receives this response, Amazon Cognito replaces the `codeParameter` string with the actual verification code.

Note

A custom message Lambda function with the `CustomMessage_AdminCreateUser` trigger returns a user name and verification code. The request must include both `request.usernameParameter` and `request.codeParameter`.

Topics

- [Custom message Lambda trigger sources \(p. 131\)](#)
- [Custom message Lambda trigger parameters \(p. 132\)](#)
- [Custom message for sign-up example \(p. 133\)](#)
- [Custom message for admin create user example \(p. 135\)](#)

Custom message Lambda trigger sources

triggerSource value	Triggering event
<code>CustomMessage_SignUp</code>	Custom message – To send the confirmation code post sign-up.
<code>CustomMessage_AdminCreateUser</code>	Custom message – To send the temporary password to a new user.
<code>CustomMessage_ResendCode</code>	Custom message – To resend the confirmation code to an existing user.
<code>CustomMessage_ForgotPassword</code>	Custom message – To send the confirmation code for Forgot Password request.

triggerSource value	Triggering event
CustomMessage_UpdateUserAttribute	Custom message – When a user's email or phone number is changed, this trigger sends a verification code automatically to the user. Cannot be used for other attributes.
CustomMessage_VerifyUserAttribute	Custom message – This trigger sends a verification code to the user when they manually request it for a new email or phone number.
CustomMessage_Authentication	Custom message – To send MFA code during authentication.

Custom message Lambda trigger parameters

These are the parameters that Amazon Cognito passes to this Lambda function along with the event information in the [common parameters](#).

JSON

```
{
  "request": {
    "userAttributes": {
      "string": "string",
      ...
    }
    "codeParameter": "####",
    "usernameParameter": "string",
    "clientMetadata": {
      "string": "string",
      ...
    }
  },
  "response": {
    "smsMessage": "string",
    "emailMessage": "string",
    "emailSubject": "string"
  }
}
```

Custom message request parameters

userAttributes

One or more name-value pairs representing user attributes.

codeParameter

A string for you to use as the placeholder for the verification code in the custom message.

usernameParameter

The username parameter. It is a required request parameter for the admin create user flow.

clientMetadata

One or more key-value pairs that you can provide as custom input to the Lambda function that you specify for the custom message trigger. The request that invokes a custom message function doesn't

include data passed in the ClientMetadata parameter in [AdminInitiateAuth](#) and [InitiateAuth](#) API operations. To pass this data to your Lambda function, you can use the ClientMetadata parameter in the following API actions:

- [AdminResetUserPassword](#)
- [AdminRespondToAuthChallenge](#)
- [AdminUpdateUserAttributes](#)
- [ForgotPassword](#)
- [GetUserAttributeVerificationCode](#)
- [ResendConfirmationCode](#)
- [SignUp](#)
- [UpdateUserAttributes](#)

Custom message response parameters

In the response, specify the custom text to use in messages to your users.

smsMessage

The custom SMS message to be sent to your users. Must include the `codeParameter` value that you received in the request.

emailMessage

The custom email message to send to your users. You can use HTML formatting in the `emailMessage` parameter. Must include the `codeParameter` value that you received in the request as the variable `{#####}`. Amazon Cognito can use the `emailMessage` parameter only if the `EmailSendingAccount` attribute of the user pool is `DEVELOPER`. If the `EmailSendingAccount` attribute of the user pool isn't `DEVELOPER` and an `emailMessage` parameter is returned, Amazon Cognito generates a 400 error code `com.amazonaws.cognito.identity.idp.model.InvalidLambdaResponseException`. When you choose Amazon Simple Email Service (Amazon SES) to send email messages, the `EmailSendingAccount` attribute of a user pool is `DEVELOPER`. Otherwise, the value is `COGNITO_DEFAULT`.

emailSubject

The subject line for the custom message. You can only use the `emailSubject` parameter if the `EmailSendingAccount` attribute of the user pool is `DEVELOPER`. If the `EmailSendingAccount` attribute of the user pool isn't `DEVELOPER` and Amazon Cognito returns an `emailSubject` parameter, Amazon Cognito generates a 400 error code `com.amazonaws.cognito.identity.idp.model.InvalidLambdaResponseException`. The `EmailSendingAccount` attribute of a user pool is `DEVELOPER` when you choose to use Amazon Simple Email Service (Amazon SES) to send email messages. Otherwise, the value is `COGNITO_DEFAULT`.

Custom message for sign-up example

This example Lambda function customizes an email or SMS message when the service requires an app to send a verification code to the user.

Amazon Cognito can invoke a Lambda trigger at multiple events: post-registration, resending a verification code, recovering a forgotten password, or verifying a user attribute. The response includes messages for both SMS and email. The message must include the `code` parameter "`#####`". This parameter is the placeholder for the verification code that the user receives.

The maximum length for an email message is 20,000 UTF-8 characters,. This length includes the verification code. You can use HTML tags in these email messages.

The maximum length of SMS messages is 140 UTF-8 characters. This length includes the verification code.

Node.js

```
exports.handler = (event, context, callback) => {
    //
    if(event.userPoolId === "theSpecialUserPool") {
        // Identify why was this function invoked
        if(event.triggerSource === "CustomMessage_SignUp") {
            // Ensure that your message contains event.request.codeParameter. This is
            // the placeholder for code that will be sent
            event.response.smsMessage = "Welcome to the service. Your confirmation code
            is " + event.request.codeParameter;
            event.response.emailSubject = "Welcome to the service";
            event.response.emailMessage = "Thank you for signing up. " +
            event.request.codeParameter + " is your verification code";
        }
        // Create custom message for other events
    }
    // Customize messages for other user pools

    // Return to Amazon Cognito
    callback(null, event);
};
```

Amazon Cognito passes event information to your Lambda function. The function then returns the same event object to Amazon Cognito, with any changes in the response. In the Lambda console, you can set up a test event with data that is relevant to your Lambda trigger. The following is a test event for this code sample:

JSON

```
{
    "version": 1,
    "triggerSource": "CustomMessage_SignUp/CustomMessage_ResendCode/
    CustomMessage_ForgotPassword/CustomMessage_VerifyUserAttribute",
    "region": "<region>",
    "userPoolId": "<userPoolId>",
    "userName": "<userName>",
    "callerContext": {
        "awsSdk": "<calling aws sdk with version>",
        "clientId": "<apps client id>",
        ...
    },
    "request": {
        "userAttributes": {
            "phone_number_verified": false,
            "email_verified": true,
            ...
        },
        "codeParameter": "####"
    },
    "response": {
        "smsMessage": "<custom message to be sent in the message with code parameter>",
        "emailMessage": "<custom message to be sent in the message with code parameter>",
        "emailSubject": "<custom email subject>"
    }
}
```

}

Custom message for admin create user example

A custom message Lambda function with the `CustomMessage_AdminCreateUser` trigger returns a user name and verification code. For this reason, include both `request.usernameParameter` and `request.codeParameter` in the message body.

The code parameter value ##### is a placeholder for the temporary password, and "username" is a placeholder for the username that your user receives.

The maximum length of an email message is 20,000 UTF-8 characters. This length includes the verification code. You can use HTML tags in these emails. The maximum length of SMS messages is 140 UTF-8 characters. This length includes the verification code.

The response includes messages for both SMS and email.

Node.js

```
exports.handler = (event, context, callback) => {
    //
    if(event.userPoolId === "theSpecialUserPool") {
        // Identify why was this function invoked
        if(event.triggerSource === "CustomMessage_AdminCreateUser") {
            // Ensure that your message contains event.request.codeParameter
            event.request.usernameParameter. This is the placeholder for the code and username
            that will be sent to your user.
            event.response.smsMessage = "Welcome to the service. Your user
            name is " + event.request.usernameParameter + " Your temporary password is " +
            event.request.codeParameter;
            event.response.emailSubject = "Welcome to the service";
            event.response.emailMessage = "Welcome to the service. Your user
            name is " + event.request.usernameParameter + " Your temporary password is " +
            event.request.codeParameter;
        }
        // Create custom message for other events
    }
    // Customize messages for other user pools

    // Return to Amazon Cognito
    callback(null, event);
};
```

Amazon Cognito passes event information to your Lambda function. The function then returns the same event object to Amazon Cognito, with any changes in the response. In the Lambda console, you can set up a test event with data that is relevant to your Lambda trigger. The following is a test event for this code sample:

JSON

```
{
    "version": 1,
    "triggerSource": "CustomMessage_AdminCreateUser",
    "region": "<region>",
    "userPoolId": "<userPoolId>",
    "userName": "<userName>",
    "callerContext": {
        "awsSdk": "<calling aws sdk with version>",
    }
}
```

```
        "clientId": "<apps client id>",
        ...
    },
    "request": {
        "userAttributes": {
            "phone_number_verified": false,
            "email_verified": true,
            ...
        },
        "codeParameter": "#####",
        "usernameParameter": "username"
    },
    "response": {
        "smsMessage": "<custom message to be sent in the message with code parameter and username parameter>"  

        "emailMessage": "<custom message to be sent in the message with code parameter and username parameter>"  

        "emailSubject": "<custom email subject>"
    }
}
```

Custom sender Lambda triggers

Amazon Cognito user pools provide the Lambda triggers `CustomEmailSender` and `CustomSMSSender` to activate third-party email and SMS notifications. You can choose SMS and email providers to send notifications to users from within your Lambda function code. When Amazon Cognito must send notifications like confirmation codes, verification codes, or temporary passwords to users, the events activate your configured Lambda functions. Amazon Cognito sends the code and temporary passwords (secrets) to your activated Lambda functions. Amazon Cognito encrypts these secrets with an AWS KMS customer managed key and the AWS Encryption SDK. The AWS Encryption SDK is a client-side encryption library that helps you to encrypt and decrypt generic data.

Note

To configure your user pools to use these Lambda triggers, you can use the AWS CLI or SDK. These configurations aren't available from Amazon Cognito console.

[CustomEmailSender \(p. 137\)](#)

Amazon Cognito invokes this trigger to send email notifications to users.

[CustomSMSSender \(p. 141\)](#)

Amazon Cognito invokes this trigger to send SMS notifications to users.

Resources

The following resources can help you to use the `CustomEmailSender` and `CustomSMSSender` triggers.

AWS KMS

AWS KMS is a managed service to create and control AWS KMS keys. These keys encrypt your data. For more information see, [What is AWS Key Management Service?](#)

KMS key

A KMS key is a logical representation of a cryptographic key. The KMS key includes metadata, such as the key ID, creation date, description, and key state. The KMS key also contains the key material used to encrypt and decrypt data. For more information see, [AWS KMS keys](#).

Symmetric KMS key

A symmetric KMS key is a 256-bit encryption key that doesn't exit AWS KMS unencrypted. To use a symmetric KMS key, you must call AWS KMS. Amazon Cognito uses symmetric keys. The same key encrypts and decrypts. For more information see, [Symmetric KMS keys](#).

Custom email sender Lambda trigger

Amazon Cognito invokes the custom email sender trigger so that a third-party provider can send email notifications to your users from your AWS Lambda function code. Amazon Cognito sends email message events as requests to a Lambda function. The custom code of your function must then process and deliver the message.

Note

Currently, you can't assign a custom email sender trigger in the Amazon Cognito console. You can assign a trigger with the `LambdaConfig` parameter in a `CreateUserPool` or `UpdateUserPool` API request.

Set up the custom email sender trigger as follows:

1. Create a Lambda function that you want to assign as your custom email sender trigger.
2. Create an encryption key in AWS Key Management Service (AWS KMS). Amazon Cognito generates secrets (temporary passwords and authorization codes), then uses this key to encrypt them. You can then use the AWS Encryption SDK in your Lambda function to decrypt the codes and send them to the user in plaintext.
3. Grant Amazon Cognito service principal `cognito-idp.amazonaws.com` permission to invoke the Lambda function.
4. Write Lambda function code that directs your email messages to custom delivery methods or third-party providers.
5. Update the user pool so that it uses a custom sender Lambda trigger.

Important

When you configure a custom email sender or a custom SMS sender function for your user pool, configure a symmetric AWS KMS key for additional security. Amazon Cognito uses your configured KMS key to encrypt codes or temporary passwords. Amazon Cognito sends the base64 encoded ciphertext to your Lambda functions. For more information, see [Symmetric KMS keys](#).

Custom email sender Lambda trigger parameters

These are the parameters that Amazon Cognito passes to this Lambda function along with the event information in the [common parameters](#).

JSON

```
{  
    "request": {  
        "type": "customEmailSenderRequestV1",  
        "code": "string",  
        "clientMetadata": {  
            "string": "string",  
            . . .  
        },  
        "userAttributes": {  
            "string": "string",  
            . . .  
        }  
    }  
}
```

```
}
```

Custom email sender request parameters

type

The request version. For a custom email sender event, the value of this string is always `customEmailSenderRequestV1`.

code

The encrypted code that your function can decrypt and send to your user.

clientMetadata

One or more key-value pairs that you can provide as custom input to the custom email sender Lambda function trigger. To pass this data to your Lambda function, you can use the `ClientMetadata` parameter in the [AdminRespondToAuthChallenge](#) and [RespondToAuthChallenge](#) API actions. Amazon Cognito doesn't include data from the `ClientMetadata` parameter in [AdminInitiateAuth](#) and [InitiateAuth](#) API operations in the request that it passes to the post authentication function.

userAttributes

One or more key-value pairs that represent user attributes.

Custom email sender response parameters

Amazon Cognito doesn't expect any additional return information in the custom email sender response. Your function can use API operations to query and modify your resources, or record event metadata to an external system.

Activating the custom email sender Lambda trigger

To set up a custom email sender trigger that uses custom logic to send email messages for your user pool, activate the trigger as follows.

Step 1: Create a Lambda function

Create a Lambda function for the custom email sender trigger. Amazon Cognito uses the [AWS encryption SDK](#) to encrypt the secrets (temporary passwords or authorization codes).

Step 2: Create an encryption key in AWS KMS

Create an encryption key in AWS KMS. Amazon Cognito uses this key to encrypt temporary passwords and authorization codes that Amazon Cognito generates. You can then decrypt these secrets in the custom sender Lambda function and send them to the user in plaintext.

Step 3: Grant Amazon Cognito permission to invoke the Lambda function

Use the following command:

```
aws lambda add-permission --function-name lambda_arn --statement-id "CognitoLambdaInvokeAccess" --action lambda:InvokeFunction --principal cognito-idp.amazonaws.com
```

Step 4: Edit the code to use custom sender

Amazon Cognito uses AWS Encryption SDK to encrypt secrets (temporary passwords and authorization codes) before Amazon Cognito sends the secrets to the custom sender Lambda function. Decrypt these secrets before you send them to users. You can choose a custom provider to send the email message. To use the AWS Encryption SDK with your Lambda function, you must package the SDK with your function. For more information, see [Installing the AWS encryption SDK for JavaScript](#). You can also update the Lambda package as follows:

1. Export the Lambda function package from the console.
2. Unzip the package.
3. Add the AWS Encryption SDK to the package. For example, if you are using Node.js, then add the `node_modules` directory and include the libraries from `@aws-crypto/client-node`.
4. Recreate the package.
5. Update the Lambda function code from the modified directory.

Step 5: Update user pool to add custom sender Lambda triggers

Update the user pool with a `CustomEmailSender` parameter in an `UpdateUserPool` API operation. `UpdateUserPool` requires all the parameters of your user pool as well as the parameters that you want to change. If you don't provide all relevant parameters, Amazon Cognito sets the values of any missing parameters to their defaults. For more information, see [Updating user pool configuration \(p. 37\)](#).

```
#Send this parameter in an 'aws cognito-identity update-user-pool' CLI command, along with
any existing user pool configurations.

--lambda-config "CustomEmailSender={LambdaVersion=V1_0,LambdaArn= lambda-arn
},KMSKeyID= key-id"
```

To remove a custom email sender Lambda trigger with the AWS CLI, omit the `CustomEmailSender` parameter from `--lambda-config` and include all other triggers that you want to use with your user pool. To remove a custom email sender Lambda trigger with an `UpdateUserPool` API request, remove `CustomEmailSender` from the request body that contains the rest of your user pool configuration. For more information, see [Updating a user pool with the Amazon Cognito API or AWS CLI \(p. 37\)](#).

The following Node.js example shows how to process an email message event in your custom email sender Lambda function.

```
const AWS = require('aws-sdk');
const b64 = require('base64-js');
const encryptionSdk = require('@aws-crypto/client-node');

#Configure the encryption SDK client with the KMS key from the environment
variables.

const { encrypt, decrypt } =
encryptionSdk.buildClient(encryptionSdk.CommitmentPolicy.REQUIRE_ENCRYPT_ALLOW_DECRYPT);
const generatorKeyId = process.env.KEY_ALIAS;
const keyIds = [ process.env.KEY_ID ];
const keyring = new encryptionSdk.KmsKeyringNode({ generatorKeyId, keyIds })
exports.handler = async (event) => {
```

```

#Decrypt the secret code using encryption SDK.
let plainTextCode;
if(event.request.code){
const { plaintext, messageHeader } = await decrypt(keyring,
b64.toByteArray(event.request.code));
plainTextCode = plaintext
}

#PlainTextCode now has the decrypted secret.

if(event.triggerSource == 'CustomEmailSender_SignUp'){

#Send email to end-user using custom or 3rd party provider.
#include temporary password in the email.

}else if(event.triggerSource == 'CustomEmailSender_ResendCode'){

}else if(event.triggerSource == 'CustomEmailSender_ForgotPassword'){

}else if(event.triggerSource == 'CustomEmailSender_UpdateUserAttribute'){

}else if(event.triggerSource == 'CustomEmailSender_VerifyUserAttribute'){

}else if(event.triggerSource == 'CustomEmailSender_AdminCreateUser'){

}else if(event.triggerSource ==
'CustomEmailSender_AccountTakeOverNotification'){

}

return;
};

```

Custom email sender Lambda trigger sources

The following table shows the triggering events for custom email trigger sources in your Lambda code.

TriggerSource value	Triggering event
CustomEmailSender_SignUp	A user signs up and Amazon Cognito sends a welcome message.
CustomEmailSender_ForgotPassword	A user requests a code to reset their password.
CustomEmailSender_ResendCode	A user requests a replacement code to reset their password.
CustomEmailSender_UpdateUserAttribute	A user updates an email address or phone number attribute and Amazon Cognito sends a code to verify the attribute.
CustomEmailSender_VerifyUserAttribute	A user creates a new email address or phone number attribute and Amazon Cognito sends a code to verify the attribute.
CustomEmailSender_AdminCreateUser	You create a new user in your user pool and Amazon Cognito sends them a temporary password.

TriggerSource value	Triggering event
CustomEmailSender_AccountTakeOverNotification	Amazon Cognito detects an attempt to take over a user account and sends the user a notification.

Custom SMS sender Lambda trigger

Amazon Cognito invokes the custom SMS sender trigger so that a third-party provider can send SMS notifications to your users from your AWS Lambda function code. Amazon Cognito sends SMS message events as requests to a Lambda function. The custom code of your function must then process and deliver the message.

Note

Currently, you can't assign a custom SMS sender trigger in the Amazon Cognito console. You can assign a trigger with the `LambdaConfig` parameter in a `CreateUserPool` or `UpdateUserPool` API request.

To set up this trigger, perform the following steps:

1. Create a Lambda function that you want to assign as your custom SMS sender trigger.
2. Create an encryption key in AWS Key Management Service (AWS KMS). Amazon Cognito generates secrets (temporary passwords and authorization codes), then uses this key to encrypt the secrets. You can then use the AWS Encryption SDK in your Lambda function to decrypt the codes and send them to the user in plaintext.
3. Grant Amazon Cognito service principal `cognito-idp.amazonaws.com` access to invoke the Lambda function.
4. Write Lambda function code that directs your SMS messages to custom delivery methods or third-party providers.
5. Update the user pool so that it uses a custom sender Lambda trigger.

Important

Create a new symmetric AWS KMS key when you configure a custom email or custom SMS sender function. Amazon Cognito uses your configured KMS key to encrypt codes or temporary passwords. Amazon Cognito sends the base64 encoded ciphertext to your Lambda functions. For more information, see [Symmetric KMS keys](#).

Custom SMS sender Lambda trigger parameters

These are the parameters that Amazon Cognito passes to this Lambda function along with the event information in the [common parameters](#).

JSON

```
{
  "request": {
    "type": "customSMSenderRequestV1",
    "code": "string",
    "clientMetadata": {
      "string": "string",
      ...
    },
    "userAttributes": {
      "string": "string",
      ...
    }
  }
}
```

Custom SMS sender request parameters

type

The request version. For a custom SMS sender event, the value of this string is always `customSMSSenderRequestV1`.

code

The encrypted code that your function can decrypt and send to your user.

clientMetadata

One or more key-value pairs that you can provide as custom input to the custom SMS sender Lambda function trigger. To pass this data to your Lambda function, you can use the `ClientMetadata` parameter in the [AdminRespondToAuthChallenge](#) and [RespondToAuthChallenge](#) API actions. Amazon Cognito doesn't include data from the `ClientMetadata` parameter in [AdminInitiateAuth](#) and [InitiateAuth](#) API operations in the request that it passes to the post authentication function.

userAttributes

One or more key-value pairs that represent user attributes.

Custom SMS sender response parameters

Amazon Cognito doesn't expect any additional return information in the response. Your function can use API operations to query and modify your resources, or record event metadata to an external system.

Activating the custom SMS sender Lambda trigger

To set up a custom SMS sender trigger that uses custom logic to send SMS messages for your user pool, activate the trigger as follows.

Step 1: Create a Lambda function

Create a Lambda function for the custom SMS sender trigger. Amazon Cognito uses the [AWS encryption SDK](#) to encrypt the secrets (temporary passwords or authorization codes).

Step 2: Create an encryption key in AWS KMS

Create an encryption key in AWS KMS. This key encrypts temporary passwords and authorization codes that Amazon Cognito generates. You can then decrypt these secrets in the custom sender Lambda function and send them to the user in plaintext.

Step 3: Grant Amazon Cognito service principal `cognito-idp.amazonaws.com` access to invoke the Lambda function

Use the following command to grant access to the Lambda function:

```
aws lambda add-permission --function-name lambda_arn --statement-id "CognitoLambdaInvokeAccess" --action lambda:InvokeFunction --principal cognito-idp.amazonaws.com
```

Step 4: Edit the code to use custom sender

Amazon Cognito uses AWS Encryption SDK to encrypt secrets (temporary passwords and authorization codes) before Amazon Cognito sends the secrets to the custom sender Lambda function. Decrypt these secrets before you send them to users through the custom provider of your choice. To use the

AWS Encryption SDK with your Lambda function, you must package the SDK with your function. For information, see [Installing the AWS encryption SDK for JavaScript](#). To update the Lambda package, complete the following steps:

1. Export the Lambda function package from the console.
2. Unzip the package.
3. Add the AWS Encryption SDK to the package. For example, if you are using Node.js, then add the `node_modules` directory and include the libraries from `@aws-crypto/client-node`.
4. Recreate the package.
5. Update the Lambda function code from the modified directory.

Step 5: Update user pool to add custom sender Lambda triggers

Update the user pool with a `CustomSMSender` parameter in an `UpdateUserPool` API operation. `UpdateUserPool` requires all the parameters of your user pool as well as the parameters that you want to change. If you don't provide all relevant parameters, Amazon Cognito sets the values of any missing parameters to their defaults. For more information, see [Updating user pool configuration \(p. 37\)](#).

```
#Send this parameter in an 'aws cognito-identity update-user-pool' CLI command, along with
any existing user pool configurations.

--lambda-config "CustomSMSender={LambdaVersion=V1_0,LambdaArn= lambda-arn
},KMSKeyID= key-id"
```

To remove a custom SMS sender Lambda trigger with the AWS CLI, omit the `CustomSMSender` parameter from `--lambda-config` and include all other triggers that you want to use with your user pool. To remove a custom SMS sender Lambda trigger with an `UpdateUserPool` API request, remove `CustomSMSender` from the request body that contains the rest of your user pool configuration. For more information, see [Updating a user pool with the Amazon Cognito API or AWS CLI \(p. 37\)](#).

Code examples

The following Node.js example shows how to process an SMS message event in your custom SMS sender Lambda function.

```
const AWS = require('aws-sdk');
const b64 = require('base64-js');
const encryptionSdk = require('@aws-crypto/client-node');

#Configure the encryption SDK client with the KMS key from the environment
variables.

const { encrypt, decrypt } =
encryptionSdk.buildClient(encryptionSdk.CommitmentPolicy.REQUIRE_ENCRYPT_ALLOW_DECRYPT);
const generatorKeyId = process.env.KEY_ALIAS;
const keyIds = [ process.env.KEY_ID ];
const keyring = new encryptionSdk.KmsKeyringNode({ generatorKeyId, keyIds })
exports.handler = async (event) => {

#Decrypt the secret code using encryption SDK.

let plainTextCode;
if(event.request.code){
const { plaintext, messageHeader } = await decrypt(keyring,
b64.toByteArray(event.request.code));
```

```
plainTextCode = plaintext
}

#PlainTextCode now has the decrypted secret.

if(event.triggerSource == 'CustomSMSender_SignUp'){

#Send sms to end-user using custom or 3rd party provider.
#include temporary password in the email.

}else if(event.triggerSource == 'CustomSMSender_ResendCode'){

}else if(event.triggerSource == 'CustomSMSender_ForgotPassword'){

}else if(event.triggerSource == 'CustomSMSender_UpdateUserAttribute'){

}else if(event.triggerSource == 'CustomSMSender_VerifyUserAttribute'){

}else if(event.triggerSource == 'CustomSMSender_AdminCreateUser'){

}else if(event.triggerSource == 'CustomSMSender_AccountTakeOverNotification'){

}

return;
};
```

Topics

- [Evaluate SMS message capabilities with a custom SMS sender function \(p. 144\)](#)
- [Custom SMS sender Lambda trigger sources \(p. 140\)](#)

Evaluate SMS message capabilities with a custom SMS sender function

A custom SMS sender Lambda function accepts the SMS messages that your user pool would send, and the function delivers the content based on your custom logic. Amazon Cognito sends the [Custom SMS sender Lambda trigger parameters \(p. 141\)](#) to your function. Your function can do what you want with this information. For example, you can send the code to an Amazon Simple Notification Service (Amazon SNS) topic. An Amazon SNS topic subscriber can be an SMS message, an HTTPS endpoint, or an email address.

To create a test environment for Amazon Cognito SMS messaging with a custom SMS sender Lambda function, see [amazon-cognito-user-pool-development-and-testing-with-sms-redirected-to-email](#) in the [aws-samples library on GitHub](#). The repository contains AWS CloudFormation templates that can create a new user pool, or work with a user pool that you already have. These templates create Lambda functions and an Amazon SNS topic. The Lambda function that the template assigns as a custom SMS sender trigger, redirects your SMS messages to the subscribers to the Amazon SNS topic.

When you deploy this solution to a user pool, all messages that Amazon Cognito usually sends through SMS messaging, the Lambda function instead sends to a central email address. Use this solution to customize and preview SMS messages, and to test the user pool events that cause Amazon Cognito to send an SMS message. After you complete your tests, roll back the CloudFormation stack, or remove the custom SMS sender function assignment from your user pool.

Important

Don't use the templates in [amazon-cognito-user-pool-development-and-testing-with-sms-redirected-to-email](#) to build a production environment. The custom SMS sender Lambda function in the solution *simulates* SMS messages, but the Lambda function sends them all to a single central email address. Before you can send SMS messages in a production Amazon

Cognito user pool, you must complete the requirements shown at [SMS message settings for Amazon Cognito user pools \(p. 187\)](#).

Custom SMS sender Lambda trigger sources

The following table shows the triggering event for custom SMS trigger sources in your Lambda code.

TriggerSource value	Triggering event
CustomSMSender_SignUp	A user signs up and Amazon Cognito sends a welcome message.
CustomSMSender_ForgotPassword	A user requests a code to reset their password.
CustomSMSender_ResendCode	A user requests a replacement code to reset their password.
CustomSMSender_VerifyUserAttribute	A user creates a new email address or phone number attribute and Amazon Cognito sends a code to verify the attribute.
CustomSMSender_UpdateUserAttribute	A user updates an email address or phone number attribute and Amazon Cognito sends a code to verify the attribute.
CustomSMSender_Authentication	A user configured with SMS multi-factor authentication (MFA) signs in.
CustomSMSender_AdminCreateUser	You create a new user in your user pool and Amazon Cognito sends them a temporary password.

Using Amazon Pinpoint analytics with Amazon Cognito user pools

Amazon Cognito User Pools are integrated with Amazon Pinpoint to provide analytics for Amazon Cognito user pools and to enrich the user data for Amazon Pinpoint campaigns. Amazon Pinpoint provides analytics and targeted campaigns to drive user engagement in mobile apps using push notifications. With Amazon Pinpoint analytics support in Amazon Cognito user pools, you can track user pool sign-ups, sign-ins, failed authentications, daily active users (DAUs), and monthly active users (MAUs) in the Amazon Pinpoint console. You can drill into the data for different date ranges or attributes, such as device platform, device locale, and app version.

You can also set up user attributes that are specific to your app using the AWS Mobile SDK for Android or AWS Mobile SDK for iOS. Those can then be used to segment your users on Amazon Pinpoint and send them targeted push notifications. If you choose **Share user attribute data with Amazon Pinpoint** in the **Analytics** tab in the Amazon Cognito console, additional endpoints are created for user email addresses and phone numbers.

Find Amazon Cognito and Amazon Pinpoint Region mappings

The following table shows Region mappings between Amazon Cognito and Amazon Pinpoint. Use the table to find the Region where you built your Amazon Cognito user pool and the corresponding Amazon

Pinpoint Region. Next, use these Regions to integrate Amazon Cognito and your Amazon Pinpoint project.

Amazon Cognito regions that support Amazon Pinpoint	Amazon Pinpoint project regions
ap-northeast-1	us-east-1
ap-northeast-2	us-east-1
ap-south-1	us-east-1, ap-south-1
ap-southeast-1	us-east-1
ap-southeast-2	us-east-1, ap-southeast-2
ca-central-1	us-east-1
eu-central-1	us-east-1, eu-central-1
eu-west-1	us-east-1, eu-west-1
eu-west-2	us-east-1
us-east-1	us-east-1
us-east-2	us-east-1
us-west-2	us-east-1, us-west-2

Region mapping examples

- If you create a user pool in ap-northeast-1, you have to create your Amazon Pinpoint project in us-east-1.
- If you create a user pool in ap-south-1, you have to create your Amazon Pinpoint project in either us-east-1 or ap-south-1.

Note

Amazon Pinpoint is available in several AWS Regions in North America, Europe, Asia, and Oceania. Besides the exceptions in the table, Amazon Cognito will only support in-Region Amazon Pinpoint integrations. If Amazon Pinpoint is available in the same Region as Amazon Cognito, then Amazon Cognito sends events to Amazon Pinpoint projects within the same Region. If Amazon Pinpoint isn't available in the Region, then Amazon Cognito doesn't support Amazon Pinpoint integrations in that Region until Amazon Pinpoint becomes available. For Amazon Pinpoint detailed Region information, see [Amazon Pinpoint endpoints and quotas](#).

Specifying Amazon Pinpoint analytics settings (AWS Management Console)

To specify analytics settings

1. Sign in to the [Amazon Cognito console](#).
2. In the navigation pane, **Manage User Pools**, and choose the user pool you want to edit.
3. Choose the **Analytics** tab.
4. Choose **Add analytics and campaigns**.

5. Choose a **Cognito app client** from the list.
6. To map your Amazon Cognito app to an **Amazon Pinpoint project**, choose the Amazon Pinpoint project from the list.

Note

The Amazon Pinpoint project ID is a 32-character string that is unique to your Amazon Pinpoint project. It is listed in the Amazon Pinpoint console.

You can map multiple Amazon Cognito apps to a single Amazon Pinpoint project. However, each Amazon Cognito app can only be mapped to one Amazon Pinpoint project.

In Amazon Pinpoint, each project should be a single app. For example, if a game developer has two games, each game should be a separate Amazon Pinpoint project, even if both games use the same Amazon Cognito user pool. For more information on Amazon Pinpoint projects, see [Create a project in Amazon Pinpoint](#).

7. Choose **Share user attribute data with Amazon Pinpoint** if you want Amazon Cognito to send email addresses and phone numbers to Amazon Pinpoint in order to create additional endpoints for users. After the account phone number and email address are verified, they are only shared with Amazon Pinpoint if they are available to the user account.

Note

An *endpoint* uniquely identifies a user device to which you can send push notifications with Amazon Pinpoint. For more information about endpoints, see [Adding endpoints](#) in the *Amazon Pinpoint Developer Guide*.

8. Choose **Save changes**.
9. To specify additional app mappings, choose **Add another app mapping**.
10. Choose **Save changes**.

Specifying Amazon Pinpoint analytics settings (AWS CLI and AWS API)

Use the following commands to specify Amazon Pinpoint analytics settings for your user pool.

To specify the analytics settings for your user pool's existing client app at app creation time

- AWS CLI: `aws cognito-identity create-user-pool-client`
- AWS API: [CreateUserPoolClient](#)

To update the analytics settings for your user pool's existing client app

- AWS CLI: `aws cognito-identity update-user-pool-client`
- AWS API: [UpdateUserPoolClient](#)

Note

Amazon Cognito supports in-Region integrations when you use `ApplicationArn`

Managing users in your user pool

After you create a user pool, you can create, confirm, and manage users accounts. With Amazon Cognito user pools groups you can manage your users and their access to resources by mapping IAM roles to groups.

You can import your users into a user pool with a user migration Lambda trigger. This approach enables seamless migration of users from your existing user directory to user pools when they sign in to your user pool for the first time.

Topics

- [Signing up and confirming user accounts \(p. 148\)](#)
- [Creating user accounts as administrator \(p. 157\)](#)
- [Adding groups to a user pool \(p. 162\)](#)
- [Managing and searching for user accounts \(p. 165\)](#)
- [Recovering user accounts \(p. 169\)](#)
- [Importing users into a user pool \(p. 170\)](#)

Signing up and confirming user accounts

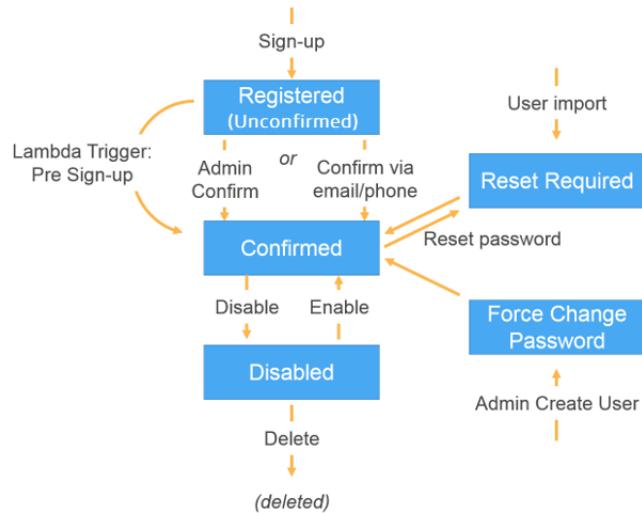
User accounts are added to your user pool in one of the following ways:

- The user signs up in your user pool's client app. This can be a mobile or web app.
- You can import the user's account into your user pool. For more information, see [Importing users into user pools from a CSV file \(p. 171\)](#).
- You can create the user's account in your user pool and invite the user to sign in. For more information, see [Creating user accounts as administrator \(p. 157\)](#).

Users who sign themselves up need to be confirmed before they can sign in. Imported and created users are already confirmed, but they must create their password the first time they sign in. The following sections explain the confirmation process and email and phone verification.

Overview of user account confirmation

The following diagram illustrates the confirmation process:



A user account can be in any of the following states:

Registered (Unconfirmed)

The user has successfully signed up, but cannot sign in until the user account is confirmed. The user is enabled but not confirmed in this state.

New users who sign themselves up start in this state.

Confirmed

The user account is confirmed and the user can sign in. When a user enters a code or follows an email link to confirm their user account, that email or phone number is automatically verified. The code or link is valid for 24 hours.

If the user account was confirmed by the administrator or a pre sign-up Lambda trigger, there might not be a verified email or phone number associated with the account.

Password Reset Required

The user account is confirmed, but the user must request a code and reset their password before they can sign in.

User accounts that are imported by an administrator or developer start in this state.

Force Change Password

The user account is confirmed and the user can sign in using a temporary password, but on first sign-in, the user must change their password to a new value before doing anything else.

User accounts that are created by an administrator or developer start in this state.

Disabled

Before you can delete a user account, you must disable sign-in access for that user.

Verifying contact information at sign-up

When new users sign up in your app, you probably want them to provide at least one contact method. For example, with your users' contact information, you might:

- Send a temporary password when a user chooses to reset their password.
- Notify users when their personal or financial information is updated.
- Send promotional messages, such as special offers or discounts.
- Send account summaries or billing reminders.

For use cases like these, it's important that you send your messages to a verified destination. Otherwise, you might send your messages to an invalid email address or phone number that was typed incorrectly. Or worse, you might send sensitive information to bad actors who pose as your users.

To help ensure that you send messages only to the right individuals, configure your Amazon Cognito user pool so that users must provide the following when they sign up:

- a. An email address or phone number.
- b. A verification code that Amazon Cognito sends to that email address or phone number.

By providing the verification code, a user proves that they have access to the mailbox or phone that received the code. After the user provides the code, Amazon Cognito updates the information about the user in your user pool by:

- Setting the user's status to CONFIRMED.
- Updating the user's attributes to indicate that the email address or phone number is verified.

To view this information, you can use the Amazon Cognito console. Or, you can use the `Admin GetUser` API action, the `admin-get-user` command with the AWS CLI, or a corresponding action in one of the AWS SDKs.

If a user has a verified contact method, Amazon Cognito automatically sends a message to the user when the user requests a password reset.

To configure your user pool to require email or phone verification

When you verify your users' email addresses and phone numbers, you ensure that you can contact your users. Complete the following steps in the AWS Management Console to configure your user pool to require that your users confirm their email addresses or phone numbers.

Note

If you don't yet have a user pool in your account, see [Getting started with user pools \(p. 23\)](#).

Original console

To configure your user pool

1. Sign in to the AWS Management Console and open the Amazon Cognito console at <https://console.aws.amazon.com/cognito>. If prompted, enter your AWS credentials.
2. Choose **Manage User Pools**.
3. On the **Your User Pools** page, choose the user pool that you want to configure.
4. In the navigation menu on the left, choose **MFA and verifications**.
5. Under **Which attributes do you want to verify?**, choose one of the following options:



Email

If you choose this option, Amazon Cognito emails a verification code when the user signs up. Choose this option if you typically communicate with your users through email. For example, you will want to use verified email addresses if you send billing statements, order summaries, or special offers.

Phone number

If you choose this option, Amazon Cognito sends a verification code through SMS message when the user signs up. Choose this option if you typically communicate with your users through SMS messages. For example, you will want to use verified phone numbers if you send delivery notifications, appointment confirmations, or alerts.

Email or phone number

Choose this option if you don't require all users to have the same verified contact method. In this case, the sign-up page in your app could ask users to verify only their preferred contact method. When Amazon Cognito sends a verification code, it sends the code to the contact method provided in the `SignUp` request from your app. If a user provides both an email address and a phone number, and your app provides both contact methods in the `SignUp` request, Amazon Cognito sends a verification code only to the phone number.

If you require users to verify both an email address and a phone number, choose this option. Amazon Cognito verifies one contact method when the user signs up, and your app must verify the other contact method after the user signs in. For more information, see [If you require users to confirm both email addresses and phone numbers \(p. 153\)](#).

None

If you choose this option, Amazon Cognito doesn't send verification codes when users sign up. Choose this option if you are using a custom authentication flow that verifies at least

one contact method without using verification codes from Amazon Cognito. For example, you might use a pre sign-up Lambda trigger that automatically verifies email addresses that belong to a specific domain.

If you don't verify your users' contact information, they may be unable to use your app. Remember that users require verified contact information to:

- **Reset their passwords** — When a user chooses an option in your app that calls the `ForgotPassword` API action, Amazon Cognito sends a temporary password to the user's email address or phone number. Amazon Cognito sends this password only if the user has at least one verified contact method.
- **Sign in by using an email address or phone number as an alias** — If you configure your user pool to allow these aliases, then a user can sign in with an alias only if the alias is verified. For more information, see [Aliases \(p. 209\)](#).

6. Choose **Save changes**.

New console

To configure your user pool

1. Navigate to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. From the navigation pane, choose **User Pools**. Choose an existing user pool from the list, or [create a user pool](#).
3. Choose the **Sign-up experience** tab and locate **Attribute verification and user account confirmation**. Choose **Edit**.
4. Choose whether you will activate **Cognito-assisted verification and confirmation** to have Amazon Cognito send messages to the user contact attributes you choose when a user signs up, or you create a user profile. The messages that Amazon Cognito sends provide users with a code or link that, after they have confirmed they received it, verifies the attribute and confirms the user profile for sign-in.

Note

You can also disable **Cognito-assisted verification and confirmation** and use authenticated API actions or Lambda triggers to verify attributes and confirm users. If you choose this option, Amazon Cognito doesn't send verification codes when users sign up. Choose this option if you are using a custom authentication flow that verifies at least one contact method without using verification codes from Amazon Cognito. For example, you might use a pre sign-up Lambda trigger that automatically verifies email addresses that belong to a specific domain.

If you don't verify your users' contact information, they may be unable to use your app. Remember that users require verified contact information to:

- **Reset their passwords** — When a user chooses an option in your app that calls the `ForgotPassword` API action, Amazon Cognito sends a temporary password to the user's email address or phone number. Amazon Cognito sends this password only if the user has at least one verified contact method.
- **Sign in by using an email address or phone number as an alias** — If you configure your user pool to allow these aliases, then a user can sign in with an alias only if the alias is verified. For more information, see [Aliases \(p. 209\)](#).

5. Choose your **Attributes to verify**:

Send SMS message, verify phone number

Amazon Cognito sends a verification code in an SMS message when the user signs up. Choose this option if you typically communicate with your users through SMS messages. For example, you will want to use verified phone numbers if you send delivery notifications, appointment confirmations, or alerts. User phone numbers will be the verified attribute

when accounts are confirmed; you must take additional action to verify and communicate with user email addresses.

Send email message, verify email address

Amazon Cognito sends a verification code through an email message when the user signs up. Choose this option if you typically communicate with your users through email. For example, you will want to use verified email addresses if you send billing statements, order summaries, or special offers. User email addresses will be the verified attribute when accounts are confirmed; you must take additional action to verify and communicate with user phone numbers.

Send SMS message if phone number is available, otherwise send email message

Choose this option if you don't require all users to have the same verified contact method. In this case, the sign-up page in your app could ask users to verify only their preferred contact method. When Amazon Cognito sends a verification code, it sends the code to the contact method provided in the `SignUp` request from your app. If a user provides both an email address and a phone number, and your app provides both contact methods in the `SignUp` request, Amazon Cognito sends a verification code only to the phone number.

If you require users to verify both an email address and a phone number, choose this option. Amazon Cognito verifies one contact method when the user signs up, and your app must verify the other contact method after the user signs in. For more information, see [If you require users to confirm both email addresses and phone numbers \(p. 153\)](#).

6. Choose **Save changes**.

Authentication flow with email or phone verification

If your user pool requires users to verify their contact information, your app must facilitate the following flow when a user signs up:

1. A user signs up in your app by entering a username, phone number and/or email address, and possibly other attributes.
2. The Amazon Cognito service receives the sign-up request from the app. After verifying that the request contains all attributes required for sign-up, the service completes the sign-up process and sends a confirmation code to the user's phone (in an SMS message) or email. The code is valid for 24 hours.
3. The service returns to the app that sign-up is complete and that the user account is pending confirmation. The response contains information about where the confirmation code was sent. At this point the user's account is in an unconfirmed state, and the user's email address and phone number are unverified.
4. The app can now prompt the user to enter the confirmation code. It is not necessary for the user to enter the code immediately. However, the user will not be able to sign in until after they enter the confirmation code.
5. The user enters the confirmation code in the app.
6. The app calls `ConfirmSignUp` to send the code to the Amazon Cognito service, which verifies the code and, if the code is correct, sets the user's account to the confirmed state. After successfully confirming the user account, the Amazon Cognito service automatically marks the attribute that was used to confirm (email address or phone number) as verified. Unless the value of this attribute is changed, the user will not have to verify it again.
7. At this point the user's account is in a confirmed state, and the user can sign in.

If you require users to confirm both email addresses and phone numbers

Amazon Cognito verifies only one contact method when a user signs up. In cases where Amazon Cognito must choose between verifying an email address or phone number, it chooses to verify the phone number by sending a verification code through SMS message. For example, if you configure your user pool to allow users to verify either email addresses or phone numbers, and if your app provides both of these attributes upon sign-up, Amazon Cognito verifies only the phone number. After a user verifies his or her phone number, Amazon Cognito sets the user's status to CONFIRMED, and the user is allowed to sign in to your app.

After the user signs in, your app can provide the option to verify the contact method that wasn't verified during sign-up. To verify this second method, your app calls the `VerifyUserAttribute` API action.

Note that this action requires an `AccessToken` parameter, and Amazon Cognito only provides access tokens for authenticated users. Therefore, you can verify the second contact method only after the user signs in.

If you require your users to verify both email addresses and phone numbers, do the following:

1. Configure your user pool to allow users to verify email address or phone numbers.
2. In the sign-up flow for your app, require users to provide both an email address and a phone number. Call the `SignUp` API action, and provide the email address and phone number for the `UserAttributes` parameter. At this point, Amazon Cognito sends a verification code to the user's phone.
3. In your app interface, present a confirmation page where the user enters the verification code. Confirm the user by calling the `ConfirmSignUp` API action. At this point, the user's status is CONFIRMED, and the user's phone number is verified, but the email address is not verified.
4. Present the sign-in page, and authenticate the user by calling the `InitiateAuth` API action. After the user is authenticated, Amazon Cognito returns an access token to your app.
5. Call the `GetUserAttributeVerificationCode` API action. Specify the following parameters in the request:
 - `AccessToken` – The access token returned by Amazon Cognito when the user signed in.
 - `AttributeName` – Specify "email" as the attribute value.

Amazon Cognito sends a verification code to the user's email address.

6. Present a confirmation page where the user enters the verification code. When the user submits the code, call the `VerifyUserAttribute` API action. Specify the following parameters in the request:
 - `AccessToken` – The access token returned by Amazon Cognito when the user signed in.
 - `AttributeName` – Specify "email" as the attribute value.
 - `Code` – The verification code that the user provided.

At this point, the email address is verified.

Allowing users to sign up in your app but confirming them as administrator

1. A user signs up in your app by entering a username, phone number and/or email address, and possibly other attributes.
2. The Amazon Cognito service receives the sign-up request from the app. After verifying that the request contains all attributes required for sign-up, the service completes the sign-up process and returns to the app that sign-up is complete, pending confirmation. At this point the user's account is in an unconfirmed state. The user cannot sign in until the account is confirmed.

3. The administrator confirms the user's account, either in the Amazon Cognito console (by finding the user account in the **Users** tab and choosing the **Confirm** button) or in the CLI (by using the `admin-confirm-sign-up` command). Both the **Confirm** button and the `admin-confirm-sign-up` command use the `AdminConfirmSignUp` API to perform the confirmation.
4. At this point the user's account is in a confirmed state, and the user can sign in.

Computing SecretHash values

The following Amazon Cognito User Pools APIs have a `SecretHash` parameter:

- [ConfirmForgotPassword](#)
- [ConfirmSignUp](#)
- [ForgotPassword](#)
- [ResendConfirmationCode](#)
- [SignUp](#)

The `SecretHash` value is a Base 64-encoded keyed-hash message authentication code (HMAC) calculated using the secret key of a user pool client and username plus the client ID in the message. The following pseudocode shows how this value is calculated. In this pseudocode, `+` indicates concatenation, `HMAC_SHA256` represents a function that produces an HMAC value using HmacSHA256, and `Base64` represents a function that produces Base-64-encoded version of the hash output.

```
Base64 ( HMAC_SHA256 ( "Client Secret Key", "Username" + "Client Id" ) )
```

For a detailed overview of how to calculate and use the `SecretHash` parameter, see [How do I troubleshoot "Unable to verify secret hash for client <client-id>" errors from my Amazon Cognito user pools API?](#) in the AWS Knowledge Center.

You can use the following code example in your server-side Java application code:

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

public static String calculateSecretHash(String userPoolClientId, String
userPoolClientSecret, String userName) {
    final String HMAC_SHA256_ALGORITHM = "HmacSHA256";

    SecretKeySpec signingKey = new SecretKeySpec(
        userPoolClientSecret.getBytes(StandardCharsets.UTF_8),
        HMAC_SHA256_ALGORITHM);

    try {
        Mac mac = Mac.getInstance(HMAC_SHA256_ALGORITHM);
        mac.init(signingKey);
        mac.update(userName.getBytes(StandardCharsets.UTF_8));
        byte[] rawHmac = mac.doFinal(userPoolClientId.getBytes(StandardCharsets.UTF_8));
        return Base64.getEncoder().encodeToString(rawHmac);
    } catch (Exception e) {
        throw new RuntimeException("Error while calculating ");
    }
}
```

Confirming user accounts without verifying email or phone number

The pre sign-up Lambda trigger can be used to auto-confirm user accounts at sign-up, without requiring a confirmation code or verifying email or phone number. Users who are confirmed this way can immediately sign in without having to receive a code.

You can also mark a user's email or phone number verified through this trigger.

Note

While this approach is convenient for users when they're getting started, we recommend auto-verifying at least one of email or phone number. Otherwise the user can be left unable to recover if they forget their password.

If you don't require the user to receive and enter a confirmation code at sign-up and you don't auto-verify email and phone number in the pre sign-up Lambda trigger, you risk not having a verified email address or phone number for that user account. The user can verify the email address or phone number at a later time. However, if the user forgets his or her password and doesn't have a verified email address or phone number, the user is locked out of the account, because the forgot-password flow requires a verified email or phone number in order to send a verification code to the user.

Verifying when users change their email or phone number

When a user updates their email address or phone number in your app, Amazon Cognito immediately sends a message with a verification code to a user if you configured your user pool to automatically verify that attribute. The user must then provide the code from the verification message to your app. Your app then submits the code in a [VerifyUserAttribute](#) API request to complete verification of the new attribute value.

If your user pool doesn't require that users verify an updated email address or phone number, Amazon Cognito immediately changes the value of an updated `email` or `phone_number` attribute and marks the attribute as unverified. Your user can't sign in with an unverified email or phone number. They must complete verification of the updated value before they can use that attribute as a sign-in alias.

If your user pool requires that users verify an updated email address or phone number, Amazon Cognito leaves the attribute verified and set to its original value until your user verifies the new attribute value. If the attribute is an alias for sign-in, your user can sign in with the original attribute value until verification changes the attribute to the new value. For more information about how to configure your user pool to require users to verify updated attributes, see [Configuring email or phone verification](#).

You can use a custom message Lambda trigger to customize the verification message. For more information, see [Custom message Lambda trigger \(p. 131\)](#). When a user's email address or phone number is unverified, your app should inform the user that they must verify the attribute, and provide a button or link for users to verify their new email address or phone number.

Confirmation and verification processes for user accounts created by administrators or developers

User accounts that are created by an administrator or developer are already in the confirmed state, so users aren't required to enter a confirmation code. The invitation message that the Amazon Cognito service sends to these users includes the username and a temporary password. The user is required to change the password before signing in. For more information, see the [Customize email and SMS messages \(p. 159\)](#) in [Creating user accounts as administrator \(p. 157\)](#) and the Custom Message trigger in [Customizing user pool workflows with Lambda triggers \(p. 92\)](#).

Confirmation and verification processes for imported user accounts

User accounts that are created by using the user import feature in the AWS Management Console, CLI, or API (see [Importing users into user pools from a CSV file \(p. 171\)](#)) are already in the confirmed state, so users aren't required to enter a confirmation code. No invitation message is sent. However, imported user accounts require users to first request a code by calling the `ForgotPassword` API and then create a password using the delivered code by calling `ConfirmForgotPassword` API before they sign in. For more information, see [Requiring imported users to reset their passwords \(p. 180\)](#).

Either the user's email or phone number must be marked as verified when the user account is imported, so no verification is required when the user signs in.

Sending emails while testing your app

Amazon Cognito sends email messages to your users when they create and manage their accounts in the client app for your user pool. If you configure your user pool to require email verification, Amazon Cognito sends an email when:

- A user signs up.
- A user updates their email address.
- A user performs an action that calls the `ForgotPassword` API action.
- You create a user account as an administrator.

Depending on the action that initiates the email, the email contains a verification code or a temporary password. Your users must receive these emails and understand the message. Otherwise, they might be unable to sign in and use your app.

To ensure that emails send successfully and that the message looks correct, test the actions in your app that initiate email deliveries from Amazon Cognito. For example, by using the sign-up page in your app, or by using the `SignUp` API action, you can initiate an email by signing up with a test email address. When you test in this way, remember the following:

Important

When you use an email address to test actions that initiate emails from Amazon Cognito, don't use a fake email address (one that has no mailbox). Use a real email address that will receive the email from Amazon Cognito without creating a *hard bounce*.

A hard bounce occurs when Amazon Cognito fails to deliver the email to the recipient's mailbox, which always happens if the mailbox doesn't exist.

Amazon Cognito limits the number of emails that can be sent by AWS accounts that persistently incur hard bounces.

When you test actions that initiate emails, use one of the following email addresses to prevent hard bounces:

- An address for an email account that you own and use for testing. When you use your own email address, you receive the email that Amazon Cognito sends. With this email, you can use the verification code to test the sign-up experience in your app. If you customized the email message for your user pool, you can check that your customizations look correct.
- The mailbox simulator address, `success@simulator.amazonaws.com`. If you use the simulator address, Amazon Cognito sends the email successfully, but you're not able to view it. This option is useful when you don't need to use the verification code and you don't need to check the email message.
- The mailbox simulator address with the addition of an arbitrary label, such as `success+user1@simulator.amazonaws.com` or `success+user2@simulator.amazonaws.com`. Amazon Cognito

emails these addresses successfully, but you're not able to view the emails that it sends. This option is useful when you want to test the sign-up process by adding multiple test users to your user pool, and each test user has a unique email address.

Creating user accounts as administrator

After you create your user pool, you can create users using the AWS Management Console, as well as the AWS Command Line Interface or the Amazon Cognito API. You can create a profile for a new user in a user pool and send a welcome message with sign-up instructions to the user via SMS or email.

Developers and administrators can perform the following tasks:

- Create a new user profile by using the AWS Management Console or by calling the `AdminCreateUser` API.
- Specify the temporary password or allow Amazon Cognito to automatically generate one.
- Specify whether provided email addresses and phone numbers are marked as verified for new users.
- Specify custom SMS and email invitation messages for new users via the AWS Management Console or a Custom Message Lambda trigger. For more information, see [Customizing user pool workflows with Lambda triggers \(p. 92\)](#).
- Specify whether invitation messages are sent via SMS, email, or both.
- Resend the welcome message to an existing user by calling the `AdminCreateUser` API, specifying `RESEND` for the `MessageAction` parameter.

Note

This action cannot currently be performed using the AWS Management Console.

- Suppress the sending of the invitation message when the user is created.
- Specify an expiration time limit for the user account (up to 90 days).
- Allow users to sign themselves up or require that new users only be added by the administrator.

Authentication flow for users created by administrators or developers

The authentication flow for these users includes the extra step to submit the new password and provide any missing values for required attributes. The steps are outlined next; steps 5, 6, and 7 are specific to these users.

1. The user starts to sign in for the first time by submitting the user name and password provided to him or her.
2. The SDK calls `InitiateAuth(Username, USER_SRP_AUTH)`.
3. Amazon Cognito returns the `PASSWORD_VERIFIER` challenge with Salt & Secret block.
4. The SDK performs the SRP calculations and calls `RespondToAuthChallenge(Username, <SRP variables>, PASSWORD_VERIFIER)`.
5. Amazon Cognito returns the `NEW_PASSWORD_REQUIRED` challenge. The body of this challenge includes the user's current attributes, and any required attributes in your user pool that don't currently have a value in the user's profile. For more information, see [RespondToAuthChallenge](#).
6. The user is prompted and enters a new password and any missing values for required attributes.
7. The SDK calls `RespondToAuthChallenge(Username, <New password>, <User attributes>)`.
8. If the user requires a second factor for MFA, Amazon Cognito returns the `SMS_MFA` challenge and the code is submitted.

9. After the user has successfully changed his or her password and optionally provided attributed values or completed MFA, the user is signed in and tokens are issued.

When the user has satisfied all challenges, the Amazon Cognito service marks the user as confirmed and issues ID, access, and refresh tokens for the user. For more information, see [Using tokens with user pools \(p. 191\)](#).

Creating a new user in the AWS Management Console

You can set user password requirements, configure the invitation and verification messages sent to users, and add new users with the Amazon Cognito console.

Set a password policy and enable self-registration

Original console

The **Policies** tab has these related settings:

- Specify the required password strength.

This screenshot shows the 'What password strength do you want to require?' configuration page. It includes a 'Minimum length' input field set to 8, and four checked checkboxes for 'Require numbers', 'Require special character', 'Require uppercase letters', and 'Require lowercase letters'.

- Specify whether to allow users to sign themselves up. This option is set by default.

This screenshot shows the 'Do you want to allow users to sign themselves up?' configuration page. It includes a note: 'You can choose to only allow administrators to create users or allow users to sign themselves up.' Below is a radio button group where 'Allow users to sign themselves up' is selected.

- Specify user account expiration time limit (in days) for new accounts. The default setting is 7 days, measured from the time when the user account is created. The maximum setting is 90 days. After the account expires, the user cannot log in to the account until the administrator updates the user's profile.

Note

Once the user has logged in, the account never expires.

How quickly should temporary passwords set by administrators expire if not used?

You can choose for how long until a temporary password set by an administrator expires if the password is not used. This includes accounts created by administrators.

This screenshot shows the 'Days to expire' configuration page. It includes a 'Days to expire' input field set to 7.

New console

You can configure settings for minimum password complexity and whether users can sign up using public APIs in your user pool.

Configure a password policy

1. Navigate to the [Amazon Cognito console](#), and choose **User Pools**.
2. Choose an existing user pool from the list, or [create a user pool](#).
3. Choose the **Sign-in experience** tab and locate **Password policy**. Choose **Edit**.
4. Choose a **Password policy mode** of **Custom**.

5. Choose a **Password minimum length**. For limits to the password length requirement, see [User pools resource quotas](#).
6. Choose a **Password complexity** requirement.
7. Choose how long password set by administrators should be valid for.
8. Choose **Save changes**.

Allow self-service sign-up

1. Navigate to the [Amazon Cognito console](#), and choose **User Pools**.
2. Choose an existing user pool from the list, or [create a user pool](#).
3. Choose the **Sign-up experience** tab and locate **Self-service sign-up**. Select **Edit**.
4. Choose whether to **Enable self-registration**. Self-registration is typically used with public app clients that need to register new users in your user pool without distributing a client secret or AWS Identity and Access Management (IAM) API credentials.

Disabling self-registration

If you do not enable self-registration, new users must be created by administrative API actions using IAM API credentials or by sign-in with federated providers.

5. Choose **Save changes**.

Customize email and SMS messages

Original console

The **Message Customizations** tab includes templates for specifying custom email verification and user invitation messages.

For email verification or user invitation messages, the maximum length for the message is 2048 UTF-8 characters, including the verification code or temporary password. For SMS verification or user invitation messages, the maximum length is 140 UTF-8 characters, including the verification code or temporary password.

Verification codes are valid for 24 hours.

Do you want to customize your email verification messages?

You can choose to send a code or a clickable link and customize the message to verify email addresses. [Learn more about email verification](#).

Verification type
 Code Link

Email subject
Your verification code

Email message
Your verification code is #####.

You can customize the message above and include HTML tags, but it must include the "(####)" placeholder, which will be replaced with the code.

Do you want to customize your user invitation messages?

SMS message
Your username is (username) and temporary password is #####.

You can customize the message above and include HTML tags, but it must include the "(username)" and "(####)" placeholder, which will be replaced with the username and temporary password respectively.

Email subject
Your temporary password

Email message
Your username is (username) and temporary password is #####.

You can customize the message above and include HTML tags, but it must include the "(username)" and "(####)" placeholder, which will be replaced with the username and temporary password respectively.

New console

Customize user messages

You can customize the messages that Amazon Cognito sends to your users when you invite them to sign in, they sign up for a user account, or they sign in and are prompted for multi-factor authentication (MFA).

Note

An **Invitation message** is sent when you create a user in your user pool and invite them to sign in. Amazon Cognito sends initial sign-in information to the user's email address or phone number.

A **Verification message** is sent when a user signs up for a user account in your user pool. Amazon Cognito sends a code to the user. When the user provides the code to Amazon Cognito, they verify their contact information and confirm their account for sign-in. Verification codes are valid for 24 hours.

An **MFA message** is sent when you enable SMS MFA in your user pool, and a user that has configured SMS MFA signs in and is prompted for MFA.

1. Navigate to the [Amazon Cognito console](#), and choose **User Pools**.
2. Choose an existing user pool from the list, or [create a user pool](#).
3. Choose the **Messaging** tab and locate **Message templates**. Select **Verification messages**, **Invitation messages**, or **MFA messages** and choose **Edit**.
4. Customize the messages for the chosen message type.

Note

All variables in message templates must be included when you customize the message. If the variable, for example `{#####}`, is not included, your user will have insufficient information to complete the message action.

For more information, see [Message templates](#).

5. a. **Verification messages**

- i. Choose a **Verification type** for **Email** messages. A **Code** verification sends a numeric code that the user must enter. A **Link** verification sends a link the user can click to verify their contact information. The text in the variable for a **Link** message is displayed as hyperlink text. For example, a message template using the variable `{##Click here##}` is displayed as [Click here](#) in the email message.
- ii. Enter an **Email subject** for **Email** messages.
- iii. Enter a custom **Email message** template for **Email** messages. You can customize this template with HTML.
- iv. Enter a custom **SMS message** template for **SMS** messages.
- v. Choose **Save changes**.

- b. **Invitation messages**

- i. Enter an **Email subject** for **Email** messages.
- ii. Enter a custom **Email message** template for **Email** messages. You can customize this template with HTML.
- iii. Enter a custom **SMS message** template for **SMS** messages.
- iv. Choose **Save changes**.

- c. **MFA messages**

- i. Enter a custom **SMS message** template for **SMS** messages.
- ii. Choose **Save changes**.

Create a user

Original console

The **Users** tab in the **Users and groups** tab has a **Create user** button.

Username	Enabled	Account status	Email verified	Phone number verified	Updated	Created
[REDACTED]-7c0dc801ee66	Enabled	CONFIRMED	true	-	Nov 19, 2018 7:44:48 PM	Nov 19, 2018 7:44:48 PM
[REDACTED]-75f3de26d2b5	Enabled	CONFIRMED	true	true	Nov 7, 2018 8:59:35 PM	Nov 7, 2018 8:59:35 PM

When you choose **Create user**, a **Create user** dialog appears, where you can enter information about the new user. Only the **Username** field is required.

Note

For user accounts that you create by using the **Create user** form in the AWS Management Console, only the attributes shown in the form can be set in the AWS Management Console. Other attributes must be set by using the AWS Command Line Interface or the Amazon Cognito API, even if you have marked them as required attributes.

New console

Create a user

You can create new users for your user pool from the Amazon Cognito console. Typically, users can sign in after they set a password. To sign in with an email address, a user must verify the `email` attribute. To sign in with a phone number, the user must verify the `phone_number` attribute. To confirm accounts as an administrator, you can also use the AWS CLI or API, or create user profiles with a federated identity provider. For more information, see the [Amazon Cognito API Reference](#).

1. Navigate to the [Amazon Cognito console](#), and choose **User Pools**.
2. Choose an existing user pool from the list, or [create a user pool](#).
3. Choose the **Users** tab, and choose **Create a user**.
4. Review the **User pool sign-in and security requirements** for guidance on password requirements, available account recovery methods, and alias attributes for your user pool.

5. Choose how you want to send an **Invitation message**. Choose SMS message, email message, or both.

Note

Before you can send invitation messages, configure a sender and an AWS Region with Amazon Simple Notification Service and Amazon Simple Email Service in the **Messaging** tab of your user pool . Recipient message and data rates apply. Amazon SES bills you for email messages separately, and Amazon SNS bills you for SMS messages separately.

6. Choose a **Username** for the new user.
7. Choose if you want to **Create a password** or have Amazon Cognito **Generate a password** for the user. Any temporary password must adhere to the user pool password policy.
8. Choose **Create**.
9. Choose the **Users** tab, and choose the **User name** entry for the user. Add and edit **User attributes** and **Group memberships**. Review **User event history**.

Adding groups to a user pool

Support for groups in Amazon Cognito user pools enables you to create and manage groups, add users to groups, and remove users from groups. Use groups to create collections of users to manage their permissions or to represent different types of users. You can assign an AWS Identity and Access Management (IAM) role to a group to define the permissions for members of a group.

You can use groups to create a collection of users in a user pool, which is often done to set the permissions for those users. For example, you can create separate groups for users who are readers, contributors, and editors of your website and app. Using the IAM role associated with a group, you can also set different permissions for those different groups so that only contributors can put content into Amazon S3 and only editors can publish content through an API in Amazon API Gateway.

You can create and manage groups in a user pool from the AWS Management Console, the APIs, and the CLI. As a developer (using AWS credentials), you can create, read, update, delete, and list the groups for a user pool. You can also add users and remove users from groups.

There is no additional cost for using groups within a user pool. See [Amazon Cognito Pricing](#) for more information.

You can see this feature used in the [SpaceFinder](#) reference app.

Assigning IAM roles to groups

You can use groups to control permissions to your resources using an IAM role. IAM roles include trust policies and permission policies. The role [trust](#) policy specifies who can use the role. The [permissions](#) policies specify the actions and resources that your group members can access. When you create an IAM role, set up the role trust policy to allow your group's users to assume the role. In the role permissions policies, specify the permissions that you want your group to have.

When you create a group in Amazon Cognito, you specify an IAM role by providing the role's [ARN](#). When group members sign in using Amazon Cognito, they can receive temporary credentials from the identity pools. Their permissions are determined by the associated IAM role.

Individual users can be in multiple groups. As a developer, you have the following options for automatically choosing the IAM role when a user is in multiple groups:

- You can assign precedence values to each group. The group with the better (lower) precedence will be chosen and its associated IAM role will be applied.

- Your app can also choose from among the available roles when requesting AWS credentials for a user through an identity pool, by specifying a role ARN in the [GetCredentialsForIdentity](#) `CustomRoleARN` parameter. The specified IAM role must match a role that is available to the user.

Assigning precedence values to groups

A user can belong to more than one group. In the user's ID token, the `cognito:groups` claim contains the list of all the groups a user belongs to. The `cognito:roles` claim contains the list of roles corresponding to the groups.

Because a user can belong to more than one group, each group can be assigned a precedence. This is a non-negative number that specifies the precedence of this group relative to the other groups that a user belongs to in the user pool. Zero is the top precedence value. Groups with lower precedence values take precedence over groups with higher or null precedence values. If a user belongs to two or more groups, the group with the lowest precedence value will have its IAM role applied to the `cognito:preferred_role` claim in the user's ID token.

Two groups can have the same precedence value. If this happens, neither group takes precedence over the other. If two groups with the same precedence value have the same role ARN, that role is used in the `cognito:preferred_role` claim in ID tokens for users in each group. If the two groups have different role ARNs, the `cognito:preferred_role` claim is not set in users' ID tokens.

Using groups to control permission with Amazon API Gateway

You can use groups in a user pool to control permission with Amazon API Gateway. The groups that a user is a member of are included in both the ID token and access token from a user pool in the `cognito:groups` claim. You can submit ID or access tokens with requests to Amazon API Gateway and use an Amazon Cognito user pool authorizer for a REST API. For more information, see [Control access to a REST API using Amazon Cognito user pools as authorizer](#) in the [API Gateway Developer Guide](#).

You can also authorize access to an Amazon API Gateway HTTP API with a custom JWT authorizer. For more information, see [Controlling access to HTTP APIs with JWT authorizers](#) in the [API Gateway Developer Guide](#).

Limitations on groups

User groups are subject to the following limitations:

- The number of groups you can create is limited by the [Amazon Cognito service limits \(p. 406\)](#).
- Groups cannot be nested.
- You cannot search for users in a group.
- You cannot search for groups by name, but you can list groups.
- Only groups with no members can be deleted.

Creating a new group in the AWS Management Console

Use the following procedure to create a new group.

Original console

The **Groups** tab in the **Users and groups** tab has a **Create group** button.



When you choose **Create group**, a **Create group** form appears. You'll enter your group's information into this form. Only the **Name** field is required. If you are integrating a user pool with an identity pool, the **IAM role** setting determines which role is assigned in the user's ID token if the identity pool is configured to choose the role from the token. If you don't have roles already defined, choose **Create new role**. If you have more than one group, and your users can be assigned to more than one group, you can set a **Precedence** value for each group. The precedence value can be any non-negative integer. Zero is the top precedence value.

New console

To create a new group

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **User Pools**.
3. Choose an existing user pool from the list.
4. Choose the **Groups** tab, and then choose **Create a group**.
5. On the **Create a group** page, in **Group name**, enter a friendly name for your new group.
6. You can optionally provide additional information about this group using any of the following fields:
 - **Description** - Enter details about what this new group will be used for.
 - **Precedence** - Amazon Cognito evaluates and applies all group permissions for a given user based on which groups that they belong to has a lower precedence value. The group with the lower precedence will be chosen and its associated IAM role will be applied. For more information, see [Assigning precedence values to groups \(p. 163\)](#).
 - **IAM role** - You can assign an IAM role to your group when you need to control permissions to your resources. If you are integrating a user pool with an identity pool, the **IAM role** setting determines which role is assigned in the user's ID token if the identity pool is configured to choose the role from the token. For more information, see [Assigning IAM roles to groups \(p. 162\)](#).
 - **Add users to this group** - Add existing users as members of this group after it is created.
7. Choose **Create** to confirm.

Managing and searching for user accounts

Once you create your user pool, you can view and manage users using the AWS Management Console, as well as the AWS Command Line Interface or the Amazon Cognito API. This topic describes how you can view and search for users using the AWS Management Console.

Viewing user attributes

Use the following procedure to view user attributes in the Amazon Cognito console.

Original console

There are a number of operations you can perform in the AWS Management Console:

- You can view the **Pool details** and edit user pool attributes, password policies, MFA settings, apps, and triggers. For more information, see [User pools reference \(AWS Management Console\) \(p. 205\)](#).
- You can view the users in your user pool and drill down for more details.
- You can also view the details for an individual user in your user pool.
- You can also search for a user in your user pool.

To view user attributes

1. From the Amazon Cognito home page in the AWS Management Console, choose **Manage user pools**.
2. Choose your user pool from the **Your User Pools** page.
3. Choose **User and Groups** to view user information.
4. Choose a user name to show more information about an individual user. From this screen, you can perform any of the following actions:
 - **Add user to group**
 - **Reset user password**
 - **Confirm user**
 - **Enable or disable MFA**
 - **Delete user**

New console

To view user attributes

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **User Pools**.
3. Choose an existing user pool from the list.
4. Choose the **Users** tab, and then select a user in the list.
5. On the user details page, under **User attributes**, you can view which attributes are associated with the user.

Resetting a user's password

Use the following procedure to reset a user's password in the Amazon Cognito console.

Original console

To reset a user's password

1. From the Amazon Cognito home page in the AWS Management Console, choose **Manage user pools**.
2. Choose your user pool from the **Your User Pools** page.
3. Choose **User and Groups** to view user information.
4. Choose the user in the list for which you want to reset their password.

The **Reset user password** action immediately sends a confirmation code to the user, and disables the user's current password by changing the user state to **RESET_REQUIRED**. The **Enable MFA** action results in a confirmation code being sent to the user when the user tries to log in. The **Reset user password** code is valid for 1 hour. The MFA code is valid for 3 minutes.

New console

To reset a user's password

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **User Pools**.
3. Choose an existing user pool from the list.
4. Choose the **Users** tab, and then select a user in the list.
5. On the user details page, choose **Actions, Reset password**.
6. In the **Reset password** dialog, review the information and when ready, choose **Reset**.

This action immediately results in a confirmation code being sent to the user and disables the user's current password by changing the user state to **RESET_REQUIRED**. The **Reset password** code is valid for 1 hour.

Searching user attributes

If you have already created a user pool, you can search from the **Users** panel in the AWS Management Console. You can also use the Amazon Cognito [ListUsers API](#), which accepts a **Filter** parameter.

You can search for any of the following standard attributes. Custom attributes are not searchable.

- `username` (case-sensitive)
- `email`
- `phone_number`
- `name`
- `given_name`
- `family_name`
- `preferred_username`
- `cognito:user_status` (called **Status** in the Console) (case-insensitive)
- `status` (called **Enabled** in the Console) (case-sensitive)
- `sub`

Note

You can also list users with a client-side filter. The server-side filter matches no more than 1 attribute. For advanced search, use a client-side filter with the `--query` parameter of the `list-users` action in the AWS Command Line Interface. When you use a client-side filter, `ListUsers`

returns a paginated list of zero or more users. You can receive multiple pages in a row with zero results. Repeat the query with each pagination token that is returned until you receive a null pagination token value, then review the combined result.

For more information about server-side and client-side filtering, see [Filtering AWS CLI output](#) in the AWS Command Line Interface User Guide.

Searching for users with the AWS Management Console

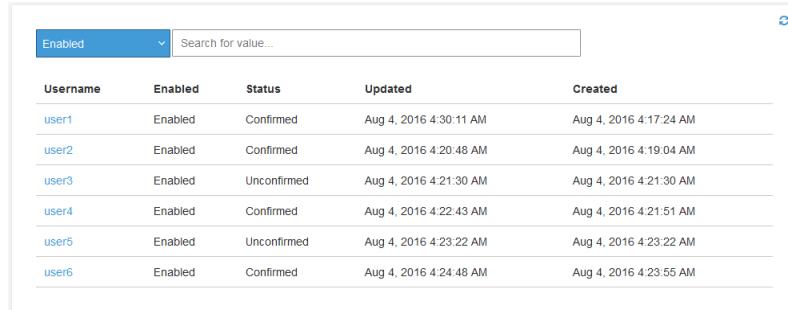
If you have already created a user pool, you can search from the **Users** panel in the AWS Management Console.

AWS Management Console searches are always prefix ("starts with") searches.

Original console

All of the following examples use the same user pool.

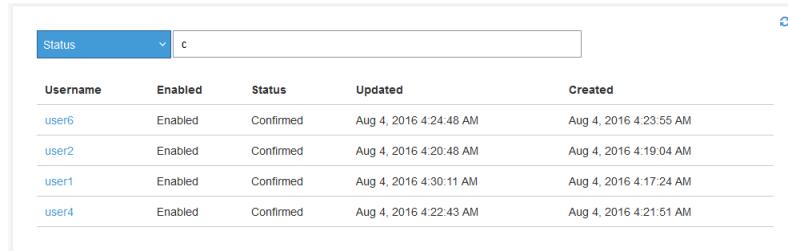
For example, if you want to list all users, leave the search box empty.



The screenshot shows a table with columns: Username, Enabled, Status, Updated, and Created. There are six rows of data:

Username	Enabled	Status	Updated	Created
user1	Enabled	Confirmed	Aug 4, 2016 4:30:11 AM	Aug 4, 2016 4:17:24 AM
user2	Enabled	Confirmed	Aug 4, 2016 4:20:48 AM	Aug 4, 2016 4:19:04 AM
user3	Enabled	Unconfirmed	Aug 4, 2016 4:21:30 AM	Aug 4, 2016 4:21:30 AM
user4	Enabled	Confirmed	Aug 4, 2016 4:22:43 AM	Aug 4, 2016 4:21:51 AM
user5	Enabled	Unconfirmed	Aug 4, 2016 4:23:22 AM	Aug 4, 2016 4:23:22 AM
user6	Enabled	Confirmed	Aug 4, 2016 4:24:48 AM	Aug 4, 2016 4:23:55 AM

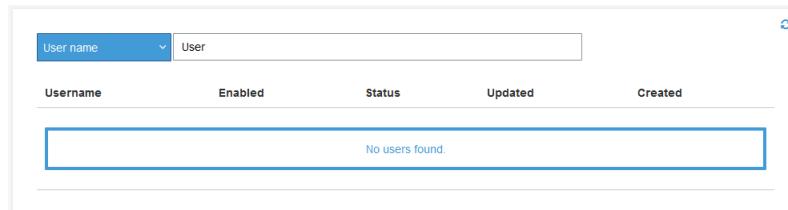
If you want to search for all confirmed users, choose **Status** from the drop-down menu. In the search box, type the first letter of the word "confirmed."



The screenshot shows a table with columns: Username, Enabled, Status, Updated, and Created. There are four rows of data:

Username	Enabled	Status	Updated	Created
user6	Enabled	Confirmed	Aug 4, 2016 4:24:48 AM	Aug 4, 2016 4:23:55 AM
user2	Enabled	Confirmed	Aug 4, 2016 4:20:48 AM	Aug 4, 2016 4:19:04 AM
user1	Enabled	Confirmed	Aug 4, 2016 4:30:11 AM	Aug 4, 2016 4:17:24 AM
user4	Enabled	Confirmed	Aug 4, 2016 4:22:43 AM	Aug 4, 2016 4:21:51 AM

Note that some attribute values are case-sensitive, such as **User name**.



The screenshot shows a table with columns: Username, Enabled, Status, Updated, and Created. The search bar at the top contains the text "User". Below the table, a message says "No users found."

Username	Enabled	Status	Updated	Created

New console

To search for a user in the Amazon Cognito console

1. Go to the [Amazon Cognito console](#). You might be prompted for your AWS credentials.
2. Choose **User Pools**.

3. Choose an existing user pool from the list.
4. Choose the **Users** tab, and then enter in the user's username in the search field. Note that some attribute values are case-sensitive (for example, **Username**).

You can also find users by adjusting the search filter to narrow the scope down to other user properties, such as **Email**, **Phone number**, or **Last name**.

Searching for users with the `ListUsers` API

To search for users from your app, use the Amazon Cognito [ListUsers API](#). This API uses the following parameters:

- **AttributesToGet**: An array of strings, where each string is the name of a user attribute to be returned for each user in the search results. If the array is empty, all attributes are returned.
- **Filter**: A filter string of the form "AttributeName Filter-TypeAttributeValue". Quotation marks within the filter string must be escaped using the backslash (\) character. For example, "family_name = \"Reddy\"". If the filter string is empty, `ListUsers` returns all users in the user pool.
- **AttributeName**: The name of the attribute to search for. You can only search for one attribute at a time.

Note

You can only search for standard attributes. Custom attributes are not searchable. This is because only indexed attributes are searchable, and custom attributes cannot be indexed.

- **Filter-Type**: For an exact match, use `=`, for example, `given_name = "Jon"`. For a prefix ("starts with") match, use `^=`, for example, `given_name ^= "Jon"`.
- **AttributeValue**: The attribute value that must be matched for each user.
- **Limit**: Maximum number of users to be returned.
- **PaginationToken**: A token to get more results from a previous search. Amazon Cognito expires the pagination token after one hour.
- **UserPoolId**: The user pool ID for the user pool on which the search should be performed.

All searches are case-insensitive. Search results are sorted by the attribute named by the `AttributeName` string, in ascending order.

Examples of using the `ListUsers` API

The following example returns all users and includes all attributes.

```
{
  "AttributesToGet": [],
  "Filter": "",
  "Limit": 10,
  "UserPoolId": "us-east-1_samplepool"
}
```

The following example returns all users whose phone numbers start with "+1312" and includes all attributes.

```
{
  "AttributesToGet": [],
  "Filter": "phone_number ^= \"+1312\\"",
```

```
        "Limit": 10,
        "UserPoolId": "us-east-1_samplepool"
    }
```

The following example returns the first 10 users whose family name is "Reddy". For each user, the search results include the user's given name, phone number, and email address. If there are more than 10 matching users in the user pool, the response includes a pagination token.

```
{
    "AttributesToGet": [
        "given_name", "phone_number", "email"
    ],
    "Filter": "family_name = \"Reddy\"",
    "Limit": 10,
    "UserPoolId": "us-east-1_samplepool"
}
```

If the previous example returns a pagination token, the following example returns the next 10 users that match the same filter string.

```
{
    "AttributesToGet": [
        "given_name", "phone_number", "email"
    ],
    "Filter": "family_name = \"Reddy\"",
    "Limit": 10,
    "PaginationToken": "pagination_token_from_previous_search",
    "UserPoolId": "us-east-1_samplepool"
}
```

Recovering user accounts

The `AccountRecoverySetting` parameter enables you to customize which method a user can use to recover their password when they call the [ForgotPassword](#) API. `ForgotPassword` sends a recovery code to a verified email or a verified phone number. The recovery code is valid for one hour. When you specify an `AccountRecoverySetting` for your user pool, Amazon Cognito chooses the code delivery destination based on the priority that you set.

When you define `AccountRecoverySetting` and a user has SMS MFA configured, SMS cannot be used as an account recovery mechanism. The priority for this setting is determined with 1 being of the highest priority. Cognito sends a verification to only one of the specified methods.

For example, `admin_only` is a value used when the administrator does not want the user to recover their account themselves, and would instead require them to contact the administrator to reset their account. You cannot use `admin_only` with any other account recovery mechanism.

If you do not specify `AccountRecoverySetting`, Amazon Cognito uses the legacy mechanism to determine the password recovery method. In this case, Cognito uses a verified phone first. If the verified phone is not found for the user, Cognito falls back and will use verified email next.

For more information about `AccountRecoverySetting`, see [CreateUserPool](#) and [UpdateUserPool](#) in the *Amazon Cognito Identity Provider API Reference*.

Forgot password behavior

In a given hour, we allow between 5 and 20 attempts for a user to request or enter a password reset code as part of forgot-password and confirm-forgot-password actions. The exact value depends on the risk parameters associated with the requests. Please note that this behavior is subject to change.

Importing users into a user pool

There are two ways you can import or migrate users from your existing user directory or user database into Amazon Cognito user pools. You can migrate users when they sign-in using Amazon Cognito for the first time with a user migration Lambda trigger. With this approach, users can continue using their existing passwords and will not have to reset them after the migration to your user pool. Alternatively, you can migrate users in bulk by uploading a CSV file containing the user profile attributes for all users. The following sections describe both these approaches.

Topics

- [Importing users into user pools with a user migration Lambda trigger \(p. 170\)](#)
- [Importing users into user pools from a CSV file \(p. 171\)](#)

Importing users into user pools with a user migration Lambda trigger

With this approach, you can seamlessly migrate users from your existing user directory to user pools when a user signs in for the first time with your app or requests a password reset. Add a [Migrate user Lambda trigger \(p. 127\)](#) function to your user pool and it receives metadata about users who try to sign in, and returns user profile information from an external identity source. For details and example code for this Lambda trigger, including request and response parameters, see [Migrate user Lambda trigger parameters \(p. 128\)](#).

Before you start to migrate users, create a user migration Lambda function in your AWS account, and set the Lambda function as the user migration trigger in your user pool. Add an authorization policy to your Lambda function that permits only the Amazon Cognito service account principal, `cognito-idp.amazonaws.com` to invoke the Lambda function, and only in the context of your own user pool. For more information, see [Using resource-based policies for AWS Lambda \(Lambda function policies\)](#).

Sign-in process

1. The user opens your app and signs in with the Amazon Cognito user pools API or through the Amazon Cognito hosted UI. For more information about how to facilitate sign-in with Amazon Cognito APIs, see [Integrating Amazon Cognito with web and mobile apps \(p. 17\)](#).
2. Your app sends the user name and password to Amazon Cognito. If your app has a custom sign-in UI that you built with an AWS SDK, your app must use `InitiateAuth` or `AdminInitiateAuth` with the `USER_PASSWORD_AUTH` or `ADMIN_USER_PASSWORD_AUTH` flow. When your app uses one of these flows, the SDK sends the password to the server.

Note

Before you add a user migration trigger, activate the `USER_PASSWORD_AUTH` or `ADMIN_USER_PASSWORD_AUTH` flow in the settings of your app client. You must use these flows instead of the default `USER_SRP_AUTH` flow. Amazon Cognito must send a password to your Lambda function so that it can verify your user's authentication in the other directory. An SRP obscures your user's password from your Lambda function.

3. Amazon Cognito checks if the submitted user name matches a user name or alias in the user pool. You can set the user's email address, phone number, or preferred user name as an alias in your user pool. If the user doesn't exist, Amazon Cognito sends parameters, including the user name and password, to your [Migrate user Lambda trigger \(p. 127\)](#) function.
4. Your [Migrate user Lambda trigger \(p. 127\)](#) function checks for or authenticates the user with your existing user directory or user database. The function returns user attributes that Amazon Cognito stores in the user's profile in the user pool. You can return a `username` parameter only if the submitted user name matches an alias attribute. If you want users to continue to use their existing passwords, your function sets the attribute `finalUserStatus` to `CONFIRMED` in the Lambda

response. Your app must return all "response" parameters shown at [Migrate user Lambda trigger parameters \(p. 128\)](#).

Important

Do not log the entire request event object in your user migration Lambda code. This request event object includes the user's password. If you don't sanitize the logs, passwords appear in CloudWatch Logs.

5. Amazon Cognito creates the user profile in your user pool, and returns tokens to your app client.
6. Your app performs token intake, accepts the user authentication, and proceeds to the requested content.

After you migrate your users, use `USER_SRP_AUTH` for sign-in. The Secure Remote Password (SRP) protocol doesn't send the password across the network, and provides security benefits over the `USER_PASSWORD_AUTH` flow that you use during migration.

In case of errors during migration, including client device or network issues, your app receives error responses from the Amazon Cognito user pools API. When this happens, Amazon Cognito might or might not create the user account in your user pool. The user should then attempt to sign in again. If sign-in fails repeatedly, attempt to reset the user's password with the forgot-password flow in your app.

The forgot-password flow also invokes your [Migrate user Lambda trigger \(p. 127\)](#) function with a `UserMigration_ForgotPassword` event source. Because the user doesn't submit a password when they request a password reset, Amazon Cognito doesn't include a password in the event that it sends to your Lambda function. Your function can only look up the user in your existing user directory and return attributes to add to the user profile in your user pool. Amazon Cognito receives the response from your function and sends the user a password reset code by email or SMS. When the user verifies the code, they set a new password in your app. If your app has a custom UI, build pages that support the forgot-password flow. The Amazon Cognito hosted UI has built-in pages for the forgot-password flow.

Importing users into user pools from a CSV file

You can import users into an Amazon Cognito user pool. The user information is imported from a specially formatted .csv file. The import process sets values for all user attributes except **password**. Password import is not supported, because security best practices require that passwords are not available as plain text, and we don't support importing hashes. This means that your users must change their passwords the first time they sign in. So, your users will be in a `RESET_REQUIRED` state when imported using this method.

You can use a [Pre authentication Lambda trigger \(p. 108\)](#) to set an imported user's password. In your function code, set your user's password with an `AdminSetUserPassword` API request that sets the `Permanent` parameter to `true`.

Note

The creation date for each user is the time when that user was imported into the user pool. Creation date is not one of the imported attributes.

The basic steps are:

1. Create an Amazon CloudWatch Logs role in the AWS Identity and Access Management (IAM) console.
2. Create the user import .csv file.
3. Create and run the user import job.
4. Upload the user import .csv file.
5. Start and run the user import job.
6. Use CloudWatch to check the event log.
7. Require the imported users to reset their passwords.

Topics

- [Creating the CloudWatch Logs IAM role \(AWS CLI, API\) \(p. 172\)](#)
- [Creating the user import .csv file \(p. 173\)](#)
- [Creating and running the Amazon Cognito user pool import job \(p. 175\)](#)
- [Viewing the user pool import results in the CloudWatch console \(p. 179\)](#)
- [Requiring imported users to reset their passwords \(p. 180\)](#)

Creating the CloudWatch Logs IAM role (AWS CLI, API)

If you're using the Amazon Cognito CLI or API, then you need to create a CloudWatch IAM role. The following procedure describes how to enable Amazon Cognito to record information in CloudWatch Logs about your user pool import job.

Note

You don't need to use this procedure if you are using the [Amazon Cognito console](#), because the console creates the role for you.

To create the CloudWatch Logs IAM Role for user pool import (AWS CLI, API)

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane of the IAM console, choose **Roles**.
3. For a new role, choose **Create role**.
4. For **Select type of trusted entity**, choose **AWS service**.
5. In **Common use cases**, choose **EC2**.
6. Choose **Next: Permissions**.
7. In **Attach permissions policy**, choose **Create policy** to open a new browser tab and create a new policy from scratch.
8. On the **Create policy** page, choose the **JSON** tab.
9. In the **JSON** tab, copy and paste the following text as your role access policy, replacing any existing text:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "logs:CreateLogGroup",  
                "logs:CreateLogStream",  
                "logs:DescribeLogStreams",  
                "logs:PutLogEvents"  
            ],  
            "Resource": [  
                "arn:aws:logs:REGION:ACCOUNT:log-group:/aws/cognito/*"  
            ]  
        }  
    ]  
}
```

10. Choose **Review policy**. Add a name and optional description and choose **Create policy**.

After you create the policy, close that browser tab and return to your original tab.

11. In **Attach Policy**, choose **Next: Tags**.

12. (Optional) In **Tags**, add metadata to the role by entering tags as key–value pairs.
13. Choose **Next: Review**.
14. In **Review**, enter a **Role Name**.
15. (Optional) Enter a **Role Description**.
16. Choose **Create role**.
17. In the navigation pane of the IAM console, choose **Roles**.
18. In **Roles**, choose the role you created.
19. In **Summary**, choose the **Trust relationships** tab.
20. In the **Trust relationships** tab, choose **Edit trust relationship**.
21. Copy and paste the following trust relationship text into the **Policy Document** text box, replacing any existing text:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "cognito-idp.amazonaws.com"  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}
```

22. Choose **Update Trust Policy**. You are now finished creating the role.
23. Note the role ARN. You need this later when you're creating an import job.

Creating the user import .csv file

Before you can import your existing users into your user pool, you must create a `.csv` file that serves as the input. To do this, download the user import `.csv` header information, and then you edit the file to match the formatting requirements outlined in [Formatting the .csv file \(p. 174\)](#).

Downloading the .csv file header (console)

1. Navigate to the [Amazon Cognito console](#), choose **Manage User Pools**, and then choose the user pool into which you are importing the users.
2. Choose the **Users** tab.
3. Choose **Import users**.
4. Choose **Download CSV header** to get a `.csv` file containing the header row that you must include in your `.csv` file.

Downloading the .csv file header (AWS CLI)

To get a list of the correct headers, run the following CLI command, where `USER_POOL_ID` is the user pool identifier for the user pool you'll import users into:

```
aws cognito-idp get-csv-header --user-pool-id "USER_POOL_ID"
```

Sample response:

```
{
    "CSVHeader": [
        "name",
        "given_name",
        "family_name",
        "middle_name",
        "nickname",
        "preferred_username",
        "profile",
        "picture",
        "website",
        "email",
        "email_verified",
        "gender",
        "birthdate",
        "zoneinfo",
        "locale",
        "phone_number",
        "phone_number_verified",
        "address",
        "updated_at",
        "cognito:mfa_enabled",
        "cognito:username"
    ],
    "UserPoolId": "USER_POOL_ID"
}
```

Formatting the .csv file

The downloaded user import .csv header file looks like this:

```
cognito:username,name,given_name,family_name,middle_name,nickname,preferred_username,profile,picture,we
```

You'll need to edit your .csv file so that it includes this header and the attribute values for your users, and is formatted according to the following rules:

Note

For more information about attribute values, such as proper format for phone numbers, see [User pool attributes \(p. 206\)](#).

- The first row in the file is the downloaded header row, which contains the user attribute names.
- The order of columns in the .csv file doesn't matter.
- Each row after the first row contains the attribute values for a user.
- All columns in the header must be present, but you don't need to provide values in every column.
- The following attributes are required:
 - **cognito:username**
 - **cognito:mfa_enabled**
 - **email_verified** or **phone_number_verified**
 - **email** (if **email_verified** is true)
 - **phone_number** (if **phone_number_verified** is true)
 - Any attributes that you marked as required when you created the user pool
- The user pool must have at least one auto-verified attribute, either **email_verified** or **phone_number_verified**. At least one of the auto-verified attributes must be true for each user. If the user pool has no auto-verified attributes, the import job will not start. If the user pool only has one auto-verified attribute, that attribute must be verified for each user. For example, if the user pool has only **phone_number** as an auto-verified attribute, the **phone_number_verified** value must be true for each user.

Note

In order for users to reset their passwords, they must have a verified email or phone number.

Amazon Cognito sends a message containing a reset password code to the email or phone number specified in the .csv file. If the message is sent to the phone number, it is sent by SMS message.

- Attribute values that are strings should *not* be in quotation marks.
- If an attribute value contains a comma, you must put a backslash (\) before the comma. This is because the fields in a .csv file are separated by commas.
- The .csv file contents should be in UTF-8 format without byte order mark.
- The **cognito:username** field is required and must be unique within your user pool. It can be any Unicode string. However, it cannot contain spaces or tabs.
- The **birthdate** values, if present, must be in the format *mm/dd/yyyy*. This means, for example, that a birthdate of February 1, 1985 must be encoded as **02/01/1985**.
- The **cognito:mfa_enabled** field is required. If you've set multi-factor authentication (MFA) to be required in your user pool, this field must be **true** for all users. If you've set MFA to be off, this field must be **false** for all users. If you've set MFA to be optional, this field can be either **true** or **false**, but it can't be empty.
- The maximum row length is 16,000 characters.
- The maximum .csv file size is 100 MB.
- The maximum number of rows (users) in the file is 500,000. This maximum doesn't include the header row.
- The **updated_at** field value is expected to be epoch time in seconds, for example: **1471453471**.
- Any leading or trailing white space in an attribute value will be trimmed.

A complete sample user import .csv file looks like this:

```
cognito:username,name,given_name,family_name,middle_name,nickname,preferred_username,profile,picture,we
John,,John,Doe,,,,,,johndoe@example.com,TRUE,,02/01/1985,,,+12345550100,TRUE,123 Any
Street,,FALSE
Jane,,Jane,Roe,,,,,,janeroe@example.com,TRUE,,01/01/1985,,,+12345550199,TRUE,100 Main
Street,,FALSE
```

Creating and running the Amazon Cognito user pool import job

This section describes how to create and run the user pool import job by using the Amazon Cognito console and the AWS Command Line Interface.

Topics

- [Importing users from a .csv file \(console\) \(p. 175\)](#)
- [Importing users \(AWS CLI\) \(p. 176\)](#)

Importing users from a .csv file (console)

The following procedure describes how to import the users from the .csv file.

To import users from the .csv file (console)

1. Choose **Create import job**.
2. Enter a **Job name**. Job names can contain uppercase and lowercase letters (a-z, A-Z), numbers (0-9), and the following special characters: + = , . @ and -.

3. If this is your first time creating a user import job, the AWS Management Console will automatically create an IAM role for you. Otherwise, choose an existing role from the **IAM Role** list, or let the AWS Management Console create a new role for you.
4. Choose **Upload CSV** and select the .csv file to import users from.
5. Choose **Create job**.
6. To start the job, choose **Start**.

Importing users (AWS CLI)

The following CLI commands are available for importing users into a user pool:

- `create-user-import-job`
- `get-csv-header`
- `describe-user-import-job`
- `list-user-import-jobs`
- `start-user-import-job`
- `stop-user-import-job`

To get the list of command line options for these commands, use the `help` command line option. For example:

```
aws cognito-idp get-csv-header help
```

Creating a user import job

After you create your .csv file, create a user import job by running the following CLI command, where **JOB_NAME** is the name you're choosing for the job, **USER_POOL_ID** is the user pool ID for the user pool into which the new users will be added, and **ROLE_ARN** is the role ARN you received in [Creating the CloudWatch Logs IAM role \(AWS CLI, API\) \(p. 172\)](#):

```
aws cognito-idp create-user-import-job --job-name "JOB_NAME" --user-pool-id "USER_POOL_ID"  
--cloud-watch-logs-role-arn "ROLE_ARN"
```

The **PRE_SIGNED_URL** returned in the response is valid for 15 minutes. After that time, it will expire and you must create a new user import job to get a new URL.

Example Sample response:

```
{  
    "UserImportJob": {  
        "Status": "Created",  
        "SkippedUsers": 0,  
        "UserPoolId": "USER_POOL_ID",  
        "ImportedUsers": 0,  
        "JobName": "JOB_NAME",  
        "JobId": "JOB_ID",  
        "PreSignedUrl": "PRE_SIGNED_URL",  
        "CloudWatchLogsRoleArn": "ROLE_ARN",  
        "FailedUsers": 0,  
        "CreationDate": 1470957431.965  
    }  
}
```

Status values for a user import job

In the responses to your user import commands, you'll see one of the following **Status** values:

- **Created** - The job was created but not started.
- **Pending** - A transition state. You have started the job, but it has not begun importing users yet.
- **InProgress** - The job has started, and users are being imported.
- **Stopping** - You have stopped the job, but the job has not stopped importing users yet.
- **Stopped** - You have stopped the job, and the job has stopped importing users.
- **Succeeded** - The job has completed successfully.
- **Failed** - The job has stopped due to an error.
- **Expired** - You created a job, but did not start the job within 24-48 hours. All data associated with the job was deleted, and the job can't be started.

Uploading the .csv file

Use the following `curl` command to upload the .csv file containing your user data to the presigned URL that you obtained from the response of the `create-user-import-job` command.

```
curl -v -T "PATH_TO_CSV_FILE" -H "x-amz-server-side-encryption:aws:kms" "PRE_SIGNED_URL"
```

In the output of this command, look for the phrase "We are completely uploaded and fine". This phrase indicates that the file was uploaded successfully.

Describing a user import job

To get a description of your user import job, use the following command, where `USER_POOL_ID` is your user pool ID, and `JOB_ID` is the job ID that was returned when you created the user import job.

```
aws cognito-idp describe-user-import-job --user-pool-id "USER_POOL_ID" --job-id "JOB_ID"
```

Example Sample response:

```
{
    "UserImportJob": {
        "Status": "Created",
        "SkippedUsers": 0,
        "UserPoolId": "USER_POOL_ID",
        "ImportedUsers": 0,
        "JobName": "JOB_NAME",
        "JobId": "JOB_ID",
        "PreSignedUrl": "PRE_SIGNED_URL",
        "CloudWatchLogsRoleArn": "ROLE_ARN",
        "FailedUsers": 0,
        "CreationDate": 1470957431.965
    }
}
```

In the preceding sample output, the `PRE_SIGNED_URL` is the URL that you uploaded the .csv file to. The `ROLE_ARN` is the CloudWatch Logs role ARN that you received when you created the role.

Listing your user import jobs

To list your user import jobs, use the following command:

```
aws cognito-idp list-user-import-jobs --user-pool-id "USER_POOL_ID" --max-results 2
```

Example Sample response:

```
{
    "UserImportJobs": [
        {
            "Status": "Created",
            "SkippedUsers": 0,
            "UserPoolId": "USER_POOL_ID",
            "ImportedUsers": 0,
            "JobName": "JOB_NAME",
            "JobId": "JOB_ID",
            "PreSignedUrl": "PRE_SIGNED_URL",
            "CloudWatchLogsRoleArn": "ROLE_ARN",
            "FailedUsers": 0,
            "CreationDate": 1470957431.965
        },
        {
            "CompletionDate": 1470954227.701,
            "StartDate": 1470954226.086,
            "Status": "Failed",
            "UserPoolId": "USER_POOL_ID",
            "ImportedUsers": 0,
            "SkippedUsers": 0,
            "JobName": "JOB_NAME",
            "CompletionMessage": "Too many users have failed or been skipped during the import.",
            "JobId": "JOB_ID",
            "PreSignedUrl": "PRE_SIGNED_URL",
            "CloudWatchLogsRoleArn": "ROLE_ARN",
            "FailedUsers": 5,
            "CreationDate": 1470953929.313
        }
    ],
    "PaginationToken": "PAGINATION_TOKEN"
}
```

Jobs are listed in chronological order from last created to first created. The `PAGINATION_TOKEN` string after the second job indicates that there are additional results for this list command. To list the additional results, use the `--pagination-token` option as follows:

```
aws cognito-idp list-user-import-jobs --user-pool-id "USER_POOL_ID" --max-results 10 --pagination-token "PAGINATION_TOKEN"
```

Starting a user import job

To start a user import job, use the following command:

```
aws cognito-idp start-user-import-job --user-pool-id "USER_POOL_ID" --job-id "JOB_ID"
```

Only one import job can be active at a time per account.

Example Sample response:

```
{
    "UserImportJob": {
        "Status": "Pending",
        "StartDate": 1470957851.483,
```

```
        "UserPoolId": "USER_POOL_ID",  
        "ImportedUsers": 0,  
        "SkippedUsers": 0,  
        "JobName": "JOB_NAME",  
        "JobId": "JOB_ID",  
        "PreSignedUrl": "PRE_SIGNED_URL",  
        "CloudWatchLogsRoleArn": "ROLE_ARN",  
        "FailedUsers": 0,  
        "CreationDate": 1470957431.965  
    }  
}
```

Stopping a user import job

To stop a user import job while it is in progress, use the following command. After you stop the job, it cannot be restarted.

```
aws cognito-idp stop-user-import-job --user-pool-id "USER_POOL_ID" --job-id "JOB_ID"
```

Example Sample response:

```
{  
    "UserImportJob": {  
        "CompletionDate": 1470958050.571,  
        "StartDate": 1470958047.797,  
        "Status": "Stopped",  
        "UserPoolId": "USER_POOL_ID",  
        "ImportedUsers": 0,  
        "SkippedUsers": 0,  
        "JobName": "JOB_NAME",  
        "CompletionMessage": "The Import Job was stopped by the developer.",  
        "JobId": "JOB_ID",  
        "PreSignedUrl": "PRE_SIGNED_URL",  
        "CloudWatchLogsRoleArn": "ROLE_ARN",  
        "FailedUsers": 0,  
        "CreationDate": 1470957972.387  
    }  
}
```

Viewing the user pool import results in the CloudWatch console

You can view the results of your import job in the Amazon CloudWatch console.

Topics

- [Viewing the results \(p. 179\)](#)
- [Interpreting the results \(p. 180\)](#)

Viewing the results

The following steps describe how to view the user pool import results.

To view the results of the user pool import

1. Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. Choose **Logs**.
3. Choose the log group for your user pool import jobs. The log group name is in the form /aws/cognito/userpools/**USER_POOL_ID**/**USER_POOL_NAME**.

4. Choose the log for the user import job you just ran. The log name is in the form *JOB_ID/JOB_NAME*. The results in the log refer to your users by line number. No user data is written to the log. For each user, a line similar to the following appears:
 - [SUCCEEDED] Line Number 5956 – The import succeeded.
 - [SKIPPED] Line Number 5956 – The user already exists.
 - [FAILED] Line Number 5956 – The User Record does not set any of the auto verified attributes to true. (Example: email_verified to true).

Interpreting the results

Successfully imported users have their status set to "PasswordReset".

In the following cases, the user will not be imported, but the import job will continue:

- No auto-verified attributes are set to true.
- The user data doesn't match the schema.
- The user couldn't be imported due to an internal error.

In the following cases, the import job will fail:

- The Amazon CloudWatch Logs role cannot be assumed, doesn't have the correct access policy, or has been deleted.
- The user pool has been deleted.
- Amazon Cognito is unable to parse the .csv file.

Requiring imported users to reset their passwords

The first time each imported user signs in, he or she is required to enter a new password as follows:

Requiring imported users to reset their passwords

1. The user attempts to sign in, providing user name and password (via `InitiateAuth`).
2. Amazon Cognito returns `NotAuthorizedException` when `PreventUserExistenceErrors` is enabled. Otherwise, it returns `PasswordResetRequiredException`.
3. The app should direct the user into the `ForgotPassword` flow as outlined in the following procedure:
 - a. The app calls `ForgotPassword(user name)`.
 - b. Amazon Cognito sends a code to the verified email or phone number (depending on what you have provided in the .csv file for that user) and indicates to the app where the code was sent in the response to the `ForgotPassword` request.

Note

For sending reset password codes, it is important that your user pool has phone number or email verification turned on.

- c. The app indicates to the user that a code was sent and where the code was sent, and the app provides a UI to enter the code and a new password.
- d. The user enters the code and new password in the app.
- e. The app calls `ConfirmForgotPassword(code, password)`, which, if successful, sets the new password.
- f. The app should now direct the user to a sign-in page.

Email settings for Amazon Cognito user pools

Certain events in your user pool's client app can cause Amazon Cognito to email your users. For example, if you configure your user pool to require email verification, Amazon Cognito sends an email when a user signs up for a new account in your app or resets their password. Depending on the action that initiates the email, the email contains a verification code or a temporary password.

To handle email delivery, you can use either of the following options:

- [The default email functionality \(p. 181\)](#) that is built into the Amazon Cognito service.
- [Your Amazon Simple Email Service \(Amazon SES\) configuration \(p. 181\)](#).

These settings are reversible. You can update your user pool to switch between them.

Default email functionality

Amazon Cognito can use its default email functionality to handle email deliveries for you. When you use the default option, Amazon Cognito limits the number of emails it sends each day for your user pool. For information on service limits, see [Quotas in Amazon Cognito \(p. 406\)](#). For typical production environments, the default email limit is below the required delivery volume. To enable a higher delivery volume, you can use your Amazon SES email configuration.

When you use the default functionality, you use Amazon SES resources that are managed by AWS to send email messages. Amazon SES adds email addresses that return a [hard bounce](#) to an [account-level suppression list](#) or a [global suppression list](#). If an undeliverable email address becomes deliverable later, you can't control its removal from the suppression list while your user pool is configured to use the default functionality. An email address can remain on the AWS-managed suppression list indefinitely. To manage undeliverable email addresses, use your Amazon SES email configuration with an account-level suppression list, as described in the next section.

When you use the default email configuration, you can use either of the following email addresses as the FROM address:

- The default email address, no-reply@verificationemail.com.
- A custom email address. Before you can use your own email address, you must verify it with Amazon SES and grant Amazon Cognito permission to use this address.

Amazon SES email configuration

Your application might require a higher delivery volume than what is available with the default option. To increase the possible delivery volume, use your Amazon SES resources with your user pool to email your users. You can also [monitor your email sending activity](#) when you send email messages with your own Amazon SES configuration.

Before you can use your Amazon SES configuration, you must verify one or more email addresses, or a domain, with Amazon SES. Use a verified email address, or an address from a verified domain, as the FROM email address that you assign to your user pool. When Amazon Cognito sends email to a user, it calls Amazon SES for you and uses your email address.

When you use your Amazon SES configuration, the following conditions apply:

- The email delivery limits for your user pool are the same limits that apply to your Amazon SES verified email address in your AWS account.
- You can manage your messages to undeliverable email addresses with an account-level suppression list in Amazon SES that overrides the [global suppression list](#). When you use an account-level

suppression list, email message bounces affect the reputation of your account as a sender. For more information, see [Using the Amazon SES account-level suppression list](#) in the Amazon Simple Email Service Developer Guide.

Amazon SES email configuration Regions

When you choose the AWS Region that contains the Amazon SES resources that you want to use for Amazon Cognito email messages, you can choose the same Region as the one where you created your user pool. With Amazon Cognito user pools in some Regions, you can also use Amazon SES resources that are in the following alternate Regions: US East (N. Virginia), US West (Oregon), or Europe (Ireland).

To make your email operations faster and more reliable, use the Amazon SES configuration in the AWS Region where you created your user pool. Support for cross-Region Amazon SES configuration in Amazon Cognito provides continuity for user pool resources that you created to comply with Amazon Cognito requirements when the service launched. User pool resources that you created during that period could only use Amazon SES resources in a limited number of AWS Regions.

Note

In the AWS Management Console, you can only use Amazon SES resources in the same Region after you have switched to the new Amazon Cognito console experience.

If you create an Amazon Cognito user pools resource with the AWS Command Line Interface, API, or AWS CloudFormation, your user pool sends email messages with the Amazon SES identity that the `SourceArn` parameter of the [EmailConfigurationType](#) object specifies for your user pool. The Amazon SES identity must occupy a supported AWS Region. If your `EmailSendingAccount` is `COGNITO_DEFAULT` and you don't specify a `SourceArn` parameter, Amazon Cognito sends email messages from `no-reply@verificationemail.com` using resources in the Region where you created your user pool.

The following table shows the AWS Regions where you can use Amazon SES identities with Amazon Cognito.

Amazon Cognito user pools AWS Region	Amazon Simple Email Service supported AWS Regions
US East (N. Virginia)	US East (N. Virginia), US West (Oregon), Europe (Ireland)
US East (Ohio)	US East (Ohio), US East (N. Virginia), US West (Oregon), Europe (Ireland)
US West (N. California)	US West (N. California)
US West (Oregon)	US East (N. Virginia), US West (Oregon), Europe (Ireland)
Canada (Central)	Canada (Central), US East (N. Virginia), US West (Oregon), Europe (Ireland)
Asia Pacific (Tokyo)	Asia Pacific (Tokyo), US East (N. Virginia), US West (Oregon), Europe (Ireland)
Asia Pacific (Seoul)	Asia Pacific (Seoul), US East (N. Virginia), US West (Oregon), Europe (Ireland)
Asia Pacific (Mumbai)	Asia Pacific (Mumbai), US East (N. Virginia), US West (Oregon), Europe (Ireland)

Amazon Cognito user pools AWS Region	Amazon Simple Email Service supported AWS Regions
Asia Pacific (Singapore)	Asia Pacific (Singapore), US East (N. Virginia), US West (Oregon), Europe (Ireland)
Asia Pacific (Sydney)	Asia Pacific (Sydney), US East (N. Virginia), US West (Oregon), Europe (Ireland)
Europe (Ireland)	US East (N. Virginia), US West (Oregon), Europe (Ireland)
Europe (London)	Europe (London), US East (N. Virginia), US West (Oregon), Europe (Ireland)
Europe (Paris)	Europe (Paris)
Europe (Frankfurt)	Europe (Frankfurt), US East (N. Virginia), US West (Oregon), Europe (Ireland)
Europe (Stockholm)	Europe (Stockholm)
Middle East (Bahrain)	Middle East (Bahrain)
South America (São Paulo)	South America (São Paulo)

Configuring email for your user pool

Complete the following steps to configure the email settings for your user pool. Depending on the settings that you use, you might need IAM permissions in Amazon SES, AWS Identity and Access Management (IAM), and Amazon Cognito.

Note

You can't share the resources that you create in these steps across AWS accounts. For example, you can't configure a user pool in one account, and then use it with an Amazon SES email address in a different account. If you use Amazon Cognito in multiple accounts, repeat these steps for each account.

Step 1: Verify your email address or domain with Amazon SES

Before you configure your user pool, you must verify one or more domains or email addresses with Amazon SES if you want to do either of the following:

- Use your own email address as the FROM address
- Use your Amazon SES configuration to handle email delivery

By verifying your email address or domain, you confirm that you own it, which helps prevent unauthorized use.

For information on verifying an email address with Amazon SES, see [Verifying an Email Address](#) in the *Amazon Simple Email Service Developer Guide*. For information on verifying a domain with Amazon SES, see [Verifying domains](#).

Step 2: Move your account out of the Amazon SES sandbox

Omit this step if you are using the default Amazon Cognito email functionality.

When you first use Amazon SES in any AWS Region, it places your AWS account in the Amazon SES sandbox for that Region. Amazon SES uses the sandbox to prevent fraud and abuse. If you use your Amazon SES configuration to handle email delivery, you must move your AWS account out of the sandbox before Amazon Cognito can email your users.

In the sandbox, Amazon SES imposes restrictions on how many emails you can send and where you can send them. You can send emails only to addresses and domains that you have verified with Amazon SES, or you can send them to Amazon SES mailbox simulator addresses. While your AWS account remains in the sandbox, don't use your Amazon SES configuration for applications that are in production. In this situation, Amazon Cognito can't send messages to your users' email addresses.

To remove your AWS account from the sandbox, see [Moving out of the Amazon SES sandbox](#) in the *Amazon Simple Email Service Developer Guide*.

Step 3: Grant email permissions to Amazon Cognito

You might need to grant specific permissions to Amazon Cognito before it can email your users. The permissions that you grant, and the process that you use to grant them, depend on whether you are using the default email functionality, or your Amazon SES configuration.

To grant permissions to use the default email functionality

Omit this step only if you're using the default Amazon Cognito email functionality.

If you configure your user pool to use the default Amazon Cognito email functionality, you can use either of the following addresses as the FROM address from which Amazon Cognito emails your users:

- The default address
- A custom address, which must be a verified email address, or an email address in a verified domain, in Amazon SES

If you use a custom address, Amazon Cognito needs additional permissions to email users from that address. These permissions are granted by a *sending authorization policy* attached to the address or domain in Amazon SES. If you use the Amazon Cognito console to add a custom address to your user pool, the policy is automatically attached to the Amazon SES verified email address. However, if you configure your user pool outside of the console, such as using the AWS CLI or the Amazon Cognito API, you must attach the policy using the [Amazon SES console](#) or the [PutIdentityPolicy API](#).

Note

You can only configure a FROM address in a verified domain using the AWS CLI or the Amazon Cognito API.

A sending authorization policy allows or denies access based on the account resources that are using Amazon Cognito to invoke Amazon SES. For more information about resource-based policies, see the [IAM User Guide](#). You can also find example resource-based policies in the [Amazon SES Developer Guide](#).

Example Sending authorization policy

The following example sending authorization policy grants Amazon Cognito a limited ability to use an Amazon SES verified identity. Amazon Cognito can only send email messages when it does so on behalf of both the user pool in the `aws:SourceArn` condition and the account in the `aws:SourceAccount` condition.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "stmnt1234567891234",  
            "Effect": "Allow",  
            "Action": "ses:SendEmail",  
            "Resource": "arn:aws:ses:us-east-1:  
                REDACTED:identity/  
                REDACTED",  
            "Condition": {  
                "StringEquals": {  
                    "aws:SourceArn": "arn:aws:cognito-idp:  
                        us-east-1:  
                        REDACTED:userpool/  
                        REDACTED",  
                    "aws:SourceAccount": "  
                        REDACTED"  
                }  
            }  
        }  
    ]  
}
```

```
"Effect": "Allow",
"Principal": {
    "Service": [
        "email.cognito-idp.amazonaws.com"
    ]
},
>Action": [
    "SES:SendEmail",
    "SES:SendRawEmail"
],
"Resource": "<your SES identity ARN>",
"Condition": {
    "StringEquals": {
        "AWS:SourceAccount": "<your account number>"
    },
    "ArnLike": {
        "AWS:SourceArn": "<your identity pool ARN>"
    }
}
}
]
```

In this example, the "Sid" value is an arbitrary string that uniquely identifies the statement.

For more information about policy syntax, see [Amazon SES sending authorization policies](#) in the *Amazon Simple Email Service Developer Guide*.

For more examples, see [Amazon SES sending authorization policy examples](#) in the *Amazon Simple Email Service Developer Guide*.

Example

To grant permissions to use your Amazon SES configuration

If you configure your user pool to use your Amazon SES configuration, Amazon Cognito needs additional permissions to call Amazon SES on your behalf when it emails your users. This authorization is granted with the IAM service.

When you configure your user pool with this option, Amazon Cognito creates a *service-linked role*, which is a type of IAM role, in your AWS account. This role contains the permissions that allow Amazon Cognito to access Amazon SES and send email with your address.

Before Amazon Cognito can create this role, the IAM permissions that you use to set up your user pool must include the `iam:CreateServiceLinkedRole` action. For more information about updating permissions in IAM, see [Changing permissions for an IAM user](#) in the *IAM User Guide*.

For more information about the service-linked role that Amazon Cognito creates, see [Using service-linked roles for Amazon Cognito \(p. 360\)](#).

Step 4: Configure your user pool

Complete the following steps if you want to configure your user pool with any of the following:

- A custom FROM address that appears as the email sender
- A custom REPLY-TO address that receives the messages that your users send to your FROM address
- Your Amazon SES configuration

Omit this procedure if you want to use the default Amazon Cognito email functionality and address.

Original console

To configure your user pool to use a custom email address

1. Open the Amazon Cognito console at <https://console.aws.amazon.com/cognito>. If prompted, enter your AWS credentials.
2. Choose **Manage User Pools**.
3. On the **Your User Pools** page, choose the user pool that you want to configure.
4. In the navigation menu on the left, choose **Message customizations**.
5. If you want to use a custom FROM address, choose **Add custom FROM address** and complete the following steps:
 - a. For **SES Region**, choose the Region that contains your verified email address.
 - b. For **Source ARN**, choose your email address. Use an email address that you have verified with Amazon SES in the selected Region.
 - c. For **FROM email address**, choose your email address. You can provide only your email address, or your email address and a friendly name in the format Jane Doe <janedoe@example.com>.
6. Under **Do you want to send emails through your Amazon SES Configuration?**, choose **Yes - Use Amazon SES or No - Use Cognito (Default)**.

If you choose to use Amazon SES, Amazon Cognito creates a service-linked role after you save your changes.
7. If you want to use a custom REPLY-TO address, choose **Add custom REPLY-TO address**. Then enter the email address where you want to receive messages that your users send to your FROM address.
8. When you finish setting your email account options, choose **Save changes**.

New console

To configure your user pool to use a custom email address

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **User Pools**.
3. Choose an existing user pool from the list.
4. Choose the **Messaging** tab, locate **Email configuration**, choose **Edit**.
5. On the **Edit email configuration** page, select **Send email from Amazon SES** or **Send email with Amazon Cognito**. You can customize the **SES Region**, **Configuration Set**, and **FROM sender name** only when you choose **Send email from Amazon SES**.
6. To use a custom FROM address, complete the following steps:
 - a. Under **SES Region**, choose the Region that contains your verified email address.
 - b. Under **FROM email address**, choose your email address. Use an email address that you have verified with Amazon SES.
 - c. (Optional) Under **Configuration set**, choose a configuration set for Amazon SES to use. Making and saving this change creates a service-linked role.
 - d. (Optional) Under **FROM sender address**, enter an email address. You can provide only an email address, or an email address and a friendly name in the format Jane Doe <janedoe@example.com>.
 - e. (Optional) Under **REPLY-TO email address**, enter the email address where you want to receive messages that your users send to your FROM address.
7. Choose **Save changes**.

Related Topics

- [Customizing email verification messages \(p. 219\)](#)
- [Customizing user invitation messages \(p. 220\)](#)

SMS message settings for Amazon Cognito user pools

Some Amazon Cognito events for your user pool might cause Amazon Cognito to send SMS text messages to your users. For example, if you configure your user pool to require phone verification, Amazon Cognito sends an SMS text message when a user signs up for a new account in your app or resets their password. Depending on the action that initiates the SMS text message, the message contains a verification code, a temporary password, or a welcome message.

Amazon Cognito uses Amazon Simple Notification Service (Amazon SNS) for delivery of SMS text messages. If you are sending a text message through Amazon Cognito or Amazon SNS for the first time, Amazon SNS places you in a sandbox environment. In the sandbox environment, you can test your applications for SMS text messages. In the sandbox, messages can be sent only to verified phone numbers.

Setting up SMS messaging for the first time in Amazon Cognito user pools

Amazon Cognito uses Amazon SNS to send SMS messages to your user pools. You can also use a [Custom SMS sender Lambda trigger \(p. 141\)](#) to use your own resources to send SMS messages. The first time that you set up Amazon SNS to send SMS text messages in a particular AWS Region, Amazon SNS places your AWS account in the SMS sandbox for that Region. Amazon SNS uses the sandbox to prevent fraud and abuse and to meet compliance requirements. When your AWS account is in the sandbox, Amazon SNS imposes some [restrictions](#). For example, you can send text messages to a maximum of 10 phone numbers that you have verified with Amazon SNS. While your AWS account remains in the sandbox, do not use your Amazon SNS configuration for applications that are in production. When you're in the sandbox, Amazon Cognito can't send messages to your users' phone numbers.

To send SMS text messages to user pool users

1. Choose the AWS Region for Amazon SNS SMS messages.
2. Obtain an origination identity if you want to send SMS messages to US phone numbers.
3. Confirm that you are in the SMS sandbox.
4. Move your account out of Amazon SNS sandbox.
5. Verify phone numbers for Amazon Cognito in Amazon SNS.
6. Finish setting up the user pool in Amazon Cognito.

Step 1: Choose the AWS Region for Amazon SNS SMS messages

In some AWS Regions, you can choose the Region that contains the Amazon SNS resources that you want to use for Amazon Cognito SMS messages. In any AWS Region where Amazon Cognito is available, except for Asia Pacific (Seoul), you can use Amazon SNS resources in the AWS Region where you created your user pool. To make your SMS messaging faster and more reliable when you have a choice of Regions, use Amazon SNS resources in the same Region as your user pool.

Note

In the AWS Management Console, you can only change the Region for SMS resources after you have switched to the new Amazon Cognito console experience.

Choose a Region for SMS resources in the **Configure message delivery** step of the new user pool wizard. You can also choose **Edit** under **SMS** in the **Messaging** tab of an existing user pool.

At launch, for some AWS Regions, Amazon Cognito sent SMS messages with Amazon SNS resources in an alternate Region. To set your preferred Region, use the `SnsRegion` parameter of the `SmsConfigurationType` object for your user pool. When you programmatically create an Amazon Cognito user pools resource in an **Amazon Cognito Region** from the following table and you do not provide an `SnsRegion` parameter, your user pool can send SMS messages with Amazon SNS resources in a legacy **Amazon SNS Region**.

Amazon Cognito user pools in the Asia Pacific (Seoul) AWS Region must use your Amazon SNS configuration in the Asia Pacific (Tokyo) Region.

Amazon SNS sets the spending quota for all new accounts at \$1.00 (USD) per month. You might have increased your spend limit in an AWS Region that you use with Amazon Cognito. Before you change the AWS Region for Amazon SNS SMS messages, open a quota increase case in the AWS Support Center to increase your limit in the new Region. For more information, see [Requesting increases to your monthly SMS spending quota for Amazon SNS](#) in the *Amazon Simple Notification Service Developer Guide*.

You can send SMS messages for any **Amazon Cognito Region** in the following table with Amazon SNS resources in the corresponding **Amazon SNS Region**.

Amazon Cognito Region	Amazon SNS Region
US East (Ohio)	US East (Ohio), US East (N. Virginia)
Asia Pacific (Mumbai)	Asia Pacific (Mumbai), Asia Pacific (Singapore)
Canada (Central)	Canada (Central), US East (N. Virginia)
Europe (Frankfurt)	Europe (Frankfurt), Europe (Ireland)
Europe (London)	Europe (London), Europe (Ireland)
Asia Pacific (Seoul)	Asia Pacific (Tokyo)
US East (N. Virginia)	US East (N. Virginia)
US West (N. California)	US West (N. California)
US West (Oregon)	US West (Oregon)
Asia Pacific (Singapore)	Asia Pacific (Singapore)
Asia Pacific (Sydney)	Asia Pacific (Sydney)
Asia Pacific (Tokyo)	Asia Pacific (Tokyo)
Europe (Ireland)	Europe (Ireland)
Europe (Paris)	Europe (Paris)
Europe (Stockholm)	Europe (Stockholm)
Middle East (Bahrain)	Middle East (Bahrain)
South America (São Paulo)	South America (São Paulo)

Step 2: Obtain an origination identity to send SMS messages to US phone numbers

If you plan to send SMS text messages to US phone numbers, you must obtain an origination identity, regardless of whether you build an SMS sandbox testing environment, or a production environment.

Starting June 1, 2021, US carriers require an origination identity to send messages to US phone numbers. If you don't already have an origination identity, you must get one. To learn how to obtain an origination identity, see [Requesting a number](#) in the *Amazon Pinpoint User Guide*.

If you operate in the following AWS Regions, you must open an AWS Support ticket to obtain an origination identity. For instructions, see [Requesting support for SMS messaging](#) in the *Amazon Simple Notification Service Developer Guide*.

- US East (Ohio)
- Europe (Stockholm)
- Middle East (Bahrain)
- Europe (Paris)
- South America (São Paulo)
- US West (N. California)

Step 3: Confirm that you are in the SMS sandbox

Use the following procedure to confirm that you are in the SMS sandbox. Repeat for each AWS Region where you have production Amazon Cognito user pools.

[Review SMS sandbox status in the Amazon Cognito console](#)

Original console

To confirm that you are in the SMS sandbox

1. Open the Amazon Cognito console at <https://console.aws.amazon.com/cognito>. If prompted, enter your AWS credentials.
2. [Create a new user pool \(p. 23\)](#) or [edit an existing user pool \(p. 150\)](#).
3. If your account is in the SMS sandbox, you will see the following message in Amazon Cognito:

You are currently in a Sandbox environment in [Amazon SNS](#).

If you don't see this message, then someone has set up SMS messages in your account already. Skip to [Step 6: Complete user pool setup in Amazon Cognito \(p. 191\)](#).

4. In the message to open the Amazon SNS console in a new tab, choose the [Amazon SNS](#) link.
5. Verify that you are in the sandbox environment. The console message indicates your sandbox status and AWS Region, as follows:

This account is in the SMS sandbox in US East (N. Virginia).

New console

To confirm that you are in the SMS sandbox

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.

2. Choose **User Pools**.
3. Choose an existing user pool from the list.
4. Choose the **Messaging** tab.
5. In the **SMS configuration** section, expand **Move to Amazon SNS production environment**. If your account is in the SMS sandbox, you will see the following message:

You are currently in the SMS Sandbox and cannot send SMS messages to unverified numbers.

If you don't see this message, then someone has set up SMS messages in your account already. Skip to [Step 6: Complete user pool setup in Amazon Cognito \(p. 191\)](#).

6. Choose the [Amazon SNS](#) link in the message. This opens the Amazon SNS console in a new tab.
7. Verify that you are in the sandbox environment. The console message indicates your sandbox status and AWS Region, as follows:

This account is in the SMS sandbox in US East (N. Virginia).

Step 4: Move your account out of Amazon SNS sandbox

If you are testing your app and you only need to send SMS messages to phone numbers that your administrators can verify, skip this step and proceed to step 5.

To use your app in production, move your account out of the SMS sandbox and into production. After you have configured an origination identity in the AWS Region that contains the Amazon SNS resources that you want Amazon Cognito to use, you can verify US phone numbers while your AWS account remains in the SMS sandbox. When your Amazon SNS environment is in production, you don't have to verify user phone numbers in Amazon SNS to send SMS messages to your users.

For detailed instructions, see [Moving Out of the SMS Sandbox in the Amazon Simple Notification Service Developer Guide](#).

Step 5: Verify phone numbers for Amazon Cognito in Amazon SNS

If you have moved your account out of the SMS sandbox, skip this step and proceed to step 6.

When you are in the SMS sandbox, you can send messages to any phone number that you have verified with Amazon SNS.

To verify a phone number, do the following:

1. Add a **Sandbox destination phone number** in the **Text messaging (SMS)** section of the Amazon SNS console.
2. Receive an SMS message with a code at the phone number that you provided.
3. Enter the **Verification code** from the SMS message in the Amazon SNS console.

For detailed instructions, see [Adding and verifying phone numbers in the SMS sandbox](#) in the [Amazon Simple Notification Service Developer Guide](#).

Note

Amazon SNS limits the number of destination phone numbers that you can verify while you are in the SMS sandbox. See [SMS sandbox](#) in the [Amazon Simple Notification Service Developer Guide](#).

Step 6: Complete user pool setup in Amazon Cognito

Return to the browser tab where you were [creating \(p. 23\)](#) or [editing \(p. 150\)](#) your user pool. Complete the procedure.

Using tokens with user pools

Authenticate users and grant access to resources with tokens. Tokens have claims, which are pieces of information about the user. The ID token contains claims about the identity of the authenticated user, such as name and email. The Access token contains claims about the authenticated user, a list of the user's groups, and a list of scopes.

Amazon Cognito also has tokens that you can use to get new tokens or revoke existing tokens. [Refresh a token \(p. 195\)](#) to retrieve a new ID and access tokens. [Revoke a token \(p. 196\)](#) to revoke user access that is allowed by refresh tokens.

Authenticating with tokens

When a user signs into your app, Amazon Cognito verifies the login information. If the login is successful, Amazon Cognito creates a session and returns an ID, access, and refresh token for the authenticated user. You can use the tokens to grant your users access to your own server-side resources or to the Amazon API Gateway. Or you can exchange them for temporary AWS credentials to access other AWS services.



Storing tokens

Your app must be able to store tokens of varying sizes. Token size can change for reasons including, but not limited to, additional claims, changes in encoding algorithms, and changes in encryption algorithms. When you enable token revocation in your user pool, Amazon Cognito adds additional claims to JSON web tokens, increasing their size. The new claims `origin_jti` and `jti` are added to access and ID tokens. For more information about token revocation, see [Revoking tokens](#).

Important

Best practice is to secure all tokens in transit and storage in the context of your application. Tokens can contain personally-identifying information about your users, and information about the security model that you use for your user pool.

Topics

- [Using the ID token \(p. 191\)](#)
- [Using the access token \(p. 194\)](#)
- [Using the refresh token \(p. 195\)](#)
- [Revoking tokens \(p. 197\)](#)
- [Verifying a JSON web token \(p. 198\)](#)

Using the ID token

The ID token is a [JSON web token \(JWT\)](#) that contains claims about the identity of the authenticated user, such as `name`, `email`, and `phone_number`. You can use this identity information inside your

application. The ID token can also be used to authenticate users to your resource servers or server applications. You can also use an ID token outside of the application with your web API operations. In those cases, you must verify the signature of the ID token before you can trust any claims inside the ID token. See [Verifying a JSON web token \(p. 198\)](#).

You can set the ID token expiration to any value between 5 minutes and 1 day. You can set this value per app client.

Important

When your user signs in with the hosted UI or a federated identity provider (IdP), Amazon Cognito sets session cookies that are valid for 1 hour. If you use the hosted UI or federation, and specify a minimum duration of less than 1 hour for your access and ID tokens, your users will still have a valid session until the cookie expires. If the user has tokens that expire during the one-hour session, the user can refresh their tokens without the need to reauthenticate.

ID Token Header

The header contains two pieces of information: the key ID (**kid**), and the algorithm (**alg**).

```
{  
  "kid" : "1234example=",  
  "alg" : "RS256",  
}
```

Key ID (**kid**)

The **kid** parameter is a hint that indicates which key was used to secure the JSON Web Signature (JWS) of the token.

For more information about the **kid** parameter, see the [Key identifier \(kid\) header parameter](#).

Algorithm (**alg**)

The **alg** parameter represents the cryptographic algorithm that is used to secure the ID token. User pools use an RS256 cryptographic algorithm, which is an RSA signature with SHA-256.

For more information about the **alg** parameter, see [Algorithm \(alg\) header parameter](#).

ID Token Payload

This is a sample payload from an ID token. It contains claims about the authenticated user. For more information about OpenID Connect (OIDC) standard claims, see the [OIDC standard claims](#).

```
{  
  "sub": "aaaaaaaa-bbbbcccc-dddd-eeeeeeeeeee",  
  "aud": "xxxxxxxxxxxxxexample",  
  "email_verified": true,  
  "token_use": "id",  
  "auth_time": 1500009400,  
  "iss": "https://cognito-idp.us-east-1.amazonaws.com/us-east-1_example",  
  "cognito:username": "janedoe",  
  "exp": 1500013000,  
  "given_name": "Jane",  
  "iat": 1500009400,  
  "email": "janedoe@example.com",  
  "jti": "aaaaaaaa-bbbbcccc-dddd-eeeeeeeeeee",  
  "origin_jti": "aaaaaaaa-bbbbcccc-dddd-eeeeeeeeeee"  
}
```

Subject (sub)

The `sub` claim is a unique identifier (UUID) for the authenticated user. It is not the same as the user name, which might not be unique.

Issuer (iss)

The `iss` claim has the following format:

`https://cognito-idp.{region}.amazonaws.com/{userPoolId}`

For example, suppose you created a user pool in the `us-east-1` Region and its user pool ID is `u123456`. In that case, the ID token that is issued for users of your user pool have the following `iss` claim value:

`https://cognito-idp.us-east-1.amazonaws.com/u123456`

Audience (aud)

The content of the `aud` claim is the `client_id` that the user requested when they authenticated with your user pool.

Token use (token_use)

The `token_use` claim describes the intended purpose of this token. Its value is always `id` in the case of the ID token.

Authentication time (auth_time)

The `auth_time` claim contains the time when the authentication occurred. Its value is a JSON number representing the number of seconds from 1970-01-01T0:0:0Z as measured in UTC format. On refreshes, `auth_time` represents the time when the original authentication occurred, not the time when the token was issued.

Nonce (nonce)

The `nonce` claim comes from a parameter of the same name that you can add to requests to your OAuth 2.0 `authorize` endpoint. When you add the parameter, the `nonce` claim is included in the ID token that Amazon Cognito issues, and you can use it to guard against replay attacks. If you do not provide a `nonce` value in your request, Amazon Cognito automatically generates and validates a `nonce` when you authenticate through a third-party identity provider, then adds it as a `nonce` claim to the ID token. The implementation of the `nonce` claim in Amazon Cognito is based on [OIDC standards](#).

Origin jti (origin_jti)

The originating JWT identifier, from the time when the original authentication occurred.

jti (jti)

The `jti` claim is a unique identifier of the JWT.

The ID token can contain OIDC standard claims that are defined in [OIDC standard claims](#). The ID token can also contain custom attributes that you define in your user pool. Amazon Cognito writes custom attribute values to the ID token as strings regardless of attribute type.

Note

User pool custom attributes are always prefixed with a custom: prefix.

ID Token Signature

The signature of the ID token is calculated based on the header and payload of the JWT token. When used outside of an application in your web APIs, you must always verify this signature before accepting the token. See [Verifying a JSON Web Token](#). [Verifying a JSON web token \(p. 198\)](#).

Using the access token

The user pool access token contains claims about the authenticated user, a list of the user's groups, and a list of scopes. The purpose of the access token is to authorize API operations in the context of the user in the user pool. For example, you can use the access token to grant your user access to add, change, or delete user attributes.

The access token is represented as a JSON Web Token (JWT). The header for the access token has the same structure as the ID token. However, the key ID (`kid`) is different because different keys are used to sign ID tokens and access tokens. As with the ID token, you must first verify the signature of the access token in your web APIs before you can trust any of its claims. See [Verifying a JSON web token \(p. 198\)](#). You can set the access token expiration to any value between 5 minutes and 1 day. You can set this value per app client.

Important

For access and ID tokens, don't specify a minimum less than an hour. Amazon Cognito HostedUI uses cookies that are valid for an hour. If you enter a minimum less than an hour, you won't get a lower expiry time.

Access token header

The header contains two pieces of information: the key ID (`kid`), and the algorithm (`alg`).

```
{  
  "kid" : "1234example=",  
  "alg" : "RS256",  
}
```

Key ID (`kid`)

The `kid` parameter is a hint indicating which key was used to secure the JSON Web Signature (JWS) of the token.

For more information about the `kid` parameter, see the [Key identifier \(kid\) header parameter](#).

Algorithm (`alg`)

The `alg` parameter represents the cryptographic algorithm that was used to secure the access token. User pools use an RS256 cryptographic algorithm, which is an RSA signature with SHA-256.

For more information about the `alg` parameter, see [Algorithm \(alg\) header parameter](#).

Access token payload

This is a sample payload from an access token. For more information, see [JWT claims](#).

```
{  
  "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeee",  
  "device_key": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeee",  
  "cognito:groups": [  
    "admin"
```

```
],
  "token_use": "access",
  "scope": "aws.cognito.signin.user.admin",
  "auth_time": 1562190524,
  "iss": "https://cognito-idp.us-west-2.amazonaws.com/us-west-2_example",
  "exp": 1562194124,
  "iat": 1562190524,
  "origin_jti": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeee",
  "jti": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeee",
  "client_id": "57cbishk4j24pabc1234567890",
  "username": "janedoe@example.com"
}
```

Subject (sub)

The `sub` claim is a unique identifier (UUID) for the authenticated user. It is not the same as the user name, which may not be unique.

Amazon Cognito groups (cognito:groups)

The `cognito:groups` claim is a list of groups the user belongs to.

Token use (token_use)

The `token_use` claim describes the intended purpose of this token. Its value is always `access` in the case of the access token.

Scope (scope)

The scope claim is a list of OAuth 2.0 scopes that define what access the token provides.

Authentication time (auth_time)

The `auth_time` claim contains the time when the authentication occurred. Its value is a JSON number that represents the number of seconds from 1970-01-01T0:0:0Z as measured in UTC format. On refreshes, it represents the time when the original authentication occurred, not the time when the token was issued.

Issuer (iss)

The `iss` claim has the following format:

<https://cognito-idp.{region}.amazonaws.com/{userPoolId}>.

Origin jti (origin_jti)

The originating JWT identifier, from the time when the original authentication occurred.

jti (jti)

The `jti` claim is the unique identifier of the JWT.

Access token signature

The signature of the access token is calculated based on the header and payload of the JWT token. When used outside of an application in your web APIs, you must always verify this signature before accepting the token. For more information, see [JWT tokens](#).

Using the refresh token

You can use the refresh token to retrieve new ID and access tokens. By default, the refresh token expires 30 days after your application user signs into your user pool. When you create an application for your

user pool, you can set the application's refresh token expiration to any value between 60 minutes and 10 years.

The Mobile SDK for iOS, Mobile SDK for Android, Amplify for iOS, Android, and Flutter automatically refresh your ID and access tokens if a valid (unexpired) refresh token is present. The ID and access tokens have a minimum remaining validity of 2 minutes. If the refresh token is expired, your app user must re-authenticate by signing in again to your user pool. If the minimum for the access token and ID token is set to 5 minutes, and you are using the SDK, the refresh token will be continually used to retrieve new access and ID tokens. You will see expected behavior with a minimum of 7 minutes instead of 5 minutes.

Note

The AWS SDK for Android offers the option to change the minimum validity period of the ID and access tokens to a value between 0 and 30 minutes. See the `setRefreshThreshold()` method of [CognitoIdentityProviderClientConfig](#) in the Amplify Android SDK reference.

Your user's account itself never expires, as long as the user has logged in at least once before the `UnusedAccountValidityDays` time limit for new accounts.

Initiate new refresh tokens (API)

Use the API or hosted UI to initiate authentication for refresh tokens.

To use the refresh token to get new ID and access tokens with the user pool API, use the [AdminInitiateAuth](#) or [InitiateAuth](#) API operations. Pass `REFRESH_TOKEN_AUTH` for the `AuthFlow` parameter. The authorization parameter, `AuthParameters`, is a key-value map where the key is "REFRESH_TOKEN" and the value is the actual refresh token. Amazon Cognito returns new ID and access tokens after your API request passes all challenges.

Note

To change tokens for users signed in using the hosted UI, use the [InitiateAuth](#) API operation.

Revoking refresh tokens

You can revoke refresh tokens that belong to a user. For more information about revoking tokens, see [Revoking tokens \(p. 197\)](#).

Note

Revoking the refresh token will revoke all tokens that are issued with the refresh token.

Users can sign out from all devices where they are currently signed in when you revoke all of the user's tokens using the `GlobalSignOut` and `AdminUserGlobalSignOut` API operations. After the user is signed out, the following things occur:

- The user's refresh token cannot be used to get new tokens for the user.
- The user's access token cannot be used for the user pools service.
- The user must re-authenticate to get new tokens. The session cookies do not expire automatically. As a best practice, applications should redirect users to the logout endpoint to force the browser to clear session cookies.

An application can use the `GlobalSignOut` API to allow individual users to sign themselves out from all devices. Typically an application would present this option as a choice, such as **Sign out from all devices**. The application must call this API operation with the user's valid, non-expired, non-revoked access token. This operation can't be used to allow a user to sign out another user.

An application can use the `AdminUserGlobalSignOut` API to allow administrators to sign out a user from all devices. The administrator application must call this API operation with AWS

developer credentials and pass the user pool ID and the user's user name as parameters. The `AdminUserGlobalSignOut` API can sign out any user in the user pool.

Revoking tokens

You can revoke a refresh token for a user using the AWS API. When you revoke a refresh token, all access tokens that were previously issued by that refresh token become invalid. The other refresh tokens issued to the user are not affected.

Note

[JWT tokens](#) are self-contained with a signature and expiration time that was assigned when the token was created. Revoked tokens can't be used with any Amazon Cognito API calls that require a token. However, revoked tokens will still be valid if they are verified using any JWT library that verifies the signature and expiration of the token.

You can revoke a refresh token for a user pool client with token revocation enabled. When you create a new user pool client, token revocation is enabled by default.

Enable token revocation

Before you can revoke a token for an existing user pool client, you must enable token revocation. You can enable token revocation for existing user pool clients using the AWS CLI or the AWS API. To do this, call the `aws cognito update-user-pool-client` CLI command or the `UpdateUserPoolClient` API operation. When you do, set the `EnableTokenRevocation` parameter to `true`.

When you create a new user pool client using the AWS Management Console, the AWS CLI, or the AWS API, token revocation is enabled by default.

After you enable token revocation, new claims are added in the Amazon Cognito JSON web tokens. The `origin_jti` and `jti` claims are added to access and ID tokens. These claims increase the size of the application client access and ID tokens.

The following JSON example shows a request to enable token revocation using the [CreateUserPoolClient](#) API.

```
{
    "AccessTokenValidity": 123,
    "AllowedOAuthFlows": [
        "string"
    ],
    "AllowedOAuthFlowsUserPoolClient": true,
    "AllowedOAuthScopes": [
        "string"
    ],
    "AnalyticsConfiguration": {
        "ApplicationArn": "string",
        "ApplicationId": "string",
        "ExternalId": "string",
        "RoleArn": "string",
        "UserDataShared": false
    },
    "CallbackURLs": [
        "string"
    ],
    "ClientName": "string",
    "DefaultRedirectURI": "string",
    "ExplicitAuthFlows": [
        "string"
    ],
    "GenerateSecret": true,
```

```
"IdTokenValidity": 123,  
"LogoutURLs": [  
    "string"  
,  
"PreventUserExistenceErrors": "string",  
"ReadAttributes": [  
    "string"  
,  
"RefreshTokenValidity": 456,  
"SupportedIdentityProviders": [  
    "string"  
,  
"TokenValidityUnits": {  
    "AccessToken": "string",  
    "IdToken": "string",  
    "RefreshToken": "string"  
,  
"UserPoolId": "string",  
"WriteAttributes": [  
    "string"  
,  
"EnableTokenRevocation": true  
}
```

Revoke a token

You can revoke a refresh token using the [RevokeToken](#) API operation. You can also use the `aws cognito-idp revoke-token` CLI command to revoke tokens. Finally, you can revoke tokens using the [revocation endpoint](#). This endpoint is available after you add a domain to your user pool. You can use the revocation endpoint on either an Amazon Cognito hosted domain or your own custom domain.

Note

To revoke a refresh token, you must use the same client ID that was used to obtain the token.

Verifying a JSON web token

These steps describe verifying a user pool JSON Web Token (JWT).

Topics

- [Prerequisites \(p. 198\)](#)
- [Step 1: Confirm the structure of the JWT \(p. 199\)](#)
- [Step 2: Validate the JWT signature \(p. 199\)](#)
- [Step 3: Verify the claims \(p. 200\)](#)

Prerequisites

Your library, SDK, or software framework might already handle the tasks in this section. For example, Amazon Cognito SDKs provide user pool token handling and management on the client side. Likewise, the Mobile SDK for iOS and the Mobile SDK for Android automatically refresh your ID and access tokens if two conditions are met: A valid (unexpired) refresh token must present, and the ID and access tokens must have a minimum remaining validity time of 5 minutes. For information on the SDKs, and sample code for JavaScript, Android, and iOS, see [Amazon Cognito user pool SDKs](#).

Many libraries are available for decoding and verifying a JSON Web Token (JWT). If you must manually process tokens for server-side API processing or if you are using other programming languages, these libraries can help. See the [OpenID foundation list of libraries for working with JWT tokens](#).

Step 1: Confirm the structure of the JWT

A JSON Web Token (JWT) includes three sections:

1. Header
2. Payload
3. Signature

```
1111111111.2222222222.3333333333
```

These sections are encoded as base64url strings and are separated by dot (.) characters. If your JWT does not conform to this structure, consider it not valid and do not accept it.

Step 2: Validate the JWT signature

The JWT signature is a hashed combination of the header and the payload. Amazon Cognito generates two pairs of RSA cryptographic keys for each user pool. One of the private keys is used to sign the token.

To verify the signature of a JWT token

1. Decode the ID token.

You can use AWS Lambda to decode user pool JWTs. For more information, see [Decode and verify Amazon Cognito JWT tokens using AWS Lambda](#).

The OpenID Foundation also [maintains a list of libraries for working with JWT tokens](#).

2. Compare the local key ID (`kid`) to the public kid.

- a. Download and store the corresponding public JSON Web Key (JWK) for your user pool. It is available as part of a JSON Web Key Set (JWKS). You can locate it by constructing the following URL for your environment:

```
https://cognito-idp.{region}.amazonaws.com/{userPoolId}/.well-known/jwks.json
```

For more information on JWK and JWK sets, see [JSON web key \(JWK\)](#).

Note

Downloading and storing the JWK for your user pool is a one-time step before your web API operations can process tokens. After doing so, you can perform the following steps each time the ID token or the access token is used with your web API operations.

This is a sample `jwks.json` file:

```
{
  "keys": [
    {
      "kid": "1234example=",
      "alg": "RS256",
      "kty": "RSA",
      "e": "AQAB",
      "n": "1234567890",
      "use": "sig"
    },
    {
      "kid": "5678example=",
      "alg": "RS256",
      "kty": "RSA",
      "e": "AQAB",
      "n": "1234567890"
    }
  ]
}
```

```
        "n": "987654321",
        "use": "sig"
    }]
}
```

Key ID (`kid`)

The `kid` is a hint that indicates which key was used to secure the JSON web signature (JWS) of the token.

Algorithm (`alg`)

The `alg` header parameter represents the cryptographic algorithm that is used to secure the ID token. User pools use an RS256 cryptographic algorithm, which is an RSA signature with SHA-256. For more information on RSA, see [RSA cryptography](#).

Key type (`kty`)

The `kty` parameter identifies the cryptographic algorithm family that is used with the key, such as "RSA" in this example.

RSA exponent (`e`)

The `e` parameter contains the exponent value for the RSA public key. It is represented as a Base64urlUInt-encoded value.

RSA modulus (`n`)

The `n` parameter contains the modulus value for the RSA public key. It is represented as a Base64urlUInt-encoded value.

Use (`use`)

The `use` parameter describes the intended use of the public key. For this example, the `use` value `sig` represents signature.

- b. Search the public JSON web key for a `kid` that matches the `kid` of your JWT.
3. Use a JWT library, such as the [aws-jwt-verify library on GitHub](#), to compare the signature of the issuer to the signature in the token. The issuer signature is derived from the public key (the RSA modulus "`n`") of the `kid` in `jwks.json` that matches the token `kid`. You might need to convert the JWK to PEM format first. The following example takes the JWT and JWK and uses the Node.js library, `jsonwebtoken`, to verify the JWT signature:

Node.js

```
var jwt = require('jsonwebtoken');
var jwkToPem = require('jwk-to-pem');
var pem = jwkToPem(jwk);
jwt.verify(token, pem, { algorithms: ['RS256'] }, function(err, decodedToken) {
});
```

Step 3: Verify the claims

To verify JWT claims

1. Verify that the token is not expired.
2. The `aud` claim in an ID token and the `client_id` claim in an access token should match the app client ID that was created in the Amazon Cognito user pool.
3. The issuer (`iss`) claim should match your user pool. For example, a user pool created in the `us-east-1` Region will have the following `iss` value:

<https://cognito-idp.us-east-1.amazonaws.com/<userpoolID>>.

4. Check the `token_use` claim.

- If you are only accepting the access token in your web API operations, its value must be `access`.
- If you are only using the ID token, its value must be `id`.
- If you are using both ID and access tokens, the `token_use` claim must be either `id` or `access`.

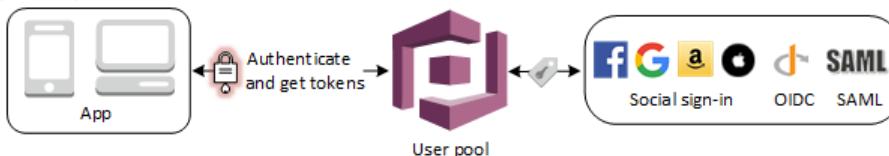
You can now trust the claims inside the token.

Accessing resources after a successful user pool authentication

Your app users can sign in either directly through a user pool, or federate through a third-party identity provider (IdP). The user pool manages the overhead of handling the tokens that are returned from social sign-in through Facebook, Google, Amazon, and Apple, and from OpenID Connect (OIDC) and SAML IdPs. For more information see [Using tokens with user pools \(p. 191\)](#).

After a successful authentication, your app will receive user pool tokens from Amazon Cognito. You can use those tokens to retrieve AWS credentials that allow your app to access other AWS services, or you might choose to use them to control access to your own server-side resources, or to the Amazon API Gateway.

For more information, see [User pool authentication flow \(p. 364\)](#) and [Using tokens with user pools \(p. 191\)](#).

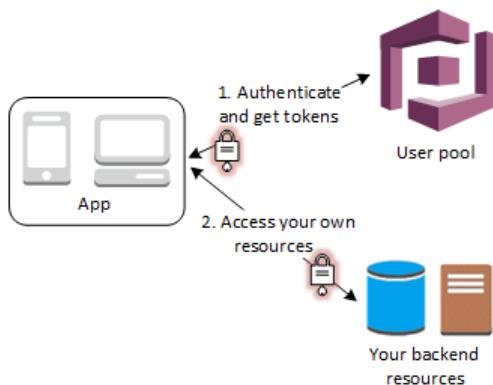


Topics

- [Accessing server-side resources after sign-in \(p. 10\)](#)
- [Accessing resources with API Gateway and Lambda after sign-in \(p. 202\)](#)
- [Accessing AWS services using an identity pool after sign-in \(p. 203\)](#)

Accessing server-side resources after sign-in

After a successful authentication, your web or mobile app will receive user pool tokens from Amazon Cognito. You can use those tokens to control access to your server-side resources. You can also create user pool groups to manage permissions, and to represent different types of users. For more information on using groups to control access to your resources see [Adding groups to a user pool \(p. 162\)](#).

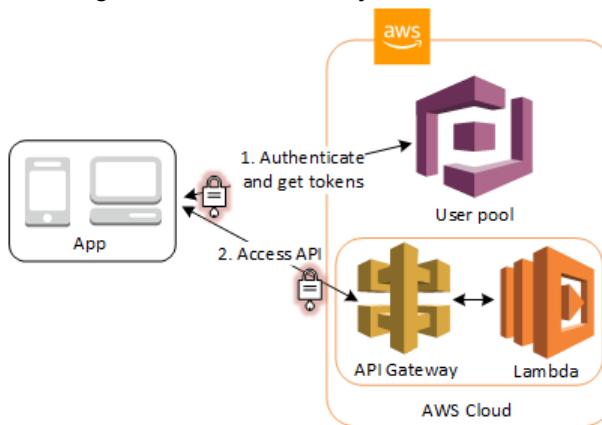


Once you configure a domain for your user pool, Amazon Cognito provisions a hosted web UI that allows you to add sign-up and sign-in pages to your app. Using this OAuth 2.0 foundation you can create your own resource server to enable your users to access protected resources. For more information see [Defining resource servers for your user pool \(p. 57\)](#).

For more information about user pool authentication see [User pool authentication flow \(p. 364\)](#) and [Using tokens with user pools \(p. 191\)](#).

Accessing resources with API Gateway and Lambda after sign-in

You can enable your users to access your API through API Gateway. API Gateway validates the tokens from a successful user pool authentication, and uses them to grant your users access to resources including Lambda functions, or your own API.



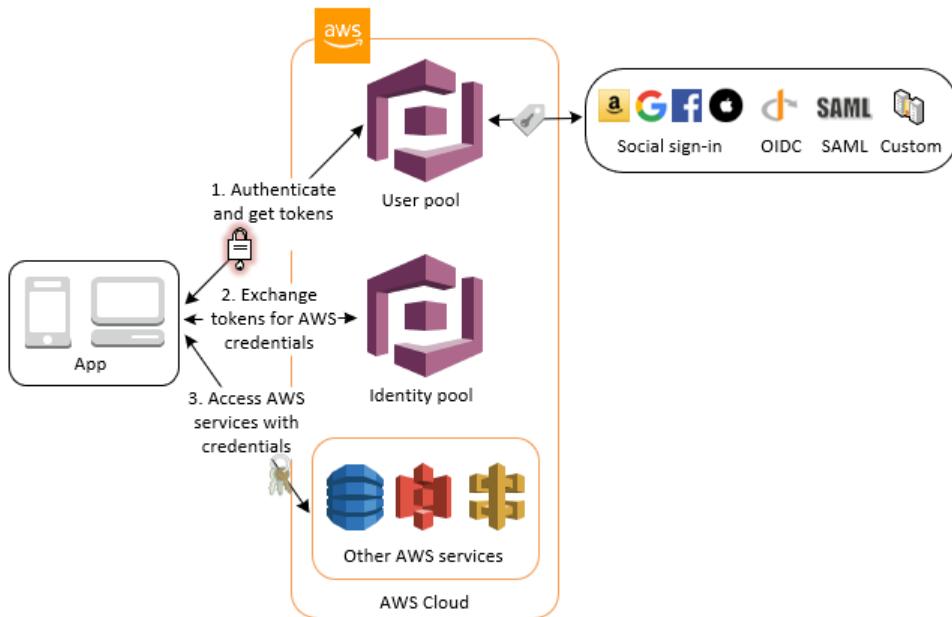
You can use groups in a user pool to control permissions with API Gateway by mapping group membership to IAM roles. The groups that a user is a member of are included in the ID token provided by a user pool when your web or mobile app user signs in. For more information on user pool groups See [Adding groups to a user pool \(p. 162\)](#).

You can submit your user pool tokens with a request to API Gateway for verification by an Amazon Cognito authorizer Lambda function. For more information on API Gateway, see [Using API Gateway with Amazon Cognito user pools](#).

Accessing AWS services using an identity pool after sign-in

You can enable your users to sign-in with a user pool, and then access AWS services using an identity pool.

After a successful authentication, your web or mobile app will receive user pool tokens from Amazon Cognito. You can use those tokens to retrieve AWS credentials that allow your app to access other AWS services. For more information, see [Getting started with Amazon Cognito identity pools \(federated identities\) \(p. 242\)](#).



For more information about using identity pools together with user pool groups to control access your AWS resources see [Adding groups to a user pool \(p. 162\)](#) and [Role-based access control \(p. 264\)](#). See also [Identity pools concepts \(federated identities\) \(p. 248\)](#) for more information about identity pools and AWS Identity and Access Management.

Setting up a user pool with the AWS Management Console

Create an Amazon Cognito user pool and make a note of the **User Pool ID** and **App Client ID** for each of your client apps. For more information about creating user pools, see [Getting started with user pools \(p. 23\)](#).

Setting up an identity pool with the AWS Management Console

The following procedure describes how to use the AWS Management Console to integrate an identity pool with one or more user pools and client apps.

Original console

To configure your identity pool

1. Open the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **Manage Identity Pools**.
3. Choose the name of the identity pool for which you want to enable Amazon Cognito user pools as a provider.

4. On the **Dashboard** page, choose **Edit identity pool**.
5. Expand the **Authentication providers** section.
6. Choose **Cognito**.
7. Enter the **User Pool ID**.
8. Enter the **App Client ID**. This must be the same client app ID that you received when you created the app in the **Your User Pools** section of the AWS Management Console for Amazon Cognito.
9. If you have additional apps or user pools, choose **Add Another Provider** and enter the **User Pool ID** and **App Client ID** for each app in each user pool.
10. When you have no more apps or user pools to add, choose **Save Changes**. If successful, you will see a **Changes saved successfully** message on the **Dashboard** page.

New console

To configure your identity pool

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **Federated identities**.
3. Choose the name of the identity pool for which you want to enable Amazon Cognito user pools as a provider.
4. On the **Dashboard** page, choose **Edit identity pool**.
5. Expand the **Authentication providers** section.
6. Choose **Cognito**.
7. Enter the **User Pool ID**.
8. Enter the **App Client ID**. This must be the same client app ID that you received when you created the app in the **User pools** section of the console.
9. If you have additional apps or user pools, choose **Add Another Provider** and enter the **User Pool ID** and **App Client ID** for each app in each user pool.
10. When you have no more apps or user pools to add, choose **Save Changes**. If successful, you will see a **Changes saved successfully** message on the **Dashboard** page.

Integrating a user pool with an identity pool

After your app user is authenticated, add that user's identity token to the logins map in the credentials provider. The provider name will depend on your Amazon Cognito user pool ID. It will have the following structure:

```
cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>
```

The value for **<region>** will be the same as the region in the **User Pool ID**. For example, `cognito-idp.us-east-1.amazonaws.com/us-east-1_123456789`.

JavaScript

```
var cognitoUser = userPool.getCurrentUser();

if (cognitoUser != null) {
    cognitoUser.getSession(function(err, result) {
        if (result) {
            console.log('You are now logged in.');

        // Add the User's Id Token to the Cognito credentials login map.
    }
}
```

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: 'YOUR_IDENTITY_POOL_ID',
    Logins: {
        'cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>':
            result.getIdToken().getJwtToken()
    }
});
```

Android

```
cognitoUser.getSessionInBackground(new AuthenticationHandler() {
    @Override
    public void onSuccess(CognitoUserSession session) {
        String idToken = session.getIdToken().getJWTToken();

        Map<String, String> logins = new HashMap<String, String>();
        logins.put("cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>",
                session.getIdToken().getJWTToken());
        credentialsProvider.setLogins(logins);
    }
});
```

iOS - objective-C

```
AWSServiceConfiguration *serviceConfiguration = [[AWSServiceConfiguration alloc]
    initWithRegion:AWSRegionUSEast1 credentialsProvider:nil];
AWSCognitoIdentityUserPoolConfiguration *userPoolConfiguration =
[[AWSCognitoIdentityUserPoolConfiguration alloc] initWithClientId:@"YOUR_CLIENT_ID"
    clientSecret:@"YOUR_CLIENT_SECRET" poolId:@"YOUR_USER_POOL_ID"];
[AWSCognitoIdentityUserPool
    registerCognitoIdentityUserPoolWithConfiguration:serviceConfiguration
    userPoolConfiguration:userPoolConfiguration forKey:@"UserPool"];
AWSCognitoIdentityUserPool *pool = [AWSCognitoIdentityUserPool
    CognitoIdentityUserPoolForKey:@"UserPool"];
AWSCognitoCredentialsProvider *credentialsProvider = [[AWSCognitoCredentialsProvider
    alloc] initWithRegionType:AWSRegionUSEast1 identityPoolId:@"YOUR_IDENTITY_POOL_ID"
    identityProviderManager:pool];
```

iOS - swift

```
let serviceConfiguration = AWSServiceConfiguration(region: .USEast1,
    credentialsProvider: nil)
let userPoolConfiguration = AWSCognitoIdentityUserPoolConfiguration(clientId:
    "YOUR_CLIENT_ID", clientSecret: "YOUR_CLIENT_SECRET", poolId: "YOUR_USER_POOL_ID")
AWSCognitoIdentityUserPool.registerCognitoIdentityUserPoolWithConfiguration(serviceConfiguration,
    userPoolConfiguration: userPoolConfiguration, forKey: "UserPool")
let pool = AWSCognitoIdentityUserPool(forKey: "UserPool")
let credentialsProvider = AWSCognitoCredentialsProvider(regionType: .USEast1,
    identityPoolId: "YOUR_IDENTITY_POOL_ID", identityProviderManager:pool)
```

User pools reference (AWS Management Console)

You can customize the user pool settings to the needs of your app. This section describes each category of settings and gives you detailed information about attributes, policies, email and phone verification, multi-factor authentication, apps, triggers, and trusted devices.

Topics

- [Adding a user pool name \(p. 206\)](#)
- [Importing and creating users and groups \(p. 206\)](#)
- [User pool attributes \(p. 206\)](#)
- [Adding user pool password requirements \(p. 214\)](#)
- [Configuring an admin create user policy \(p. 215\)](#)
- [Configuring email or phone verification \(p. 215\)](#)
- [Configuring SMS and email verification messages and user invitation messages \(p. 217\)](#)
- [Adding cost allocation tags to your user pool \(p. 222\)](#)
- [Specifying user pool device tracking settings \(p. 222\)](#)
- [Configuring a user pool app client \(p. 223\)](#)
- [Configuring user pool Lambda triggers \(p. 226\)](#)
- [Reviewing your user pool creation settings \(p. 226\)](#)
- [Configuring user pool analytics \(p. 226\)](#)
- [Configuring app client settings \(p. 227\)](#)
- [Adding a domain name for your user pool \(p. 228\)](#)
- [Customizing the built-in app UI to sign up and sign in users \(p. 229\)](#)
- [Adding resource servers for your user pool \(p. 230\)](#)
- [Configuring identity providers for your user pool \(p. 231\)](#)
- [Configuring attribute mapping for your user pool \(p. 237\)](#)

Adding a user pool name

During the user pool creation process, you must specify a user pool name. This name can't be changed after the user pool has been created.

User pool names must be between one and 128 characters long. They can contain uppercase and lowercase letters (a-z, A-Z), numbers (0-9), and the following special characters: + = , . @ and -.

Importing and creating users and groups

You can import users, create users, search for users, as well as create groups and assign users to them. For more information, see:

- [Importing users into user pools from a CSV file \(p. 171\)](#)
- [Adding groups to a user pool \(p. 162\)](#)
- [Creating user accounts as administrator \(p. 157\)](#)

User pool attributes

Attributes are pieces of information that help you identify individual users, such as name, email address, and phone number. A new user pool has a set of default *standard attributes*. You can also add custom attributes to your user pool definition in the AWS Management Console. This topic describes those attributes in detail and gives you tips on how to set up your user pool.

Don't store all information about your users in attributes. For example, keep user data that changes frequently, such as usage statistics or game scores, in a separate data store, such as Amazon Cognito Sync or Amazon DynamoDB.

Standard attributes

Amazon Cognito assigns all users the following set of standard attributes based on the [OpenID Connect specification](#).

- `address`
- `birthdate`
- `email`
- `family_name`
- `gender`
- `given_name`
- `locale`
- `middle_name`
- `name`
- `nickname`
- `phone_number`
- `picture`
- `preferred_username`
- `profile`
- `updated_at`
- `website`
- `zoneinfo`

These attributes are available as optional attributes for all users. To make an attribute required, during the user pool creation process, select the **Required** check box next to the attribute.

Note

When you mark a standard attribute as **Required**, a user can't register unless they provide a value for the attribute. To create users and not give values for required attributes, administrators can use the [AdminCreateUser](#) API. After you create a user pool, you can't switch an attribute between required and not required.

By default, standard and custom attribute values can be any string with a length of up to 2048 characters, but some attribute values, such as `updated_at`, have format restrictions. Only `email` and `phone` can be verified.

Note

Some documentation and standards refer to attributes as *members*.

Here are some additional notes about some of the these fields.

`email`

Users and administrators can verify email address values.

An administrator with proper AWS account permissions can change the user's email address and also mark it as verified. Mark an email address as verified with the [AdminUpdateUserAttributes](#) API or the [admin-update-user-attributes](#) AWS Command Line Interface (AWS CLI) command. With this command, the administrator can change the `email_verified` attribute to `true`. You can also edit a user in the **Users** tab of the AWS Management Console to mark an email address as verified.

`phone`

A user must provide a phone number if SMS multi-factor authentication (MFA) is active. For more information, see [Adding MFA to a user pool \(p. 381\)](#).

Users and administrators can verify phone number values.

An administrator with proper AWS account permissions can change the user's phone number and also mark it as verified. Mark a phone number as verified with the [AdminUpdateUserAttributes API](#) or the [admin-update-user-attributes](#) AWS CLI command. With this command, the administrator can change the `phone_number_verified` attribute to `true`. You can also edit a user in the **Users** tab of the AWS Management Console to mark a phone number as verified.

Important

Phone numbers must follow these format rules: A phone number must start with a plus (+) sign, followed immediately by the country code. A phone number can only contain the + sign and digits. Remove any other characters from a phone number, such as parentheses, spaces, or dashes (-) before you submit the value to the service. For example, a phone number based in the United States must follow this format: **+14325551212**.

preferred_username

You can select `preferred_username` as required or as an alias, but not both. If the `preferred_username` is an alias, you can use the [UpdateUserAttributes API](#) to add the attribute value after you confirm the user.

[View required attributes](#)

Use the following procedure to view required attributes for a given user pool.

Note

You can't change required attributes after you create a user pool.

Original console

To view required attributes

1. Go to [Amazon Cognito](#) in the AWS Management Console. If the console prompts you, enter your AWS credentials.
2. Choose **Manage User Pools**.
3. On the **Your User Pools** page, choose a user pool.
4. In the navigation menu on the left, choose **Attributes**.
5. Under **Which standard attributes are required?**, view the required attributes of your user pool.

New console

To view required attributes

1. Go to [Amazon Cognito](#) in the AWS Management Console. If the console prompts you, enter your AWS credentials.
2. Choose **User Pools**.
3. Choose an existing user pool from the list.
4. Choose the **Sign-up experience** tab.
5. In the **Required attributes** section, view the required attributes of your user pool.

[User name and preferred user name](#)

The `username` value is a separate attribute and not the same as the `name` attribute. Each user has a `username` attribute. Amazon Cognito automatically generates a user name for federated users. You

must provide a `username` attribute to create a native user in the Amazon Cognito directory. After you create a user, you can't change the value of the `username` attribute.

Developers can use the `preferred_username` attribute to give users user names that they can change. For more information, see [Aliases \(p. 209\)](#).

If your application doesn't require a user name, you don't need to ask users to provide one. Your app can create a unique username for users in the background. This can be useful if you want users to register and sign in with an email address and password. For more information, see [Aliases \(p. 209\)](#).

The `username` must be unique within a user pool. A `username` can be reused, but only after you delete it and it is no longer in use.

Aliases

If you want, your users can use aliases to enter other attributes when they sign in. By default, users sign in with their user name and password. The user name is a fixed value that users can't change. If you mark an attribute as an alias, users can sign in with that attribute in place of the user name. You can mark the email address, phone number, and preferred username attributes as aliases. For example, if you select email address and phone number as aliases for a user pool, users in that user pool can sign in with their user name, email address, or phone number, along with their password.

Note

When you configure your user pool to be case insensitive, a user can use either lowercase or uppercase letters to sign up or sign in with their alias. For more information, see [CreateUserPool](#) in the *Amazon Cognito user pools API Reference*.

If you select email address as an alias, Amazon Cognito doesn't accept a user name that matches a valid email address format. Similarly, if you select phone number as an alias, Amazon Cognito doesn't accept a user name for that user pool that matches a valid phone number format.

Note

Alias values must be unique in a user pool. If you configure an alias for an email address or phone number, the value that you provide can be in a verified state in only one account. During sign-up, if your user provides an email address or phone number as an alias value and another user has already used that alias value, registration succeeds. However, when a user tries to confirm the account with this email (or phone number) and enters the valid code, Amazon Cognito returns an `AliasExistsException` error. The error indicates to the user that an account with this email address (or phone number) already exists. At this point, the user can abandon their attempt to create the new account and instead try to reset the password for the old account. If the user continues to create the new account, your app must call the `ConfirmSignUp` API with the `forceAliasCreation` option. `ConfirmSignUp` with `forceAliasCreation` moves the alias from the previous account to the newly created account, and marks the attribute unverified in the previous account.

Phone numbers and email addresses only become active aliases for a user after your user verifies the phone numbers and email addresses. Therefore, we recommend that you choose automatic verification of email addresses and phone numbers if you use them as aliases.

Activate the `preferred_username` attribute so that your user can change the user name that they use to sign in while their `username` attribute value doesn't change. If you want to set up this user experience, submit the new `username` value as a `preferred_username` and choose `preferred_username` as an alias. Then users can sign in with the new value that they entered. If you select `preferred_username` as an alias, your user can provide the value only when they confirm an account. They can't provide the value during registration.

Using aliases to simplify user sign-up and sign-in

When you create a user pool, you can choose if you want your user to sign up with an email address or phone number as their user name.

Note

After you create a user pool, you can't change this setting.

Option 1: User signs up with user name and signs in with user name or alias

When the user signs up with a user name, you can choose if they can sign in with one or more of the following aliases:

- Verified email address
- Verified phone number
- Preferred user name

After the user signs up, they can change these aliases.

Include the following steps when you create the user pool so that users can sign in with an alias.

Original console

To configure a user pool for sign-in with an alias

1. Go to [Amazon Cognito](#) in the AWS Management Console. If the console prompts you, enter your AWS credentials.
2. Choose **Manage User Pools**.
3. In the top-right corner of the page, choose **Create a user pool**.
4. In the top-left corner of the page, choose **Attributes**.
5. In the **Attributes** tab, under **How do you want your end users to sign-in?**, select **Username**. Then choose one of the following options:
 - **Also allow sign in with verified email address**: Allows users to sign in with their email address.
 - **Also allow sign in with verified phone number**: Allows users to sign in with their phone number.
 - **Also allow sign in with preferred username**: Allows users to sign in with a preferred user name. This is a user name that the user can change.
6. Choose **Next step** to save, and then complete all the steps in the wizard.

New console

To configure a user pool so that users can sign in with a preferred user name

1. Go to [Amazon Cognito](#) in the AWS Management Console. If the console prompts you, enter your AWS credentials.
2. Choose **User Pools**.
3. In the top-right corner of the page, choose **Create a user pool** to start the user pool creation wizard.
4. In **Configure sign-in experience**, choose the identity **Provider types** that you want to associate with your user pool.
5. Under **Cognito user pool sign-in options**, choose any combination of **User name**, **Email**, and **Phone number**.
6. Under **User name requirements**, choose **Allow users to sign in with a preferred user name** so that your users can set an alternate user name when they sign in.
7. Choose **Next**, and then complete all of the steps in the wizard.

Option 2: User signs up and signs in with email address or phone number instead of user name

When the user signs up with an email address or phone number as their user name, you can choose if they can sign up with only email addresses, only phone numbers, or either one.

The email address or phone number must be unique, and it must not already be in use by another user. It doesn't have to be verified. After the user has signed up with an email address or phone number, the user can't create a new account with the same email address or phone number. The user can only reuse the existing account and reset the account password, if needed. However, the user can change the email address or phone number to a new email address or phone number. If the email address or phone number isn't already in use, it becomes the new user name.

Note

If a user signs up with an email address as their username, they can change the user name to another email address, but they can't change it to a phone number. If they sign up with a phone number, they can change the user name to another phone number, but they can't change it to an email address.

Use the following steps during the user pool creation process to set up sign-up and sign-in with email address or phone number.

Original console

To configure a user pool for sign-up and sign-in with email address or phone number

1. Go to [Amazon Cognito](#) in the AWS Management Console. If the console prompts you, enter your AWS credentials.
2. Choose **Manage User Pools**.
3. In the top-right corner of the page, choose **Create a user pool**.
4. In the top-left corner of the page, choose **Attributes**.
5. In the **Attributes** tab, under **How do you want your end users to sign-in?**, select **Email address or phone number**. Then choose one of the following options:
 - **Allow email addresses:** Allows your user to sign up with email as the user name.
 - **Allow phone numbers:** Allows your user to sign up with phone number as the user name.
 - **Allow both email addresses and phone number (users can choose one):** Allows your user to use either an email address or a phone number as the user name when the user signs up.
6. Choose **Next step** to save, and then complete all the steps in the wizard.

New console

To configure a user pool for sign-up and sign-in with email address or phone number

1. Go to [Amazon Cognito](#) in the AWS Management Console. If the console prompts you, enter your AWS credentials.
2. Choose **User Pools**.
3. In the top-right corner of the page, choose **Create a user pool** to start the user pool creation wizard.
4. Under **Cognito user pool sign-in options**, choose any combination of **User name**, **Email**, and **Phone number** that represents the alias attributes that the user can use to sign in.
5. Under **User name requirements**, choose one or both of the following options:

Allow users to sign in with a preferred user name

Allows the user to sign in with a preferred user name. The user can change this user name.

Make user name case sensitive

Sets the user name as case sensitive, where `ExampleUser` and `exampleuser` are unique user names.

6. Choose **Next**, and then complete all the steps in the wizard.

Note

You do not need to mark email address or phone number as required attributes for your user pool.

To implement option 2 in your app

1. Call the `CreateUserPool` API to create your user pool. Set the `UserNameAttributes` parameter to `phone_number, email, or phone_number | email`.
2. Call the `SignUp` API and pass an email address or phone number in the `username` parameter of the API. This API does the following:
 - If the `username` string is in valid email address format, the user pool automatically populates the `email` attribute of the user with the `username` value.
 - If the `username` string is in valid phone number format, the user pool automatically populates the `phone_number` attribute of the user with the `username` value.
 - If the `username` string format isn't in email address or phone number format, the `SignUp` API returns an exception.
 - The `SignUp` API generates a persistent UUID for your user, and uses it internally as the immutable user name attribute. This UUID has the same value as the `sub` claim in the user identity token.
 - If the `username` string contains an email address or phone number that is already in use, the `SignUp` API returns an exception.

You can use an email address or phone number as an alias in place of the user name in all APIs except the `ListUsers` API. When you call `ListUsers`, you can search by the `email` or `phone_number` attribute. If you search by `username`, you must supply the actual user name, not an alias.

Custom attributes

You can add up to 50 custom attributes to your user pool. You can specify a minimum and/or maximum length for custom attributes. However, the maximum length for any custom attribute can be no more than 2048 characters.

Each custom attribute has the following characteristics:

- You can define it as a string or a number. Amazon Cognito writes custom attribute values to the ID token only as strings.
- You can't require that users provide a value for the attribute.
- You can't remove or change it after you add it to the user pool.
- The character length of the attribute name is within the limit that Amazon Cognito accepts. For more information, see [Quotas in Amazon Cognito \(p. 406\)](#).
- It can be *mutable* or *immutable*. You can only write a value to an immutable attribute when you create a user. You can change the value of a mutable attribute if your app client has write permission to the attribute. See [Attribute permissions and scopes \(p. 213\)](#) for more information.

Note

In your code, and in rules settings for [Role-based access control \(p. 264\)](#), custom attributes require the `custom:` prefix to distinguish them from standard attributes.

Use the following procedure to create a new custom attribute.

Original console

To add a custom attribute through the console

1. Go to [Amazon Cognito](#) in the AWS Management Console. If the console prompts you, enter your AWS credentials.
2. Choose **Manage User Pools**.
3. On the **Your User Pools** page, choose the user pool that you want to configure.
4. From the left navigation bar, choose **Attributes**.
5. Under **Do you want to add custom attributes?**, choose **Add custom attribute**.
6. Provide the following details about the new attribute:
 - Select the data **Type** (string or number)
 - Enter a **Name** value
 - Enter a **Min length** value
 - Enter a **Max length** value
 - Select **Mutable** if you want to give users permission to change the value of a custom attribute after they set the initial value.
7. Choose **Save changes**.

New console

To add a custom attribute using the console

1. Go to [Amazon Cognito](#) in the AWS Management Console. If the console prompts you, enter your AWS credentials.
2. Choose **User Pools**.
3. Choose an existing user pool from the list.
4. Choose the **Sign-up experience** tab, and in the **Custom attributes** section, choose **Add custom attributes**.
5. On the **Add custom attributes** page, provide the following details about the new attribute:
 - Enter a **Name** value
 - Select the **Type** (string or number)
 - Enter a **Min length** value
 - Enter a **Max length** value
 - Select **Mutable** if you want to give users permission to change the value of a custom attribute after they set the initial value.
6. Choose **Save changes**.

Attribute permissions and scopes

For each app client, you can set read and write permissions for each user attribute. This way, you can control the access that any app has to read and modify each attribute that you store for your users. For example, you might have a custom attribute that indicates whether a user is a paying customer or not. Your apps might be able to see this attribute but not change it directly. Instead, you would update this attribute using an administrative tool or a background process. You can set permissions for user attributes from the Amazon Cognito console, the Amazon Cognito API, or the AWS CLI. By default, any new custom attributes aren't available until you set read and write permissions for them.

Original console

To update attribute permissions through the console

1. Go to [Amazon Cognito](#) in the AWS Management Console. If the console prompts you, enter your AWS credentials.
2. Choose **Manage User Pools**.
3. On the **Your User Pools** page, choose the user pool that you want to configure.
4. In the left navigation bar, choose **App clients**.
5. Choose **Show Details** for the app client that you want to update.
6. At the bottom of the page, choose **Set attribute read and write permissions**, and then configure your read and write permissions.
7. Choose **Save app client changes**.

New console

To update attribute permissions through the console

1. Go to [Amazon Cognito](#) in the AWS Management Console. If the console prompts you, enter your AWS credentials.
2. Choose **User Pools**.
3. Choose an existing user pool from the list.
4. Choose the **App integration** tab, and in the **App clients** section, choose an app client from the list.
5. In the **Attribute read and write permissions** section, choose **Edit**.
6. On the **Edit attribute read and write permissions** page, configure your read and write permissions, and then choose **Save changes**.

Repeat these steps for each app client that uses the custom attribute.

For each app, you can mark attributes as readable or writable. This applies to both standard and custom attributes. An app can read an attribute that you mark as readable and can write an attribute that you mark as writable. If an app tries to update an attribute that isn't writable, the app gets a `NotAuthorizedException` exception. An app that calls `GetUser` only receives the attributes that are readable for that app. Your user's ID token from an app only contains claims that correspond to the readable attributes. Required attributes in a user pool are always writable. If you use the AWS CLI or the admin API to set a writable attribute and don't provide required attributes, then you receive an `InvalidParameterException` exception.

You can change attribute permissions and scopes after you have created your user pool.

Adding user pool password requirements

To create strong passwords for your app users, specify a minimum password length of at least eight characters. Also require uppercase, numeric, and special characters. Complex passwords are harder to guess. We recommend them as a security best practice.

You can use the following characters in passwords:

- Uppercase and lowercase [Basic latin](#) letters
- Numbers

- Special characters listed in the next section.

Creating a password policy

You can specify the following password requirements in the AWS Management Console:

- **Minimum length**, which must be at least six characters, but fewer than 99 characters
- **Includes at least one:**
 - **Number**
 - **Special character** from the following set. The space character is also treated as a special character.
`^ $ * . [] { } () ? " ! @ # % & / \ , > < ' : ; | _ ~ ` = + -`
- **Uppercase Basic Latin letter**
- **Lowercase Basic Latin letter**

Configuring an admin create user policy

You can specify the following policies for Admin Create User:

- Specify whether to allow users to sign themselves up. This option is set by default. If it is not set, only administrators can create users in this pool and calls to the [SignUp](#) API fail with `NotAuthorizedException`.
- Specify the user account expiration time limit (in days) for new accounts. The default setting is 7 days, measured from the time when an administrator or the user creates the account. The maximum setting is 365 days. After the account expires, the user can't log in to the account until you update the user's profile. To do this, update an attribute or resend the password to the user.

Note

After the user logs in, the account never expires.

Configuring email or phone verification

Note

In the new Amazon Cognito console experience, you can manage verification in the **Sign-up experience** tab of your user pool.

You can choose settings for email or phone verification under the **MFA and verifications** tab. For more information on multi-factor authentication (MFA), see [SMS Text Message MFA \(p. 384\)](#).

Amazon Cognito uses Amazon SNS to send SMS messages. If you haven't sent an SMS message from Amazon Cognito or any other AWS service before, Amazon SNS might place your account in the SMS sandbox. We recommend that you send a test message to a verified phone number before you remove your account from the sandbox to production. Additionally, if you plan to send SMS messages to US destination phone numbers, you must obtain an origination or Sender ID from Amazon Pinpoint. To configure your Amazon Cognito user pool for SMS messages, see [SMS message settings for Amazon Cognito user pools \(p. 187\)](#).

Amazon Cognito can automatically verify email addresses or phone numbers. To do this verification, Amazon Cognito sends a verification code or a verification link. For email addresses, Amazon Cognito can send a code or a link in an email message. For phone numbers, Amazon Cognito sends a code in an SMS text message.

Amazon Cognito must verify a phone number or email address to confirm users and help them to recover forgotten passwords. Alternatively, you can automatically confirm users with the pre sign-up

Lambda trigger or use the [AdminConfirmSignUp](#) API operation. For more information, see [Signing up and confirming user accounts \(p. 148\)](#).

The verification code or link is valid for 24 hours.

If you choose to require verification for an email address or phone number, Amazon Cognito automatically sends the verification code or link when a user signs up. If the user pool has a [Custom SMS sender Lambda trigger \(p. 141\)](#) or [Custom email sender Lambda trigger \(p. 137\)](#) configured, that function is invoked instead.

Notes

- Amazon SNS charges separately for SMS text messaging that it uses to verify phone numbers. There is no charge to send email messages. For information about Amazon SNS pricing, see [Worldwide SMS pricing](#). For the current list of countries where SMS messaging is available, see [Supported regions and countries](#).
- When you test actions in your app that generate email messages from Amazon Cognito, use a real email address that Amazon Cognito can reach without hard bounces. For more information, see [the section called “Sending emails while testing your app” \(p. 156\)](#).
- The forgotten password flow requires either the user's email or the user's phone number to verify the user.

Important

If a user signs up with both a phone number and an email address, and your user pool settings require verification of both attributes, Amazon Cognito sends a verification code to the phone number through SMS message. Amazon Cognito hasn't yet verified the email address, so your app must call [GetUser](#) to see if an email address awaits verification. If it does require verification, the app must call [GetUserAttributeVerificationCode](#) to initiate the email verification flow. Then it must submit the verification code by calling [VerifyUserAttribute](#).

You can adjust your SMS message spend quota for an AWS account and for individual messages. The limits apply only to the cost to send SMS messages. For more information, see [What are account-level and message-level spend quotas and how do they work?](#) in the [Amazon SNS FAQs](#).

Amazon Cognito sends SMS messages using Amazon SNS resources in either the AWS Region where you created the user pool or in a **Legacy Amazon SNS alternate Region** from the following table. The exception is Amazon Cognito user pools in the Asia Pacific (Seoul) Region. These user pools use your Amazon SNS configuration in the Asia Pacific (Tokyo) Region. For more information, see [Step 1: Choose the AWS Region for Amazon SNS SMS messages \(p. 187\)](#).

Amazon Cognito Region	Legacy Amazon SNS alternate Region
US East (Ohio)	US East (N. Virginia)
Asia Pacific (Mumbai)	Asia Pacific (Singapore)
Asia Pacific (Seoul)	Asia Pacific (Tokyo)
Canada (Central)	US East (N. Virginia)
Europe (Frankfurt)	Europe (Ireland)
Europe (London)	Europe (Ireland)

Example: If your Amazon Cognito user pool is in Asia Pacific (Mumbai), and you have increased your spend limit in ap-southeast-1, you might not want to request a separate increase in ap-south-1. Instead, you can use your Amazon SNS resources in Asia Pacific (Singapore).

Verifying updates to to email addresses and phone numbers

An email address or phone number attribute can become active and unverified immediately after your user changes its value. Amazon Cognito can also require that your user verifies the new value before Amazon Cognito updates the attribute. When you require that your users first verify the new value, they can use the original value for sign-in and to receive messages until they verify the new value.

When your users can use their email address or phone number as a sign-in alias in your user pool, their sign-in name for an updated attribute depends on whether you require verification of updated attributes. When you require that users verify an updated attribute, a user can sign in with the original attribute value until they verify the new value. When you don't require that users verify an updated attribute, a user can't sign in or receive messages at either the new or the original attribute value until they verify the new value.

For example, your user pool allows sign-in with an email address alias, and requires that users verify their email address when they update. Sue, who signs in as `sue@example.com`, wants to change her email address to `sue2@example.com` but accidentally enters `ssue2@example.com`. Sue doesn't receive the verification email, so she can't verify `ssue2@example.com`. Sue signs in as `sue@example.com` and resubmits the form in your app to update her email address to `sue2@example.com`. She receives this email, provides the verification code to your app, and begins signing in as `sue2@example.com`.

To require attribute verification when users update their email address or phone number

1. Sign in to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. In the navigation pane, choose **User Pools**, and choose the user pool you want to edit.
3. In the **Sign-up experience** tab, choose **Edit** under **Attribute verification and user account confirmation**.
4. Choose **Keep original attribute value active when an update is pending**.
5. Under **Active attribute values when an update is pending**, choose the attributes that you want to require your users verify before Amazon Cognito updates the value.
6. Choose **Save changes**.

To require attribute update verification with the Amazon Cognito API, you can set the `RequireAttributesVerifiedBeforeUpdate` parameter in an [UpdateUserPool](#) request.

Authorizing Amazon Cognito to send SMS messages on your behalf

To send SMS messages to your users on your behalf, Amazon Cognito needs your permission. To grant that permission, you can create an AWS Identity and Access Management (IAM) role. Under the **MFA and verifications** tab of the Amazon Cognito console, choose **Create role**.

Configuring SMS and email verification messages and user invitation messages

Note

In the new Amazon Cognito console experience, you can customize messages in the **Messaging** tab under the **Message templates** heading.

In the **Message customizations** tab, you can customize:

- Your SMS text message multi-factor authentication (MFA) message

- Your SMS and email verification messages
- The verification type for email—code or link
- Your user invitation messages
- FROM and REPLY-TO email addresses for emails going through your user pool

Note

The SMS and email verification message templates only appear if you have chosen to require phone number and email verification in the **Verifications** tab. Similarly, the SMS MFA message template only appears if the MFA setting is **required** or **optional**.

Topics

- [Message templates \(p. 218\)](#)
- [Customizing the SMS message \(p. 219\)](#)
- [Customizing email verification messages \(p. 219\)](#)
- [Customizing user invitation messages \(p. 220\)](#)
- [Customizing your email address \(p. 220\)](#)
- [Authorizing Amazon Cognito to send Amazon SES email on your behalf \(from a custom FROM email address\) \(p. 221\)](#)

Message templates

You can use message templates to insert fields into your messages using placeholders that the corresponding values replace.

Template placeholders

Description	Token
Verification code	{#####}
Temporary password	{#####}
User name	{username}

Note

You can't use the {username} placeholder in verification email messages. You can use the {username} placeholder in invitation email messages that you generate with the [AdminCreateUser](#) operation. These invitation email messages use two placeholders: the user name, as {username}, and the temporary password, as {#####}.

You can use advanced security template placeholders to do the following:

- Include specific details about an event such as IP address, city, country, sign-in time, and device name. Amazon Cognito advanced security features can analyze these details.
- Verify whether a one-click link is valid.
- Use event ID, feedback token, and user name to build your own one-click link.

Note

To generate one-click links and use the {one-click-link-valid} and {one-click-link-invalid} placeholders in advanced security email templates, you must already have a domain configured for your user pool.

Advanced security template placeholders

Description	Token
IP address	{ip-address}
City	{city}
Country	{country}
Log-in time	{login-time}
Device name	{device-name}
One-click link is valid	{one-click-link-valid}
One-click link is not valid	{one-click-link-invalid}
Event ID	{event-id}
Feedback token	{feedback-token}

Customizing the SMS message

Note

In the new Amazon Cognito console experience, you can customize SMS messages in the **Messaging** tab under the **Message templates** heading.

You can customize the SMS message for multi-factor authentication (MFA) by editing the template under the **Do you want to customize your SMS messages?** heading.

Important

Your custom message must contain the {#####} placeholder. This placeholder is replaced with the authentication code before the message is sent.

The maximum length for the message, including the authentication code, is 140 UTF-8 characters.

Customizing SMS verification messages

You can customize the SMS message for phone number verifications by editing the template under the **Do you want to customize your SMS verification messages?** heading.

Important

Your custom message must contain the {#####} placeholder. This placeholder is replaced with the verification code before the message is sent.

The maximum length for the message, including the verification code, is 140 UTF-8 characters.

Customizing email verification messages

To verify the email address of a user in your user pool with Amazon Cognito, you can send the user an email message with a link that they can select, or you can send them a code that they can enter.

Note

In the new Amazon Cognito console, you can customize email verification messages under the **Message Templates** heading in the **Messaging** tab of your user pool.

To customize the email subject and message content for email address verification messages, edit the template under the **Do you want to customize your email verification messages?** heading.

Important

If you choose code as the verification type, your custom message must contain the {####} placeholder. When you send the message, the verification code replaces this placeholder. If you choose link as the verification type, your custom message must include a placeholder in the format {##Verify Your Email##}. A verification link titled *Verify Your Email* replaces this placeholder.

The maximum length for the message, including the verification code (if present), is 20,000 UTF-8 characters. You can use HTML tags in this message to format the contents.

Customizing user invitation messages

Note

In the new Amazon Cognito console experience, you can customize invitation messages in the **Messaging** tab under the **Message templates** heading.

You can customize the user invitation message that Amazon Cognito sends to new users by SMS or email message by editing the templates under the **Do you want to customize your user invitation messages?** heading.

Important

Your custom message must contain the {username} and {####} placeholders. These placeholders are replaced with the user's username and password before the message is sent.

For SMS, the maximum length for the message, including the verification code, is 140 UTF-8 characters. For email, the maximum length for the message, including the verification code, is 20,000 UTF-8 characters. You may use HTML tags in your email messages to format the contents.

Customizing your email address

By default, Amazon Cognito sends email messages to users in your user pools from the address **no-reply@verificationemail.com**. You can choose to specify custom FROM and REPLY-TO email addresses instead of **no-reply@verificationemail.com**.

To customize the FROM and REPLY-TO email addresses in the AWS console:

Original console

1. Navigate to the [Amazon Cognito console](#), and choose a user pool.
2. Choose **Message customizations**.
3. In **Message customizations**, choose an **SES Region**.
4. In the **FROM email address ARN** field, choose your verified email address from the list. Verify the Amazon Simple Email Service identity by choosing the [Verify an SES identity](#) link below the **FROM email address ARN** field. For more information, see [Verifying email addresses and domains in Amazon SES](#) in the *Amazon Simple Email Service Developer Guide*.
5. To customize the REPLY-TO email address, enter a valid email address in the **REPLY-TO email address** field.

New console

1. Navigate to the [Amazon Cognito console](#), and choose **User Pools**.
2. Choose an existing user pool from the list, or [create a user pool](#).
3. Choose the **Messaging** tab. Under **Email**, choose **Edit**.
4. Choose an **SES Region**.
5. Choose a **FROM email address** from the list of email addresses you have verified with Amazon SES in the **SES Region** you selected. To use an email address from a verified domain, configure

email settings in the AWS Command Line Interface or the AWS API. For more information, see [Verifying email addresses and domains in Amazon SES](#) in the *Amazon Simple Email Service Developer Guide*.

6. Choose a **Configuration set** from the list of configuration sets in your chosen **SES Region**.
7. Enter a friendly **FROM sender name** for your email messages, in the format John Stiles <johnstiles@example.com>.
8. To customize the REPLY-TO email address, enter a valid email address in the **REPLY-TO email address** field.

Authorizing Amazon Cognito to send Amazon SES email on your behalf (from a custom FROM email address)

You can configure Amazon Cognito to send email from a custom FROM email address instead of its default address. To use a custom address, you must give Amazon Cognito permission to send email message from an Amazon SES verified identity. In most cases, you can grant permission by creating a sending authorization policy. For more information, see [Using sending authorization with Amazon SES](#) in the *Amazon Simple Email Service Developer Guide*.

When you configure a user pool to use Amazon SES for email messages, Amazon Cognito creates the `AWSServiceRoleForAmazonCognitoIdpEmailService` role in your account to grant access to Amazon SES. No sending authorization policy is needed when the `AWSServiceRoleForAmazonCognitoIdpEmailService` service-linked role is used. You only need to add a sending authorization policy when you use both the default email functionality in your user pool and a verified Amazon SES identity as the FROM address.

For more information about the service-linked role that Amazon Cognito creates, see [Using service-linked roles for Amazon Cognito \(p. 360\)](#).

The following example sending authorization policy grants Amazon Cognito a limited ability to use an Amazon SES verified identity. Amazon Cognito can only send email messages when it does so on behalf of both the user pool in the `aws:SourceArn` condition and the account in the `aws:SourceAccount` condition. For more examples, see [Amazon SES sending authorization policy examples](#) in the *Amazon Simple Email Service Developer Guide*.

Note

In this example, the "Sid" value is an arbitrary string that uniquely identifies the statement. For more information about policy syntax, see [Amazon SES sending authorization policies](#) in the *Amazon Simple Email Service Developer Guide*.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "stmnt1234567891234",  
            "Effect": "Allow",  
            "Principal": {  
                "Service": [  
                    "email.cognito-idp.amazonaws.com"  
                ]  
            },  
            "Action": [  
                "SES:SendEmail",  
                "SES:SendRawEmail"  
            ],  
            "Resource": "<your SES identity ARN>",  
            "Condition": {  
                "StringEquals": {  
                    "AWS:SourceAccount": "<your account number>"  
                }  
            }  
        }  
    ]  
}
```

```
        },
        "ArnLike": {
            "AWS:SourceArn": "<your identity pool ARN>"
        }
    ]
}
```

The Amazon Cognito console adds a similar policy for you when you select an Amazon SES identity from the drop-down menu. If you use the CLI or API to configure the user pool, you must attach a policy structured like the previous example to your Amazon SES Identity.

Adding cost allocation tags to your user pool

Note

In the new Amazon Cognito console experience, you can manage tags in the **User pool properties** tab of your user pool.

In the **Tags** tab, you can add cost allocation tags to categorize and track your AWS costs. When you apply tags to your AWS resources, such as Amazon Cognito user pools, your AWS cost allocation report includes usage and costs aggregated by tags. You can apply tags that represent business categories such as cost centers, application names, and owners, to organize your costs across multiple services. For more information, see [Using cost allocation tags](#) in the [AWS Billing User Guide](#).

To add a tag, choose **Add tag**. Specify a **Tag key** and **Tag value**, following the restrictions listed in [Tag restrictions](#). Choose **Save changes** to save your tag.

Important

In order for tags to appear on your billing reports, you must activate your applied tags in the billing console. For more information, see [Activating user-defined cost allocation tags](#) in the [AWS Billing User Guide](#).

Specifying user pool device tracking settings

As a way of providing additional security, you can track devices that users have logged in to. This topic describes how to add device tracking to your Amazon Cognito user pools in the AWS Management Console.

Setting up remembered devices

With Amazon Cognito user pools, you can choose to have Amazon Cognito remember devices used to access your application and associate these remembered devices with your application's users in a user pool. You can also choose to use remembered devices to stop sending codes to your users when you have set up multi-factor authentication (MFA). You must use the `USER_SRP_AUTH` authentication flow to use the device tracking feature. You must also enable MFA for your user pool.

When setting up the remembered devices functionality through the Amazon Cognito console, you have three options: **Always**, **User Opt-In**, and **No**.

Note

If you have opted in to the new Amazon Cognito console, your device tracking options are **Always**, **User Opt-In**, and **Don't remember**.

- **No** (default) – Devices are not remembered.
- **Always** – Every device used by your application's users is remembered.
- **User Opt-In** – Your user's device is only remembered if that user opts to remember the device.

If either **Always** or **User Opt-In** is selected, a device identifier (key and secret) will be assigned to each device the first time a user signs in with that device. This key will not be used for anything other than identifying the device, but it will be tracked by the service.

If you select **Always**, Amazon Cognito will use the device identifier (key and secret) to authenticate the device on every user sign-in with that device as part of the user authentication flow.

If you select **User Opt-In**, you can remember devices only when your application's users opt to do so. When a user signs in with a new device, the response from the request to initiate tracking indicates whether the user should be prompted about remembering their device. You must create the user interface to prompt users. If the user opts to have the device remembered, the device status is updated with a 'remembered' state.

The AWS Mobile SDKs have additional APIs to see remembered devices ([ListDevices](#), [GetDevice](#)), mark a device as remembered or not remembered ([UpdateDeviceStatus](#)), and stop tracking a device ([ForgetDevice](#)). In the REST API, there are additional administrator versions of these APIs that have elevated privileges and work on any user. They have API names such as [AdminListDevices](#), [AdminGetDevice](#), and so on. They are not exposed through the SDKs.

Using remembered devices to suppress multi factor authentication (MFA)

If you have selected either **Always** or **User Opt-In**, you also can suppress MFA challenges on remembered devices for the users of your application. To use this feature, you must enable MFA for your user pool. For more information, see [Adding MFA to a user pool \(p. 381\)](#).

Note

If the device remembering feature is set to **Always** and **Do you want to use a remembered device to suppress the second factor during multi-factor authentication (MFA)?** is set to **Yes**, then the MFA settings for medium/high risks in risk-based MFA are ignored.

Configuring a user pool app client

An app is an entity within a user pool that has permission to call unauthenticated API operations. Unauthenticated API operations are those that do not have an authenticated user. Examples include operations to register, sign in, and handle forgotten passwords. To call these API operations, you need an app client ID and an optional client secret. It is your responsibility to secure any app client IDs or secrets so that only authorized client apps can call these unauthenticated operations.

You can create multiple apps for a user pool. Typically, an app corresponds to the platform of an app. For example, you might create an app for a server-side application and a different Android app. Each app has its own app client ID.

When you create an app client in Amazon Cognito, you can pre-populate options based on the standard OAuth client types **public client** and **confidential client**. Configure a **confidential client** with a **client secret**.

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 secret** for unauthenticated API operations. The app might run as a daemon or shell script on your backend server.

Client secret

A client secret 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.

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

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](#) on the AWS blog.

Creating an app client (AWS Management Console)

Original console

To create an app client (console)

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. Choose the **App integration** tab. Locate **App clients** and select **Create an app client**.
5. Choose **Add app client**.
6. Choose **Add an app client**.
7. Enter an **App client name**.
8. Specify the **Refresh token expiration** for the app. The default value is 30 days. You can change this default to any value between 1 hour and 10 years.
9. Specify the app's **Access token expiration**. The default value is 1 hour. You can change this default to any value between 5 minutes and 24 hours.
10. Specify the **ID token expiration** for the app. The default value is 1 hour. You can change this default to any value between 5 minutes and 24 hours.

Important

If you use the hosted UI and set token expiration to less than an hour, your user can get new tokens based on their session cookie, which expires after 1 hour. You can't change the duration of this cookie.

11. By default, user pools generate a client secret for your app. If you do not need a client secret, clear **Generate client secret**.
12. If your server app requires developer credentials (using [Signature Version 4](#)) and doesn't use [Secure remote password \(SRP\) authentication](#), select **Enable username password auth for admin APIs for authentication (ALLOW_ADMIN_USER_PASSWORD_AUTH)** to activate server-side authentication. For more information, see [Admin authentication flow \(p. 368\)](#).
13. Under **Prevent User Existence Errors**, choose **Legacy** or **Enabled**. For more information, see [Managing error response](#).
14. By default, user pools give your app permission to read and write all attributes. If you want to set different permissions for your app, complete the following steps. Otherwise, skip to Step 15.
 - a. Choose **Set attribute read and write permissions**.
 - b. Do either of the following to set read and write permissions:
 - Choose one or more scopes. Each scope is a set of standard attributes. For more information, see the list of [standard OIDC scopes](#).
 - Choose individual standard or custom attributes.

Note

You can't remove required attributes from write permissions in any app.

15. Choose **Create app client**.
16. If you want to create another app, choose **Add an app**.
17. Once you've created all the apps you want, choose **Return to pool details**, update any other fields, and then choose **Create pool**.

New console

To create an app client (console)

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. Select the **App integration** tab.
5. Under **App clients**, select **Create an app client**.
6. Select an **App type**: **Public client**, **Confidential client**, or **Other**.
7. Enter an **App client name**.
8. Select the **Authentication flows** you want to allow in your app client.
9. (Optional) If you want to configure token expiration, complete the following steps:
 - a. Specify the **Refresh token expiration** for the app client. The default value is 30 days. You can change it to any value between 1 hour and 10 years.
 - b. Specify the **Access token expiration** for the app client. The default value is 1 hour. You can change it to any value between 5 minutes and 24 hours.
 - c. Specify the **ID token expiration** for the app client. The default value is 1 hour. You can change it to any value between 5 minutes and 24 hours.

Important

If you use the hosted UI and configure a token lifetime of less than an hour, your user will be able to use tokens based on their session cookie duration, which is currently fixed at one hour.

10. Choose **Generate client secret** to have Amazon Cognito generate a client secret for you. Client secrets are typically associated with confidential clients.
11. Choose whether you will **Enable token revocation** for this app client. This will increase the size of tokens that Amazon Cognito issues.
12. Choose whether you will **Prevent error messages that reveal user existence** for this app client. Amazon Cognito will respond to sign-in requests for nonexistent users with a generic message stating that either the user name or password was incorrect.
13. (Optional) Configure **Attribute read and write permissions** for this app client. Your app client can have permission to read and write a limited subset of your user pool's attribute schema.
14. Choose **Create**.
15. Note the **Client id**. This will identify the app client in sign-up and sign-in requests.

To create and update app clients in a user pool (API, AWS CLI)

Do one of the following:

- **API** – Use the [CreateUserPoolClient](#) and [UpdateUserPoolClient](#) operations.

- **AWS CLI** – At the command line, run the `create-user-pool-client` and `update-user-pool-client` commands.

Configuring user pool Lambda triggers

You can use AWS Lambda triggers to customize workflows and the user experience with Amazon Cognito. You can create the following Lambda triggers: **Pre sign-up**, **Pre authentication**, **Custom message**, **Post authentication**, **Post confirmation**, **Define Auth Challenge**, **Create Auth Challenge**, **Verify Auth Challenge Response**, and **User Migration**.

A user migration Lambda trigger allows easy migration of users from your existing user management system into your user pool.

For examples of each Lambda trigger, see [Customizing user pool workflows with Lambda triggers \(p. 92\)](#).

Note

The **Custom message** AWS Lambda trigger is an advanced way to customize messages for email and SMS. For more information, see [Customizing user pool workflows with Lambda triggers \(p. 92\)](#).

Reviewing your user pool creation settings

Before you create your user pool, you can review the different settings and edit them in the AWS Management Console. Amazon Cognito validates the user pool settings and warns you if something needs to be changed. For example:

Warning

This user pool does not have an IAM role defined to allow Amazon Cognito to send SMS messages, so it will not be able to confirm phone numbers or for MFA after August 31, 2016. You can define the IAM role by selecting a role on the **Verifications** panel.

If you see a message, follow the instructions to fix them before choosing **Create pool**.

Configuring user pool analytics

Note

The **Analytics** tab appears only when you're editing an existing user pool.

Using Amazon Pinpoint Analytics, you can track Amazon Cognito user pools sign-ups, sign-ins, failed authentications, daily active users (DAUs), and monthly active users (MAUs). You can also set up user attributes specific to your app using the AWS Mobile SDK for Android or AWS Mobile SDK for iOS. Those can then be used to segment your users in Amazon Pinpoint and send them targeted push notifications.

In the **Analytics** tab, you can specify an Amazon Pinpoint project for your Amazon Cognito app client. For more information, see [Using Amazon Pinpoint Analytics with Amazon Cognito user pools \(p. 145\)](#).

Note

Amazon Pinpoint is available in several AWS Regions in North America, Europe, Asia, and Oceania. Amazon Pinpoint Regions include the Amazon Pinpoint API. If an Amazon Pinpoint region is supported by Amazon Cognito, then Amazon Cognito will send events to Amazon Pinpoint projects within the *same* Amazon Pinpoint region. If a region *isn't* supported by Amazon Pinpoint, then Amazon Cognito will *only* support sending events in us-east-1. For Amazon Pinpoint detailed region information, see [Amazon Pinpoint endpoints and quotas](#) and [Using Amazon Pinpoint analytics with Amazon Cognito user pools](#).

To add analytics and campaigns

1. Choose **Add analytics and campaigns**.

2. Choose a **Cognito app client** from the list.
3. To map your Amazon Cognito app to an **Amazon Pinpoint project**, choose the Amazon Pinpoint project from the list.

Note

The Amazon Pinpoint project ID is a 32-character string that is unique to your Amazon Pinpoint project. It is listed the Amazon Pinpoint console.

You can map multiple Amazon Cognito apps to a single Amazon Pinpoint project. However, each Amazon Cognito app can only be mapped to one Amazon Pinpoint project.

In Amazon Pinpoint, each project should be a single app. For example, if a game developer has two games, each game should be a separate Amazon Pinpoint project, even if both games use the same Amazon Cognito user pools.

4. Choose **Share user attribute data with Amazon Pinpoint** if you want Amazon Cognito to send email addresses and phone numbers to Amazon Pinpoint in order to create additional endpoints for users.

Note

An *endpoint* uniquely identifies a user device where you can send push notifications with Amazon Pinpoint. For more information about endpoints, see [Adding endpoints to Amazon Pinpoint](#) in the *Amazon Pinpoint Developer Guide*.

5. Enter an **IAM role** that you already created or choose **Create new role** to create a new role in the IAM console.
6. Choose **Save changes**.
7. To specify additional app mappings, choose **Add app mapping**.
8. Choose **Save changes**.

Configuring app client settings

Original console

Note

The **General settings** tab only appears when you're editing an existing user pool.

On the **General settings** tab, you must configure at least one identity provider (IdP) for your apps if you want to use the built-in hosted pages to sign up and sign in users, or if you want to use OAuth2.0 flows. For more information, see [Configuring a user pool app client \(p. 42\)](#).

To specify app client settings for your user pool

1. In **Enabled Identity Providers**, select the identity providers you want to use for the apps that you configured in the **App Clients** tab.
2. Enter the **Callback URLs** you want, separated by commas. These URLs apply to all of the selected identity providers.

Note

You must register the URLs in the console, or by using the AWS CLI or API, before you can use them in your app.

3. Enter the **Sign out URLs** you want, separated by commas.

Note

You must register the URLs in the console, or by using the CLI or API, before you can use them in your app.

4. Under **OAuth 2.0**, select the from the following options. For more information, see [App client settings terminology \(p. 45\)](#) and the [OAuth 2.0 specification](#).
 - For **Allowed OAuth Flows**, select **Authorized code grant** and **Implicit grant**. Select **Client credentials** only if your app needs to request access tokens on its own behalf, not on behalf of a user.

- For **Allowed OAuth Scopes**, select the scopes you want. Each scope is a set of one or more standard attributes.
- For **Allowed Custom Scopes**, select the scopes you want from any custom scopes that you have defined. Custom scopes are defined in the **Resource Servers** tab. For more information, see [Defining resource servers for your user pool \(p. 57\)](#).

New console

On the **Sign-in experience** tab, you must configure at least one **Federated identity provider sign-in** identity provider (IdP) if you want to use the built-in hosted pages to sign up and sign in users, or if you want to use OAuth2.0 flows. For more information, see [Configuring a user pool app client \(p. 42\)](#).

Configure the app

1. In the **App integration** tab, select your app client under **App clients**. Review your current **Hosted UI** information.
2. Add a **callback URL** under **Allowed callback URL(s)**. A callback URL is where the user is redirected to after a successful sign-in.
3. Add a **sign-out URL** under **Allowed sign-out URL(s)**. A sign-out URL is where your user is redirected to after signing out.
4. Add at least one from the list of **Identity providers**.
5. Under **OAuth 2.0 grant types**, select **Authorization code grant** to return an authorization code that is then exchanged for user pool tokens. Because the tokens are never exposed directly to an end user, they are less likely to become compromised. However, a custom application is required on the backend to exchange the authorization code for user pool tokens. For security reasons, we recommend that you use the authorization code grant flow, together with [Proof key for code Exchange \(PKCE\)](#), for mobile apps.
6. Under **OAuth 2.0 grant types**, select **Implicit grant** to have user pool JSON web tokens (JWT) returned to you from Amazon Cognito. You can use this flow when there's no backend available to exchange an authorization code for tokens. It's also helpful for debugging tokens.
7. You can enable both the **Authorization code** and the **Implicit code** grants, and then use each grant as needed. If neither **Authorization code** or **Implicit code** grants are selected and your app client has a client secret, you can enable **Client credentials** grants. Select **Client credentials** only if your app needs to request access tokens on its own behalf, not on behalf of a user.
8. Select the **OpenID Connect scopes** that you want to authorize for this app client.
9. Choose **Save changes**.

Adding a domain name for your user pool

Original console

Note

The **Domain name** tab only appears while you edit an existing user pool.

On the **Domain name** tab, you can enter your own prefix domain name. The domain for your app is https://<domain_prefix>.auth.<region>.amazoncognito.com.

The full URL for your app looks like this example:

https://example.auth.us-east-1.amazoncognito.com/login?redirect_uri=https://www.google.com&response_type=code&client_id=<client_id_value>

For more information, see [Configuring a user pool domain \(p. 47\)](#).

Important

Before you can access the URL for your app, you must specify app client settings such as callback and redirect URLs. For more information, see [Configuring app client settings \(p. 227\)](#).

To specify a domain name for your user pool

1. Enter the prefix domain name that you want in the **Prefix domain name** box.
2. Choose **Check availability** to verify that your desired prefix domain name is available.
3. If the prefix domain name is available for use, choose **Save changes**.

New console

Note

The **Domain name** tab only appears while you edit an existing user pool.

On the **App integration** tab, create an Amazon-owned prefix domain or use a custom domain for your user pool. The prefix domain for your app is `https://<domain_prefix>.auth.<region>.amazoncognito.com`.

The full URL for your app looks like this example:

`https://example.auth.us-east-1.amazoncognito.com/login?redirect_uri=https://www.google.com&response_type=code&client_id=<client_id_value>`

For more information, see [Configuring a user pool domain \(p. 47\)](#).

Important

Before you can access the hosted UI for your app, you must specify app client settings such as callback and redirect URLs. For more information, see [Configuring app client settings \(p. 227\)](#).

Configure a domain

1. Navigate to the **App integration** tab for your user pool.
2. Next to **Domain**, choose **Actions** and select either **Create custom domain** or **Create Cognito domain**. If you have already configured a user pool domain, choose **Delete Cognito domain** or **Delete custom domain** before creating your new custom domain.
3. Enter an available domain prefix to use with a **Cognito domain**. Your prefix can't contain the strings aws, amazon, or cognito. For information on setting up a **Custom domain**, see [Using your own Domain for the hosted UI](#)
4. Choose **Create**.

Customizing the built-in app UI to sign up and sign in users

Original console

Note

The **UI customization** tab only appears when you're editing an existing user pool.

On the **UI customization** tab, you can add your own customizations to the default app UI. For detailed information about each of the customization fields, see [Customizing the built-in sign-in and sign-up webpages \(p. 53\)](#).

Note

You can view the hosted UI with your customizations by constructing the following URL, with the specifics for your user pool, and entering it into a browser: `https://<your_domain>/login?response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>`

You may have to wait up to one minute to refresh your browser before changes made in the console appear.

Your domain is shown on the **Domain name** tab. Your app client ID and callback URL are shown on the **General settings** tab.

To customize the built-in app UI

1. Under **App client to customize**, choose from the previously created app clients in the dropdown menu.
2. To add a logo to the default app UI, choose **Choose a file**, or drag a file onto the **Logo** box.
3. Under **CSS customizations (optional)**, you can customize the appearance of the app by changing the CSS properties for the UI from their default values.
4. Choose **Save changes**.

New console

With **Hosted UI customization** in the **App integration** tab, you can add your own customizations to the default app UI.

You can also customize the hosted UI for an app client. Choose the app client from the **App integration** tab and locate **Hosted UI customization**. Choose **Use client-level settings** to enter the UI customization screen for that app client only.

For detailed information about each of the customization fields, see [Customizing the built-in sign-in and sign-up webpages \(p. 53\)](#).

Note

You can view the hosted UI with your customizations by constructing the following URL, with the specifics for your user pool, and

typing it into a browser: `https://<your_domain>/login?response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>`

You may have to wait up to one minute to refresh your browser before changes made in the console appear.

Your domain is shown on the **Domain name** tab. Your app client ID and callback URL are shown on the **General settings** tab.

You can also launch the hosted UI for an app client that has it enabled. Choose the app client in the **App integration** tab and locate **Hosted UI settings**. Choose **View hosted UI**.

To customize the built-in app UI

1. To upload your own logo image file, choose **Choose file** or **Replace current file**.
2. To customize the CSS for the hosted UI, download **CSS template.css** and modify the template with the values you wish to customize. Only the keys that are included in the template can be used with the hosted UI. Added CSS keys will not be reflected in your UI. After you have customized the CSS, choose **Choose file** or **Replace current file** to upload your custom CSS.
3. Choose **Save changes**.

Adding resource servers for your user pool

A *resource server* is a server for access-protected resources. It handles authenticated requests from an app that has an access token. A *scope* is a level of access that an app can request to a resource.

Original console

Note

The **Resource Servers** tab appears only when you're editing an existing user pool.

In the **Resource Servers** tab, you can define custom resource servers and scopes for your user pool. For more information, see [Defining resource servers for your user pool \(p. 57\)](#).

To define a custom resource server

1. Choose **Add a resource server**.
2. Enter the name of your resource server, for example, `Photo Server`.
3. Enter the identifier of your resource server, for example, `com.example.photos`.
4. Enter the names of the custom scopes for your resources, such as `read` and `write`.
5. For each of the scope names, enter a description, such as `view your photos` and `update your photos`.

Each of the custom scopes that you define appears on the **App client settings** tab, under **OAuth2.0 Allowed Custom Scopes**; for example `com.example.photos/read`.

New console

Note

You can add **Resource servers** to your user pool after you create it. Resource server configuration is not included in the new user pool wizard.

In the **App integration** tab, locate **Resource servers**, where you can define custom resource servers and scopes for your user pool. For more information, see [Defining resource servers for your user pool \(p. 57\)](#).

To define a custom resource server

1. Choose **Create a resource server**.
2. Enter a **Resource server name**, for example, `Photo Server`.
3. Enter a **Resource server identifier**, for example, `com.example.photos`.
4. Choose **Add a custom scope** and enter a **Scope name**, such as `read` and `write`.
5. For each **Scope name**, enter a **Description**, such as `view your photos` and `update your photos`.

Each of the custom scopes that you define appears on the **App integration** tab under **Resource servers**. Hover over the number in the **Custom scopes** column to display all scopes and descriptions associate with the resource server.

Configuring identity providers for your user pool

Note

The **Identity providers** tab appears only when you're editing an existing user pool.

In the **Identity providers** tab, you can specify identity providers (IdPs) for your user pool. For more information, see [Adding user pool sign-in through a third party \(p. 60\)](#).

Topics

- [Set up user sign-in with a social IdP \(p. 232\)](#)
- [Set up user sign-in with an OIDC IdP \(p. 233\)](#)
- [Set up user sign-in with a SAML IdP \(p. 235\)](#)

Set up user sign-in with a social IdP

You can use federation to integrate Amazon Cognito user pools with social identity providers such as Facebook, Google, and Login with Amazon.

To add a social identity provider, you first create a developer account with the identity provider. After you have your developer account, register your app with the identity provider. The identity provider creates an app ID and an app secret for your app, and you configure those values in your Amazon Cognito user pools.

Here are links to help you get started with social identity providers:

- [Google identity platform](#)
- [Facebook for developers](#)
- [Login with Amazon](#)
- [Sign in with Apple](#)

Original console

To integrate user sign-in with a social IdP

1. Choose a social identity provider such as **Facebook**, **Google**, **Login with Amazon**, or **Sign In with Apple**.
2. Enter your social identity provider's information by completing one of the following steps, based on your choice of IdP:

Facebook, Google, and Login with Amazon

Enter the app ID and app secret that you received when you created your client app.

Sign In with Apple

Enter the services ID that you provided to Apple, and the team ID, key ID, and private key you received when you created your app client.

3. For **App secret**, enter the app secret that you received when you created your client app.
4. For **Authorized scopes**, enter the names of the social identity provider scopes that you want to map to user pool attributes. Scopes define which user attributes, such as name and email, you want to access with your app. When entering scopes, use the following guidelines based on your choice of IdP:

- **Facebook** — Separate scopes with commas. For example:

`public_profile, email`

- **Google, Login with Amazon, and Sign In with Apple** — Separate scopes with spaces. For example:

`Google: profile email openid`

`Login with Amazon: profile postal_code`

`Sign In with Apple: name email`

Note

For Sign In with Apple (console), use the check boxes to choose scopes.

Your user will be asked to consent to providing these attributes to your app. For more information about each social identity provider's scopes, see the documentation from Google, Facebook, Login with Amazon, or Sign In with Apple.

5. Choose **Enable Facebook**, **Enable Google**, **Enable Login with Amazon**, or **Enable Sign in with Apple**.

New console

To integrate user sign-in with a social IdP

1. Sign in to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. In the navigation pane, choose **User Pools**, and choose the user pool you want to edit.
3. Choose the **Sign-in experience** tab and locate **Federated sign-in**.
4. Choose **Add an identity provider**, or choose the **Facebook**, **Google**, **Amazon**, or **Apple** identity provider you have configured, locate **Identity provider information**, and choose **Edit**. For more information about adding a social identity provider, see [Adding social identity providers to a user pool \(p. 62\)](#).
5. Enter your social identity provider's information by completing one of the following steps, based on your choice of IdP:

Facebook, Google, and Login with Amazon

Enter the app ID and app secret that you received when you created your client app.

Sign In with Apple

Enter the service ID that you provided to Apple, and the team ID, key ID, and private key you received when you created your app client.

6. For **Authorized scopes**, enter the names of the social identity provider scopes that you want to map to user pool attributes. Scopes define which user attributes, such as name and email, that you want to access with your app. When entering scopes, use the following guidelines based on your choice of IdP:

- **Facebook** — Separate scopes with commas. For example:

`public_profile, email`

- **Google, Login with Amazon, and Sign In with Apple** — Separate scopes with spaces. For example:

- **Google:** profile email openid

- **Login with Amazon:** profile postal_code

- **Sign In with Apple:** name email

Note

For Sign In with Apple (console), use the check boxes to choose scopes.

7. Choose **Save changes**.
8. From the **App client integration** tab, choose one of the **App clients** in the list and then choose **Edit hosted UI settings**. Add the new social identity provider to the app client under **Identity providers**.
9. Choose **Save changes**.

For more information on social IdPs, see [Adding social identity providers to a user pool \(p. 62\)](#).

Set up user sign-in with an OIDC IdP

You can integrate user sign-in with an OpenID Connect (OIDC) identity provider (IdP) such as Salesforce or Ping Identity.

Original console

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **Manage User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. On the left navigation bar, choose **Identity providers**.
5. Choose **OpenId Connect**.
6. Enter a unique name into **Provider name**.
7. Enter the OIDC IdP's client ID into **Client ID**.
8. Enter the OIDC IdP's client secret into **Client secret**.
9. In the drop-down list, select the HTTP method (either GET or POST) that Amazon Cognito uses to fetch user details from the **userInfo endpoint** into **Attributes request method**.
10. Enter the names of the scopes that you want to authorize. Scopes define which user attributes (such as `name` and `email`) that you want to access with your application. Scopes are separated by spaces, according to the [OAuth 2.0 specification](#).

Your app user is asked to consent to providing these attributes to your application.

11. Enter the URL of your IdP and choose **Run discovery**. For example, Salesforce uses this URL:

`https://login.salesforce.com`

Note

The URL should start with `https://`, and shouldn't end with a slash `/`.

- If **Run discovery** isn't successful, then you must provide the **Authorization endpoint**, **Token endpoint**, **Userinfo endpoint**, and **Jwks uri** (the location of the [JSON web key](#)).

12. Choose **Create provider**.

13. On the left navigation bar, choose **App client settings**.

14. Select your OIDC provider as one of the **Enabled Identity Providers**.

15. Enter a callback URL for the Amazon Cognito authorization server to call after users are authenticated. This is the URL of the page where Amazon Cognito directs your user after a successful sign-in.

`https://www.example.com`

16. Under **Allowed OAuth Flows**, activate both the **Authorization code grant** and the **Implicit code grant**.

Unless you specifically want to exclude one, select the check boxes for all of the **Allowed OAuth scopes**.

17. Choose **Save changes**.

18. On the **Attribute mapping** tab on the left navigation bar, add mappings of OIDC claims to user pool attributes.

- a. As a default, the OIDC claim **sub** is mapped to the user pool attribute **Username**. You can map other OIDC [claims](#) to user pool attributes. Enter the OIDC claim, and select the corresponding user pool attribute from the drop-down list. For example, the claim `email` is typically mapped to the user pool attribute `Email`.
- b. In the drop-down list, select the destination user pool attribute.
- c. Choose **Save changes**.
- d. Choose **Go to summary**.

New console

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **User Pools** from the navigation menu.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. Choose the **Sign-in experience** tab. Locate **Federated sign-in** and select **Add an identity provider**.
5. Choose an **OpenID Connect** identity provider.
6. Enter a unique name into **Provider name**.
7. Enter the client ID that you received from your provider into **Client ID**.
8. Enter the client secret that you received from your provider into **Client secret**.
9. Enter **Authorized scopes** for this provider. Scopes define which groups of user attributes (such as name and email) that your application will request from your provider. Scopes must be separated by spaces, following the [OAuth 2.0 specification](#).

Your user must consent to provide these attributes to your application.

10. Choose an **Attribute request method** to provide Amazon Cognito with the HTTP method (either GET or POST) that Amazon Cognito uses to fetch the details of the user from the **userInfo** endpoint operated by your provider.
11. Choose a **Setup method** to retrieve OpenID Connect endpoints either by **Auto fill through issuer URL** or **Manual input**. Use **Auto fill through issuer URL** when your provider has a public `.well-known/openid-configuration` endpoint where Amazon Cognito can retrieve the URLs of the `authorization`, `token`, `userInfo`, and `jwks_uri` endpoints.
12. Enter the issuer URL or `authorization`, `token`, `userInfo`, and `jwks_uri` endpoint URLs from your IdP.

Note

You can use only port numbers 443 and 80 with discovery, auto-filled, and manually entered URLs. User logins fail if your OIDC provider uses any nonstandard TCP ports.

The issuer URL must start with `https://`, and must not end with a / character. For example, Salesforce uses this URL:

`https://login.salesforce.com`

The `openid-configuration` document associated with your issuer URL must provide HTTPS URLs for the following values: `authorization_endpoint`, `token_endpoint`, `userinfo_endpoint`, and `jwks_uri`. Similarly, when you choose **Manual input**, you can only enter HTTPS URLs.

13. The OIDC claim **sub** is mapped to the user pool attribute **Username** by default. You can map other OIDC [claims](#) to user pool attributes. Enter the OIDC claim, and select the corresponding user pool attribute from the drop-down list. For example, the claim **email** is often mapped to the user pool attribute **Email**.
14. Map additional attributes from your identity provider to your user pool. For more information, see [Specifying Identity Provider attribute mappings for your user pool](#).
15. Choose **Create**.
16. From the **App client integration** tab, select one of the **App clients** in the list and **Edit hosted UI settings**. Add the new OIDC identity provider to the app client under **Identity providers**.
17. Choose **Save changes**.

For more information on OIDC IdPs, see [Adding OIDC identity providers to a user pool \(p. 79\)](#).

Set up user sign-in with a SAML IdP

You can use federation for Amazon Cognito user pools to integrate with a SAML identity provider (IdP). You supply a metadata document, either by uploading the file or by entering a metadata document

endpoint URL. For information about obtaining metadata documents for third-party SAML IdPs, see [Integrating third-party SAML identity providers with Amazon Cognito user pools \(p. 76\)](#).

Original console

To integrate user sign-in with a SAML IdP

1. Choose **SAML** to display the SAML identity provider options.
2. To upload a metadata document, choose **Select file**, or enter a metadata document endpoint URL. The metadata document must be a valid XML file.
3. Enter your **SAML Provider name**, for example, "SAML_provider_1", and any **Identifiers** you want. The provider name is required; the identifiers are optional. For more information, see [Adding SAML identity providers to a user pool \(p. 68\)](#).
4. Select **Enable IdP sign out flow** when you want your user to be logged out from a SAML IdP when logging out from Amazon Cognito.

Enabling this flow sends a signed logout request to the SAML IdP when the [Logout endpoint \(p. 432\)](#) is called.

Note

If you select this option and your SAML IdP expects a signed logout request, you also must configure the signing certificate provided by Amazon Cognito with your SAML IdP. The SAML IdP will process the signed logout request and logout your user from the Amazon Cognito session.

5. Choose **Create provider**.
6. To create additional providers, repeat the previous steps.

New console

To configure a SAML 2.0 identity provider in your user pool

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. Choose the **Sign-in experience** tab. Locate **Federated sign-in** and select **Add an identity provider**.
5. Choose a **SAML** identity provider.
6. Enter **Identifiers** separated by commas. An identifier directs Amazon Cognito to check the user sign-in email address, and then direct the user to the provider that corresponds to their domain.
7. Choose **Add sign-out flow** if you want Amazon Cognito to send signed sign-out requests to your provider when a user logs out. Configure your SAML 2.0 identity provider to send sign-out responses to the `https://<your Amazon Cognito domain>/saml2/logout` endpoint that Amazon Cognito creates when you configure the hosted UI. The `saml2/logout` endpoint uses POST binding.

Note

If you select this option and your SAML identity provider expects a signed logout request, you also must configure the signing certificate provided by Amazon Cognito with your SAML IdP.

The SAML IdP will process the signed logout request and logout your user from the Amazon Cognito session.

8. Choose a **Metadata document source**. If your identity provider offers SAML metadata at a public URL, you can choose **Metadata document URL** and enter that public URL. Otherwise, choose **Upload metadata document** and select a metadata file you downloaded from your provider earlier.

Note

If your provider has a public endpoint, we recommend that you enter a metadata document URL, rather than uploading a file. If you use the URL, Amazon Cognito refreshes metadata automatically. Typically, metadata refresh happens every 6 hours or before the metadata expires, whichever is earlier.

9. **Map attributes between your SAML provider and your app** to map SAML provider attributes to the user profile in your user pool. Include your user pool required attributes in your attribute map.

For example, when you choose **User pool attribute** `email`, enter the SAML attribute name as it appears in the SAML assertion from your identity provider. Your identity provider might offer sample SAML assertions for reference. Some identity providers use simple names, such as `email`, while others use URL-formatted attribute names similar to:

```
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
```

10. Choose **Create**.

Note

If you see `InvalidArgumentException` while creating a SAML IdP with an HTTPS metadata endpoint URL, make sure that the metadata endpoint has SSL correctly set up and that there is a valid SSL certificate associated with it. One example of such an exception would be "Error retrieving metadata from <metadata endpoint>".

To set up the SAML IdP to add a signing certificate

- To get the certificate containing the public key that the IdP uses to verify the signed logout request, choose **Show signing certificate** under **Active SAML Providers** on the **SAML** dialog under **Identity providers** on the **Federation** console page.

For more information on SAML IdPs see [Adding SAML identity providers to a user pool \(p. 68\)](#).

Configuring attribute mapping for your user pool

Original console

Note

The **Attribute mapping** tab only appears when you're editing an existing user pool.

On the **Attribute mapping** tab, you can map identity provider (IdP) attributes or assertions to user pool attributes. For more information, see [Specifying identity provider attribute mappings for your user pool \(p. 86\)](#).

Note

Currently, only the Facebook `id`, Google `sub`, Login with Amazon `user_id`, and Sign in with Apple `sub` attributes can be mapped to the Amazon Cognito user pools `username` attribute.

Note

The attribute in the user pool must be large enough to contain the values of the mapped identity provider attributes, or an error will occur when users sign in. Custom attributes should be set to the maximum 2048 character size if mapped to identity provider tokens. You must create mappings for any attributes that are required for your user pool.

To specify a social identity provider attribute mapping for your user pool

1. Choose the **Facebook**, **Google**, **Amazon**, or **Apple** tab.

2. For each attribute you need to map, complete the following steps:
 - a. Select the **Capture** check box.
 - b. In the **User pool attribute** field, select the user pool attribute from the drop-down list to map to the social identity provider attribute.
 - c. For Facebook, Google, and Login with Amazon, if you need more attributes, choose either **Add Facebook attribute**, **Add Google attribute** or **Add Amazon attribute**, and then perform the following steps:

Note
Sign in with Apple does not provide additional attributes at this time.

 - i. In the **Facebook attribute**, **Google attribute** or **Amazon attribute** field, enter the name of the attribute to map.
 - ii. For **User pool attribute**, choose the user pool attribute from the drop-down list that you want to map to the social identity provider attribute. - d. Choose **Save changes**.

To specify a SAML provider attribute mapping for your user pool

1. Choose the **SAML** tab.
2. For each attribute you need to map, complete the following steps:
 - a. Choose **Add SAML attribute**.
 - b. In the **SAML attribute** field, enter the name of the SAML attribute to map.
 - c. For **User pool attribute**, choose the user pool attribute from the drop-down list that you want to map to the SAML attribute.
 - d. Choose **Save changes**.

New console

On the **Attribute mapping** tab, you can map identity provider (IdP) attributes or assertions to user pool attributes. For more information, see [Specifying identity provider attribute mappings for your user pool \(p. 86\)](#).

Note

Currently, only the Facebook `id`, Google `sub`, Login with Amazon `user_id`, and Sign in with Apple `sub` attributes can be mapped to the Amazon Cognito User Pools `username` attribute.

Note

The attribute in the user pool must be large enough to contain the values of the mapped identity provider attributes, or an error will occur when users sign in. Custom attributes should be set to the maximum 2048 character size if mapped to identity provider tokens. You must create mappings for any attributes that are required for your user pool.

To specify a social identity provider attribute mapping for your user pool

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **User Pools** from the navigation menu.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. Choose the **Sign-in experience** tab. Locate **Federated sign-in**.
5. Choose **Add an identity provider**, or choose the **Facebook**, **Google**, **Amazon** or **Apple** identity provider you have configured. Locate **Attribute mapping**, and choose **Edit**. For more information about adding a social identity provider, see [Adding social identity providers to a user pool \(p. 62\)](#).

6. For each attribute you need to map, complete the following steps:
 - a. Select an attribute from the **User pool attribute** column. This is the attribute that is assigned to the user profile in your user pool. Custom attributes are listed after standard attributes.
 - b. Select an attribute from the **<provider> attribute** column. This will be the attribute passed from the provider directory. Known attributes from the social provider are provided in a drop-down list.
 - c. To map additional attributes between your IdP and Amazon Cognito, choose **Add another attribute**.
7. Choose **Save changes**.

To specify a SAML provider attribute mapping for your user pool

1. Sign in to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. In the navigation pane, choose **User Pools**, and choose the user pool you want to edit.
3. Choose the **Sign-in experience** tab and locate **Federated sign-in**.
4. Choose **Add an identity provider**, or choose the SAML identity provider you have configured. Locate **Attribute mapping**, and choose **Edit**. For more information about adding a SAML identity provider, see [Adding SAML identity providers to a user pool \(p. 68\)](#).
5. For each attribute you need to map, complete the following steps:
 - a. Select an attribute from the **User pool attribute** column. This is the attribute that is assigned to the user profile in your user pool. Custom attributes are listed after standard attributes.
 - b. Select an attribute from the **SAML attribute** column. This will be the attribute passed from the provider directory.
6. Choose **Save changes**.

`http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

- c. To map additional attributes between your IdP and Amazon Cognito, choose **Add another attribute**.

Managing error responses

Amazon Cognito supports customizing error responses returned by User Pools. Custom error responses are available for authentication, confirmation, and password recovery-related operations. Use the `PreventUserExistenceErrors` setting of a user pool app client to enable or disable user existence related errors.

When you enable custom error responses, Amazon Cognito authentication APIs return a generic authentication failure response. The error response tells you the user name or password is incorrect. Amazon Cognito account confirmation and password recovery APIs return a response indicating a code was sent to a simulated delivery medium. The error response works when the status is `ENABLED` and the user doesn't exist. Below are the detailed behaviors for the Amazon Cognito operations when `PreventUserExistenceErrors` is set to `ENABLED`.

User authentication operations

You can use either authentication flow method with the following operations.

- AdminInitiateAuth
- AdminRespondToAuthChallenge
- InitiateAuth
- RespondToAuthChallenge

User name password based authentication

In the authentication flows for ADMIN_USER_PASSWORD_AUTH and USER_PASSWORD_AUTH the user name and password returns with a single call of `InitiateAuth`. Amazon Cognito returns a generic `NotAuthorizedException` error indicating either the user name or password is incorrect.

Secure Remote Password (SRP) based authentication

In the USER_SRP_AUTH authentication flow Amazon Cognito receives a user name and SRP parameter 'A' in the first step. In response, Amazon Cognito returns SRP parameter 'B' and 'salt' for the user as per SRP protocol. When a user isn't found, Amazon Cognito returns a simulated response in the first step as described in [RFC 5054](#). Amazon Cognito returns the same 'salt' and internal user id in [Universally Unique Identifier \(UUID\)](#) format for the same user name and user pool combination. When the next operation of `RespondToAuthChallenge` proof of password runs, Amazon Cognito returns a generic `NotAuthorizedException` error indicating either user name or password was incorrect.

Note

You can use `UsernamePassword` to simulate a generic response if you are using verification based aliases and the format of immutable user name isn't a UUID.

`ForgotPassword`

When a user isn't found, is deactivated, or doesn't have a verified delivery mechanism to recover their password, Amazon Cognito returns `CodeDeliveryDetails` with a simulated delivery medium for a user. The simulated delivery medium is determined by the input user name format and verification settings of the user pool.

`ConfirmForgotPassword`

Amazon Cognito returns the `CodeMismatchException` error for users that don't exist or are disabled. If a code isn't requested when using `ForgotPassword`, Amazon Cognito returns the `ExpiredCodeException` error.

`ResendConfirmationCode`

Amazon Cognito returns `CodeDeliveryDetails` for a disabled user or a user that doesn't exist. Amazon Cognito sends a confirmation code to the existing user's email or phone number.

`ConfirmSignUp`

`ExpiredCodeException` returns if a code has expired. Amazon Cognito returns `NotAuthorizedException` when a user isn't authorized. If the code doesn't match what the server expects Amazon Cognito returns `CodeMismatchException`.

`SignUp`

The `SignUp` operation returns `UsernameExistsException` when a user name is already taken. To prevent the `UsernameExistsException` error for email or phone number during `SignUp`, you can use verification based aliases. For more information, see [AliasAttributes](#) Amazon Cognito API Reference guide. For more information about aliases see [Overview of Aliases](#).

Imported users

If `PreventUserExistenceErrors` is enabled, during authentication of imported users a generic `NotAuthorizedException` error is returned indicating either the user name or password was

incorrect instead of returning `PasswordResetRequiredException`. See [Requiring imported users to reset their passwords](#) for more information.

Migrate user Lambda trigger

Amazon Cognito returns a simulated response for users that don't exist when an empty response was set in the original event context by the Lambda trigger. For more information, see [Migrate User Lambda Trigger](#).

Custom Authentication Challenge Lambda trigger

If you use [Custom Authentication Challenge Lambda Trigger](#) and you enable error responses, then `LambdaChallenge` returns a boolean parameter named `UserNotFound`. Then it's passed in the request of `DefineAuthChallenge`, `VerifyAuthChallenge`, and `CreateAuthChallenge` Lambda triggers. You can use this trigger to simulate custom authorization challenges for a user that doesn't exist. If you call the Pre-Authentication Lambda trigger for a user that doesn't exist, then Amazon Cognito returns `UserNotFound`.

Amazon Cognito identity pools (federated identities)

Amazon Cognito identity pools (federated identities) enable you to create unique identities for your users and federate them with identity providers. With an identity pool, you can obtain temporary, limited-privilege AWS credentials to access other AWS services. Amazon Cognito identity pools support the following identity providers:

- Public providers: [Login with Amazon \(identity pools\) \(p. 281\)](#), [Facebook \(identity pools\) \(p. 276\)](#), [Google \(identity pools\) \(p. 284\)](#), [Sign in with Apple \(identity pools\) \(p. 291\)](#).
- Amazon Cognito user pools ([p. 22](#))
- Open ID Connect providers (identity pools) ([p. 295](#))
- SAML identity providers (identity pools) ([p. 297](#))
- Developer authenticated identities (identity pools) ([p. 298](#))

For information about Amazon Cognito identity pools region availability, see [AWS Service Region Availability](#).

For more information about Amazon Cognito identity pools, see the following topics.

Topics

- [Getting started with Amazon Cognito identity pools \(federated identities\) \(p. 242\)](#)
- [Using identity pools \(federated identities\) \(p. 244\)](#)
- [Identity pools concepts \(federated identities\) \(p. 248\)](#)
- [Using attributes for access control as a form of attribute-based access control \(p. 260\)](#)
- [Role-based access control \(p. 264\)](#)
- [Getting credentials \(p. 268\)](#)
- [Accessing AWS services \(p. 274\)](#)
- [Identity pools \(federated identities\) external identity providers \(p. 276\)](#)
- [Developer authenticated identities \(identity pools\) \(p. 298\)](#)
- [Switching unauthenticated users to authenticated users \(identity pools\) \(p. 309\)](#)

Getting started with Amazon Cognito identity pools (federated identities)

With Amazon Cognito identity pools, you can create unique identities and assign permissions for users. Your identity pool can include:

- Users in an Amazon Cognito user pool
- Users who authenticate with external identity providers such as Facebook, Google, Apple, or an OIDC or SAML identity provider.
- Users authenticated via your own existing authentication process

With an identity pool, you can obtain temporary AWS credentials with permissions you define to directly access other AWS services or to access resources through Amazon API Gateway.

Topics

- [Sign up for an AWS account \(p. 243\)](#)
- [Create an identity pool in Amazon Cognito \(p. 243\)](#)
- [Install the Mobile or JavaScript SDK \(p. 243\)](#)
- [Integrate the identity providers \(p. 244\)](#)
- [Get credentials \(p. 244\)](#)

Sign up for an AWS account

To use Amazon Cognito identity pools, you need an AWS account. If you don't already have one, use the following procedure to sign up:

To sign up for an AWS account

1. Open <https://portal.aws.amazon.com/billing/signup>.
2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

Create an identity pool in Amazon Cognito

You can create an identity pool through the Amazon Cognito console, or you can use the AWS Command Line Interface (CLI) or the Amazon Cognito APIs.

To create a new identity pool in the console

1. Sign in to the [Amazon Cognito console](#), choose **Manage identity pools**, and then choose **Create new identity pool**.
2. Type a name for your identity pool.
3. To enable unauthenticated identities, select **Enable access to unauthenticated identities** from the **Unauthenticated identities** collapsible section.
4. If desired, configure an authentication provider in the **Authentication providers** section.
5. Choose **Create Pool**.

Note

At least one identity is required for a valid identity pool.

6. You will be prompted for access to your AWS resources.

Choose **Allow** to create the two default roles associated with your identity pool—one for unauthenticated users and one for authenticated users. These default roles provide your identity pool access to Amazon Cognito Sync. You can modify the roles associated with your identity pool in the AWS Identity and Access Management (IAM) console.

Install the Mobile or JavaScript SDK

To use Amazon Cognito identity pools, you must install and configure the AWS Mobile or JavaScript SDK. For more information, see the following topics:

- [Set Up the AWS Mobile SDK for Android](#)
- [Set Up the AWS Mobile SDK for iOS](#)
- [Set Up the AWS SDK for JavaScript](#)
- [Set Up the AWS Mobile SDK for Unity](#)
- [Set Up the AWS Mobile SDK for .NET and Xamarin](#)

Integrate the identity providers

Amazon Cognito identity pools (federated identities) support user authentication through Amazon Cognito user pools, federated identity providers—including Amazon, Facebook, Google, Apple, and SAML identity providers—as well as unauthenticated identities. This feature also supports [Developer authenticated identities \(identity pools\) \(p. 298\)](#), which lets you register and authenticate users via your own backend authentication process.

To learn more about using an Amazon Cognito user pool to create your own user directory, see [Amazon Cognito user pools \(p. 22\)](#) and [Accessing AWS services using an identity pool after sign-in \(p. 203\)](#).

To learn more about using external identity providers, see [Identity pools \(federated identities\) external identity providers \(p. 276\)](#).

To learn more about integrating your own backend authentication process, see [Developer authenticated identities \(identity pools\) \(p. 298\)](#).

Get credentials

Amazon Cognito identity pools provide temporary AWS credentials for users who are guests (unauthenticated) and for users who have authenticated and received a token. With those AWS credentials, your app can securely access a back end in AWS or outside AWS through Amazon API Gateway. See [Getting credentials \(p. 268\)](#).

Using identity pools (federated identities)

Amazon Cognito identity pools provide temporary AWS credentials for users who are guests (unauthenticated) and for users who have been authenticated and received a token. An identity pool is a store of user identity data specific to your account.

To create a new identity pool in the console

1. Sign in to the [Amazon Cognito console](#), choose **Manage identity pools**, and then choose **Create new identity pool**.
2. Type a name for your identity pool.
3. To enable unauthenticated identities, select **Enable access to unauthenticated identities** from the **Unauthenticated identities** collapsible section.
4. If desired, configure an authentication provider in the **Authentication providers** section.
5. Choose **Create Pool**.

Note

At least one identity is required for a valid identity pool.

6. You will be prompted for access to your AWS resources.

Choose **Allow** to create the two default roles associated with your identity pool—one for unauthenticated users, and one for authenticated users. These default roles provide your identity

pool access to Amazon Cognito Sync. You can modify the roles associated with your identity pool in the IAM console. For additional instructions on working with the Amazon Cognito console, see [Using the Amazon Cognito console \(p. 3\)](#).

User IAM roles

An IAM role defines the permissions for your users to access AWS resources, like [Amazon Cognito Sync \(p. 312\)](#). Users of your application will assume the roles you create. You can specify different roles for authenticated and unauthenticated users. To learn more about IAM roles, see [IAM roles \(p. 254\)](#).

Authenticated and unauthenticated identities

Amazon Cognito identity pools support both authenticated and unauthenticated identities. Authenticated identities belong to users who are authenticated by any supported identity provider. Unauthenticated identities typically belong to guest users.

- To configure authenticated identities with a public login provider, see [Identity pools \(federated identities\) external identity providers \(p. 276\)](#).
- To configure your own backend authentication process, see [Developer authenticated identities \(identity pools\) \(p. 298\)](#).

Enable or disable unauthenticated identities

Amazon Cognito identity pools can support unauthenticated identities by providing a unique identifier and AWS credentials for users who do not authenticate with an identity provider. If your application allows users who do not log in, you can enable access for unauthenticated identities. To learn more, see [Getting started with Amazon Cognito identity pools \(federated identities\) \(p. 242\)](#).

Choose **Manage identity pools** from the [Amazon Cognito console](#):

1. Select the name of the identity pool for which you want to enable or disable unauthenticated identities. The **Dashboard page** for your identity pool appears.
2. In the top-right corner of the Dashboard page, select **Edit identity pool**. The **Edit identity pool** page appears.
3. Scroll down and select **Unauthenticated identities** to expand it.
4. Select the check box to enable or disable access to unauthenticated identities.
5. Select **Save Changes**.

Change the role associated with an identity type

Every identity in your identity pool is either authenticated or unauthenticated. Authenticated identities belong to users who are authenticated by a public login provider (Amazon Cognito user pools, Login with Amazon, Sign in with Apple, Facebook, Google, SAML, or any OpenID Connect Providers) or a developer provider (your own backend authentication process). Unauthenticated identities typically belong to guest users.

For each identity type, there is an assigned role. This role has a policy attached to it which dictates which AWS services that role can access. When Amazon Cognito receives a request, the service determines the identity type, determines the role assigned to that identity type, and uses the policy attached to that role to respond. By modifying a policy or assigning a different role to an identity type, you can control which AWS services an identity type can access. To view or modify the policies associated with the roles in your identity pool, see the [AWS IAM Console](#).

You can change which role is associated with an identity type using the Amazon Cognito identity pool (federated identities) console. Choose **Manage identity pools** from the [Amazon Cognito console](#):

1. Select the name of the identity pool for which you want to modify roles. The **Dashboard page** for your identity pool appears.
2. In the top-right corner of the Dashboard page, select **Edit identity pool**. The **Edit identity pool** page appears.
3. Use the dropdown lists next to **Unauthenticated role** and **Authenticated role** to change roles. Select **Create new role** to create or modify the roles associated with each identity type in the [AWS IAM console](#). For more information, see [IAM Roles](#).

Enable or edit authentication providers

If you allow your users to authenticate using public identity providers (for example, Amazon Cognito user pools, Login with Amazon, Sign in with Apple, Facebook, or Google), you can specify your application identifiers in the Amazon Cognito identity pools (federated identities) console. This associates the application ID (provided by the public login provider) with your identity pool.

You can also configure authentication rules for each provider from this page. Each provider allows up to 25 rules. The rules are applied in the order you save for each provider. For more information, see [Role-based access control \(p. 264\)](#).

Warning

Changing the application ID to which an identity pool is linked disables existing users from authenticating with that identity pool. Learn more about [Identity pools \(federated identities\)](#) [external identity providers \(p. 276\)](#).

Choose **Manage identity pools** from the [Amazon Cognito console](#):

1. Select the name of the identity pool for which you want to enable the external provider. The **Dashboard page** for your identity pool appears.
2. In the top-right corner of the Dashboard page, select **Edit identity pool**. The **Edit identity pool** page appears.
3. Scroll down and select **Authentication providers** to expand it.
4. Select the tab for the appropriate provider and enter the required information associated with that authentication provider.

Delete an identity pool

Choose **Manage identity pools** from the [Amazon Cognito console](#):

1. Select the name of the identity pool that you want to delete. The **Dashboard page** for your identity pool appears.
2. In the top-right corner of the Dashboard page, select **Edit identity pool**. The **Edit identity pool** page appears.
3. Scroll down and select **Delete identity pool** to expand it.
4. Select **Delete identity pool**.
5. Select **Delete pool**.

Warning

When you select the Delete button, you will permanently delete your identity pool and all the user data it contains. Deleting an identity pool will cause applications and other services utilizing the identity pool to stop working.

Delete an identity from an identity pool

Choose **Manage identity pools** from the [Amazon Cognito console](#):

1. Select the name of the identity pool that contains the identity that you want to delete. The **Dashboard page** for your identity pool appears.
2. In the left-hand navigation on the Dashboard page, select **Identity browser**. The **Identities** page appears.
3. On the **Identities** page, enter the identity ID that you want to delete, and then select **Search**.
4. On the **Identity details** page, select the **Delete identity** button, and then select **Delete**.

Managing datasets

If you have implemented Amazon Cognito Sync functionality in your application, the Amazon Cognito identity pools console enables you to manually create and delete datasets and records for individual identities. Any change you make to an identity's dataset or records in the Amazon Cognito identity pools console isn't saved until you select **Synchronize** in the console. The change isn't visible to the end user until the identity calls **Synchronize**. The data being synchronized from other devices for individual identities is visible once you refresh the list datasets page for a particular identity.

Create a dataset for an identity

Choose **Manage identity pools** from the [Amazon Cognito console](#):

1. Select the name of the identity pool that contains the identity for which you want to create a dataset. The **Dashboard page** for your identity pool appears.
2. In the left-hand navigation on the Dashboard page, select **Identity browser**. The **Identities** page appears.
3. On the **Identities** page, enter the identity ID for which you want to create a dataset, and then select **Search**.
4. On the **Identity details** page for that identity, select the **Create dataset** button, enter a dataset name, and then select **Create and edit dataset**.
5. On the **Current dataset** page, select **Create record** to create a record to store in that dataset.
6. Enter a key for that dataset, the valid JSON value or values to store, and then select **Format as JSON** to format the value you entered for readability, and to confirm that it is well-formed JSON. When finished, select **Save Changes**.
7. Select **Synchronize** to synchronize the dataset. Your changes will not be saved until you select **Synchronize** and will not be visible to the user until the identity calls **Synchronize**. To discard unsynchronized changes, select the change you wish to discard, and then select **Discard changes**.

Delete a dataset associated with an identity

Choose **Manage identity pools** from the [Amazon Cognito console](#):

1. Select the name of the identity pool that contains the identity for which you want to delete a dataset. The **Dashboard page** for your identity pool appears.
2. In the left-hand navigation on the Dashboard page, select **Identity browser**. The **Identities** page appears.
3. On the **Identities** page, enter the identity ID containing the dataset which you want to delete, and then select **Search**.
4. On the **Identity details** page, select the check box next to the dataset or datasets that you want to delete, select **Delete selected**, and then select **Delete**.

Bulk publish data

Bulk publish can be used to export data already stored in your Amazon Cognito Sync store to a Amazon Kinesis stream. For instructions on how to bulk publish all of your streams, see [Amazon Cognito Streams \(p. 338\)](#).

Enable push synchronization

Amazon Cognito automatically tracks the association between identity and devices. Using the Push Sync feature, you can ensure that every instance of a given identity is notified when identity data changes. Push Sync ensures that, whenever the sync store data changes for a particular identity, all devices associated with that identity receive a silent push notification informing them of the change.

You can enable Push Sync via the Amazon Cognito console. Choose **Manage identity pools** from the [Amazon Cognito console](#):

1. Select the name of the identity pool for which you want to enable Push Sync. The **Dashboard** page for your identity pool appears.
2. In the top-right corner of the Dashboard page, select **Edit identity pool**. The **Edit identity pool** page appears.
3. Scroll down and select **Push synchronization** to expand it.
4. In the **Service role** dropdown list, select the IAM role that grants Amazon Cognito permission to send an Amazon Simple Notification Service (Amazon SNS) notification. Select **Create role** to create or modify the roles associated with your identity pool in the [AWS IAM console](#).
5. Select a platform application, and then select **Save Changes**.

Set up Amazon Cognito Streams

Amazon Cognito Streams gives developers control and insight into their data stored in Amazon Cognito Sync. Developers can now configure a Kinesis stream to receive events as data. Amazon Cognito can push each dataset change to a Kinesis stream you own in real time. For instructions on how to set up Amazon Cognito Streams in the Amazon Cognito console, see [Amazon Cognito Streams \(p. 338\)](#).

Set up Amazon Cognito Events

Amazon Cognito Events allows you to execute an AWS Lambda function in response to important events in Amazon Cognito Sync. Amazon Cognito Sync raises the Sync Trigger event when a dataset is synchronized. You can use the Sync Trigger event to take an action when a user updates data. For instructions on setting up Amazon Cognito Events from the console, see [Amazon Cognito Events \(p. 340\)](#).

To learn more about AWS Lambda, see [AWS Lambda](#).

Identity pools concepts (federated identities)

You can use Amazon Cognito identity pools to create unique identities for your users and authenticate them with identity providers. With an identity, you can obtain temporary, limited-privilege AWS credentials to access other AWS services. Amazon Cognito identity pools support public identity providers—Amazon, Apple, Facebook, and Google—and unauthenticated identities. It also supports developer authenticated identities, which let you register and authenticate users via your own backend authentication process.

For information about Amazon Cognito identity pools Region availability, see [AWS Service Region Availability](#). For more information about Amazon Cognito identity pools concepts, see the following topics.

Topics

- [Identity pools \(federated identities\) authentication flow \(p. 249\)](#)
- [IAM roles \(p. 254\)](#)
- [Role trust and permissions \(p. 259\)](#)

Identity pools (federated identities) authentication flow

Amazon Cognito helps you create unique identifiers for your end users that are kept consistent across devices and platforms. Amazon Cognito also delivers temporary, limited-privilege credentials to your application to access AWS resources. This page covers the basics of how authentication in Amazon Cognito works and explains the lifecycle of an identity inside your identity pool.

External provider authflow

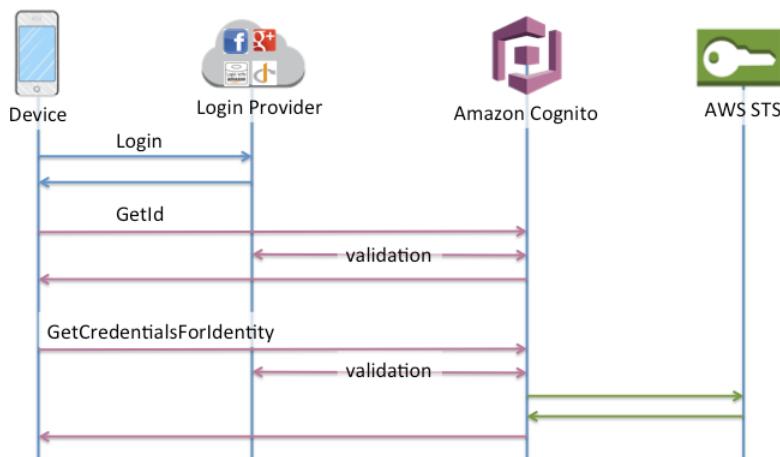
A user authenticating with Amazon Cognito goes through a multi-step process to bootstrap their credentials. Amazon Cognito has two different flows for authentication with public providers: enhanced and basic.

Once you complete one of these flows, you can access other AWS services as defined by your role's access policies. By default, the [Amazon Cognito console](#) creates roles with access to the Amazon Cognito Sync store and to Amazon Mobile Analytics. For more information on how to grant additional access, see [IAM roles \(p. 254\)](#).

Enhanced (simplified) authflow

When you use the enhanced authflow, your app first presents an ID token from an authorized Amazon Cognito user pool or third-party identity provider in a [GetID](#) request. The app exchanges the token for an identity ID in your identity pool. The identity ID is then used with the same identity provider token in a [GetCredentialsForIdentity](#) request. The enhanced workflow simplifies credential retrieval by performing [GetOpenIdToken](#) and [AssumeRoleWithWebIdentity](#) in the background for you. [GetCredentialsForIdentity](#) returns AWS API credentials.

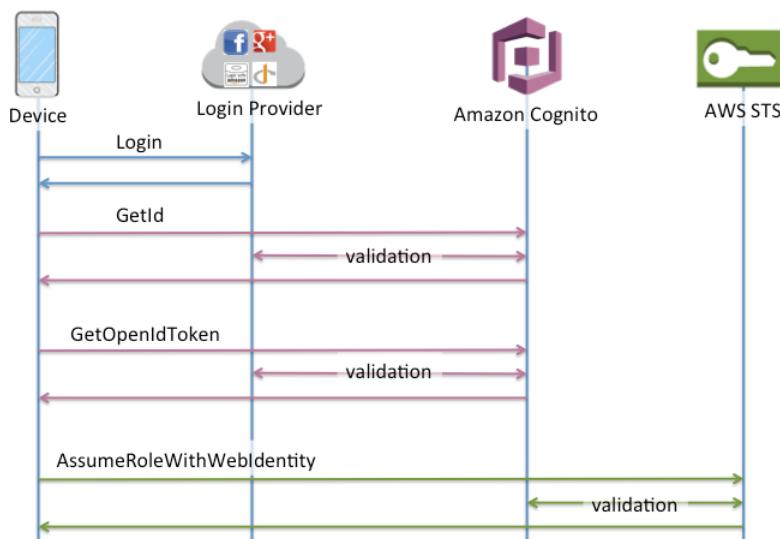
1. [GetId](#)
2. [GetCredentialsForIdentity](#)



Basic (classic) authflow

When you use the basic authflow, your app first presents an ID token from an authorized Amazon Cognito user pool or third-party identity provider in a [GetID](#) request. The app exchanges the token for an identity ID in your identity pool. The identity ID is then used with the same identity provider token in a [GetOpenIdToken](#) request. [GetOpenIdToken](#) returns a new OAuth 2.0 token that is issued by your identity pool. You can then use the new token in an [AssumeRoleWithWebIdentity](#) request to retrieve AWS API credentials. The basic workflow gives you more granular control over the credentials that you distribute to your users. The [GetCredentialsForIdentity](#) request of the enhanced authflow requests a role based on the contents of an access token. The [AssumeRoleWithWebIdentity](#) request in the classic workflow grants your app a greater ability to request credentials for any AWS Identity and Access Management role that you have configured with a sufficient trust policy.

1. [GetId](#)
2. [GetOpenIdToken](#)
3. [AssumeRoleWithWebIdentity](#)

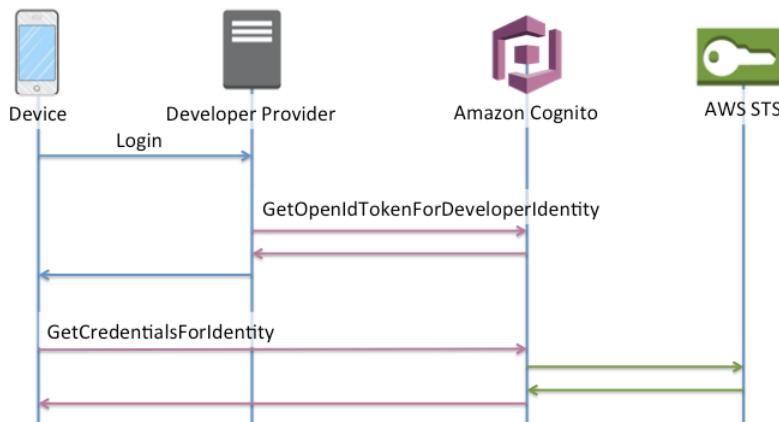


Developer authenticated identities authflow

When using [Developer authenticated identities \(identity pools\) \(p. 298\)](#), the client uses a different authflow that includes code outside of Amazon Cognito to validate the user in your own authentication system. Code outside of Amazon Cognito is indicated as such.

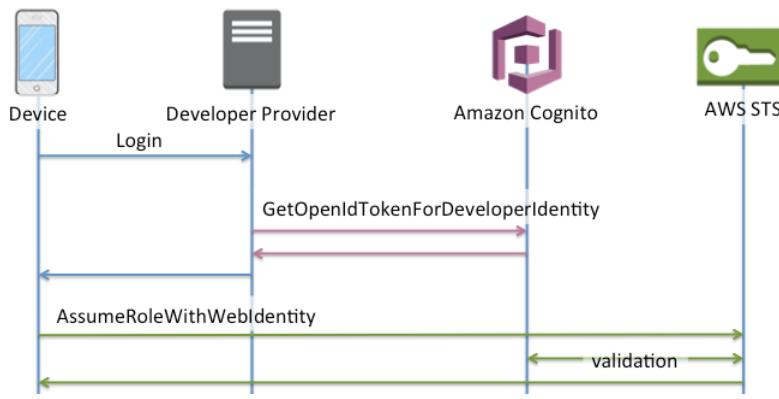
Enhanced authflow

1. Login via Developer Provider (code outside of Amazon Cognito)
2. Validate the user login (code outside of Amazon Cognito)
3. [GetOpenIdTokenForDeveloperIdentity](#)
4. [GetCredentialsForIdentity](#)



Basic authflow

1. Login via Developer Provider (code outside of Amazon Cognito)
2. Validate the user login (code outside of Amazon Cognito)
3. [GetOpenIdTokenForDeveloperIdentity](#)
4. [AssumeRoleWithWebIdentity](#)



Which authflow should I use?

For most customers, the Enhanced Flow is the correct choice, as it offers many benefits over the Basic Flow:

- One fewer network call to get credentials on the device.

- All calls are made to Amazon Cognito, meaning it is also one less network connection.
- Roles no longer need to be embedded in your application, only an identity pool id and Region are necessary to start bootstrapping credentials.

Since February 2015, the [Amazon Cognito console](#) has displayed example code that used the Enhanced Flow. Additionally, if your identity pool does not have the role association necessary to use the Enhanced Flow, the console displays a notification.

The following are the minimum SDK versions where the Enhanced Flow is supported:

SDK (minimum version)

- AWS SDK for iOS (2.0.14)
- AWS SDK for Android (2.1.8)
- AWS SDK for JavaScript (2.1.7)
- AWS SDK for Unity (1.0.3)
- AWS SDK for Xamarin (3.0.0.5)

If you want to use more than the two default roles configured when you create a new identity pool in the console, use the Basic Flow.

API summary

GetId

The `GetId` API call is the first call necessary to establish a new identity in Amazon Cognito.

Unauthenticated access

Amazon Cognito can grant unauthenticated guest access in your applications. If this feature is enabled in your identity pool, users can request a new identity ID at any time via the `GetId` API. The application is expected to cache this identity ID to make subsequent calls to Amazon Cognito. The AWS Mobile SDKs and the AWS SDK for JavaScript in the Browser have credentials providers that handle this caching for you.

Authenticated access

When you've configured your application with support for a public login provider (Facebook, Google+, Login with Amazon, or Sign in with Apple), users can also supply tokens (OAuth or OpenID Connect) that identify them in those providers. When used in a call to `GetId`, Amazon Cognito creates a new authenticated identity or returns the identity already associated with that particular login. Amazon Cognito does this by validating the token with the provider and making sure of the following:

- The token is valid and from the configured provider.
- The token is not expired.
- The token matches the application identifier created with that provider (for example, Facebook app ID).
- The token matches the user identifier.

GetCredentialsForIdentity

The `GetCredentialsForIdentity` API can be called after you establish an identity ID. This API is functionally equivalent to calling `GetOpenIdToken` followed by `AssumeRoleWithWebIdentity`.

For Amazon Cognito to call `AssumeRoleWithWebIdentity` on your behalf, your identity pool must have IAM roles associated with it. You can do this via the Amazon Cognito console or manually via the [SetIdentityPoolRoles](#) operation.

GetOpenIdToken

The `GetOpenIdToken` API call is called after you establish an identity ID. If you have a cached identity ID, this can be the first call you make during an app session.

Unauthenticated access

To obtain a token for an unauthenticated identity, you only need the identity ID itself. It is not possible to get an unauthenticated token for authenticated or disabled identities.

Authenticated access

If you have an authenticated identity, you must pass at least one valid token for a login already associated with that identity. All tokens passed in during the `GetOpenIdToken` call must pass the same validation mentioned earlier; if any of the tokens fail, the whole call fails. The response from the `GetOpenIdToken` call also includes the identity ID. This is because the identity ID that you pass in may not be the one that is returned.

Linking logins

If you pass in a token for a login that is not already associated with any identity, the login is considered to be "linked" to the associated identity. You may only link one login per public provider. Attempts to link more than one login with a public provider results in a `ResourceConflictException` error response. If a login is merely linked to an existing identity, the identity ID returned from `GetOpenIdToken` is the same as the one that you passed in.

Merging identities

If you pass in a token for a login that is not currently linked to the given identity, but is linked to another identity, the two identities are merged. Once merged, one identity becomes the parent/owner of all associated logins and the other is disabled. In this case, the identity ID of the parent/owner is returned. You must update your local cache if this value differs. The providers in the AWS Mobile SDKs or AWS SDK for JavaScript in the Browser perform this operation for you.

GetOpenIdTokenForDeveloperIdentity

The `GetOpenIdTokenForDeveloperIdentity` API replaces the use of `GetId` and `GetOpenIdToken` from the device when using developer authenticated identities. Because this API call is signed by your AWS credentials, Amazon Cognito can trust that the user identifier supplied in the API call is valid. This replaces the token validation Amazon Cognito performs with external providers.

The payload for this API includes a `logins` map that must contain the key of your developer provider and a value as an identifier for the user in your system. If the user identifier isn't already linked to an existing identity, Amazon Cognito creates a new identity and returns the new identity id and an OpenId Connect token for that identity. If the user identifier is already linked, Amazon Cognito returns the pre-existing identity id and an OpenId Connect token.

Linking logins

As with external providers, supplying additional logins that are not already associated with an identity implicitly links those logins to that identity. If you link an external provider login to an identity, the user can use the external provider authflow with that provider, but they cannot use your developer provider name in the `logins` map when calling `GetId` or `GetOpenIdToken`.

Merging identities

With developer authenticated identities, Amazon Cognito supports both implicit merging and explicit merging through the `MergeDeveloperIdentities` API. With explicit merging, you can mark two identities with user identifiers in your system as a single identity. If you supply the source and destination user identifiers, Amazon Cognito merges them. The next time you request an OpenId Connect token for either user identifier, the same identity id is returned.

AssumeRoleWithWebIdentity

After you have an OpenID Connect token, you can then trade this for temporary AWS credentials through the `AssumeRoleWithWebIdentity` API call in AWS Security Token Service (STS). This call is no different than if you were using Facebook, Google+, Login with Amazon, or Sign in with Apple directly. The only exception is that you are passing an Amazon Cognito token instead of a token from one of the other public providers.

Because there is no restriction on the number of identities that you can create, it is important to understand the permissions that you're granting to your users. Set up different roles for your application: one for unauthenticated users, and one for authenticated users. The Amazon Cognito console creates these for you by default when you first set up your identity pool. The access policy for these two roles are exactly the same: it grants users access to Amazon Cognito Sync, and they can submit events to Amazon Mobile Analytics. You can modify these roles to meet your needs.

Learn more about [Role trust and permissions \(p. 259\)](#).

IAM roles

While creating an identity pool, you're prompted to update the IAM roles that your users assume. IAM roles work like this: When a user logs in to your app, Amazon Cognito generates temporary AWS credentials for the user. These temporary credentials are associated with a specific IAM role. With the IAM role, you can define a set of permissions to access your AWS resources.

You can specify default IAM roles for authenticated and unauthenticated users. In addition, you can define rules to choose the role for each user based on claims in the user's ID token. For more information, see [Role-based access control \(p. 264\)](#).

By default, the Amazon Cognito console creates IAM roles that provide access to Amazon Mobile Analytics and to Amazon Cognito Sync. Alternatively, you can choose to use existing IAM roles.

Modify IAM roles to allow or restrict access to other services. To do so, [log in to the IAM Console](#). Then select **Roles**, and select a role. The policies attached to the selected role are listed in the **Permissions** tab. You can customize an access policy by selecting the corresponding **Manage Policy** link. To learn more about using and defining policies, see [Overview of IAM Policies](#).

Note

As a best practice, define policies that follow the principle of granting *least privilege*. In other words, the policies include only the permissions that users require to perform their tasks. For more information, see [Grant Least Privilege](#) in the *IAM User Guide*.

Remember that unauthenticated identities are assumed by users who do not log in to your app. Typically, the permissions that you assign for unauthenticated identities should be more restrictive than those for authenticated identities.

Topics

- [Set up a trust policy \(p. 254\)](#)
- [Access policies \(p. 255\)](#)

Set up a trust policy

Amazon Cognito uses IAM roles to generate temporary credentials for your application's users. Access to permissions is controlled by a role's trust relationships. Learn more about [Role trust and permissions \(p. 259\)](#).

Reuse roles across identity pools

To reuse a role across multiple identity pools, because they share a common permission set, you can include multiple identity pools, like this:

```
"StringEquals": {  
    "cognito-identity.amazonaws.com:aud": [  
        "us-east-1:12345678-abcd-abcd-abcd-123456790ab",  
        "us-east-1:98765432-dcba-dcba-dcba-123456790ab"  
    ]  
}
```

Limit access to specific identities

To create a policy limited to a specific set of app users, check the value of `cognito-identity.amazonaws.com:sub`:

```
"StringEquals": {  
    "cognito-identity.amazonaws.com:aud": "us-east-1:12345678-abcd-abcd-abcd-123456790ab",  
    "cognito-identity.amazonaws.com:sub": [  
        "us-east-1:12345678-1234-1234-1234-123456790ab",  
        "us-east-1:98765432-1234-1234-1243-123456790ab"  
    ]  
}
```

Limit access to specific providers

To create a policy limited to users who have logged in with a specific provider (perhaps your own login provider), check the value of `cognito-identity.amazonaws.com:amr`:

```
"ForAnyValue:StringLike": {  
    "cognito-identity.amazonaws.com:amr": "login.myprovider.myapp"  
}
```

For example, an app that trusts only Facebook would have the following amr clause:

```
"ForAnyValue:StringLike": {  
    "cognito-identity.amazonaws.com:amr": "graph.facebook.com"  
}
```

Access policies

The permissions attached to a role are effective across all users that assume that role. If you want to partition your users' access, you can do so through policy variables. Be careful when including your users' identity IDs in your access policies, particularly for unauthenticated identities, as these may change if the user logs in.

For additional security protection, Amazon Cognito applies a scope-down policy to credentials vended by `GetCredentialForIdentity` to prevent access to services other than the following ones for your unauthenticated users. You can create an identity using these credentials with access to only the following services:

- Amazon API Gateway
- AWS AppSync
- Amazon CloudWatch
- Amazon CloudWatch Logs
- Amazon Cognito Identity
- Amazon Cognito Sync

- Amazon Cognito user pools
- Amazon Comprehend
- Amazon DynamoDB
- Amazon GameLift
- AWS IoT
- AWS Key Management Service (AWS KMS)
- Amazon Kinesis Data Firehose
- Amazon Kinesis Data Streams
- AWS Lambda
- Amazon Lex
- Amazon Location Service
- Amazon Machine Learning
- Amazon Mobile Analytics
- Amazon Personalize
- Amazon Pinpoint
- Amazon Polly
- Amazon Rekognition
- Amazon SageMaker
- Amazon SimpleDB
- Amazon Simple Email Service (Amazon SES)
- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Queue Service (Amazon SQS)
- Amazon Simple Storage Service (Amazon S3)
- Amazon Sumerian
- Amazon Textract
- Amazon Transcribe
- Amazon Translate

In addition, there are services that grant access to unauthenticated users, but don't permit all the service's actions for these users. The following table shows the services with restricted actions.

Service	Permission granted for unauthenticated users
AWS Key Management Service	Encrypt, Decrypt, ReEncrypt , GenerateDataKey
Amazon SageMaker	InvokeEndpoint
Amazon Textract	DetectDocumentText, AnalyzeDocument
Amazon Sumerian	View*

If you need access to something other than these services for your unauthenticated users, you must use the basic authentication flow. If you are getting `NotAuthorizedException` and you have enabled access to the service in your unauthenticated role policy, this is likely the reason.

Access policy examples

In this section, you can find example Amazon Cognito access policies that grant only the permissions your identities must complete a specific operation. You can further limit the permissions for

a given identity ID by using policy variables where possible. For example, using \${cognito-identity.amazonaws.com:sub}. For more information, see [Understanding Amazon Cognito Authentication Part 3: Roles and Policies on the AWS Mobile Blog](#).

Note

As a security best practice, policies should include only the permissions that users require to perform their tasks. This means that you should try to always scope access to an individual identity for objects whenever possible.

Example 1: Grant an identity read access to a single object in Amazon S3

The following access policy grants read permissions to an identity to retrieve a single object from a given S3 bucket.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket/assets/my_picture.jpg"]
    }
  ]
}
```

Example 2: Grant an identity both read and write access to identity specific paths in Amazon S3

The following access policy grants read and write permissions to access a specific prefix "folder" in an S3 bucket by mapping the prefix to the \${cognito-identity.amazonaws.com:sub} variable.

With this policy, an identity such as us-east-1:12345678-1234-1234-1234-123456790ab inserted via \${cognito-identity.amazonaws.com:sub} can get, put, and list objects into arn:aws:s3:::mybucket/us-east-1:12345678-1234-1234-1234-123456790ab. However, the identity would not be granted access to other objects in arn:aws:s3:::mybucket.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3>ListBucket"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket"],
      "Condition": {"StringLike": {"s3:prefix": ["${cognito-identity.amazonaws.com:sub}/*"]}}
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket/${cognito-identity.amazonaws.com:sub}/*"]
    }
  ]
}
```

Example 3: Assign identities fine-grained access to Amazon DynamoDB

The following access policy provides fine-grained access control to DynamoDB resources using Amazon Cognito environment variables. These variables grant access to items in DynamoDB by identity ID. For

more information, see [Using IAM Policy Conditions for Fine-Grained Access Control in the Amazon DynamoDB Developer Guide](#).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:BatchGetItem",
                "dynamodb:Query",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem",
                "dynamodb:BatchWriteItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-west-2:123456789012:table/MyTable"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "dynamodb:LeadingKeys": ["${cognito-identity.amazonaws.com:sub}"]
                }
            }
        }
    ]
}
```

Example 4: Grant an identity permission to invoke a Lambda function

The following access policy grants an identity permissions to invoke a Lambda function.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": [
                "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
            ]
        }
    ]
}
```

Example 5: Grant an identity permission to publish records to Kinesis Data Streams

The following access policy allows an identity to use the PutRecord operation with any of the Kinesis Data Streams. It can be applied to users that need to add data records to all streams in an account. For more information, see [Controlling Access to Amazon Kinesis Data Streams Resources Using IAM](#) in the [Amazon Kinesis Data Streams Developer Guide](#).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "kinesis:PutRecord",
            "Resource": [
                "arn:aws:kinesis:us-east-1:111122223333:stream/stream1"
            ]
        }
    ]
}
```

```
        ]
    }
```

Example 6: Grant an identity access to their data in the Amazon Cognito Sync store

The following access policy grants an identity permissions to access only their own data in the Amazon Cognito Sync store.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "cognito-sync:*",
      "Resource": ["arn:aws:cognito-sync:us-east-1:123456789012:identitypool/${cognito-identity.amazonaws.com:aud}/identity/${cognito-identity.amazonaws.com:sub}/*"]
    }
  ]
}
```

Role trust and permissions

The way these roles differ is in their trust relationships. The following is an example trust policy for an unauthenticated role:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.amazonaws.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com:aud": "us-east-1:12345678-corner-cafe-123456790ab"
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com:amr": "unauthenticated"
        }
      }
    }
  ]
}
```

This policy grants federated users from `cognito-identity.amazonaws.com` (the issuer of the OpenID Connect token) permission to assume this role. Additionally, the policy restricts the `aud` of the token, in this case the identity pool ID, to match the identity pool. Finally, the policy specifies that one of the array members of the multi-value `amr` claim of the token issued by the Amazon Cognito `GetOpenIdToken` API operation has the value `unauthenticated`.

When Amazon Cognito creates a token, it sets the `amr` of the token as either `unauthenticated` or `authenticated`. If `amr` is `authenticated`, the token includes any providers used during authentication. This means that you can create a role that trusts only users that logged in via Facebook by changing the `amr` condition as shown:

```
"ForAnyValue:StringLike": {
```

```
    "cognito-identity.amazonaws.com:amr": "graph.facebook.com"
}
```

Be careful when changing your trust relationships on your roles, or when trying to use roles across identity pools. If you don't configure your role correctly to trust your identity pool, an exception from STS results, like the following:

```
AccessDenied -- Not authorized to perform sts:AssumeRoleWithWebIdentity
```

If you see this message, check that your identity pool and authentication type have an appropriate role.

Using attributes for access control as a form of attribute-based access control

You can use IAM policies to control access to AWS resources through Amazon Cognito identity pools based on user attributes. These attributes can be drawn from social and corporate identity providers. You can map attributes within providers' access and ID tokens or SAML assertions to tags that can be referenced in the IAM permissions policies.

You can choose default mappings or create your own custom mappings in Amazon Cognito identity pools. The default mappings allow you to write IAM policies based on a fixed set of user attributes. Custom mappings allow you to select a custom set of user attributes that are referenced in the IAM permissions policies. The **Attribute names** in the Amazon Cognito console are mapped to **Tag key for principal**, which are the tags that are referenced in the IAM permissions policy.

For example, let's say that you own a media streaming service with a free and paid membership. You store the media files in Amazon S3 and tag them with free or premium tags. You can use attributes for access control to allow access to free and paid content based on user membership level, which is part of user's profile. You can map the membership attribute to a tag key for principal to be passed on to the IAM permissions policy. This way you can create a single permissions policy and conditionally allow access to premium content based on the value of membership level and tag on the content files.

Topics

- [Using attributes for access control with Amazon Cognito identity pools \(p. 261\)](#)
- [Using attributes for access control policy example \(p. 261\)](#)
- [Disable attributes for access control \(console\) \(p. 263\)](#)
- [Default provider mappings \(p. 263\)](#)

Using attributes to control access has several benefits:

- Permissions management is easier when you use attributes for access control. You can create a basic permissions policy that uses user attributes instead of creating multiple policies for different job functions.
- You don't need to update your policies whenever you add or remove resources or users for your application. The permissions policy will only grant the access to users with the matching user attributes. For example, you might need to control the access to certain S3 buckets based on the job title of users. In that case, you can create a permissions policy to allow access to these files only for users within the defined job title. For more information, see [IAM Tutorial: Use SAML session tags for ABAC](#).
- Attributes can be passed as principal tags to a policy that allows or denies permissions based on the values of those attributes.

Using attributes for access control with Amazon Cognito identity pools

Before you can use attributes for access control, ensure that you meet the following prerequisites:

- [An AWS account](#)
- [User pool](#)
- [Identity pool](#)
- [The mobile or JavaScript SDK](#)
- [Integrated identity providers](#)
- [Credentials](#)

To use attributes for access control you have to configure **Tag Key for Principal** and **Attribute name**. In **Tag Key for Principal** the value is used to match the **PrincipalTag** condition in the permissions policies. The value in **Attribute name** is the name of the attribute whose value will be evaluated in the policy.

To use attributes for access control with identity pools

1. Open the Amazon Cognito console.
2. Choose **Manage Identity Pools**.
3. On the dashboard, choose the name of the identity pool that you want to use attributes for access control on.
4. Choose **Edit identity pool**.
5. Expand the **Authentication providers** section.
6. In the **Authentication providers** section, choose the provider tab you want to use.
7. In **Attributes for access control**, choose either **Default attribute mappings** or **Custom attribute mappings**. Default mappings are different for each provider. For more information, see [Default provider mappings \(p. 263\)](#) for attributes for access control.
8. If you choose **Custom attribute mappings**, complete the following steps.
 1. In **Tag Key for Principal**, enter your custom text. There is a maximum length of 128 characters.
 2. In **Attribute name**, enter the attribute names from your providers tokens or SAML assertions. You can get your attribute names for your IdPs from your providers developer guide. Attribute names have a maximum of 256 characters. In addition, the aggregated character limit for all attributes is 460 bytes.
 3. (Optional) **Add another provider**. You can add multiple providers for Amazon Cognito user pools, OIDC, and SAML providers in the console. For example, you can add two Amazon Cognito user pools as two separate identity providers. Amazon Cognito treats each tab as different IdPs. You can configure attributes for access control separately for each IdP.
 4. To finish, use the IAM console to create a permissions policy that includes the default mappings or the custom text mappings that you provided in **Tag Key for Principal**. For a tutorial on creating a permissions policy in IAM, see [IAM tutorial: Define permissions to access AWS resources based on tags](#) in the *IAM User Guide*.

Using attributes for access control policy example

Consider a scenario where an employee from the legal department of a company needs to list all files in buckets that belong to their department and are classified with their security level. Assume the token that this employee gets from the identity provider contains the following claims.

Claims

```
{
  .
  .
  "sub" : "57e7b692-4f66-480d-98b8-45a6729b4c88",
  "department" : "legal",
  "clearance" : "confidential",
  .
  .
}
```

These attributes can be mapped to tags and referenced in IAM permissions policies as principal tags. You can now manage access by changing the user profile on the identity provider's end. Alternatively, you can change attributes on the resource side by using names or tags without changing the policy itself.

The following permissions policy does two things:

- Allows list access to all s3 buckets that end with a prefix that matches the user's department name.
- Allows read access on files in these buckets as long as the clearance tag on the file matches user's clearance attribute.

Permissions policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3>List*",
      "Resource": "arn:aws:s3:::*$aws:PrincipalTag/department"
    },
    {
      "Effect": "Allow",
      "Action": "s3:GetObject*",
      "Resource": "arn:aws:s3:::*$aws:PrincipalTag/department}/*",
      "Condition": {
        "StringEquals": {
          "s3:ExistingObjectTag/clearance": "${aws:PrincipalTag/clearance}"
        }
      }
    }
  ]
}
```

The trust policy determines who can assume this role. The trust relationship policy allows the use of `sts:AssumeRoleWithWebIdentity` and `sts:TagSession` to allow access. It adds conditions to restrict the policy to the identity pool that you created and it ensures that it's for an authenticated role.

Trust policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```

        "Effect": "Allow",
        "Principal": {
            "Federated": "cognito-identity.amazonaws.com"
        },
        "Action": [
            "sts:AssumeRoleWithWebIdentity",
            "sts:TagSession"
        ],
        "Condition": {
            "StringEquals": {
                "cognito-identity.amazonaws.com:aud": "IDENTITY-POOL-ID"
            },
            "ForAnyValue:StringLike": {
                "cognito-identity.amazonaws.com:amr": "authenticated"
            }
        }
    }
}

```

Disable attributes for access control (console)

Follow this procedure to disable attributes for access control.

To disable attributes for access control

1. Open the Amazon Cognito console.
2. Choose **Manage Identity Pools**.
3. On the dashboard, choose the name of the identity pool whose attributes for access control you want to disable.
4. Choose **Edit identity pool** in the upper-right corner of the page.
5. Expand the **Authentication providers** section.
6. Under **Authenticated role selection**, choose **Disable**.
7. To finish, scroll to the bottom of the page and choose **Save Changes**.

Default provider mappings

The following table has the default mapping information for the authentication providers that Amazon Cognito supports.

Provider	Token type	Principal tag values	Example
Amazon Cognito user pool	ID token	aud(client ID) and sub(user ID)	"6jk8ltokc7ac9es6jrtg9q572f", "57e7b692-4f66-480d-98b8-45a6729
Facebook	Access token	aud(app_id), sub(user_id)	"492844718097981", "112177216992379"
Google	ID token	aud(client ID) and sub(user ID)	"620493171733-eebk7c0hcp5lj3e1tlqp1gnntt3k0rnvc.ap "109220063452404746097"
SAML	Assertions	"http://schemas.xmlsoap.org/ws/2005/05/identity"	"auth0 5e28d196f8f55a0eaaa95de3", "user123@gmail.com"

Provider	Token type	Principal tag values	Example
		claims/nameidentifier", "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name"	
Apple	ID token	aud(client ID) and sub (user ID)	"com.amazonaws.ec2-54-80-172-243.001968.a6ca34e9c1e742458a26cf80"
Amazon	Access token	aud (Client ID on Amzn Dev Ac), user_id(user ID)	"amzn1.application-oa2-client.9d70d9382d3446108aaee3dd77" "amzn1.account.AGHNIFJQMFSBG3G6"
Standard OIDC providers	ID and access tokens	aud (as client_id), sub (as user ID)	"620493171733-eebk7c0hcp5lj3e1tlqp1gntt3k0rncv.ap" "109220063452404746097"
Twitter	Access token	aud (app ID; app Secret), sub (user ID)	"DfwifTtKEX1FilBRnOTIROCFK;Xgj5xb8" "1269003884292222976"
DevAuth	Map	Not applicable	"tag1", "tag2"

Note

The default attribute mappings option is automatically populated for the **Tag Key for Principal** and **Attribute** names. You can't change default mappings.

Role-based access control

Amazon Cognito identity pools assign your authenticated users a set of temporary, limited-privilege credentials to access your AWS resources. The permissions for each user are controlled through [IAM roles](#) that you create. You can define rules to choose the role for each user based on claims in the user's ID token. You can define a default role for authenticated users. You can also define a separate IAM role with limited permissions for guest users who are not authenticated.

Creating roles for role mapping

It is important to add the appropriate trust policy for each role so that it can only be assumed by Amazon Cognito for authenticated users in your identity pool. Here is an example of such a trust policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.amazonaws.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com:aud": "us-east-1:12345678-corner-cafe-123456790ab"
        }
      }
    }
  ]
}
```

```

        },
        "ForAnyValue:StringLike": {
            "cognito-identity.amazonaws.com:amr": "authenticated"
        }
    }
}
]
}
}

```

This policy allows federated users from `cognito-identity.amazonaws.com` (the issuer of the OpenID Connect token) to assume this role. Additionally, the policy restricts the aud of the token, in this case the identity pool ID, to match the identity pool. Finally, the policy specifies that one of the array members of the multi-value `amr` claim of the token issued by the Amazon Cognito GetOpenIdToken API action has the value `authenticated`.

Granting pass role permission

To allow an IAM user to set roles with permissions in excess of the user's existing permissions on an identity pool, you grant that `iam:PassRole` permission to pass the role to the `set-identity-pool-roles` API. For example, if the user cannot write to Amazon S3, but the IAM role that the user sets on the identity pool grants write permission to Amazon S3, the user can only set this role if `iam:PassRole` permission is granted for the role. The following example policy shows how to allow `iam:PassRole` permission.

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myS3WriteAccessRole"
            ]
        }
    ]
}

```

In this policy example, the `iam:PassRole` permission is granted for the `myS3WriteAccessRole` role. The role is specified using the role's Amazon Resource Name (ARN). You must also attach this policy to your IAM user or role to which your user belongs. For more information, see [Working with Managed Policies](#).

Note

Lambda functions use resource-based policy, where the policy is attached directly to the Lambda function itself. When creating a rule that invokes a Lambda function, you do not pass a role, so the user creating the rule does not need the `iam:PassRole` permission. For more information about Lambda function authorization, see [Manage Permissions: Using a Lambda Function Policy](#).

Using tokens to assign roles to users

For users who log in through Amazon Cognito user pools, roles can be passed in the ID token that was assigned by the user pool. The roles appear in the following claims in the ID token:

- The `cognito:preferred_role` claim is the role ARN.

- The `cognito:roles` claim is a comma-separated string containing a set of allowed role ARNs.

The claims are set as follows:

- The `cognito:preferred_role` claim is set to the role from the group with the best (lowest) Precedence value. If there is only one allowed role, `cognito:preferred_role` is set to that role. If there are multiple roles and no single role has the best precedence, this claim is not set.
- The `cognito:roles` claim is set if there is at least one role.

When using tokens to assign roles, if there are multiple roles that can be assigned to the user, Amazon Cognito identity pools (federated identities) chooses the role as follows:

- Use the [GetCredentialsForIdentity](#) `CustomRoleArn` parameter if it is set and it matches a role in the `cognito:roles` claim. If this parameter doesn't match a role in `cognito:roles`, deny access.
- If the `cognito:preferred_role` claim is set, use it.
- If the `cognito:preferred_role` claim is not set, the `cognito:roles` claim is set, and `CustomRoleArn` is not specified in the call to [GetCredentialsForIdentity](#), then the **Role resolution** setting in the console or the `AmbiguousRoleResolution` field (in the `RoleMappings` parameter of the [SetIdentityPoolRoles](#) API) is used to determine the role to be assigned.

Using rule-based mapping to assign roles to users

Rules allow you to map claims from an identity provider token to IAM roles.

Each rule specifies a token claim (such as a user attribute in the ID token from an Amazon Cognito user pool), match type, a value, and an IAM role. The match type can be `Equals`, `NotEqual`, `StartsWith`, or `Contains`. If a user has a matching value for the claim, the user can assume that role when the user gets credentials. For example, you can create a rule that assigns a specific IAM role for users with a `custom:dept` custom attribute value of `Sales`.

Note

In the rule settings, custom attributes require the `custom:` prefix to distinguish them from standard attributes.

Rules are evaluated in order, and the IAM role for the first matching rule is used, unless `CustomRoleArn` is specified to override the order. For more information about user attributes in Amazon Cognito user pools, see [User pool attributes \(p. 206\)](#).

You can set multiple rules for an authentication provider in the identity pool (federated identities) console. Rules are applied in order. You can drag the rules to change their order. The first matching rule takes precedence. If the match type is `NotEqual` and the claim doesn't exist, the rule is not evaluated. If no rules match, the **Role resolution** setting is applied to either **Use default Authenticated role** or **DENY**.

In the API and CLI, you can specify the role to be assigned when no rules match in the `AmbiguousRoleResolution` field of the `RoleMapping` type, which is specified in the `RoleMappings` parameter of the [SetIdentityPoolRoles](#) API.

You can set up rule-based mapping for OpenID Connect (OIDC) and SAML identity providers in the AWS CLI or API with the `RulesConfiguration` field of the `RoleMapping` type. You can specify this field in the `RoleMappings` parameter of the [SetIdentityPoolRoles](#) API. The AWS Management Console currently doesn't allow you to add rules for OIDC or SAML providers.

For example, the following AWS CLI command adds a rule that assigns the role `arn:aws:iam::123456789012:role/Sacramento_team_S3_admin` to users in your Sacramento location who were authenticated by OIDC IdP `arn:aws:iam::123456789012:oidc-provider/myOIDCIdP`:

```
aws cognito-identity set-identity-pool-roles --region us-east-1 --cli-input-json file://role-mapping.json
```

Contents of role-mapping.json:

```
{
    "IdentityPoolId": "us-east-1:12345678-corner-cafe-123456790ab",
    "Roles": {
        "authenticated": "arn:aws:iam::123456789012:role/myS3WriteAccessRole",
        "unauthenticated": "arn:aws:iam::123456789012:role/myS3ReadAccessRole"
    },
    "RoleMappings": {
        "arn:aws:iam::123456789012:oidc-provider/myOIDCIdP": {
            "Type": "Rules",
            "AmbiguousRoleResolution": "AuthenticatedRole",
            "RulesConfiguration": {
                "Rules": [
                    {
                        "Claim": "locale",
                        "MatchType": "Equals",
                        "Value": "Sacramento",
                        "RoleARN": "arn:aws:iam::123456789012:role/
Sacramento_team_S3_admin"
                    }
                ]
            }
        }
    }
}
```

For each user pool or other authentication provider that you configure for an identity pool, you can create up to 25 rules. This limit is not adjustable. For more information, see [Quotas in Amazon Cognito](#).

Token claims to use in rule-based mapping

Amazon Cognito

An Amazon Cognito ID token is represented as a JSON Web Token (JWT). The token contains claims about the identity of the authenticated user, such as `name`, `family_name`, and `phone_number`. For more information about standard claims, see the [OpenID Connect specification](#). Apart from standard claims, the following are the additional claims specific to Amazon Cognito:

- `cognito:groups`
- `cognito:roles`
- `cognito:preferred_role`

Amazon

The following claims, along with possible values for those claims, can be used with Login with Amazon:

- `iss: www.amazon.com`
- `aud: App Id`
- `sub: sub from the Login with Amazon token`

Facebook

The following claims, along with possible values for those claims, can be used with Facebook:

- iss: graph.facebook.com
- aud: App Id
- sub: sub from the Facebook token

Google

A Google token contains standard claims from the [OpenID Connect specification](#). All of the claims in the OpenID token are available for rule-based mapping. See Google's [OpenID Connect](#) site to learn about the claims available from the Google token.

Apple

An Apple token contains standard claims from the [OpenID Connect specification](#). See [Authenticating Users with Sign in with Apple](#) in Apple's documentation to learn more about the claim available from the Apple token. Apple's token doesn't always contain email.

OpenID

All of the claims in the Open Id token are available for rule-based mapping. For more information about standard claims, see the [OpenID Connect specification](#). Refer to your OpenID provider documentation to learn about any additional claims that are available.

SAML

Claims are parsed from the received SAML assertion. All the claims that are available in the SAML assertion can be used in rule-based mapping.

Best practices for role-based access control

Important

If the claim that you are mapping to a role can be modified by the end user, any end user can assume your role and set the policy accordingly. Only map claims that cannot be directly set by the end user to roles with elevated permissions. In an Amazon Cognito user pool, you can set per-app read and write permissions for each user attribute.

Important

If you set roles for groups in an Amazon Cognito user pool, those roles are passed through the user's ID token. To use these roles, you must also set **Choose role from token** for the authenticated role selection for the identity pool.

You can use the **Role resolution** setting in the console and the `RoleMappings` parameter of the [SetIdentityPoolRoles](#) API to specify what the default behavior is when the correct role cannot be determined from the token.

Getting credentials

You can use Amazon Cognito to deliver temporary, limited-privilege credentials to your application, so that your users can access AWS resources. This section describes how to get credentials and how to retrieve an Amazon Cognito identity from an identity pool.

Amazon Cognito supports both authenticated and unauthenticated identities. Unauthenticated users do not have their identity verified, making this role appropriate for guest users of your app or in cases when it doesn't matter if users have their identities verified. Authenticated users log in to your application through a third-party identity provider, or a user pool, that verifies their identities. Make sure you scope the permissions of resources appropriately so you don't grant access to them from unauthenticated users.

Amazon Cognito identities are not credentials. They are exchanged for credentials using web identity federation support in the AWS Security Token Service (AWS STS). The recommended way to obtain AWS credentials for your app users is to use `AWS.CognitoIdentityCredentials`. The identity in the `credentials` object is then exchanged for credentials using AWS STS.

Note

If you created your identity pool before February 2015, you must reassociate your roles with your identity pool in order to use the `AWS.CognitoIdentityCredentials` constructor without the roles as parameters. To do so, open the [Amazon Cognito console](#), choose **Manage identity pools**, select your identity pool, choose **Edit identity Pool**, specify your authenticated and unauthenticated roles, and save the changes.

Android

You can use Amazon Cognito to deliver temporary, limited-privilege credentials to your application, so that your users can access AWS resources. Amazon Cognito supports both authenticated and unauthenticated identities. To provide AWS credentials to your app, follow the steps below.

1. Choose **Manage identity pools** from the [Amazon Cognito console](#), create an identity pool, and copy the starter code snippets.
2. If you haven't already done so, add the AWS Mobile SDK for Android to your project. For instructions, see [Set Up the Mobile SDK for Android](#).
3. Include the following import statements:

```
import com.amazonaws.auth.CognitoCachingCredentialsProvider;
import com.amazonaws.regions.Regions;
```

4. Initialize the Amazon Cognito credentials provider using the code snippet generated by the Amazon Cognito console. The value for `IDENTITY_POOL_ID` will be specific to your account:

```
CognitoCachingCredentialsProvider credentialsProvider = new
CognitoCachingCredentialsProvider(
    getApplicationContext(), // Context
    "IDENTITY_POOL_ID", // Identity Pool ID
    Regions.US_EAST_1 // Region
);
```

5. Pass the initialized Amazon Cognito credentials provider to the constructor of the AWS client to be used. The code required depends on the service to be initialized. The client will use this provider to get credentials with which it will access AWS resources.

Note

If you created your identity pool before February 2015, you must reassociate your roles with your identity pool in order to use this constructor without the roles as parameters. To do so, open the [Amazon Cognito console](#), choose **Manage Federated Identities**, select your identity pool, and choose **Edit identity Pool**, specify your authenticated and unauthenticated roles, and save the changes.

Retrieving an Amazon Cognito identity

If you're allowing unauthenticated users, you can retrieve a unique Amazon Cognito identifier (identity ID) for your end user immediately. If you're authenticating users, you can retrieve the identity ID after you've set the login tokens in the credentials provider:

```
String identityId = credentialsProvider.getIdentityId();
Log.d("LogTag", "my ID is " + identityId);
```

Note

Do not call `getIdentityId()`, `refresh()`, or `getCredentials()` in the main thread of your application. As of Android 3.0 (API Level 11), your app will automatically fail and throw a [NetworkOnMainThreadException](#) if you perform network I/O on the main application thread. You must move your code to a background thread using `AsyncTask`. For more information, consult the [Android documentation](#). You can also call `getCachedIdentityId()` to retrieve an ID, but only if one is already cached locally. Otherwise, the method will return null.

iOS - Objective-C

You can use Amazon Cognito to deliver temporary, limited-privilege credentials to your application, so that your users can access AWS resources. Amazon Cognito identity pools support both authenticated and unauthenticated identities. To provide AWS credentials to your app, follow the steps below.

1. Choose **Manage identity pools** from the [Amazon Cognito console](#), create an identity pool, and copy the starter code snippets.
2. If you haven't already done so, add the AWS Mobile SDK for iOS to your project. For instructions, see [Set Up the Mobile SDK for iOS](#).
3. In your source code, include the `AWSCore` header:

```
#import <AWSCore/AWSCore.h>
```

4. Initialize the Amazon Cognito credentials provider using the code snippet generated by the Amazon Cognito console. The value for `IDENTITY_POOL_ID` will be specific to your account:

```
AWSCognitoCredentialsProvider *credentialsProvider = [[AWSCognitoCredentialsProvider alloc]
initWithRegionType:AWSRegionUSEast1 identityPoolId:@"IDENTITY_POOL_ID"];
AWSServiceConfiguration *configuration = [[AWSServiceConfiguration alloc]
initWithRegion:AWSRegionUSEast1 credentialsProvider:credentialsProvider];
AWSServiceManager.defaultServiceManager.defaultServiceConfiguration = configuration;
```

Note

If you created your identity pool before February 2015, you must reassociate your roles with your identity pool in order to use this constructor without the roles as parameters. To do so, open the [Amazon Cognito console](#), choose **Manage identity pools**, select your identity pool, choose **Edit identity Pool**, specify your authenticated and unauthenticated roles, and save the changes.

Retrieving an Amazon Cognito identity

You can retrieve a unique Amazon Cognito identifier (identity ID) for your end user immediately if you're allowing unauthenticated users or after you've set the login tokens in the credentials provider if you're authenticating users:

```
// Retrieve your Amazon Cognito ID
[[credentialsProvider getIdentityId] continueWithBlock:^id(AWSTask *task) {
    if (task.error) {
        NSLog(@"Error: %@", task.error);
    }
    else {
        // the task result will contain the identity id
        NSString *cognitoId = task.result;
    }
    return nil;
}];
```

```
 }];
```

Note

`getIdentityId` is an asynchronous call. If an identity ID is already set on your provider, you can call `credentialsProvider.identityId` to retrieve that identity, which is cached locally. However, if an identity ID is not set on your provider, calling `credentialsProvider.identityId` will return `nil`. For more information, consult the [Amplify iOS SDK reference](#).

iOS - Swift

You can use Amazon Cognito to deliver temporary, limited-privilege credentials to your application so that your users can access AWS resources. Amazon Cognito supports both authenticated and unauthenticated identities. To provide AWS credentials to your app, follow the steps below.

1. Choose **Manage identity pools** from the [Amazon Cognito console](#), create an identity pool, and copy the starter code snippets.
2. If you haven't already done so, add the Mobile SDK for iOS to your project. For instructions, see [Set Up the SDK for iOS](#).
3. In your source code, include the `AWSCore` header:

```
import AWSCore
```

4. Initialize the Amazon Cognito credentials provider using the code snippet generated by the Amazon Cognito console. The value for `IDENTITY_POOL_ID` will be specific to your account:

```
let credentialsProvider = AWSCognitoCredentialsProvider(regionType: .USEast1,
    identityPoolId: "IDENTITY_POOL_ID")
let configuration = AWSServiceConfiguration(region: .USEast1, credentialsProvider:
    credentialsProvider)
AWSServiceManager.default().defaultServiceConfiguration = configuration
```

Note

If you created your identity pool before February 2015, you must reassociate your roles with your identity pool in order to use this constructor without the roles as parameters. To do so, open the [Amazon Cognito console](#), choose **Manage identity pools**, select your identity pool, choose **Edit identity Pool**, specify your authenticated and unauthenticated roles, and save the changes.

Retrieving an Amazon Cognito identity

You can retrieve a unique Amazon Cognito identifier (identity ID) for your end user immediately if you're allowing unauthenticated users or after you've set the login tokens in the credentials provider if you're authenticating users:

```
// Retrieve your Amazon Cognito ID
credentialsProvider.getIdentityId().continueWith(block: { (task) -> AnyObject? in
    if (task.error != nil) {
        print("Error: " + task.error!.localizedDescription)
    }
    else {
        // the task result will contain the identity id
        let cognitoId = task.result!
        print("Cognito id: \(cognitoId)")
    }
    return task;
}
```

})

Note

`getIdentityId` is an asynchronous call. If an identity ID is already set on your provider, you can call `credentialsProvider.identityId` to retrieve that identity, which is cached locally. However, if an identity ID is not set on your provider, calling `credentialsProvider.identityId` will return `nil`. For more information, consult the [Amplify iOS SDK reference](#).

JavaScript

If you have not yet created one, create an identity pool in the [Amazon Cognito console](#) before using `AWS.CognitoIdentityCredentials`.

After you configure an identity pool with your identity providers, you can use `AWS.CognitoIdentityCredentials` to authenticate users. To configure your application credentials to use `AWS.CognitoIdentityCredentials`, set the `credentials` property of either `AWS.Config` or a per-service configuration. The following example uses `AWS.Config`:

```
// Set the region where your identity pool exists (us-east-1, eu-west-1)
AWS.config.region = 'us-east-1';

// Configure the credentials provider to use your identity pool
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: 'IDENTITY_POOL_ID',
    Logins: { // optional tokens, used for authenticated login
        'graph.facebook.com': 'FBTOKEN',
        'www.amazon.com': 'AMAZONTOKEN',
        'accounts.google.com': 'GOOGLETOKEN',
        'appleid.apple.com': 'APPLETOKEN'
    }
});

// Make the call to obtain credentials
AWS.config.credentials.get(function(){

    // Credentials will be available when this function is called.
    var accessKeyId = AWS.config.credentials.accessKeyId;
    var secretAccessKey = AWS.config.credentials.secretAccessKey;
    var sessionToken = AWS.config.credentials.sessionToken;

});
```

The optional `Logins` property is a map of identity provider names to the identity tokens for those providers. How you get the token from your identity provider depends on the provider you use. For example, if Facebook is one of your identity providers, you might use the `FB.login` function from the [Facebook SDK](#) to get an identity provider token:

```
FB.login(function (response) {
    if (response.authResponse) { // logged in
        AWS.config.credentials = new AWS.CognitoIdentityCredentials({
            IdentityPoolId: 'us-east-1:1699ebc0-7900-4099-b910-2df94f52a030',
            Logins: {
                'graph.facebook.com': response.authResponse.accessToken
            }
        });

        console.log('You are now logged in.');
    } else {
        console.log('There was a problem logging you in.');
    }
});
```

```
}); }
```

Retrieving an Amazon Cognito identity

You can retrieve a unique Amazon Cognito identifier (identity ID) for your end user immediately if you're allowing unauthenticated users or after you've set the login tokens in the credentials provider if you're authenticating users:

```
var identityId = AWS.config.credentials.identityId;
```

Unity

You can use Amazon Cognito to deliver temporary, limited-privilege credentials to your application, so that your users can access AWS resources. Amazon Cognito supports both authenticated and unauthenticated identities. To provide AWS credentials to your app, follow the steps below.

1. Choose **Manage identity pools**, from the [Amazon Cognito console](#), create an identity pool, and copy the starter code snippets.
2. If you haven't already done so, download and import the [AWS Mobile SDK for Unity](#) package into your project. You can do so from the menu Assets > Import Package > Custom Package.
3. Paste the starter code snippet from the Console into the script from which you want to call Amazon Cognito. The value for `IDENTITY_POOL_ID` will be specific to your account:

```
CognitoAWSCredentials credentials = new CognitoAWSCredentials (
    "IDENTITY_POOL_ID",      // Cognito identity Pool ID
    RegionEndpoint.USEast1 // Region
);
```

4. Pass the initialized Amazon Cognito credentials to the constructor of the AWS client to be used. The code required depends on the service to be initialized. The client will use this provider to get credentials with which it will access AWS resources.

Note

If you created your identity pool before February 2015, you must to reassociate your roles with your identity pool in order to use this constructor without the roles as parameters. To do so, open the [Amazon Cognito console](#), choose **Manage identity pools**, select your identity pool, choose **Edit identity Pool**, specify your authenticated and unauthenticated roles, and save the changes.

Retrieving an Amazon Cognito identity

You can retrieve a unique Amazon Cognito identifier (identity ID) for your end user immediately if you're allowing unauthenticated users or after you've set the login tokens in the credentials provider if you're authenticating users:

```
credentials.GetIdentityIdAsync(delegate(AmazonCognitoIdentityResult<string> result) {
    if (result.Exception != null) {
        //Exception!
    }
    string identityId = result.Response;
});
```

Xamarin

You can use Amazon Cognito to deliver temporary, limited-privilege credentials to your application so that your users can access AWS resources. Amazon Cognito supports both authenticated and unauthenticated identities. To provide AWS credentials to your app, follow the steps below.

1. Choose **Manage identity pools**, from the [Amazon Cognito console](#), create an identity pool, and copy the starter code snippets.
2. If you haven't already done so, add the AWS Mobile SDK for Xamarin to your project. For instructions, see [Set Up the SDK for Xamarin](#).
3. Include the following using statements:

```
using Amazon.CognitoIdentity;
```

4. Paste the starter code snippet from the Console into the script from which you want to call Amazon Cognito. The value for `IDENTITY_POOL_ID` will be specific to your account:

```
CognitoAWSCredentials credentials = new CognitoAWSCredentials (
    "IDENTITY_POOL_ID",           // Cognito identity Pool ID
    RegionEndpoint.USEast1        // Region
);
```

5. Pass the initialized Amazon Cognito credentials to the constructor of the AWS client to be used. The code required depends on the service to be initialized. The client will use this provider to get credentials with which it will access AWS resources.

Note

Note: If you created your identity pool before February 2015, you must reassociate your roles with your identity pool in order to use this constructor without the roles as parameters. To do so, open the [Amazon Cognito console](#), choose **Manage identity pools**, select your identity pool, choose **Edit identity Pool**, specify your authenticated and unauthenticated roles, and save the changes.

Retrieving an Amazon Cognito identity

You can retrieve a unique Amazon Cognito identifier (identity ID) for your end user immediately if you're allowing unauthenticated users or after you've set the login tokens in the credentials provider if you're authenticating users:

```
var identityId = await credentials.GetIdentityIdAsync();
```

Accessing AWS services

Once the Amazon Cognito credentials provider is initialized and refreshed, you can pass it directly to the initializer for an AWS client. For example, the following snippet initializes an Amazon DynamoDB client:

Android

```
// Create a service client with the provider
AmazonDynamoDB client = new AmazonDynamoDBClient(credentialsProvider);
```

The credentials provider communicates with Amazon Cognito, retrieving both the unique identifier for authenticated and unauthenticated users as well as temporary, limited privilege AWS credentials for the AWS Mobile SDK. The retrieved credentials are valid for one hour, and the provider refreshes them when they expire.

iOS - Objective-C

```
// create a configuration that uses the provider
AWSServiceConfiguration *configuration = [AWSServiceConfiguration
    configurationWithRegion:AWSRegionUSEast1 provider:credentialsProvider];

// get a client with the default service configuration
AWSDynamoDB *dynamodb = [AWSDynamoDB defaultDynamoDB];
```

The credentials provider communicates with Amazon Cognito, retrieving both the unique identifier for authenticated and unauthenticated users as well as temporary, limited privilege AWS credentials for the AWS Mobile SDK. The retrieved credentials are valid for one hour, and the provider refreshes them when they expire.

iOS - Swift

```
// get a client with the default service configuration
let dynamoDB = AWSDynamoDB.default()

// get a client with a custom configuration
AWSDynamoDB.register(with: configuration!, forKey: "USWest2DynamoDB");
let dynamoDBCustom = AWSDynamoDB(forKey: "USWest2DynamoDB")
```

The credentials provider communicates with Amazon Cognito, retrieving both the unique identifier for authenticated and unauthenticated users as well as temporary, limited privilege AWS credentials for the AWS Mobile SDK. The retrieved credentials are valid for one hour, and the provider refreshes them when they expire.

JavaScript

```
// Create a service client with the provider
var dynamodb = new AWS.DynamoDB({region: 'us-west-2'});
```

The credentials provider communicates with Amazon Cognito, retrieving both the unique identifier for authenticated and unauthenticated users as well as temporary, limited-privilege AWS credentials for the AWS Mobile SDK. The retrieved credentials are valid for one hour, and the provider refreshes them when they expire.

Unity

```
// create a service client that uses credentials provided by Cognito
AmazonDynamoDBClient client = new AmazonDynamoDBClient(credentials, REGION);
```

The credentials provider communicates with Amazon Cognito, retrieving both the unique identifier for authenticated and unauthenticated users as well as temporary, limited-privilege AWS credentials for the AWS Mobile SDK. The retrieved credentials are valid for one hour, and the provider refreshes them when they expire.

Xamarin

```
// create a service client that uses credentials provided by Cognito
```

```
var client = new AmazonDynamoDBClient(credentials, REGION)
```

The credentials provider communicates with Amazon Cognito, retrieving both the unique identifier for authenticated and unauthenticated users as well as temporary, limited-privilege AWS credentials for the AWS Mobile SDK. The retrieved credentials are valid for one hour, and the provider refreshes them when they expire.

Identity pools (federated identities) external identity providers

Using the `logins` property, you can set credentials received from an identity provider (IdP). Furthermore, you can associate an identity pool with multiple IdPs. For example, you can set both the Facebook and Google tokens in the `logins` property to associate the unique Amazon Cognito identity with both IdP logins. The user can authenticate with either account, but Amazon Cognito returns the same user identifier.

The instructions below guide you through authentication with the IdPs that Amazon Cognito identity pools support.

Topics

- [Facebook \(identity pools\) \(p. 276\)](#)
- [Login with Amazon \(identity pools\) \(p. 281\)](#)
- [Google \(identity pools\) \(p. 284\)](#)
- [Sign in with Apple \(identity pools\) \(p. 291\)](#)
- [Open ID Connect providers \(identity pools\) \(p. 295\)](#)
- [SAML identity providers \(identity pools\) \(p. 297\)](#)

Facebook (identity pools)

Amazon Cognito identity pools integrate with Facebook to provide federated authentication for your mobile application users. This section explains how to register and set up your application with Facebook as an IdP.

Set up Facebook

Register your application with Facebook before you authenticate Facebook users and interact with Facebook APIs.

The [Facebook Developers portal](#) helps you to set up your application. Do this procedure before you integrate Facebook in your Amazon Cognito identity pool:

Setting up Facebook

1. At the [Facebook Developers portal](#), log in with your Facebook credentials.
2. From the **Apps** menu, select **Add a New App**.
3. Select a platform and complete the quick start process.

Android

For more information about how to integrate Android apps with Facebook Login, see the [Facebook Getting Started Guide](#).

iOS - Objective-C

For more information about how to integrate iOS Objective-C apps with Facebook Login, see the [Facebook Getting Started Guide](#).

iOS - Swift

For more information about how to integrate iOS Swift apps with Facebook Login, see the [Facebook Getting Started Guide](#).

JavaScript

For more information about how to integrate JavaScript web apps with Facebook Login, see the [Facebook Getting Started Guide](#).

Unity

For more information about how to integrate Unity apps with Facebook Login, see the [Facebook Getting Started Guide](#).

Xamarin

To add Facebook authentication, first follow the appropriate flow below to integrate the Facebook SDK into your application. Amazon Cognito identity pools use the Facebook access token to generate a unique user identifier that is associated with an Amazon Cognito identity.

- [Facebook iOS SDK by Xamarin](#)
- [Facebook Android SDK by Xamarin](#)

Configure the external provider in the Amazon Cognito federated identities console

Use the following procedure to configure your external provider.

1. Choose **Manage Identity Pools** from the [Amazon Cognito console home page](#).
2. Choose the name of the identity pool where you want to enable Facebook as an external provider. The **Dashboard** page for your identity pool appears.
3. In the top-right corner of the **Dashboard** page, choose **Edit identity pool**. The **Edit identity pool** page appears.
4. Scroll down and choose **Authentication providers** to expand the section.
5. Choose the **Facebook** tab.
6. Choose **Unlock**.
7. Enter the Facebook App ID you obtained from Facebook, and then choose **Save Changes**.

Using Facebook

Android

To add Facebook authentication, first follow the [Facebook guide](#) and integrate the Facebook SDK into your application. Then add a [Login with Facebook button](#) to your Android user interface. The Facebook SDK uses a session object to track its state. Amazon Cognito uses the access token from this session

object to authenticate the user, generate the unique identifier, and, if needed, grant the user access to other AWS resources.

After you authenticate your user with the Facebook SDK, add the session token to the Amazon Cognito credentials provider.

Facebook SDK 4.0 or later:

```
Map<String, String> logins = new HashMap<String, String>();
logins.put("graph.facebook.com", AccessToken.getCurrentAccessToken().getAccessToken());
credentialsProvider.setLogins(logins);
```

Facebook SDK before 4.0:

```
Map<String, String> logins = new HashMap<String, String>();
logins.put("graph.facebook.com", Session.getActiveSession().getAccessToken());
credentialsProvider.setLogins(logins);
```

The Facebook login process initializes a singleton session in its SDK. The Facebook session object contains an OAuth token that Amazon Cognito uses to generate AWS credentials for your authenticated end user. Amazon Cognito also uses the token to check against your user database for the existence of a user that matches this particular Facebook identity. If the user already exists, the API returns the existing identifier. Otherwise, the API returns a new identifier. The client SDK automatically caches identifiers on the local device.

Note

After you set the logins map, make a call to `refresh` or `get` to retrieve the AWS credentials.

iOS - Objective-C

To add Facebook authentication, first follow the [Facebook guide](#) and integrate the Facebook SDK into your application. Then add a [Login with Facebook button](#) to your user interface. The Facebook SDK uses a session object to track its state. Amazon Cognito uses the access token from this session object to authenticate the user and bind them to a unique Amazon Cognito identity pools (federated identities).

To provide the Facebook access token to Amazon Cognito, implement the [AWSIdentityProviderManager](#) protocol.

When you implement the `logins` method, return a dictionary that contains `AWSIdentityProviderFacebook`. This dictionary acts as the key, and the current access token from the authenticated Facebook user acts as the value, as shown in the following code example.

```
- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    FBSDKAccessToken* fbToken = [FBSDKAccessToken currentAccessToken];
    if(fbToken){
        NSString *token = fbToken.tokenString;
        return [AWSTask taskWithResult: @{@"AWSIdentityProviderFacebook": token }];
    }else{
        return [AWSTask taskWithError:[NSError errorWithDomain:@"Facebook Login"
                                         code:-1
                                         userInfo:@{"error":@"No current
Facebook access token"}]];
    }
}
```

When you instantiate the `AWSCognitoCredentialsProvider`, pass the class that implements `AWSIdentityProviderManager` as the value of `identityProviderManager` in the constructor. For more information, go to the [AWSCognitoCredentialsProvider](#) reference page and choose `initWithRegionType:identityPoolId:identityProviderManager`.

iOS - Swift

To add Facebook authentication, first follow the [Facebook guide](#) and integrate the Facebook SDK into your application. Then add a [Login with Facebook button](#) to your user interface. The Facebook SDK uses a session object to track its state. Amazon Cognito uses the access token from this session object to authenticate the user and bind them to a unique Amazon Cognito identity pools (federated identities).

To provide the Facebook access token to Amazon Cognito, implement the [AWSIdentityProviderManager](#) protocol.

When you implement the `logins` method, return a dictionary containing `AWSIdentityProviderFacebook`. This dictionary acts as the key, and the current access token from the authenticated Facebook user acts as the value, as shown in the following code example.

```
class FacebookProvider: NSObject, AWSIdentityProviderManager {
    func logins() -> AWSTask<NSDictionary> {
        if let token = AccessToken.current?.authenticationToken {
            return AWSTask(result: [AWSIdentityProviderFacebook:token])
        }
        return AWSTask(error:NSError(domain: "Facebook Login", code: -1 , userInfo: [
            "Facebook" : "No current Facebook access token"
        ]))
    }
}
```

When you instantiate the `AWSCognitoCredentialsProvider`, pass the class that implements `AWSIdentityProviderManager` as the value of `identityProviderManager` in the constructor. For more information, go to the [AWSCognitoCredentialsProvider](#) reference page and choose `initWithRegionType:identityPoolId:identityProviderManager`.

JavaScript

To add Facebook authentication, follow the [Facebook Login for the Web](#) and add the [Login with Facebook](#) button on your website. The Facebook SDK uses a session object to track its state. Amazon Cognito uses the access token from this session object to authenticate the user, generate the unique identifier, and, if needed, grant the user access to other AWS resources.

After you authenticate your user with the Facebook SDK, add the session token to the Amazon Cognito credentials provider.

```
FB.login(function (response) {
    // Check if the user logged in successfully.
    if (response.authResponse) {
        console.log('You are now logged in.');

        // Add the Facebook access token to the Amazon Cognito credentials login map.
        AWS.config.credentials = new AWS.CognitoIdentityCredentials({
            IdentityPoolId: 'IDENTITY_POOL_ID',
            Logins: {
                'graph.facebook.com': response.authResponse.accessToken
            }
        });

        // Obtain AWS credentials
        AWS.config.credentials.get(function(){
            // Access AWS resources here.
        });
    } else {
        console.log('There was a problem logging you in.');
    }
});
```

```
    }
});
```

The Facebook SDK obtains an OAuth token that Amazon Cognito uses to generate AWS credentials for your authenticated end user. Amazon Cognito also uses the token to check against your user database for the existence of a user matching this particular Facebook identity. If the user already exists, the API returns the existing identifier. Otherwise a new identifier is returned. Identifiers are automatically cached by the client SDK on the local device.

Note

After you set the logins map, make a call to `refresh` or `get` to get the credentials. For a code example, see "Use Case 17, Integrating User Pools with Cognito Identity," in the [JavaScript README file](#).

Unity

To add Facebook authentication, first follow the [Facebook guide](#) and integrate the Facebook SDK into your application. Amazon Cognito uses the Facebook access token from the `FB` object to generate a unique user identifier that is associated with an Amazon Cognito identity.

After you authenticate your user with the Facebook SDK, add the session token to the Amazon Cognito credentials provider:

```
void Start()
{
    FB.Init(delegate() {
        if (FB.IsLoggedIn) { //User already logged in from a previous session
            AddFacebookTokenToCognito();
        } else {
            FB.Login ("email", FacebookLoginCallback);
        }
    });
}

void FacebookLoginCallback(FBResult result)
{
    if (FB.IsLoggedIn)
    {
        AddFacebookTokenToCognito();
    }
    else
    {
        Debug.Log("FB Login error");
    }
}

void AddFacebookTokenToCognito()
{
    credentials.AddLogin ("graph.facebook.com",
    AccessToken.CurrentAccessToken.TokenString);
}
```

Before you use `FB.AccessToken`, call `FB.Login()` and make sure `FB.IsLoggedIn` is true.

Xamarin

Xamarin for Android:

```
public void InitializeFacebook() {
    FacebookSdk.SdkInitialize(this.ApplicationContext);
```

```
callbackManager = CallbackManagerFactory.Create();
LoginManager.Instance.RegisterCallback(callbackManager, new FacebookCallback &lt;
LoginResult &gt; () {
    HandleSuccess = loginResult = &gt; {
        var accessToken = loginResult.AccessToken;
        credentials.AddLogin("graph.facebook.com", accessToken.Token);
        //open new activity
    },
    HandleCancel = () = &gt; {
        //throw error message
    },
    HandleError = loginError = &gt; {
        //throw error message
    }
});
LoginManager.Instance.LogInWithReadPermissions(this, new List &lt; string &gt; {
    "public_profile"
});
```

Xamarin for iOS:

```
public void InitializeFacebook() {
    LoginManager login = new LoginManager();
    login.LogInWithReadPermissions(readPermissions.ToArray(),
delegate(LoginManagerLoginResult result, NSError error) {
    if (error != null) {
        //throw error message
    } else if (result.IsCancelled) {
        //throw error message
    } else {
        var accessToken = loginResult.AccessToken;
        credentials.AddLogin("graph.facebook.com", accessToken.Token);
        //open new view controller
    }
});
```

Login with Amazon (identity pools)

Amazon Cognito integrates with Login with Amazon to provide federated authentication for your mobile and web app users. This section explains how to register and set up your application with Login with Amazon as an identity provider (IdP).

Set up Login with Amazon to work with Amazon Cognito in the [Developer Portal](#). For more information, see [Setting Up Login with Amazon](#) in the Login with Amazon FAQ.

Note

To integrate Login with Amazon into a Xamarin application, follow the [Xamarin Getting Started Guide](#).

Note

You can't natively integrate Login with Amazon on the Unity platform. Instead, use a web view and go through the browser sign-in flow.

Setting up Login with Amazon

Implement Login with Amazon

In the [Amazon developer portal](#), you can set up an OAuth application to integrate with your identity pool, find Login with Amazon documentation, and download SDKs. Choose **Developer**

console, then **Login with Amazon** in the developer portal. You can create a security profile for your application and then build Login with Amazon authentication mechanisms into your app. See [Getting credentials \(p. 268\)](#) for more information about how to integrate Login with Amazon authentication with your app.

Amazon issues an OAuth 2.0 **client ID** for your new security profile. You can find the **client ID** on the security profile **Web Settings** tab. Enter the **client ID** in the **Amazon App ID** field of the Login with Amazon IdP in your identity pool.

Configure the external provider in the Amazon Cognito console

Choose **Manage Identity Pools** from the [Amazon Cognito console home page](#):

1. Choose the name of the identity pool where you want to enable Login with Amazon as an external provider. The **Dashboard** page for your identity pool appears.
2. In the top-right corner of the **Dashboard** page, choose **Edit identity pool**. The Edit identity pool page appears.
3. Scroll down and choose **Authentication providers** to expand the section.
4. Choose the **Amazon** tab.
5. Choose **Unlock**.
6. Enter the Amazon App ID you obtained from Amazon, and then choose **Save Changes**.

Use Login with Amazon: Android

After you authenticate Amazon login, you can pass the token to the Amazon Cognito credentials provider in the `onSuccess` method of the `TokenListener` interface. The code looks like this:

```
@Override
public void onSuccess(Bundle response) {
    String token = response.getString(AuthzConstants.BUNDLE_KEY.TOKEN.val);
    Map<String, String> logins = new HashMap<String, String>();
    logins.put("www.amazon.com", token);
    credentialsProvider.setLogins(logins);
}
```

Use Login with Amazon: iOS - Objective-C

After you authenticate Amazon login, you can pass the token to the Amazon Cognito credentials provider in the `requestDidSucceed` method of the `AMZNAccessTokenDelegate`:

```
- (void)requestDidSucceed:(APIResult *)apiResult {
    if (apiResult.api == kAPIAuthorizeUser) {
        [AIMobileLib getAccessTokenForScopes:[NSArray arrayWithObject:@"profile"]
        withOverrideParams:nil delegate:self];
    }
    else if (apiResult.api == kAPIGetAccessToken) {
        credentialsProvider.logins = @{@"AWSLoginProviderKeyLoginWithAmazon": apiResult.result};
    }
}
```

Use Login with Amazon: iOS - Swift

After you authenticate Amazon login, you can pass the token to the Amazon Cognito credentials provider in the `requestDidSucceed` method of the `AMZNAccessTokenDelegate`:

```
func requestDidSucceed(apiResult: APIResult!) {
    if apiResult.api == API.AuthorizeUser {
        AIMobileLib.getAccessTokenForScopes(["profile"], withOverrideParams: nil, delegate:
    self)
    } else if apiResult.api == API.GetAccessToken {
        credentialsProvider.logins = [AWSCognitoLoginProviderKey.LoginWithAmazon.rawValue:
    apiResult.result]
    }
}
```

Use Login with Amazon: JavaScript

After the user authenticates with Login with Amazon and is redirected back to your website, the Login with Amazon access_token is provided in the query string. Pass that token into the credentials login map.

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: 'IDENTITY_POOL_ID',
    Logins: {
        'www.amazon.com': 'Amazon Access Token'
    }
});
```

Use Login with Amazon: Xamarin

Xamarin for Android

```
AmazonAuthorizationManager manager = new AmazonAuthorizationManager(this, Bundle.Empty);

var tokenListener = new APIListener {
    Success = response => {
        // Get the auth token
        var token = response.GetString(AuthzConstants.BUNDLE_KEY.Token.Val);
        credentials.AddLogin("www.amazon.com", token);
    }
};

// Try and get existing login
manager.GetToken(new[] {
    "profile"
}, tokenListener);
```

Xamarin for iOS

In `AppDelegate.cs`, insert the following:

```
public override bool OpenUrl (UIApplication application, NSURL url, string
sourceApplication, NSObject annotation)
{
    // Pass on the url to the SDK to parse authorization code from the url
    bool isValidRedirectSignInURL = AIMobileLib.HandleOpenUrl (url, sourceApplication);
    if(!isValidRedirectSignInURL)
        return false;

    // App may also want to handle url
    return true;
}
```

Then, in `ViewController.cs`, do the following:

```
public override void ViewDidLoad ()
{
    base.LoadView ();

    // Here we create the Amazon Login Button
    btnLogin = UIButton.FromType (UIButtonType.RoundedRect);
    btnLogin.Frame = new RectangleF (55, 206, 209, 48);
    btnLoginSetTitle ("Login using Amazon", UIControlState.Normal);
    btnLogin.TouchUpInside += (sender, e) => {
        AIMobileLib.AuthorizeUser (new [] { "profile"}, new AMZNAuthorizationDelegate ());
    };
    View.AddSubview (btnLogin);
}

// Class that handles Authentication Success/Failure
public class AMZNAuthorizationDelegate : AIAuthenticationDelegate
{
    public override void RequestDidSucceed(ApiResult apiResult)
    {
        // Your code after the user authorizes application for requested scopes
        var token = apiResult["access_token"];
        credentials.AddLogin("www.amazon.com", token);
    }

    public override void RequestDidFail(ApiError errorResponse)
    {
        // Your code when the authorization fails
        InvokeOnMainThread(() => new UIAlertView("User Authorization Failed",
errorResponse.Error.Message, null, "Ok", null).Show());
    }
}
```

Google (identity pools)

Amazon Cognito integrates with Google to provide federated authentication for your mobile application users. This section explains how to register and set up your application with Google as an IdP.

Android

Note

If your app uses Google and is available on multiple mobile platforms, you should configure it as an [OpenID Connect Provider \(p. 295\)](#). Add all created client IDs as additional audience values for better integration. To learn more about Google's cross-client identity model, see [Cross-client Identity](#).

Setting up Google

To activate Google Sign-in for Android, create a Google Developers console project for your application.

1. Go to the [Google Developers console](#) and create a new project.
2. Choose **APIs & Services**, then **OAuth consent screen**. Customize the information that Google shows to your users when Google asks for their consent to share their profile data with your app.
3. Choose **Credentials**, then **Create credentials**. Choose **OAuth client ID**. Select **Android** as the **Application type**. Create a separate client ID for each platform where you develop your app.
4. From **Credentials**, choose **Manage service accounts**. Choose **Create service account**. Enter your service account details, and then choose **Create and continue**.
5. Grant the service account access to your project. Grant users access to the service account as your app requires.

6. Choose your new service account, choose the **Keys** tab, and **Add key**. Create and download a new JSON key.

For more information about how to use the Google Developers console, see [Creating and managing projects](#) in the Google Cloud documentation.

For more information about how to integrate Google into your Android app, see [Google Sign-In for Android](#) in the Google Identity documentation.

Configuring the external provider in the Amazon Cognito Console

Choose **Manage Identity Pools** from the [Amazon Cognito Console home page](#):

1. Choose the name of the identity pool where you want to activate Google as an external provider. The **Dashboard** page for your identity pool appears.
2. In the top-right corner of the **Dashboard** page, choose **Edit identity pool**. The Edit identity pool page appears.
3. Scroll down and choose **Authentication providers**. The Edit identity pool page expands to show additional Authentication provider options.
4. Choose the **Google** tab.
5. Choose **Unlock**.
6. Enter the Google Client ID that you created in the Google Cloud console, and then choose **Save Changes**.

Use Google

To enable login with Google in your application, follow the instructions in the [Google documentation for Android](#). When a user signs in, they request an OpenID Connect authentication token from Google. Amazon Cognito then uses the token to authenticate the user and generate a unique identifier.

The following example code shows how to retrieve the authentication token from the Google Play service:

```
GooglePlayServicesUtil.isGooglePlayServicesAvailable(getApplicationContext());
AccountManager am = AccountManager.get(this);
Account[] accounts = am.getAccountsByType(GoogleAuthUtil.GOOGLE_ACCOUNT_TYPE);
String token = GoogleAuthUtil.getToken(getApplicationContext(), accounts[0].name,
    "audience:server:client_id:YOUR_GOOGLE_CLIENT_ID");
Map<String, String> logins = new HashMap<String, String>();
logins.put("accounts.google.com", token);
credentialsProvider.setLogins(logins);
```

iOS - Objective-C

Note

If your app uses Google and is available on multiple mobile platforms, configure Google as an [OpenID Connect Provider](#) (p. 295). Add all created client IDs as additional audience values for better integration. To learn more about Google's cross-client identity model, see [Cross-client Identity](#).

Setting up Google

To enable Google Sign-in for iOS, create a Google Developers console project for your application.

1. Go to the [Google Developers console](#) and create a new project.

2. Choose **APIs & Services**, then **OAuth consent screen**. Customize the information that Google shows to your users when Google asks for their consent to share their profile data with your app.
3. Choose **Credentials**, then **Create credentials**. Choose **OAuth client ID**. Select **iOS** as the **Application type**. Create a separate client ID for each platform where you develop your app.
4. From **Credentials**, choose **Manage service accounts**. Choose **Create service account**. Enter your service account details, and choose **Create and continue**.
5. Grant the service account access to your project. Grant users access to the service account as your app requires.
6. Choose your new service account. Choose the **Keys** tab, and **Add key**. Create and download a new JSON key.

For more information about how to use the Google Developers console, see [Creating and managing projects](#) in the Google Cloud documentation.

For more information about how to integrate Google into your iOS app, see [Google Sign-In for iOS](#) in the Google Identity documentation.

Choose **Manage Identity Pools** from the [Amazon Cognito Console home page](#):

Configuring the external provider in the Amazon Cognito Console

1. Choose the name of the identity pool where you want to enable Google as an external provider. The **Dashboard** page for your identity pool appears.
2. In the top-right corner of the **Dashboard** page, choose **Edit identity pool**. The Edit identity pool page appears.
3. Scroll down and choose **Authentication providers** to expand the section.
4. Choose the **Google** tab.
5. Choose **Unlock**.
6. Enter the Google Client ID that you obtained from Google, and then choose **Save Changes**.

Use Google

To enable login with Google in your application, follow the [Google documentation for iOS](#). Successful authentication results in an OpenID Connect authentication token, which Amazon Cognito uses to authenticate the user and generate a unique identifier.

Successful authentication results in a `GTMOAuth2Authentication` object, which contains an `id_token`, which Amazon Cognito uses to authenticate the user and generate a unique identifier:

```
- (void)finishedWithAuth: (GTMOAuth2Authentication *)auth error: (NSError *) error {
    NSString *idToken = [auth.parameters objectForKey:@"id_token"];
    credentialsProvider.logins = @{@"@{AWSLoginProviderKeyGoogle}": idToken };
}
```

iOS - Swift

Note

If your app uses Google and is available on multiple mobile platforms, configure Google as an [OpenID Connect Provider \(p. 295\)](#). Add all created client IDs as additional audience values for better integration. To learn more about Google's cross-client identity model, see [Cross-client Identity](#).

Setting up Google

To enable Google Sign-in for iOS, create a Google Developers console project for your application.

1. Go to the [Google Developers console](#) and create a new project.
2. Choose **APIs & Services**, then **OAuth consent screen**. Customize the information that Google shows to your users when Google asks for their consent to share their profile data with your app.
3. Choose **Credentials**, then **Create credentials**. Choose **OAuth client ID**. Select **iOS** as the **Application type**. Create a separate client ID for each platform where you develop your app.
4. From **Credentials**, choose **Manage service accounts**. Choose **Create service account**. Enter your service account details, and choose **Create and continue**.
5. Grant the service account access to your project. Grant users access to the service account as your app requires.
6. Choose your new service account, choose the **Keys** tab, and **Add key**. Create and download a new JSON key.

For more information about how to use the Google Developers console, see [Creating and managing projects](#) in the Google Cloud documentation.

For more information about how to integrate Google into your iOS app, see [Google Sign-In for iOS](#) in the Google Identity documentation.

Choose **Manage Identity Pools** from the [Amazon Cognito Console home page](#):

Configuring the external provider in the Amazon Cognito Console

1. Choose the name of the identity pool where you want to enable Google as an external provider. The **Dashboard** page for your identity pool appears.
2. In the top-right corner of the **Dashboard** page, choose **Edit identity pool**. The Edit identity pool page appears.
3. Scroll down and choose **Authentication providers** to expand the section.
4. Choose the **Google** tab.
5. Choose **Unlock**.
6. Enter the Google Client ID that you obtained from Google, and then choose **Save Changes**.

Use Google

To enable login with Google in your application, follow the [Google documentation for iOS](#). Successful authentication results in an OpenID Connect authentication token that Amazon Cognito uses to authenticate the user and generate a unique identifier.

Successful authentication results in a `GTMOAuth2Authentication` object that contains an `id_token`. Amazon Cognito uses this token to authenticate the user and generate a unique identifier:

```
func finishedWithAuth(auth: GTMOAuth2Authentication!, error: NSError!) {
    if error != nil {
        print(error.localizedDescription)
    } else {
        let idToken = auth.parameters.objectForKey("id_token")
        credentialsProvider.logins = [AWSCognitoLoginProviderKey.Google.rawValue: idToken!]
    }
}
```

JavaScript

Note

If your app uses Google and is available on multiple mobile platforms, you should configure Google as an [OpenID Connect Provider \(p. 295\)](#). Add all created client IDs as additional

audience values for better integration. To learn more about Google's cross-client identity model, see [Cross-client Identity](#).

Setting up Google

To enable Google Sign-in for a JavaScript web app, create a Google Developers console project for your application.

1. Go to the [Google Developers console](#) and create a new project.
2. Choose **APIs & Services**, then **OAuth consent screen**. Customize the information that Google shows to your users when Google asks their consent to share their profile data with your app.
3. Choose **Credentials**, then **Create credentials**. Choose **OAuth client ID**. Select **Web application** as the **Application type**. Create a separate client ID for each platform where you develop your app.
4. From **Credentials**, choose **Manage service accounts**. Choose **Create service account**. Enter your service account details, and choose **Create and continue**.
5. Grant the service account access to your project. Grant users access to the service account as your app requires.
6. Choose your new service account, choose the **Keys** tab, and **Add key**. Create and download a new JSON key.

For more information about how to use the Google Developers console, see [Creating and managing projects](#) in the Google Cloud documentation.

For more information about how to integrate Google into your web app, see [Sign in With Google](#) in the Google Identity documentation.

Configure the External Provider in the Amazon Cognito Console

Choose **Manage Identity Pools** from the [Amazon Cognito Console home page](#):

1. Choose the name of the identity pool where you want to enable Google as an external provider. The **Dashboard** page for your identity pool appears.
2. In the top-right corner of the **Dashboard** page, choose **Edit identity pool**. The Edit identity pool page appears.
3. Scroll down and choose **Authentication providers** to expand the section.
4. Choose the **Google** tab.
5. Choose **Unlock**.
6. Enter the Google Client ID you obtained from Google, and then choose **Save Changes**.

Use Google

To enable login with Google in your application, follow the [Google documentation for Web](#).

Successful authentication results in a response object that contains an `id_token` that Amazon Cognito uses to authenticate the user and generate a unique identifier:

```
function signinCallback(authResult) {
  if (authResult['status']['signed_in']) {

    // Add the Google access token to the Amazon Cognito credentials login map.
    AWS.config.credentials = new AWS.CognitoIdentityCredentials({
      IdentityPoolId: 'IDENTITY_POOL_ID',
      Logins: {
        'accounts.google.com': authResult['id_token']
      }
    });
  }
}
```

```
        }
    });

    // Obtain AWS credentials
    AWS.config.credentials.get(function(){
        // Access AWS resources here.
    });
}

}
```

Unity

Setting up Google

To enable Google Sign-in for a Unity app, create a Google Developers console project for your application.

1. Go to the [Google Developers console](#) and create a new project.
2. Choose **APIs & Services**, then **OAuth consent screen**. Customize the information that Google shows to your users when Google asks for their consent to share their profile data with your app.
3. Choose **Credentials**, then **Create credentials**. Choose **OAuth client ID**. Select **Web application** as the **Application type**. Create a separate client ID for each platform where you develop your app.
4. For Unity, create an additional **OAuth client ID** for **Android**, and another for **iOS**.
5. From **Credentials**, choose **Manage service accounts**. Choose **Create service account**. Enter your service account details, and choose **Create and continue**.
6. Grant the service account access to your project. Grant users access to the service account as your app requires.
7. Choose your new service account, choose the **Keys** tab, and **Add key**. Create and download a new JSON key.

For more information about how to use the Google Developers console, see [Creating and managing projects](#) in the Google Cloud documentation.

Create an OpenID Provider in the IAM Console

1. Create an OpenID Provider in the IAM Console. For information about how to set up an OpenID Provider, see [Using OpenID Connect Identity Providers \(p. 295\)](#).
2. When prompted for your Provider URL, enter "<https://accounts.google.com>".
3. When prompted to enter a value in the **Audience** field, enter any one of the three client IDs that you created in the previous steps.
4. Choose the provider name and add two more audiences with the two other client IDs.

Configure the External Provider in the Amazon Cognito Console

Choose **Manage Identity Pools** from the [Amazon Cognito Console home page](#):

1. Choose the name of the identity pool where you want to enable Google as an external provider. The **Dashboard** page for your identity pool appears.
2. In the top-right corner of the **Dashboard** page, choose **Edit identity pool**. The Edit identity pool page appears.
3. Scroll down and choose **Authentication providers** to expand the section.
4. Choose the **Google** tab.
5. Choose **Unlock**.

6. Enter the Google Client ID you obtained from Google, and then choose **Save Changes**.

Install the Unity Google Plugin

1. Add the [Google Play Games plugin for Unity](#) to your Unity project.
2. In Unity, from the **Windows** menu, use the three IDs for the Android and iOS platforms to configure the plugin.

Use Google

The following example code shows how to retrieve the authentication token from the Google Play service:

```
void Start()
{
    PlayGamesClientConfiguration config = new PlayGamesClientConfiguration.Builder().Build();
    PlayGamesPlatform.InitializeInstance(config);
    PlayGamesPlatform.DebugLogEnabled = true;
    PlayGamesPlatform.Activate();
    Social.localUser.Authenticate(GoogleLoginCallback);
}

void GoogleLoginCallback(bool success)
{
    if (success)
    {
        string token = PlayGamesPlatform.Instance.GetIdToken();
        credentials.AddLogin("accounts.google.com", token);
    }
    else
    {
        Debug.LogError("Google login failed. If you are not running in an actual Android/iOS device, this is expected.");
    }
}
```

Xamarin

Note

Amazon Cognito doesn't natively support Google on the Xamarin platform. Integration currently requires the use of a web view to go through the browser sign-in flow. To learn how Google integration works with other SDKs, please select another platform.

To enable login with Google in your application, authenticate your users and obtain an OpenID Connect token from them. Amazon Cognito uses this token to generate a unique user identifier that is associated with an Amazon Cognito identity. Unfortunately, the Google SDK for Xamarin doesn't allow you to retrieve the OpenID Connect token, so use an alternative client or the web flow in a web view.

After you have the token, you can set it in your `CognitoAWSCredentials`:

```
credentials.AddLogin("accounts.google.com", token);
```

Note

If your app uses Google and is available on multiple mobile platforms, you should configure Google as an [OpenID Connect Provider \(p. 295\)](#). Add all created client IDs as additional audience values for better integration. To learn more about Google's cross-client identity model, see [Cross-client Identity](#).

Sign in with Apple (identity pools)

Amazon Cognito integrates with Sign in with Apple to provide federated authentication for your mobile application and web application users. This section explains how to register and set up your application using Sign in with Apple as an identity provider (IdP).

To add Sign in with Apple as an authentication provider to an identity pool, you must complete two procedures. First, integrate Sign in with Apple in an application, and then configure Sign in with Apple in identity pools.

Set up Sign in with Apple

To configure Sign in with Apple as an IdP, register your application with the Apple to receive client ID.

1. Create a [developer account with Apple](#).
2. [Sign in](#) with your Apple credentials.
3. In the left navigation pane, choose **Certificates, IDs & Profiles**.
4. In the left navigation pane, choose **Identifiers**.
5. On the **Identifiers** page, choose the + icon.
6. On the **Register a New Identifier** page, choose **App IDs**, and then choose **Continue**.
7. On the **Register an App ID** page, do the following:
 - a. Under **Description**, type a description.
 - b. Under **Bundle ID**, type an identifier. Make a note of this **Bundle ID** as you need this value to configure Apple as a provider in the identity pool.
 - c. Under **Capabilities**, choose **Sign In with Apple**, and then choose **Edit**.
 - d. On the **Sign in with Apple: App ID Configuration** page, select the appropriate setting for your app. Then choose **Save**.
 - e. Choose **Continue**.
8. On the **Confirm your App ID** page, choose **Register**.
9. Proceed to step 10 if you want to integrate Sign in with Apple with a native iOS application. Step 11 is for applications that you want to integrate with Sign in with Apple JS.
10. On the **Identifiers** page, choose the **App IDs** menu, then **Services IDs**. Choose the + icon.
11. On the **Register a New Identifier** page, choose **Services IDs**, and then choose **Continue**.
12. On the **Register a Services ID** page, do the following:
 - a. Under **Description**, type a description.
 - b. Under **Identifier**, type an identifier. Make a note of the services ID as you need this value to configure Apple as a provider in your identity pool.
 - c. Select **Sign In with Apple** and then choose **Configure**.
 - d. On the **Web Authentication Configuration** page, choose a **Primary App ID**. Under **Website URLs**, choose the + icon. For **Domains and Subdomains**, enter the domain name of your app. In **Return URLs**, enter the callback URL where the authorization redirects the user after they authenticate through Sign in with Apple.
 - e. Choose **Next**.
 - f. Choose **Continue**, and then choose **Register**.
13. In the left navigation pane, choose **Keys**.
14. On the **Keys** page, choose the + icon.
15. On the **Register a New Key** page, do the following:
 - a. Under **Key Name**, type a key name.

- b. Choose **Sign In with Apple**, and then choose **Configure**.
- c. On the **Configure Key** page, choose a **Primary App ID** and then choose **Save**.
- d. Choose **Continue**, and then choose **Register**.

Note

To integrate Sign in with Apple with a native iOS application, see [Implementing User Authentication with Sign in with Apple](#).

To integrate Sign in with Apple in a platform other than native iOS, see [Sign in with Apple JS](#).

Configure the external provider in the Amazon Cognito federated identities console

Use the following procedure to configure your external provider.

1. Choose **Manage Identity Pools** from the [Amazon Cognito console home page](#).
2. Choose the name of the identity pool where you want to enable Apple as an external provider.
3. In the top-right corner of the dashboard, choose **Edit identity pool**.
4. Scroll down and choose **Authentication providers** to expand the section.
5. Choose the **Apple** tab.
6. Enter the Bundle ID that you obtained from <https://developer.apple.com>. Then choose **Save Changes**.
7. If you use Sign in with Apple with native iOS applications, enter the **Bundle ID** you obtained from developer.apple.com. If you use Sign in with Apple with web or other applications, enter the service ID. Then choose **Save Changes**.

Sign in with Apple as a provider in the Amazon Cognito federated identities CLI examples

This example creates an identity pool named `MyIdentityPool` with Sign in with Apple as an IdP.

```
aws cognito-identity create-identity-pool --identity-pool-name MyIdentityPool  
--supported-login-providers appleid.apple.com="sameple.apple.clientid"
```

For more information, see [Create identity pool](#)

Generate an Amazon Cognito identity ID

This example generates (or retrieves) an Amazon Cognito ID. This is a public API so you don't need any credentials to call this API.

```
aws cognito-identity get-id --identity-pool-id SampleIdentityPoolId --logins  
appleid.apple.com="SignInWithAppleIdToken"
```

For more information, see [get-id](#).

Get credentials for an Amazon Cognito identity ID

This example returns credentials for the provided identity ID and Sign in with Apple login. This is a public API so you don't need any credentials to call this API.

```
aws cognito-identity get-credentials-for-identity --identity-id  
SampleIdentityId --logins appleid.apple.com="SignInWithAppleIdToken"
```

For more information, see [get-credentials-for-identity](#)

Use Sign in with Apple: Android

Apple doesn't provide an SDK that supports Sign in with Apple for Android. You can use the web flow in a web view instead.

- To configure Sign in with Apple in your application, follow [Configuring Your Web page for Sign In with Apple](#) in the Apple documentation.
- To add a **Sign in with Apple** button to your Android user interface, follow [Displaying and Configuring Sign In with Apple Buttons](#) in the Apple documentation.
- To securely authenticate users with Sign in with Apple, follow [Authenticating Users with Sign in with Apple](#) in the Apple documentation.

Sign in with Apple uses a session object to track its state. Amazon Cognito uses the ID token from this session object to authenticate the user, generate the unique identifier, and, if needed, grant the user access to other AWS resources.

```
@Override
public void onSuccess(Bundle response) {
    String token = response.getString("id_token");
    Map<String, String> logins = new HashMap<String, String>();
    logins.put("appleid.apple.com", token);
    credentialsProvider.setLogins(logins);
}
```

Use Sign in with Apple: iOS - Objective-C

Apple provided SDK support for Sign in with Apple in native iOS applications. To implement user authentication with Sign in with Apple in native iOS devices, follow [Implementing User Authentication with Sign in with Apple](#) in the Apple documentation.

Amazon Cognito uses the ID token to authenticate the user, generate the unique identifier, and, if needed, grant the user access to other AWS resources.

```
(void)finishedWithAuth: (ASAAuthorizationAppleIDCredential *)auth error: (NSError *) error {
    NSString *idToken = [ASAAuthorizationAppleIDCredential
        objectForKey:@"identityToken"];
    credentialsProvider.logins = @{@"appleid.apple.com": idToken};
}
```

Use Sign in with Apple: iOS - Swift

Apple provided SDK support for Sign in with Apple in native iOS applications. To implement user authentication with Sign in with Apple in native iOS devices, follow [Implementing User Authentication with Sign in with Apple](#) in the Apple documentation.

Amazon Cognito uses the ID token to authenticate the user, generate the unique identifier, and, if needed, grant the user access to other AWS resources.

For more information about how to set up Sign in with Apple in iOS, see [Set up Sign in with Apple](#)

```
func finishedWithAuth(auth: ASAAuthorizationAppleIDCredential!, error: NSError!) {
    if error != nil {
        print(error.localizedDescription)
```

```

    }
    else {
        let idToken = auth.identityToken,
            credentialsProvider.logins = ["appleid.apple.com": idToken!]
    }
}

```

Use Sign in with Apple: JavaScript

Apple doesn't provide an SDK that supports Sign in with Apple for JavaScript. You can use the web flow in a web view instead.

- To configure Sign in with Apple in your application, follow [Configuring Your Web page for Sign In with Apple](#) in the Apple documentation.
- To add a **Sign in with Apple** button to your JavaScript user interface, follow [Displaying and Configuring Sign In with Apple Buttons](#) in the Apple documentation.
- To securely authenticate users through Sign in with Apple, follow [Configuring Your Web page for Sign In with Apple](#) in the Apple documentation.

Sign in with Apple uses a session object to track its state. Amazon Cognito uses the ID token from this session object to authenticate the user, generate the unique identifier, and, if needed, grant the user access to other AWS resources.

```

function signinCallback(authResult) {
    // Add the apple's id token to the Amazon Cognito credentials login map.
    AWS.config.credentials = new AWS.CognitoIdentityCredentials({
        IdentityPoolId: 'IDENTITY_POOL_ID',
        Logins: {
            'appleid.apple.com': authResult['id_token']
        }
    });

    // Obtain AWS credentials
    AWS.config.credentials.get(function(){
        // Access AWS resources here.
    });
}

```

Use Sign in with Apple: Xamarin

We don't have an SDK that supports Sign in with Apple for Xamarin. You can use the web flow in a web view instead.

- To configure Sign in with Apple in your application, follow [Configuring Your Web page for Sign In with Apple](#) in the Apple documentation.
- To add a **Sign in with Apple** button to your Xamarin user interface, follow [Displaying and Configuring Sign In with Apple Buttons](#) in the Apple documentation.
- To securely authenticate users through Sign in with Apple, follow [Configuring Your Web page for Sign In with Apple](#) in the Apple documentation.

Sign in with Apple uses a session object to track its state. Amazon Cognito uses the ID token from this session object to authenticate the user, generate the unique identifier, and, if needed, grant the user access to other AWS resources.

After you have the token, you can set it in your `CognitoAWSCredentials`:

```
credentials.AddLogin("appleid.apple.com", token);
```

Open ID Connect providers (identity pools)

[OpenID Connect](#) is an open standard for authentication that a number of login providers support. Amazon Cognito supports you to link identities with OpenID Connect providers that you configure through [AWS Identity and Access Management](#).

Adding an OpenID Connect provider

For information about how to create an OpenID Connect provider, see the [IAM documentation](#).

Associating a provider with Amazon Cognito

After you create an OpenID Connect provider in the IAM Console, you can associate it with an identity pool. All providers that you configure are visible in the Edit Identity Pool screen in the Amazon Cognito Console under the OpenID Connect Providers header.

▼ OpenID Connect providers ⓘ

Amazon Cognito can authenticate users through any OpenID Connect provider. Once a provider has been configured with IAM, you can select the provider from the list below. [Learn more about using OpenID Connect providers](#).

accounts.google.com

login.salesforce.com

You can associate multiple OpenID Connect providers with a single identity pool.

Using OpenID Connect

Refer to your provider's documentation for how to login and receive an ID token.

After you have a token, add the token to the logins map. Use the URI of your provider as the key.

Validating an OpenID Connect token

When you first integrate with Amazon Cognito, you might receive an `InvalidToken` exception. It is important to understand how Amazon Cognito validates OpenID Connect (OIDC) tokens.

Note

As specified here (<https://tools.ietf.org/html/rfc7523>), Amazon Cognito provides a grace period of 5 minutes to handle any clock skew between systems.

1. The `iss` parameter must match the key that the logins map uses (such as `login.provider.com`).
2. The signature must be valid. The signature must be verifiable via an RSA public key.
3. The fingerprint of the certificate public key matches the fingerprint that you set in IAM when you created your OIDC provider.
4. If the `azp` parameter is present, check this value against listed client IDs in your OIDC provider.
5. If the `azp` parameter isn't present, check the `aud` parameter against listed client IDs in your OIDC provider.

The website jwt.io is a valuable resource that you can use to decode tokens and verify these values.

Android

```
Map<String, String> logins = new HashMap<String, String>();
logins.put("login.provider.com", token);
credentialsProvider.setLogins(logins);
```

iOS - Objective-C

```
credentialsProvider.logins = @{@"login.provider.com": token}
```

iOS - Swift

To provide the OIDC ID token to Amazon Cognito, implement the `AWSIdentityProviderManager` protocol.

When you implement the `logins` method, return a dictionary that contains the OIDC provider name that you configured. This dictionary acts as the key, and the current ID token from the authenticated user acts as the value, as shown in the following code example.

```
class OIDCProvider: NSObject, AWSIdentityProviderManager {
    func logins() -> AWSTask<NSDictionary> {
        let completion = AWSTaskCompletionSource<NSString>()
        getToken(tokenCompletion: completion)
        return completion.task.continueOnSuccessWith { (task) -> AWSTask<NSDictionary>? in
            //login.provider.name is the name of the OIDC provider as setup in the Amazon
            Cognito console
            return AWSTask(result:[ "login.provider.name":task.result!])
        } as! AWSTask<NSDictionary>
    }

    func getToken(tokenCompletion: AWSTaskCompletionSource<NSString>) -> Void {
        //get a valid oidc token from your server, or if you have one that hasn't expired
        cached, return it

        //TODO code to get token from your server
        //...

        //if error getting token, set error appropriately
        tokenCompletion.set(error:NSError(domain: "OIDC Login", code: -1 , userInfo:
        ["Unable to get OIDC token" : "Details about your error"]))
        //else
        tokenCompletion.set(result:"result from server id token")
    }
}
```

When you instantiate the `AWSCognitoCredentialsProvider`, pass the class that implements `AWSIdentityProviderManager` as the value of `identityProviderManager` in the constructor.

For more information, go to the [AWSCognitoCredentialsProvider](#) reference page and choose `initWithRegionType:identityPoolId:identityProviderManager`.

JavaScript

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: 'IDENTITY_POOL_ID',
    Logins: {
        'login.provider.com': token
    }
})
```

```
});
```

Unity

```
credentials.AddLogin("login.provider.com", token);
```

Xamarin

```
credentials.AddLogin("login.provider.com", token);
```

SAML identity providers (identity pools)

Amazon Cognito supports authentication with identity providers (IdPs) through Security Assertion Markup Language 2.0 (SAML 2.0). You can use an IdP that supports SAML with Amazon Cognito to provide a simple onboarding flow for your users. Your SAML-supporting IdP specifies the IAM roles that your users can assume. This way, different users can receive different sets of permissions.

Configuring your identity pool for a SAML IdP

The following steps describe how to configure your identity pool to use a SAML-based IdP.

Note

Before you configure your identity pool to support a SAML provider, first configure the SAML IdP in the [IAM console](#). For more information, see [Integrating third-party SAML solution providers with AWS](#) in the *IAM User Guide*.

Configuring your identity pool to support a SAML provider

1. Sign in to the [Amazon Cognito console](#), choose **Manage Identity Pools**, and choose **Create new identity pool**.
2. In the **Authentication providers** section, choose the **SAML** tab.
3. Choose the ARN of the SAML provider, and then choose **Create Pool**.

Configuring your SAML IdP

After you create the SAML provider, configure your SAML IdP to add relying party trust between your IdP and AWS. With many IdPs, you can specify a URL that the IdP can use to read relying party information and certificates from an XML document. For AWS, you can use <https://signin.aws.amazon.com/static/saml-metadata.xml>. The next step is to configure the SAML assertion response from your IdP to populate the claims that AWS needs. For details on the claim configuration, see [Configuring SAML assertions for authentication response](#).

Customizing your user role with SAML

When you use SAML with Amazon Cognito Identity, you can customize the role for the end user. Amazon Cognito only supports the [enhanced flow \(p. 249\)](#) with the SAML-based IdP. You don't need to specify an authenticated or unauthenticated role for the identity pool to use a SAML-based IdP. The `https://aws.amazon.com/SAML/Attributes/Role` claim attribute specifies one or more pairs of comma-delimited role and provider ARN. These are the roles that the user can assume. You can configure the SAML IdP to populate the role attributes based on the user attribute information available from the IdP.

If you receive multiple roles in the SAML assertion, populate the optional `customRoleArn` parameter when you call `getCredentialsForIdentity`. The user assumes this `customRoleArn` if the role matches one in the claim in the SAML assertion.

Authenticating users with a SAML IdP

To federate with the SAML-based IdP, determine the URL where the user initiates the login. AWS federation uses IdP-initiated login. In AD FS 2.0 the URL takes the form of `https://<fqdn>/adfs/ls/IdpInitiatedSignOn.aspx?loginToRp=urn:amazon:webservices`.

To add support for your SAML IdP in Amazon Cognito, first authenticate users with your SAML identity provider from your iOS or Android application. The code that you use to integrate and authenticate with the SAML IdP is specific to SAML providers. After you authenticate your user, you can use Amazon Cognito APIs to provide the resulting SAML assertion to Amazon Cognito Identity .

Android

If you use the Android SDK, you can populate the logins map with the SAML assertion as follows.

```
Map logins = new HashMap();
logins.put("arn:aws:iam::aws account id:saml-provider/name", "base64 encoded assertion
response");
// Now this should be set to CognitoCachingCredentialsProvider object.
CognitoCachingCredentialsProvider credentialsProvider = new
    CognitoCachingCredentialsProvider(context, identity pool id, region);
credentialsProvider.setLogins(logins);
// If SAML assertion contains multiple roles, resolve the role by setting the custom role
credentialsProvider.setCustomRoleArn("arn:aws:iam::aws account id:role/customRoleName");
// This should trigger a call to the Amazon Cognito service to get the credentials.
credentialsProvider.getCredentials();
```

iOS

If you are using the iOS SDK, you can provide the SAML assertion in `AWSIdentityProviderManager` as follows.

```
- (AWSTask<NSDictionary<NSString*, NSString*> *> *) logins {
    //this is hardcoded for simplicity, normally you would asynchronously go to your SAML
    provider
    //get the assertion and return the logins map using a AWSTaskCompletionSource
    return [AWSTask taskWithResult:@{@"arn:aws:iam::aws account id:saml-provider/
name":@"base64 encoded assertion response"}];
}

// If SAML assertion contains multiple roles, resolve the role by setting the custom role.
// Implementing this is optional if there is only one role.
- (NSString *)customRoleArn {
    return @"arn:aws:iam::accountId:role/customRoleName";
}
```

Developer authenticated identities (identity pools)

Amazon Cognito supports developer authenticated identities, in addition to web identity federation through [Facebook \(identity pools\) \(p. 276\)](#), [Google \(identity pools\) \(p. 284\)](#), [Login with Amazon \(identity pools\) \(p. 281\)](#), and [Sign in with Apple \(identity pools\) \(p. 291\)](#). With developer

authenticated identities, you can register and authenticate users via your own existing authentication process, while still using Amazon Cognito to synchronize user data and access AWS resources. Using developer authenticated identities involves interaction between the end user device, your backend for authentication, and Amazon Cognito. For more details, see [Understanding Amazon Cognito Authentication Part 2: Developer Authenticated Identities](#) in the AWS blog.

Understanding the authentication flow

For information on the developer authenticated identities authflow and how it differs from the external provider authflow, see [Identity pools \(federated identities\) authentication flow \(p. 249\)](#).

Define a developer provider name and associate it with an identity pool

To use developer authenticated identities, you'll need an identity pool associated with your developer provider. To do so, follow these steps:

1. Log in to the [Amazon Cognito console](#).
2. Create a new identity pool and, as part of the process, define a developer provider name in the **Custom** tab in **Authentication Providers**.
3. Alternatively, edit an existing identity pool and define a developer provider name in the **Custom** tab in **Authentication Providers**.

Note: Once the provider name has been set, it cannot be changed.

For additional instructions on working with the Amazon Cognito Console, see [Using the Amazon Cognito console \(p. 3\)](#).

Implement an identity provider

Android

To use developer authenticated identities, implement your own identity provider class which extends `AWSAbstractCognitoIdentityProvider`. Your identity provider class should return a response object containing the token as an attribute.

Below is a simple example of an identity provider.

```
public class DeveloperAuthenticationProvider extends
    AWSAbstractCognitoDeveloperIdentityProvider {

    private static final String developerProvider = "<Developer_provider_name>";

    public DeveloperAuthenticationProvider(String accountId, String identityPoolId, Regions
        region) {
        super(accountId, identityPoolId, region);
        // Initialize any other objects needed here.
    }

    // Return the developer provider name which you choose while setting up the
    // identity pool in the &COG; Console

    @Override
    public String getProviderName() {
```

```

        return developerProvider;
    }

    // Use the refresh method to communicate with your backend to get an
    // identityId and token.

    @Override
    public String refresh() {

        // Override the existing token
        setToken(null);

        // Get the identityId and token by making a call to your backend
        // (Call to your backend)

        // Call the update method with updated identityId and token to make sure
        // these are ready to be used from Credentials Provider.

        update(identityId, token);
        return token;
    }

    // If the app has a valid identityId return it, otherwise get a valid
    // identityId from your backend.

    @Override
    public String getIdentityId() {

        // Load the identityId from the cache
        identityId = cachedIdentityId;

        if (identityId == null) {
            // Call to your backend
        } else {
            return identityId;
        }
    }
}

```

To use this identity provider, you have to pass it into `CognitoCachingCredentialsProvider`. Here's an example:

```

DeveloperAuthenticationProvider developerProvider = new
    DeveloperAuthenticationProvider( null, "IDENTITYPOOLID", context, Regions.US_EAST_1 );
CognitoCachingCredentialsProvider credentialsProvider = new
    CognitoCachingCredentialsProvider( context, developerProvider, Regions.US_EAST_1 );

```

iOS - objective-C

To use developer authenticated identities, implement your own identity provider class which extends [AWS Cognito Credentials Provider Helper](#). Your identity provider class should return a response object containing the token as an attribute.

```

@implementation DeveloperAuthenticatedIdentityProvider
/*
 * Use the token method to communicate with your backend to get an
 * identityId and token.
 */
- (AWSTask<NSString*> *) token {

```

```

//Write code to call your backend:
//Pass username/password to backend or some sort of token to authenticate user
//If successful, from backend call getOpenIdTokenForDeveloperIdentity with logins map
//containing "your.provider.name":"enduser.username"
//Return the identity id and token to client
//You can use AWSTaskCompletionSource to do this asynchronously

// Set the identity id and return the token
self.identityId = response.identityId;
return [AWSTask taskWithResult:response.token];
}

@end

```

To use this identity provider, pass it into `AWSCognitoCredentialsProvider` as shown in the following example:

```

DeveloperAuthenticatedIdentityProvider * devAuth = [[DeveloperAuthenticatedIdentityProvider
alloc] initWithRegionType:AWSRegionYOUR_IDENTITY_POOL_REGION
                           identityPoolId:@"YOUR_IDENTITY_POOL_ID"
                           useEnhancedFlow:YES
                           identityProviderManager:nil];
AWSCognitoCredentialsProvider *credentialsProvider = [[AWSCognitoCredentialsProvider alloc]
initWithRegionType:AWSRegionYOUR_IDENTITY_POOL_REGION
                           identityProvider:devAuth];

```

If you want to support both unauthenticated identities and developer authenticated identities, override the `logins` method in your `AWSCognitoCredentialsProviderHelper` implementation.

```

- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    if(/*logic to determine if user is unauthenticated*/) {
        return [AWSTask taskWithResult:nil];
    }else{
        return [super logins];
    }
}

```

If you want to support developer authenticated identities and social providers, you must manage who the current provider is in your `logins` implementation of `AWSCognitoCredentialsProviderHelper`.

```

- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    if(/*logic to determine if user is unauthenticated*/) {
        return [AWSTask taskWithResult:nil];
    }else if (/*logic to determine if user is Facebook*/){
        return [AWSTask taskWithResult: @{@"AWSIdentityProviderFacebook : [FBSDKAccessToken
currentAccessToken] }];
    }else {
        return [super logins];
    }
}

```

iOS - swift

To use developer authenticated identities, implement your own identity provider class which extends `AWSCognitoCredentialsProviderHelper`. Your identity provider class should return a response object containing the token as an attribute.

```
import AWSCore
```

```

/*
 * Use the token method to communicate with your backend to get an
 * identityId and token.
 */
class DeveloperAuthenticatedIdentityProvider : AWSIdentityProviderHelper {
    override func token() -> AWSTask<NSString> {
        //Write code to call your backend:
        //pass username/password to backend or some sort of token to authenticate user, if
        successful,
        //from backend call getOpenIdTokenForDeveloperIdentity with logins map containing
        "your.provider.name": "enduser.username"
        //return the identity id and token to client
        //You can use AWSTaskCompletionSource to do this asynchronously

        // Set the identity id and return the token
        self.identityId = resultFromAbove.identityId
        return AWSTask(result: resultFromAbove.token)
    }
}

```

To use this identity provider, pass it into `AWSIdentityProvider` as shown in the following example:

```

let devAuth =
    DeveloperAuthenticatedIdentityProvider(regionType: .YOUR_IDENTITY_POOL_REGION,
    identityPoolId: "YOUR_IDENTITY_POOL_ID", useEnhancedFlow: true,
    identityProviderManager:nil)
let credentialsProvider =
    AWSIdentityProvider(regionType: .YOUR_IDENTITY_POOL_REGION,
    identityProvider:devAuth)
let configuration = AWSServiceConfiguration(region: .YOUR_IDENTITY_POOL_REGION,
    credentialsProvider:credentialsProvider)
AWSServiceManager.default().defaultServiceConfiguration = configuration

```

If you want to support both unauthenticated identities and developer authenticated identities, override the `logins` method in your `AWSIdentityProviderHelper` implementation.

```

override func logins () -> AWSTask<NSDictionary> {
    if(/*logic to determine if user is unauthenticated*/) {
        return AWSTask(result:nil)
    }else {
        return super.logins()
    }
}

```

If you want to support developer authenticated identities and social providers, you must manage who the current provider is in your `logins` implementation of `AWSIdentityProviderHelper`.

```

override func logins () -> AWSTask<NSDictionary> {
    if(/*logic to determine if user is unauthenticated*/) {
        return AWSTask(result:nil)
    }else if /*logic to determine if user is Facebook*/{
        if let token = AccessToken.current?.authenticationToken {
            return AWSTask(result: [AWSIdentityProviderFacebook:token])
        }
        return AWSTask(error:NSError(domain: "Facebook Login", code: -1 , userInfo: [
        "Facebook" : "No current Facebook access token"]))
    }else {
        return super.logins()
    }
}

```

JavaScript

Once you obtain an identity ID and session token from your backend, you will pass them into the `AWS.CognitoIdentityCredentials` provider. Here's an example:

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: 'IDENTITY_POOL_ID',
    IdentityId: 'IDENTITY_ID_RETURNED_FROM_YOUR_PROVIDER',
    Logins: {
        'cognito-identity.amazonaws.com': 'TOKEN_RETURNED_FROM_YOUR_PROVIDER'
    }
});
```

Unity

To use developer-authenticated identities you have to extend `CognitoAWSCredentials` and override the `RefreshIdentity` method to retrieve the user identity id and token from your backend and return them. Below is a simple example of an identity provider that would contact a hypothetical backend at 'example.com':

```
using UnityEngine;
using System.Collections;
using Amazon.CognitoIdentity;
using System.Collections.Generic;
using ThirdParty.Json.LitJson;
using System;
using System.Threading;

public class DeveloperAuthenticatedCredentials : CognitoAWSCredentials
{
    const string PROVIDER_NAME = "example.com";
    const string IDENTITY_POOL = "IDENTITY_POOL_ID";
    static readonly RegionEndpoint REGION = RegionEndpoint.USEast1;

    private string login = null;

    public DeveloperAuthenticatedCredentials(string loginAlias)
        : base(IDENTITY_POOL, REGION)
    {
        login = loginAlias;
    }

    protected override IdentityState RefreshIdentity()
    {
        IdentityState state = null;
        ManualResetEvent waitLock = new ManualResetEvent(false);
        MainThreadDispatcher.ExecuteCoroutineOnMainThread(ContactProvider((s) =>
        {
            state = s;
            waitLock.Set();
        }));
        waitLock.WaitOne();
        return state;
    }

    IEnumerator ContactProvider(Action<IdentityState> callback)
    {
        WWW www = new WWW("http://example.com/?username=" + login);
        yield return www;
        string response = www.text;

        JsonData json = JsonMapper.ToObject(response);
```

```

    //The backend has to send us back an Identity and a OpenID token
    string identityId = json["IdentityId"].ToString();
    string token = json["Token"].ToString();

    IdentityState state = new IdentityState(identityId, PROVIDER_NAME, token, false);
    callback(state);
}
}

```

The code above uses a thread dispatcher object to call a coroutine. If you don't have a way to do this in your project, you can use the following script in your scenes:

```

using System;
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class MainThreadDispatcher : MonoBehaviour
{
    static Queue<IEnumerator> _coroutineQueue = new Queue<IEnumerator>();
    static object _lock = new object();

    public void Update()
    {
        while (_coroutineQueue.Count > 0)
        {
            StartCoroutine(_coroutineQueue.Dequeue());
        }
    }

    public static void ExecuteCoroutineOnMainThread(IEnumerator coroutine)
    {
        lock (_lock) {
            _coroutineQueue.Enqueue(coroutine);
        }
    }
}

```

Xamarin

To use developer-authenticated identities you have to extend `CognitoAWSCredentials` and override the `RefreshIdentity` method to retrieve the user identity id and token from your backend and return them. Below is a simple example of an identity provider that would contact a hypothetical backend at 'example.com':

```

public class DeveloperAuthenticatedCredentials : CognitoAWSCredentials
{
    const string PROVIDER_NAME = "example.com";
    const string IDENTITY_POOL = "IDENTITY_POOL_ID";
    static readonly RegionEndpoint REGION = RegionEndpoint.USEast1;
    private string login = null;

    public DeveloperAuthenticatedCredentials(string loginAlias)
        : base(IDENTITY_POOL, REGION)
    {
        login = loginAlias;
    }

    protected override async Task<IdentityState> RefreshIdentityAsync()
    {
        IdentityState state = null;
    }
}

```

```
//get your identity and set the state  
return state;  
}  
}
```

Updating the logins map (Android and iOS only)

Android

After successfully authenticating the user with your authentication system, update the logins map with the developer provider name and a developer user identifier, which is an alphanumeric string that uniquely identifies a user in your authentication system. Be sure to call the `refresh` method after updating the logins map as the `identityId` might have changed:

```
HashMap<String, String> loginsMap = new HashMap<String, String>();  
loginsMap.put(developerAuthenticationProvider.getProviderName(), developerUserIdentifer);  
  
credentialsProvider.setLogins(loginsMap);  
credentialsProvider.refresh();
```

iOS - objective-C

The iOS SDK only calls your `logins` method to get the latest logins map if there are no credentials or they have expired. If you want to force the SDK to obtain new credentials (e.g., your end user went from unauthenticated to authenticated and you want credentials against the authenticated user), call `clearCredentials` on your `credentialsProvider`.

```
[credentialsProvider clearCredentials];
```

iOS - swift

The iOS SDK only calls your `logins` method to get the latest logins map if there are no credentials or they have expired. If you want to force the SDK to obtain new credentials (e.g., your end user went from unauthenticated to authenticated and you want credentials against the authenticated user), call `clearCredentials` on your `credentialsProvider`.

```
credentialsProvider.clearCredentials()
```

Getting a token (server side)

You obtain a token by calling [GetOpenIdTokenForDeveloperIdentity](#). This API must be invoked from your backend using AWS developer credentials. It must not be invoked from the client SDK. The API receives the Cognito identity pool ID; a logins map containing your identity provider name as the key and identifier as the value; and optionally a Cognito identity ID (i.e., you are making an unauthenticated user authenticated). The identifier can be the username of your user, an email address, or a numerical value. The API responds to your call with a unique Cognito ID for your user and an OpenID Connect token for the end user.

A few things to keep in mind about the token returned by `GetOpenIdTokenForDeveloperIdentity`:

- You can specify a custom expiration time for the token so you can cache it. If you don't provide any custom expiration time, the token is valid for 15 minutes.
- The maximum token duration you can set is 24 hours.

- Be mindful of the security implications of increasing the token duration. If an attacker obtains this token, they can exchange it for AWS credentials for the end user for the token duration.

The following Java snippet shows how to initialize an Amazon Cognito client and retrieve a token for a developer authenticated identity.

```
// authenticate your end user as appropriate
// ....

// if authenticated, initialize a cognito client with your AWS developer credentials
AmazonCognitoIdentity identityClient = new AmazonCognitoIdentityClient(
    new BasicAWSCredentials("access_key_id", "secret_access_key")
);

// create a new request to retrieve the token for your end user
GetOpenIdTokenForDeveloperIdentityRequest request =
    new GetOpenIdTokenForDeveloperIdentityRequest();
request.setIdentityPoolId("YOUR_COGNITO_IDENTITY_POOL_ID");

request.setIdentityId("YOUR_COGNITO_IDENTITY_ID"); //optional, set this if your client has
an
                                                //identity ID that you want to link to
this
                                                //developer account

// set up your logins map with the username of your end user
HashMap<String, String> logins = new HashMap<>();
logins.put("YOUR_IDENTITY_PROVIDER_NAME", "YOUR_END_USER_IDENTIFIER");
request.setLogins(logins);

// optionally set token duration (in seconds)
request.setTokenDuration(60 * 15l);
GetOpenIdTokenForDeveloperIdentityResult response =
    identityClient.getOpenIdTokenForDeveloperIdentity(request);

// obtain identity id and token to return to your client
String identityId = response.getIdentityId();
String token = response.getToken();

//code to return identity id and token to client
//...
```

Following the steps above, you should be able to integrate developer authenticated identities in your app. If you have any issues or questions please feel free to post in our [forums](#).

Connect to an existing social identity

All linking of providers when you are using developer authenticated identities must be done from your backend. To connect a custom identity to a user's social identity (Login with Amazon, Sign in with Apple, Facebook, or Google), add the identity provider token to the logins map when you call [GetOpenIdTokenForDeveloperIdentity](#). To make this possible, when you call your backend from your client SDK to authenticate your end user, additionally pass the end user's social provider token.

For example, if you are trying to link a custom identity to Facebook, you would add the Facebook token in addition to your identity provider identifier to the logins map when you call [GetOpenIdTokenForDeveloperIdentity](#).

```
logins.put("YOUR_IDENTITY_PROVIDER_NAME", "YOUR_END_USER_IDENTIFIER");
logins.put("graph.facebook.com", "END_USERS_FACEBOOK_ACESSTOKEN");
```

Supporting transition between providers

Android

Your application might require supporting unauthenticated identities or authenticated identities using public providers (Login with Amazon, Sign in with Apple, Facebook, or Google) along with developer authenticated identities. The essential difference between developer authenticated identities and other identities (unauthenticated identities and authenticated identities using public provider) is the way the identityId and token are obtained. For other identities the mobile application will interact directly with Amazon Cognito instead of contacting your authentication system. So the mobile application should be able to support two distinct flows depending on the choice made by the app user. For this you will have to make some changes to the custom identity provider.

The `refresh` method should check the `logins` map, if the map is not empty and has a key with developer provider name, then you should call your backend; otherwise just call the `getIdentityId` method and return null.

```
public String refresh() {  
  
    setToken(null);  
  
    // If the logins map is not empty make a call to your backend  
    // to get the token and identityId  
    if (getProviderName() != null &&  
        !this.loginsMap.isEmpty() &&  
        this.loginsMap.containsKey(getProviderName())) {  
  
        /**  
         * This is where you would call your backend  
         **/  
  
        // now set the returned identity id and token in the provider  
        update(identityId, token);  
        return token;  
  
    } else {  
        // Call getIdentityId method and return null  
        this.getIdentityId();  
        return null;  
    }  
}
```

Similarly the `getIdentityId` method will have two flows depending on the contents of the `logins` map:

```
public String getIdentityId() {  
  
    // Load the identityId from the cache  
    identityId = cachedIdentityId;  
  
    if (identityId == null) {  
  
        // If the logins map is not empty make a call to your backend  
        // to get the token and identityId  
  
        if (getProviderName() != null && !this.loginsMap.isEmpty()  
            && this.loginsMap.containsKey(getProviderName())) {  
  
            /**  
             * This is where you would call your backend  
             **/  
        }  
    }  
}
```

```

        // now set the returned identity id and token in the provider
        update(identityId, token);
        return token;

    } else {
        // Otherwise call &COG; using getIdentityId of super class
        return super.getIdentityId();
    }

} else {
    return identityId;
}

}

```

iOS - objective-C

Your application might require supporting unauthenticated identities or authenticated identities using public providers (Login with Amazon, Sign in with Apple, Facebook, or Google) along with developer authenticated identities. To do this, override the [AWSIdentityProviderHelper](#) `logins` method to be able to return the correct logins map based on the current identity provider. This example shows you how you might pivot between unauthenticated, Facebook and developer authenticated.

```

- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    if(/*logic to determine if user is unauthenticated*/) {
        return [AWSTask taskWithResult:nil];
    }else if (/*logic to determine if user is Facebook*/){
        return [AWSTask taskWithResult: @{@"AWSIdentityProviderFacebook" : [FBSDKAccessToken currentAccessToken]}];
    }else {
        return [super logins];
    }
}

```

When you transition from unauthenticated to authenticated, you should call `[credentialsProvider clearCredentials]`; to force the SDK to get new authenticated credentials. When you switch between two authenticated providers and you aren't trying to link the two providers (i.e. you are not providing tokens for multiple providers in your logins dictionary), you should call `[credentialsProvider clearKeychain]`. This will clear both the credentials and identity and force the SDK to get new ones.

iOS - swift

Your application might require supporting unauthenticated identities or authenticated identities using public providers (Login with Amazon, Sign in with Apple, Facebook, or Google) along with developer authenticated identities. To do this, override the [AWSIdentityProviderHelper](#) `logins` method to be able to return the correct logins map based on the current identity provider. This example shows you how you might pivot between unauthenticated, Facebook and developer authenticated.

```

override func logins () -> AWSTask<NSDictionary> {
    if(/*logic to determine if user is unauthenticated*/) {
        return AWSTask(result:nil)
    }else if (/*logic to determine if user is Facebook*/){
        if let token = AccessToken.current?.authenticationToken {
            return AWSTask(result: [AWSIdentityProviderFacebook:token])
        }
        return AWSTask(error:NSError(domain: "Facebook Login", code: -1 , userInfo: ["Facebook" : "No current Facebook access token"]))
    }else {
        return super.logins()
    }
}

```

```
    }  
}
```

When you transition from unauthenticated to authenticated, you should call `credentialsProvider.clearCredentials()` to force the SDK to get new authenticated credentials. When you switch between two authenticated providers and you aren't trying to link the two providers (i.e. you are not providing tokens for multiple providers in your logins dictionary), you should call `credentialsProvider.clearKeychain()`. This will clear both the credentials and identity and force the SDK to get new ones.

Unity

Your application might require supporting unauthenticated identities or authenticated identities using public providers (Login with Amazon, Sign in with Apple, Facebook, or Google) along with developer authenticated identities. The essential difference between developer authenticated identities and other identities (unauthenticated identities and authenticated identities using public provider) is the way the `identityId` and `token` are obtained. For other identities the mobile application will interact directly with Amazon Cognito instead of contacting your authentication system. So the mobile application should be able to support two distinct flows depending on the choice made by the app user. For this you will have to make some changes to the custom identity provider.

The recommended way to do it in Unity is to extend your identity provider from `AmazonCognitoEnhancedIdentityProvider` instead of `AbstractCognitoIdentityProvider`, and call the parent `RefreshAsync` method instead of your own in case the user is not authenticated with your own backend. If the user is authenticated, you can use the same flow explained before.

Xamarin

Your application might require supporting unauthenticated identities or authenticated identities using public providers (Login with Amazon, Sign in with Apple, Facebook, or Google) along with developer authenticated identities. The essential difference between developer authenticated identities and other identities (unauthenticated identities and authenticated identities using public provider) is the way the `identityId` and `token` are obtained. For other identities the mobile application will interact directly with Amazon Cognito instead of contacting your authentication system. So the mobile application should be able to support two distinct flows depending on the choice made by the app user. For this you will have to make some changes to the custom identity provider.

Switching unauthenticated users to authenticated users (identity pools)

Amazon Cognito identity pools support both authenticated and unauthenticated users. Unauthenticated users receive access to your AWS resources even if they aren't logged in with any of your identity providers (IdPs). This degree of access is useful to display content to users before they log in. Each unauthenticated user has a unique identity in the identity pool, even though they haven't been individually logged in and authenticated.

This section describes the case where your user chooses to switch from logging in with an unauthenticated identity to using an authenticated identity.

Android

Users can log in to your application as unauthenticated guests. Eventually they might decide to log in using one of the supported IdPs. Amazon Cognito makes sure that an old identity retains the same unique identifier as the new one, and that the profile data is merged automatically.

Your application is informed of a profile merge through the `IdentityChangedListener` interface. Implement the `identityChanged` method in the interface to receive these messages:

```
@override
public void identityChanged(String oldIdentityId, String newIdentityId) {
    // handle the change
}
```

iOS - objective-C

Users can log in to your application as unauthenticated guests. Eventually they might decide to log in using one of the supported IdPs. Amazon Cognito makes sure that an old identity retains the same unique identifier as the new one, and that the profile data is merged automatically.

`NSNotificationCenter` informs your application of a profile merge:

```
[[NSNotificationCenter defaultCenter] addObserver:self
                                         selector:@selector(identityDidChange:)
                                         name:AWSCognitoIdentityIdChangedNotification
                                         object:nil];

-(void)identityDidChange:(NSNotification*)notification {
    NSDictionary *userInfo = notification.userInfo;
    NSLog(@"%@", userInfo);
    [userInfo objectForKey:AWSCognitoNotificationPreviousId];
    [userInfo objectForKey:AWSCognitoNotificationNewId];
}
```

iOS - swift

Users can log in to your application as unauthenticated guests. Eventually they might decide to log in using one of the supported IdPs. Amazon Cognito makes sure that an old identity retains the same unique identifier as the new one, and that the profile data is merged automatically.

`NSNotificationCenter` informs your application of a profile merge:

```
[NSNotificationCenter.defaultCenter().addObserver(observer: self
                                                 selector:"identityDidChange"
                                                 name:AWSCognitoIdentityIdChangedNotification
                                                 object:nil)

func identityDidChange(notification: NSNotification!) {
    if let userInfo = notification.userInfo as? [String: AnyObject] {
        print("identity changed from: \(userInfo[AWSCognitoNotificationPreviousId])"
              to: \(userInfo[AWSCognitoNotificationNewId]))"
    }
}
```

JavaScript

Initially unauthenticated user

Users typically start with the unauthenticated role. For this role, you set the `credentials` property of your configuration object without a `Logins` property. In this case, your default configuration might look like the following:

```
// set the default config object
var creds = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: 'us-east-1:1699ebc0-7900-4099-b910-2df94f52a030'
});
AWS.config.credentials = creds;
```

Switch to authenticated user

When an unauthenticated user logs in to an IdP and you have a token, you can switch the user from unauthenticated to authenticated by calling a custom function that updates the credentials object and adds the Logins token:

```
// Called when an identity provider has a token for a logged in user
function userLoggedIn(providerName, token) {
    creds.params.Logins = creds.params.Logins || {};
    creds.params.Logins[providerName] = token;

    // Expire credentials to refresh them on the next request
    creds.expired = true;
}
```

You can also create a `CognitoIdentityCredentials` object. If you do, you must reset the credentials properties of any existing service objects to reflect the updated credentials configuration information. See [Using the global configuration object](#).

For more information about the `CognitoIdentityCredentials` object, see [AWS.CognitoIdentityCredentials](#) in the AWS SDK for JavaScript API Reference.

Unity

Users can log in to your application as unauthenticated guests. Eventually they might decide to log in using one of the supported IdPs. Amazon Cognito makes sure that an old identity retains the same unique identifier as the new one, and that the profile data is merged automatically.

You can subscribe to the `IdentityChangedEvent` to be notified of profile merges:

```
credentialsProvider.IdentityChangedEvent += delegate(object sender,
    CognitoAWSCredentials.IdentityChangedEventArgs e)
{
    // handle the change
    Debug.log("Identity changed from " + e.OldIdentityId + " to " + e.NewIdentityId);
};
```

Xamarin

Users can log in to your application as unauthenticated guests. Eventually they might decide to log in using one of the supported IdPs. Amazon Cognito makes sure that an old identity retains the same unique identifier as the new one, and that the profile data is merged automatically.

```
credentialsProvider.IdentityChangedEvent += delegate(object sender,
    CognitoAWSCredentials.IdentityChangedEventArgs e){
    // handle the change
    Console.WriteLine("Identity changed from " + e.OldIdentityId + " to " +
    e.NewIdentityId);
};
```

Amazon Cognito Sync

If you're new to Amazon Cognito Sync, use [AWS AppSync](#). Like Amazon Cognito Sync, AWS AppSync is a service for synchronizing application data across devices.

It enables user data like app preferences or game state to be synchronized. It also extends these capabilities by allowing multiple users to synchronize and collaborate in real time on shared data.

Amazon Cognito Sync is an AWS service and client library that makes it possible to sync application-related user data across devices. Amazon Cognito Sync can synchronize user profile data across mobile devices and the web without using your own backend. The client libraries cache data locally so that your app can read and write data regardless of device connectivity status. When the device is online, you can synchronize data. If you set up push sync, you can notify other devices immediately that an update is available.

For information about Amazon Cognito Identity region availability, see [AWS Service Region Availability](#).

To learn more about Amazon Cognito Sync, see the following topics.

Topics

- [Getting started with Amazon Cognito Sync \(p. 312\)](#)
- [Synchronizing data \(p. 313\)](#)
- [Handling callbacks \(p. 320\)](#)
- [Push sync \(p. 332\)](#)
- [Amazon Cognito Streams \(p. 338\)](#)
- [Amazon Cognito Events \(p. 340\)](#)

Getting started with Amazon Cognito Sync

If you're new to Amazon Cognito Sync, use [AWS AppSync](#). Like Amazon Cognito Sync, AWS AppSync is a service for synchronizing application data across devices.

It enables user data like app preferences or game state to be synchronized. It also extends these capabilities by allowing multiple users to synchronize and collaborate in real time on shared data.

Amazon Cognito Sync is an AWS service and client library that enable cross-device syncing of application-related user data. You can use it to synchronize user profile data across mobile devices and web applications. The client libraries cache data locally so your app can read and write data regardless of device connectivity status. When the device is online, you can synchronize data, and if you set up push sync, notify other devices immediately that an update is available.

Sign up for an AWS account

To use Amazon Cognito Sync, you need an AWS account. If you don't already have one, use the following procedure to sign up:

To sign up for an AWS account

1. Open <https://portal.aws.amazon.com/billing/signup>.
2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

Set up an identity pool in Amazon Cognito

Amazon Cognito Sync requires an Amazon Cognito identity pool to provide user identities. Before you use Amazon Cognito Sync you must first set up an identity pool. To create an identity pool and install the SDK, see [Getting started with Amazon Cognito identity pools \(federated identities\) \(p. 242\)](#).

Store and sync data

After you have set up your identity pool and installed the SDK, you can start storing and syncing data between devices. For more information, see [Synchronizing data \(p. 313\)](#).

Synchronizing data

If you're new to Amazon Cognito Sync, use [AWS AppSync](#). Like Amazon Cognito Sync, AWS AppSync is a service for synchronizing application data across devices.

It enables user data like app preferences or game state to be synchronized. It also extends these capabilities by allowing multiple users to synchronize and collaborate in real time on shared data.

With Amazon Cognito, you can save user data in datasets that contain key-value pairs. Amazon Cognito associates this data with an identity in your identity pool so that your app can access it across logins and devices. To sync this data between the Amazon Cognito service and an end user's devices, invoke the synchronize method. Each dataset can have a maximum size of 1 MB. You can associate up to 20 datasets with an identity.

The Amazon Cognito Sync client creates a local cache for the identity data. When your app reads and writes keys, it communicates with this local cache. This communication guarantees that all changes you make on the device are immediately available on the device, even when you are offline. When the synchronize method is called, changes from the service are pulled to the device, and any local changes are pushed to the service. At this point, the changes are available to other devices to synchronize.

Initializing the Amazon Cognito Sync client

To initialize the Amazon Cognito Sync client, you must first create a credentials provider. The credentials provider acquires temporary AWS credentials to make it possible for your app to access your AWS resources. You also must import the necessary header files. Use the following steps to initialize the Amazon Cognito Sync client.

Android

1. Create a credentials provider, following the instructions in [Getting credentials \(p. 268\)](#).
2. Import the Amazon Cognito package as follows: `import com.amazonaws.mobileconnectors.cognito.*;`
3. Initialize Amazon Cognito Sync. Pass in the Android app context, the identity pool ID, an AWS Region, and an initialized Amazon Cognito credentials provider as follows:

```
CognitoSyncManager client = new CognitoSyncManager(  
    getApplicationContext(),
```

```
Regions.YOUR_REGION,  
credentialsProvider);
```

iOS - Objective-C

1. Create a credentials provider, following the instructions in [Getting credentials \(p. 268\)](#).
2. Import AWSCore and Cognito, and initialize AWSCognito as follows:

```
#import <AWSSiOSSDKv2/AWSCore.h>  
#import <AWSCognitoSync/Cognito.h>  
  
AWSCognito *syncClient = [AWSCognito defaultCognito];
```

3. If you're using CocoaPods, replace `<AWSSiOSSDKv2/AWSCore.h>` with `AWSCore.h`. Follow the same syntax for the Amazon Cognito import.

iOS - Swift

1. Create a credentials provider, following the instructions in [Getting credentials \(p. 268\)](#).
2. Import and initialize AWSCognito as follows:

```
import AWSCognito  
let syncClient = AWSCognito.default()!
```

JavaScript

1. Download the [Amazon Cognito Sync Manager for JavaScript](#).
2. Include the Sync Manager library in your project.
3. Create a credentials provider, following the instructions in [Getting credentials \(p. 268\)](#).
4. Initialize the Sync Manager as follows:

```
var syncManager = new AWS.CognitoSyncManager();
```

Unity

1. Create an instance of `CognitoAWSCredentials`, following the instructions in [Getting credentials \(p. 268\)](#).
2. Create an instance of `CognitoSyncManager`. Pass the `CognitoAwsCredentials` object and a `AmazonCognitoSyncConfig`, and include at least the `Region` set, as follows:

```
AmazonCognitoSyncConfig clientConfig = new AmazonCognitoSyncConfig { RegionEndpoint =  
    REGION };  
CognitoSyncManager syncManager = new CognitoSyncManager(credentials, clientConfig);
```

Xamarin

1. Create an instance of `CognitoAWSCredentials`, following the instructions in [Getting credentials \(p. 268\)](#).

2. Create an instance of `CognitoSyncManager`. Pass the `CognitoAwsCredentials` object and a `AmazonCognitoSyncConfig`, and include at least the Region set, as follows:

```
AmazonCognitoSyncConfig clientConfig = new AmazonCognitoSyncConfig { RegionEndpoint = REGION };
CognitoSyncManager syncManager = new CognitoSyncManager(credentials, clientConfig);
```

Understanding datasets

Amazon Cognito organizes user profile data into datasets. Each dataset can contain up to 1MB of data in the form of key-value pairs. A dataset is the most granular entity that you can synchronize. Read and write operations performed on a dataset only affect the local store until the `synchronize` method is invoked. Amazon Cognito identifies a dataset by a unique string. You can create a new dataset or open an existing one as follows.

Android

```
Dataset dataset = client.openOrCreateDataset("datasetname");
```

To delete a dataset, first call the method to remove it from local storage, then call the `synchronize` method to delete the dataset from Amazon Cognito as follows:

```
dataset.delete();
dataset.synchronize(syncCallback);
```

iOS - Objective-C

```
AWSCognitoDataset *dataset = [syncClient openOrCreateDataset:@"myDataSet"];
```

To delete a dataset, first call the method to remove it from local storage, then call the `synchronize` method to delete the dataset from Amazon Cognito as follows:

```
[dataset clear];
[dataset synchronize];
```

iOS - Swift

```
let dataset = syncClient.openOrCreateDataset("myDataSet")!
```

To delete a dataset, first call the method to remove it from local storage, then call the `synchronize` method as follows: to delete the dataset from Amazon Cognito:

```
dataset.clear()
dataset.synchronize()
```

JavaScript

```
syncManager.openOrCreateDataset('myDatasetName', function(err, dataset) {
    // ...
```

```
});
```

Unity

```
string myValue = dataset.Get("myKey");
dataset.Put("myKey", "newValue");
```

To delete a key from a dataset, use Remove as follows:

```
dataset.Remove("myKey");
```

Xamarin

```
Dataset dataset = syncManager.OpenOrCreateDataset("myDatasetName");
```

To delete a dataset, first call the method to remove it from local storage, then call the synchronize method to delete the dataset from Amazon Cognito as follows:

```
dataset.Delete();
dataset.SynchronizeAsync();
```

Reading and writing data in datasets

Amazon Cognito datasets function as dictionaries, with values accessible by key. You can read, add, or modify keys and values of a dataset just as if the dataset were a dictionary, as shown in the following examples.

Note that values you write to a dataset only affect the local cached copy of the data until you call the synchronize method.

Android

```
String value = dataset.get("myKey");
dataset.put("myKey", "my value");
```

iOS - Objective-C

```
[dataset setString:@"my value" forKey:@"myKey"];
NSString *value = [dataset stringForKey:@"myKey"];
```

iOS - Swift

```
dataset.setString("my value", forKey:"myKey")
let value = dataset.string(forKey: "myKey")
```

JavaScript

```
dataset.get('myKey', function(err, value) {
  console.log('myRecord: ' + value);
```

```
});  
  
dataset.put('newKey', 'newValue', function(err, record) {  
    console.log(record);  
});  
  
dataset.remove('oldKey', function(err, record) {  
    console.log(success);  
});
```

Unity

```
string myValue = dataset.Get("myKey");  
dataset.Put("myKey", "newValue");
```

Xamarin

```
//obtain a value  
string myValue = dataset.Get("myKey");  
  
// Create a record in a dataset and synchronize with the server  
dataset.OnSyncSuccess += SyncSuccessCallback;  
dataset.Put("myKey", "myValue");  
dataset.SynchronizeAsync();  
  
void SyncSuccessCallback(object sender, SyncSuccessEventArgs e) {  
    // Your handler code here  
}
```

Android

To remove keys from a dataset, use the `remove` method as follows:

```
dataset.remove("myKey");
```

iOS - Objective-C

To delete a key from a dataset, use `removeObjectForKey` as follows:

```
[dataset removeObjectForKey:@"myKey"];
```

iOS - Swift

To delete a key from a dataset, use `removeObjectForKey` as follows:

```
dataset removeObjectForKey("myKey")
```

Unity

To delete a key from a dataset, use `Remove` as follows:

```
dataset.Remove("myKey");
```

Xamarin

You can use `Remove` to delete a key from a dataset:

```
dataset.Remove("myKey");
```

Synchronizing local data with the sync store

Android

The `synchronize` method compares local cached data to the data stored in the Amazon Cognito Sync store. Remote changes are pulled from the Amazon Cognito Sync store; conflict resolution is invoked if any conflicts occur; and updated values on the device are pushed to the service. To synchronize a dataset, call its `synchronize` method:

```
dataset.synchronize(syncCallback);
```

The `synchronize` method receives an implementation of the `SyncCallback` interface, discussed below.

The `synchronizeOnConnectivity()` method attempts to synchronize when connectivity is available. If connectivity is immediately available, `synchronizeOnConnectivity()` behaves like `synchronize()`. Otherwise it monitors for connectivity changes and performs a sync once connectivity is available. If `synchronizeOnConnectivity()` is called multiple times, only the last synchronize request is kept, and only the last callback will fire. If either the dataset or the callback is garbage-collected, this method won't perform a sync, and the callback won't fire.

To learn more about dataset synchronization and the different callbacks, see [Handling callbacks \(p. 320\)](#).

iOS - Objective-C

The `synchronize` method compares local cached data to the data stored in the Amazon Cognito Sync store. Remote changes are pulled from the Amazon Cognito Sync store; conflict resolution is invoked if any conflicts occur; and updated values on the device are pushed to the service. To synchronize a dataset, call its `synchronize` method:

The `synchronize` method is asynchronous and returns an `AWSTask` object to handle the response:

```
[[dataset synchronize] continueWithBlock:^id(AWSTask *task) {
    if (task.isCancelled) {
        // Task cancelled.
    } else if (task.error) {
        // Error while executing task.
    } else {
        // Task succeeded. The data was saved in the sync store.
    }
    return nil;
}];
```

The `synchronizeOnConnectivity` method attempts to synchronize when the device has connectivity. First, `synchronizeOnConnectivity` checks for connectivity and, if the device is online, immediately invokes `synchronize` and returns the `AWSTask` object associated with the attempt.

If the device is offline, `synchronizeOnConnectivity` 1) schedules a `synchronize` for the next time the device comes online and 2) returns an `AWSTask` with a nil result. The scheduled `synchronize` is only

valid for the lifecycle of the dataset object. The data will not be synchronized if the app is exited before connectivity is regained. If you want to be notified when events occur during the scheduled synchronize, you must add observers of the notifications found in [AWS Cognito](#).

To learn more about dataset synchronization and the different callbacks, see [Handling callbacks \(p. 320\)](#).

iOS - Swift

The `synchronize` method compares local cached data to the data stored in the Amazon Cognito Sync store. Remote changes are pulled from the Amazon Cognito Sync store; conflict resolution is invoked if any conflicts occur; and updated values on the device are pushed to the service. To synchronize a dataset, call its `synchronize` method:

The `synchronize` method is asynchronous and returns an `AWSTask` object to handle the response:

```
dataset.synchronize().continueWith(block: { (task) -> AnyObject? in
    if task.isCancelled {
        // Task cancelled.
    } else if task.error != nil {
        // Error while executing task
    } else {
        // Task succeeded. The data was saved in the sync store.
    }
    return task
})
```

The `synchronizeOnConnectivity` method attempts to synchronize when the device has connectivity. First, `synchronizeOnConnectivity` checks for connectivity and, if the device is online, immediately invokes `synchronize` and returns the `AWSTask` object associated with the attempt.

If the device is offline, `synchronizeOnConnectivity` 1) schedules a synchronize for the next time the device comes online and 2) returns an `AWSTask` object with a nil result. The scheduled synchronize is only valid for the lifecycle of the dataset object. The data will not be synchronized if the app is exited before connectivity is regained. If you want to be notified when events occur during the scheduled synchronize, you must add observers of the notifications found in [AWS Cognito](#).

To learn more about dataset synchronization and the different callbacks, see [Handling callbacks \(p. 320\)](#).

JavaScript

The `synchronize` method compares local cached data to the data stored in the Amazon Cognito Sync store. Remote changes are pulled from the Amazon Cognito Sync store; conflict resolution is invoked if any conflicts occur; and updated values on the device are pushed to the service. To synchronize a dataset, call its `synchronize` method:

```
dataset.synchronize();
```

To learn more about dataset synchronization and the different callbacks, see [Handling callbacks \(p. 320\)](#).

Unity

The `synchronize` method compares local cached data to the data stored in the Amazon Cognito Sync store. Remote changes are pulled from the Amazon Cognito Sync store; conflict resolution is invoked if

any conflicts occur; and updated values on the device are pushed to the service. To synchronize a dataset, call its `synchronize` method:

```
dataset.Synchronize();
```

Synchronize will run asynchronously and will end up calling one of the several callbacks you can specify in the Dataset.

To learn more about dataset synchronization and the different callbacks, see [Handling callbacks \(p. 320\)](#).

Xamarin

The `synchronize` method compares local cached data to the data stored in the Amazon Cognito Sync store. Remote changes are pulled from the Amazon Cognito Sync store; conflict resolution is invoked if any conflicts occur; and updated values on the device are pushed to the service. To synchronize a dataset, call its `synchronize` method:

```
dataset.SynchronizeAsync();
```

To learn more about dataset synchronization and the different callbacks, see [Handling callbacks \(p. 320\)](#).

Handling callbacks

If you're new to Amazon Cognito Sync, use [AWS AppSync](#). Like Amazon Cognito Sync, AWS AppSync is a service for synchronizing application data across devices.

It enables user data like app preferences or game state to be synchronized. It also extends these capabilities by allowing multiple users to synchronize and collaborate in real time on shared data.

This section describes how to handle callbacks.

Android

SyncCallback Interface

By implementing the `SyncCallback` interface, you can receive notifications on your app about dataset synchronization. Your app can then make active decisions about deleting local data, merging unauthenticated and authenticated profiles, and resolving sync conflicts. You should implement the following methods, which are required by the interface:

- `onSuccess()`
- `onFailure()`
- `onConflict()`
- `onDatasetDeleted()`
- `onDatasetsMerged()`

Note that, if you don't want to specify all the callbacks, you can also use the class `DefaultSyncCallback` which provides default, empty implementations for all of them.

onSuccess

The `onSuccess()` callback is triggered when a dataset is successfully downloaded from the sync store.

```
@Override
public void onSuccess(Dataset dataset, List<Record> newRecords) {
}
```

onFailure

`onFailure()` is called if an exception occurs during synchronization.

```
@Override
public void onFailure(DataStorageException dse) {
}
```

onConflict

Conflicts may arise if the same key has been modified on the local store and in the sync store. The `onConflict()` method handles conflict resolution. If you don't implement this method, the Amazon Cognito Sync client defaults to using the most recent change.

```
@Override
public boolean onConflict(Dataset dataset, final List<SyncConflict> conflicts) {
    List<Record> resolvedRecords = new ArrayList<Record>();
    for (SyncConflict conflict : conflicts) {
        /* resolved by taking remote records */
        resolvedRecords.add(conflict.resolveWithRemoteRecord());

        /* alternately take the local records */
        // resolvedRecords.add(conflict.resolveWithLocalRecord());

        /* or customer logic, say concatenate strings */
        // String newValue = conflict.getRemoteRecord().getValue()
        //     + conflict.getLocalRecord().getValue();
        // resolvedRecords.add(conflict.resolveWithValue(newValue));
    }
    dataset.resolve(resolvedRecords);

    // return true so that synchronize() is retried after conflicts are resolved
    return true;
}
```

onDatasetDeleted

When a dataset is deleted, the Amazon Cognito client uses the `SyncCallback` interface to confirm whether the local cached copy of the dataset should be deleted too. Implement the `onDatasetDeleted()` method to tell the client SDK what to do with the local data.

```
@Override
public boolean onDatasetDeleted(Dataset dataset, String datasetName) {
    // return true to delete the local copy of the dataset
    return true;
}
```

onDatasetMerged

When two previously unconnected identities are linked together, all of their datasets are merged. Applications are notified of the merge through the `onDatasetsMerged()` method:

```
@Override
public boolean onDatasetsMerged(Dataset dataset, List<String> datasetNames) {
    // return false to handle Dataset merge outside the synchronization callback
}
```

```
    return false;
}
```

iOS - Objective-C

Sync Notifications

The Amazon Cognito client will emit a number of `NSNotification` events during a synchronize call. You can register to monitor these notifications via the standard `NSNotificationCenter`:

```
[NSNotificationCenter defaultCenter]
addObserver:self
selector:@selector(myNotificationHandler:)
name:NOTIFICATION_TYPE
object:nil];
```

Amazon Cognito supports five notification types, listed below.

AWSCognitoDidStartSynchronizeNotification

Called when a synchronize operation is starting. The `userInfo` will contain the key dataset which is the name of the dataset being synchronized.

AWSCognitoDidEndSynchronizeNotification

Called when a synchronize operation completes (successfully or otherwise). The `userInfo` will contain the key dataset which is the name of the dataset being synchronized.

AWSCognitoDidFailToSynchronizeNotification

Called when a synchronize operation fails. The `userInfo` will contain the key dataset which is the name of the dataset being synchronized and the key `error` which will contain the error that caused the failure.

AWSCognitoDidChangeRemoteValueNotification

Called when local changes are successfully pushed to Amazon Cognito. The `userInfo` will contain the key dataset which is the name of the dataset being synchronized and the key `keys` which will contain an `NSArray` of record keys that were pushed.

AWSCognitoDidChangeLocalValueFromRemoteNotification

Called when a local value changes due to a synchronize operation. The `userInfo` will contain the key dataset which is the name of the dataset being synchronized and the key `keys` which will contain an `NSArray` of record keys that changed.

Conflict Resolution Handler

During a sync operation, conflicts may arise if the same key has been modified on the local store and in the sync store. If you haven't set a conflict resolution handler, Amazon Cognito defaults to choosing the most recent update.

By implementing and assigning an `AWSCognitoRecordConflictHandler` you can alter the default conflict resolution. The `AWSCognitoConflict` input parameter `conflict` contains an `AWSCognitoRecord` object for both the local cached data and for the conflicting record in the sync store. Using the `AWSCognitoConflict` you can resolve the conflict with the local record: `[conflict resolveWithLocalRecord]`, the remote record: `[conflict resolveWithRemoteRecord]` or a brand new value: `[conflict resolveWithValue:value]`. Returning `nil` from this method prevents synchronization from continuing and the conflicts will be presented again the next time the sync process starts.

You can set the conflict resolution handler at the client level:

```
client.conflictHandler = ^AWSCognitoResolvedConflict* (NSString *datasetName,
    AWSCognitoConflict *conflict) {
    // always choose local changes
    return [conflict resolveWithLocalRecord];
};
```

Or at the dataset level:

```
dataset.conflictHandler = ^AWSCognitoResolvedConflict* (NSString *datasetName,
    AWSCognitoConflict *conflict) {
    // override and always choose remote changes
    return [conflict resolveWithRemoteRecord];
};
```

Dataset Deleted Handler

When a dataset is deleted, the Amazon Cognito client uses the `AWSCognitoDatasetDeletedHandler` to confirm whether the local cached copy of the dataset should be deleted too. If no `AWSCognitoDatasetDeletedHandler` is implemented, the local data will be purged automatically. Implement an `AWSCognitoDatasetDeletedHandler` if you wish to keep a copy of the local data before wiping, or to keep the local data.

You can set the dataset deleted handler at the client level:

```
client.datasetDeletedHandler = ^BOOL (NSString *datasetName) {
    // make a backup of the data if you choose
    ...
    // delete the local data (default behavior)
    return YES;
};
```

Or at the dataset level:

```
dataset.datasetDeletedHandler = ^BOOL (NSString *datasetName) {
    // override default and keep the local data
    return NO;
};
```

Dataset Merge Handler

When two previously unconnected identities are linked together, all of their datasets are merged. Applications are notified of the merge through the `DatasetMergeHandler`. The handler will receive the name of the root dataset as well as an array of dataset names that are marked as merges of the root dataset.

If no `DatasetMergeHandler` is implemented, these datasets will be ignored, but will continue to use up space in the identity's 20 maximum total datasets.

You can set the dataset merge handler at the client level:

```
client.datasetMergedHandler = ^(NSString *datasetName, NSArray *datasets) {
    // Blindly delete the datasets
    for (NSString *name in datasets) {
        AWSCognitoDataset *merged = [[AWSCognito defaultCognito] openOrCreateDataset:name];
        [merged clear];
        [merged synchronize];
    }
};
```

Or at the dataset level:

```
dataset.datasetMergedHandler = ^(NSString *datasetName, NSArray *datasets) {
    // Blindly delete the datasets
    for (NSString *name in datasets) {
        AWSCognitoDataset *merged = [[AWSCognito defaultCognito] openOrCreateDataset:name];
        // do something with the data if it differs from existing dataset
        ...
        // now delete it
        [merged clear];
        [merged synchronize];
    }
};
```

iOS - Swift

Sync Notifications

The Amazon Cognito client will emit a number of `NSNotification` events during a `synchronize` call. You can register to monitor these notifications via the standard `NSNotificationCenter`:

```
NSNotificationCenter.defaultCenter().addObserver(observer: self,
    selector: "myNotificationHandler",
    name:NOTIFICATION_TYPE,
    object:nil)
```

Amazon Cognito supports five notification types, listed below.

AWSCognitoDidStartSynchronizeNotification

Called when a synchronize operation is starting. The `userInfo` will contain the key dataset which is the name of the dataset being synchronized.

AWSCognitoDidEndSynchronizeNotification

Called when a synchronize operation completes (successfully or otherwise). The `userInfo` will contain the key dataset which is the name of the dataset being synchronized.

AWSCognitoDidFailToSynchronizeNotification

Called when a synchronize operation fails. The `userInfo` will contain the key dataset which is the name of the dataset being synchronized and the key error which will contain the error that caused the failure.

AWSCognitoDidChangeRemoteValueNotification

Called when local changes are successfully pushed to Amazon Cognito. The `userInfo` will contain the key dataset which is the name of the dataset being synchronized and the key keys which will contain an `NSArray` of record keys that were pushed.

AWSCognitoDidChangeLocalValueFromRemoteNotification

Called when a local value changes due to a synchronize operation. The `userInfo` will contain the key dataset which is the name of the dataset being synchronized and the key keys which will contain an `NSArray` of record keys that changed.

Conflict Resolution Handler

During a sync operation, conflicts may arise if the same key has been modified on the local store and in the sync store. If you haven't set a conflict resolution handler, Amazon Cognito defaults to choosing the most recent update.

By implementing and assigning an `AWSCognitoRecordConflictHandler` you can alter the default conflict resolution. The `AWSCognitoConflict` input parameter `conflict` contains an `AWSCognitoRecord` object for both the local cached data and for the conflicting record in the sync store. Using the `AWSCognitoConflict` you can resolve the conflict with the local record: `[conflict resolveWithLocalRecord]`, the remote record: `[conflict resolveWithRemoteRecord]` or a brand new value: `[conflict resolveWithValue:value]`. Returning `nil` from this method prevents synchronization from continuing and the conflicts will be presented again the next time the sync process starts.

You can set the conflict resolution handler at the client level:

```
client.conflictHandler = {
    (datasetName: String?, conflict: AWSCognitoConflict?) -> AWSCognitoResolvedConflict? in
        return conflict.resolveWithLocalRecord()
}
```

Or at the dataset level:

```
dataset.conflictHandler = {
    (datasetName: String?, conflict: AWSCognitoConflict?) -> AWSCognitoResolvedConflict? in
        return conflict.resolveWithLocalRecord()
}
```

Dataset Deleted Handler

When a dataset is deleted, the Amazon Cognito client uses the `AWSCognitoDatasetDeletedHandler` to confirm whether the local cached copy of the dataset should be deleted too. If no `AWSCognitoDatasetDeletedHandler` is implemented, the local data will be purged automatically. Implement an `AWSCognitoDatasetDeletedHandler` if you wish to keep a copy of the local data before wiping, or to keep the local data.

You can set the dataset deleted handler at the client level:

```
client.datasetDeletedHandler = {
    (datasetName: String!) -> Bool in
        // make a backup of the data if you choose
        ...
        // delete the local data (default behaviour)
        return true
}
```

Or at the dataset level:

```
dataset.datasetDeletedHandler = {
    (datasetName: String!) -> Bool in
        // make a backup of the data if you choose
        ...
        // delete the local data (default behaviour)
        return true
}
```

Dataset merge handler

When two previously unconnected identities are linked together, all of their datasets are merged. Applications are notified of the merge through the `DatasetMergeHandler`. The handler will receive the name of the root dataset as well as an array of dataset names that are marked as merges of the root dataset.

If no `DatasetMergeHandler` is implemented, these datasets will be ignored, but will continue to use up space in the identity's 20 maximum total datasets.

You can set the dataset merge handler at the client level:

```
client.datasetMergedHandler = {
    (datasetName: String!, datasets: [AnyObject]!) -> Void in
    for nameObject in datasets {
        if let name = nameObject as? String {
            let merged = AWSognito.defaultCognito().openOrCreateDataset(name)
            merged.clear()
            merged.synchronize()
        }
    }
}
```

Or at the dataset level:

```
dataset.datasetMergedHandler = {
    (datasetName: String!, datasets: [AnyObject]!) -> Void in
    for nameObject in datasets {
        if let name = nameObject as? String {
            let merged = AWSognito.defaultCognito().openOrCreateDataset(name)
            // do something with the data if it differs from existing dataset
            ...
            // now delete it
            merged.clear()
            merged.synchronize()
        }
    }
}
```

JavaScript

Synchronization callbacks

When performing a synchronize() on a dataset, you can optionally specify callbacks to handle each of the following states:

```
dataset.synchronize({
    onSuccess: function(dataset, newRecords) {
        //...
    },
    onFailure: function(err) {
        //...
    },
    onConflict: function(dataset, conflicts, callback) {
        //...
    },
    onDatasetDeleted: function(dataset, datasetName, callback) {
        //...
    },
    onDatasetMerged: function(dataset, datasetNames, callback) {
        //...
    }
});
```

onSuccess()

The `onSuccess()` callback is triggered when a dataset is successfully updated from the sync store. If you do not define a callback, the synchronization will succeed silently.

```
onSuccess: function(dataset, newRecords) {
    console.log('Successfully synchronized ' + newRecords.length + ' new records.');
}
```

onFailure()

`onFailure()` is called if an exception occurs during synchronization. If you do not define a callback, the synchronization will fail silently.

```
onFailure: function(err) {
    console.log('Synchronization failed.');
    console.log(err);
}
```

onConflict()

Conflicts may arise if the same key has been modified on the local store and in the sync store. The `onConflict()` method handles conflict resolution. If you don't implement this method, the synchronization will be aborted when there is a conflict.

```
onConflict: function(dataset, conflicts, callback) {

    var resolved = [];

    for (var i=0; i<conflicts.length; i++) {

        // Take remote version.
        resolved.push(conflicts[i].resolveWithRemoteRecord());

        // Or... take local version.
        // resolved.push(conflicts[i].resolveWithLocalRecord());

        // Or... use custom logic.
        // var newValue = conflicts[i].getRemoteRecord().getValue() +
        conflicts[i].getLocalRecord().getValue();
        // resolved.push(conflicts[i].resovleWithValue(newValue));

    }

    dataset.resolve(resolved, function() {
        return callback(true);
    });

    // Or... callback false to stop the synchronization process.
    // return callback(false);
}

}
```

onDatasetDeleted()

When a dataset is deleted, the Amazon Cognito client uses the `onDatasetDeleted()` callback to decide whether the local cached copy of the dataset should be deleted too. By default, the dataset will not be deleted.

```
onDatasetDeleted: function(dataset, datasetName, callback) {

    // Return true to delete the local copy of the dataset.
```

```

    // Return false to handle deleted datasets outside the synchronization callback.

    return callback(true);

}

```

onDatasetMerged()

When two previously unconnected identities are linked together, all of their datasets are merged. Applications are notified of the merge through the `onDatasetsMerged()` callback.

```

onDatasetMerged: function(dataset, datasetNames, callback) {

    // Return true to continue the synchronization process.
    // Return false to handle dataset merges outside the synchronization callback.

    return callback(false);

}

```

Unity

After you open or create a dataset, you can set different callbacks to it that will be triggered when you use the `Synchronize` method. This is the way to register your callbacks to them:

```

dataset.OnSyncSuccess += this.HandleSyncSuccess;
dataset.OnSyncFailure += this.HandleSyncFailure;
dataset.OnSyncConflict = this.HandleSyncConflict;
dataset.OnDatasetMerged = this.HandleDatasetMerged;
dataset.OnDatasetDeleted = this.HandleDatasetDeleted;

```

Note that `SyncSuccess` and `SyncFailure` use `+=` instead of `=` so you can subscribe more than one callback to them.

OnSyncSuccess

The `OnSyncSuccess` callback is triggered when a dataset is successfully updated from the cloud. If you do not define a callback, the synchronization will succeed silently.

```

private void HandleSyncSuccess(object sender, SyncSuccessEvent e)
{
    // Continue with your game flow, display the loaded data, etc.
}

```

OnSyncFailure

`OnSyncFailure` is called if an exception occurs during synchronization. If you do not define a callback, the synchronization will fail silently.

```

private void HandleSyncFailure(object sender, SyncFailureEvent e)
{
    Dataset dataset = sender as Dataset;
    if (dataset.Metadata != null) {
        Debug.Log("Sync failed for dataset : " + dataset.Metadata.DatasetName);
    } else {
        Debug.Log("Sync failed");
    }
    // Handle the error
    Debug.LogException(e.Exception);
}

```

}

OnSyncConflict

Conflicts may arise if the same key has been modified on the local store and in the sync store. The `OnSyncConflict` callback handles conflict resolution. If you don't implement this method, the synchronization will be aborted when there is a conflict.

```
private bool HandleSyncConflict(Dataset dataset, List < SyncConflict > conflicts)
{
    if (dataset.Metadata != null) {
        Debug.LogWarning("Sync conflict " + dataset.Metadata.DatasetName);
    } else {
        Debug.LogWarning("Sync conflict");
    }
    List < Amazon.CognitoSync.SyncManager.Record > resolvedRecords = new List <
    Amazon.CognitoSync.SyncManager.Record > ();
    foreach(SyncConflict conflictRecord in conflicts) {
        // SyncManager provides the following default conflict resolution methods:
        //     ResolveWithRemoteRecord - overwrites the local with remote records
        //     ResolveWithLocalRecord - overwrites the remote with local records
        //     ResolveWithValue - to implement your own logic
        resolvedRecords.Add(conflictRecord.ResolveWithRemoteRecord());
    }
    // resolves the conflicts in local storage
    dataset.Resolve(resolvedRecords);
    // on return true the synchronize operation continues where it left,
    // returning false cancels the synchronize operation
    return true;
}
```

OnDatasetDeleted

When a dataset is deleted, the Amazon Cognito client uses the `OnDatasetDeleted` callback to decide whether the local cached copy of the dataset should be deleted too. By default, the dataset will not be deleted.

```
private bool HandleDatasetDeleted(Dataset dataset)
{
    Debug.Log(dataset.Metadata.DatasetName + " Dataset has been deleted");
    // Do clean up if necessary
    // returning true informs the corresponding dataset can be purged in the local
    storage and return false retains the local dataset
    return true;
}
```

OnDatasetMerged

When two previously unconnected identities are linked together, all of their datasets are merged. Applications are notified of the merge through the `OnDatasetsMerged` callback.

```
public bool HandleDatasetMerged(Dataset localDataset, List<string> mergedDatasetNames)
{
    foreach (string name in mergedDatasetNames)
    {
        Dataset mergedDataset = syncManager.OpenOrCreateDataset(name);
        //Lambda function to delete the dataset after fetching it
        EventHandler<SyncSuccessEvent> lambda;
        lambda = (object sender, SyncSuccessEvent e) => {
            ICollection<string> existingValues = localDataset.GetAll().Values;
            ICollection<string> newValues = mergedDataset.GetAll().Values;
```

```

        //Implement your merge logic here

        mergedDataset.Delete(); //Delete the dataset locally
        mergedDataset.OnSyncSuccess -= lambda; //We don't want this callback to be
        fired again
        mergedDataset.OnSyncSuccess += (object s2, SyncSuccessEventArgs e2) => {
            localDataset.Synchronize(); //Continue the sync operation that was
            interrupted by the merge
        };
        mergedDataset.Synchronize(); //Synchronize it as deleted, failing to do so will
        leave us in an inconsistent state
    };
    mergedDataset.OnSyncSuccess += lambda;
    mergedDataset.Synchronize(); //Asynchronously fetch the dataset
}

// returning true allows the Synchronize to continue and false stops it
return false;
}

```

Xamarin

After you open or create a dataset, you can set different callbacks to it that will be triggered when you use the `Synchronize` method. This is the way to register your callbacks to them:

```

dataset.OnSyncSuccess += this.HandleSyncSuccess;
dataset.OnSyncFailure += this.HandleSyncFailure;
dataset.OnSyncConflict = this.HandleSyncConflict;
dataset.OnDatasetMerged = this.HandleDatasetMerged;
dataset.OnDatasetDeleted = this.HandleDatasetDeleted;

```

Note that `SyncSuccess` and `SyncFailure` use `+=` instead of `=` so you can subscribe more than one callback to them.

OnSyncSuccess

The `OnSyncSuccess` callback is triggered when a dataset is successfully updated from the cloud. If you do not define a callback, the synchronization will succeed silently.

```

private void HandleSyncSuccess(object sender, SyncSuccessEventArgs e)
{
    // Continue with your game flow, display the loaded data, etc.
}

```

OnSyncFailure

`OnSyncFailure` is called if an exception occurs during synchronization. If you do not define a callback, the synchronization will fail silently.

```

private void HandleSyncFailure(object sender, SyncFailureEventArgs e)
{
    Dataset dataset = sender as Dataset;
    if (dataset.Metadata != null) {
        Console.WriteLine("Sync failed for dataset : " + dataset.Metadata.DatasetName);
    } else {
        Console.WriteLine("Sync failed");
    }
}

```

OnSyncConflict

Conflicts may arise if the same key has been modified on the local store and in the sync store. The `OnSyncConflict` callback handles conflict resolution. If you don't implement this method, the synchronization will be aborted when there is a conflict.

```
private bool HandleSyncConflict(Dataset dataset, List < SyncConflict > conflicts)
{
    if (dataset.Metadata != null) {
        Console.WriteLine("Sync conflict " + dataset.Metadata.DatasetName);
    } else {
        Console.WriteLine("Sync conflict");
    }
    List < Amazon.CognitoSync.SyncManager.Record > resolvedRecords = new List <
    Amazon.CognitoSync.SyncManager.Record > ();
    foreach(SyncConflict conflictRecord in conflicts) {
        // SyncManager provides the following default conflict resolution methods:
        //     ResolveWithRemoteRecord - overwrites the local with remote records
        //     ResolveWithLocalRecord - overwrites the remote with local records
        //     ResolveWithValue - to implement your own logic
        resolvedRecords.Add(conflictRecord.ResolveWithRemoteRecord());
    }
    // resolves the conflicts in local storage
    dataset.Resolve(resolvedRecords);
    // on return true the synchronize operation continues where it left,
    // returning false cancels the synchronize operation
    return true;
}
```

OnDatasetDeleted

When a dataset is deleted, the Amazon Cognito client uses the `OnDatasetDeleted` callback to decide whether the local cached copy of the dataset should be deleted too. By default, the dataset will not be deleted.

```
private bool HandleDatasetDeleted(Dataset dataset)
{
    Console.WriteLine(dataset.Metadata.DatasetName + " Dataset has been deleted");
    // Do clean up if necessary
    // returning true informs the corresponding dataset can be purged in the local storage
    and return false retains the local dataset
    return true;
}
```

OnDatasetMerged

When two previously unconnected identities are linked together, all of their datasets are merged. Applications are notified of the merge through the `OnDatasetsMerged` callback.

```
public bool HandleDatasetMerged(Dataset localDataset, List<string> mergedDatasetNames)
{
    foreach (string name in mergedDatasetNames)
    {
        Dataset mergedDataset = syncManager.OpenOrCreateDataset(name);

        //Implement your merge logic here

        mergedDataset.OnSyncSuccess += lambda;
        mergedDataset.SynchronizeAsync(); //Asynchronously fetch the dataset
    }

    // returning true allows the Synchronize to continue and false stops it
}
```

```
    return false;  
}
```

Push sync

If you're new to Amazon Cognito Sync, use [AWS AppSync](#). Like Amazon Cognito Sync, AWS AppSync is a service for synchronizing application data across devices. It enables user data like app preferences or game state to be synchronized. It also extends these capabilities by allowing multiple users to synchronize and collaborate in real time on shared data.

Amazon Cognito automatically tracks the association between identity and devices. Using the push synchronization, or push sync, feature, you can ensure that every instance of a given identity is notified when identity data changes. Push sync ensures that, whenever the sync store data changes for a particular identity, all devices associated with that identity receive a silent push notification informing them of the change.

Note

Push sync is not supported for JavaScript, Unity, or Xamarin.

Before you can use push sync, you must first set up your account for push sync and enable push sync in the Amazon Cognito console.

Create an Amazon Simple Notification Service (Amazon SNS) app

Create and configure an Amazon SNS app for your supported platforms, as described in the [SNS Developer Guide](#).

Enable push sync in the Amazon Cognito console

You can enable push sync via the Amazon Cognito console. From the [console home page](#):

1. Click the name of the identity pool for which you want to enable push sync. The **Dashboard** page for your identity pool appears.
2. In the top-right corner of the **Dashboard** page, click **Manage Identity Pools**. The **Federated Identities** page appears.
3. Scroll down and click **Push synchronization** to expand it.
4. In the **Service role** dropdown menu, select the IAM role that grants Cognito permission to send an SNS notification. Click **Create role** to create or modify the roles associated with your identity pool in the [AWS IAM Console](#).
5. Select a platform application, and then click **Save Changes**.
6. Grant SNS Access to Your Application

In the AWS Identity and Access Management console, configure your IAM roles to have full Amazon SNS access, or create a new role that has full Amazon SNS access. The following example role trust policy grants Amazon Cognito Sync a limited ability to assume an IAM role. Amazon Cognito Sync can only assume the role when it does so on behalf of both the identity pool in the `aws:SourceArn` condition and the account in the `aws:SourceAccount` condition.

```
{  
  "Version": "2012-10-17",
```

```

    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "cognito-sync.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "AWS:SourceArn": "arn:aws:cognito-identity:us-
east-1:123456789012:identitypool/us-east-1:177a950c-2c08-43f0-9983-28727EXAMPLE"
                }
            }
        }
    ]
}

```

To learn more about IAM roles, see [Roles \(Delegation and Federation\)](#).

Use push sync in your app: Android

Your application will need to import the Google Play services. You can download the latest version of the Google Play SDK via the [Android SDK manager](#). Follow the Android documentation on [Android Implementation](#) to register your app and receive a registration ID from GCM. Once you have the registration ID, you need to register the device with Amazon Cognito, as shown in the snippet below:

```

String registrationId = "MY_GCM_REGISTRATION_ID";
try {
    client.registerDevice("GCM", registrationId);
} catch (RegistrationFailedException rfe) {
    Log.e(TAG, "Failed to register device for silent sync", rfe);
} catch (AmazonClientException ace) {
    Log.e(TAG, "An unknown error caused registration for silent sync to fail", ace);
}

```

You can now subscribe a device to receive updates from a particular dataset:

```

Dataset trackedDataset = client.openOrCreateDataset("myDataset");
if (client.isDeviceRegistered()) {
    try {
        trackedDataset.subscribe();
    } catch (SubscribeFailedException sfe) {
        Log.e(TAG, "Failed to subscribe to datasets", sfe);
    } catch (AmazonClientException ace) {
        Log.e(TAG, "An unknown error caused the subscription to fail", ace);
    }
}

```

To stop receiving push notifications from a dataset, simply call the `unsubscribe` method. To subscribe to all datasets (or a specific subset) in the `CognitoSyncManager` object, use `subscribeAll()`:

```

if (client.isDeviceRegistered()) {
    try {
        client.subscribeAll();
    } catch (SubscribeFailedException sfe) {
        Log.e(TAG, "Failed to subscribe to datasets", sfe);
    } catch (AmazonClientException ace) {

```

```

        Log.e(TAG, "An unknown error caused the subscription to fail", ace);
    }
}
```

In your implementation of the [Android BroadcastReceiver](#) object, you can check the latest version of the modified dataset and decide if your app needs to synchronize again:

```

@Override
public void onReceive(Context context, Intent intent) {

    PushSyncUpdate update = client.getPushSyncUpdate(intent);

    // The update has the source (cognito-sync here), identityId of the
    // user, identityPoolId in question, the non-local sync count of the
    // data set and the name of the dataset. All are accessible through
    // relevant getters.

    String source = update.getSource();
    String identityPoolId = update.getIdentityPoolId();
    String identityId = update.getIdentityId();
    String datasetName = update.getDatasetName();
    long syncCount = update.getSyncCount();

    Dataset dataset = client.openOrCreateDataset(datasetName);

    // need to access last sync count. If sync count is less or equal to
    // last sync count of the dataset, no sync is required.

    long lastSyncCount = dataset.getLastSyncCount();
    if (lastSyncCount < syncCount) {
        dataset.synchronize(new SyncCallback() {
            // ...
        });
    }
}
```

The following keys are available in the push notification payload:

- **source:** cognito-sync. This can serve as a differentiating factor between notifications.
- **identityPoolId:** The identity pool ID. This can be used for validation or additional information, though it's not integral from the receiver's point of view.
- **identityId:** The identity ID within the pool.
- **datasetName:** The name of the dataset that was updated. This is available for the sake of the openOrCreateDataset call.
- **syncCount:** The sync count for the remote dataset. You can use this as a way to make sure that the local dataset is out of date, and that the incoming synchronization is new.

Use push sync in your app: iOS - Objective-C

To obtain a device token for your app, follow the Apple documentation on Registering for Remote Notifications. Once you've received the device token as an `NSData` object from APNs, you'll need to register the device with Amazon Cognito using the `registerDevice:` method of the sync client, as shown below:

```

AWSIdentityProvider *syncClient = [AWSIdentityProvider defaultIdentityProvider];
[[syncClient registerDevice: devToken] continueWithBlock:^id(AWSTask *task) {
    if(task.error){
```

```

        NSLog(@"Unable to registerDevice: %@", task.error);
    } else {
        NSLog(@"Successfully registered device with id: %@", task.result);
    }
    return nil;
}
];

```

In debug mode, your device will register with the APNs sandbox; in release mode, it will register with APNs. To receive updates from a particular dataset, use the `subscribe` method:

```

[[[syncClient openOrCreateDataset:@"MyDataset"] subscribe] continueWithBlock:^id(AWSTask
*task) {
    if(task.error){
        NSLog(@"Unable to subscribe to dataset: %@", task.error);
    } else {
        NSLog(@"Successfully subscribed to dataset: %@", task.result);
    }
    return nil;
}
];

```

To stop receiving push notifications from a dataset, simply call the `unsubscribe` method:

```

[[[syncClient openOrCreateDataset:@"MyDataset"] unsubscribe] continueWithBlock:^id(AWSTask
*task) {
    if(task.error){
        NSLog(@"Unable to unsubscribe from dataset: %@", task.error);
    } else {
        NSLog(@"Successfully unsubscribed from dataset: %@", task.result);
    }
    return nil;
}
];

```

To subscribe to all datasets in the `AWSCognito` object, call `subscribeAll`:

```

[[syncClient subscribeAll] continueWithBlock:^id(AWSTask *task) {
    if(task.error){
        NSLog(@"Unable to subscribe to all datasets: %@", task.error);
    } else {
        NSLog(@"Successfully subscribed to all datasets: %@", task.result);
    }
    return nil;
}
];

```

Before calling `subscribeAll`, be sure to synchronize at least once on each dataset, so that the datasets exist on the server.

To react to push notifications, you need to implement the `didReceiveRemoteNotification` method in your app delegate:

```

- (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary
*)userInfo
{
    [[NSNotificationCenter defaultCenter]
postNotificationName:@"CognitoPushNotification" object:userInfo];
}

```

If you post a notification using notification handler, you can then respond to the notification elsewhere in the application where you have a handle to your dataset. If you subscribe to the notification like this ...

```
[[NSNotificationCenter defaultCenter] addObserver:self
    selector:@selector(didReceivePushSync:)
    name: @"CognitoPushNotification" object:nil];
```

...you can act on the notification like this:

```
- (void)didReceivePushSync:(NSNotification*)notification
{
    NSDictionary * data = [(NSDictionary *)[notification object] objectForKey:@"data"];
    NSString * identityId = [data objectForKey:@"identityId"];
    NSString * datasetName = [data objectForKey:@"datasetName"];
    if([self.dataset.name isEqualToString:datasetName] && [self.identityId isEqualToString:identityId]){
        [[self.dataset synchronize] continueWithBlock:^id(AWSTask *task) {
            if(!task.error){
                NSLog(@"Successfully synced dataset");
            }
            return nil;
        }];
    }
}
```

The following keys are available in the push notification payload:

- **source:** cognito-sync. This can serve as a differentiating factor between notifications.
- **identityPoolId:** The identity pool ID. This can be used for validation or additional information, though it's not integral from the receiver's point of view.
- **identityId:** The identity ID within the pool.
- **datasetName:** The name of the dataset that was updated. This is available for the sake of the `openOrCreateDataset` call.
- **syncCount:** The sync count for the remote dataset. You can use this as a way to make sure that the local dataset is out of date, and that the incoming synchronization is new.

Use push sync in your app: iOS - Swift

To obtain a device token for your app, follow the Apple documentation on Registering for Remote Notifications. Once you've received the device token as an `NSData` object from APNs, you'll need to register the device with Amazon Cognito using the `registerDevice:` method of the sync client, as shown below:

```
let syncClient = AWS Cognito.default()
syncClient.registerDevice(devToken).continueWith(block: { (task: AWSTask!) -> AnyObject! in
    if (task.error != nil) {
        print("Unable to register device: " + task.error.localizedDescription)
    } else {
        print("Successfully registered device with id: \(task.result)")
    }
    return task
})
```

In debug mode, your device will register with the APNs sandbox; in release mode, it will register with APNs. To receive updates from a particular dataset, use the `subscribe` method:

```
syncClient.openOrCreateDataset("MyDataset").subscribe().continueWith(block: { (task: AWSTask!) -> AnyObject! in
    if (task.error != nil) {
        print("Unable to subscribe to dataset: " + task.error.localizedDescription)
    } else {
        print("Successfully subscribed to dataset: \(task.result)")
    }
    return task
})
```

To stop receiving push notifications from a dataset, call the `unsubscribe` method:

```
syncClient.openOrCreateDataset("MyDataset").unsubscribe().continueWith(block: { (task: AWSTask!) -> AnyObject! in
    if (task.error != nil) {
        print("Unable to unsubscribe to dataset: " + task.error.localizedDescription)
    } else {
        print("Successfully unsubscribed to dataset: \(task.result)")
    }
    return task
})
```

To subscribe to all datasets in the `AWSCognito` object, call `subscribeAll`:

```
syncClient.openOrCreateDataset("MyDataset").subscribeAll().continueWith(block: { (task: AWSTask!) -> AnyObject! in
    if (task.error != nil) {
        print("Unable to subscribe to all datasets: " + task.error.localizedDescription)
    } else {
        print("Successfully subscribed to all datasets: \(task.result)")
    }
    return task
})
```

Before calling `subscribeAll`, be sure to synchronize at least once on each dataset, so that the datasets exist on the server.

To react to push notifications, you need to implement the `didReceiveRemoteNotification` method in your app delegate:

```
func application(application: UIApplication, didReceiveRemoteNotification userInfo: [NSObject : AnyObject],
                fetchCompletionHandler completionHandler: (UIBackgroundFetchResult) -> Void) {

    NSNotificationCenter.defaultCenter().postNotificationName("CognitoPushNotification",
        object: userInfo)
}
```

If you post a notification using notification handler, you can then respond to the notification elsewhere in the application where you have a handle to your dataset. If you subscribe to the notification like this ...

```
NSNotificationCenter.defaultCenter().addObserver(observer:self,
    selector:"didReceivePushSync:",
    name:"CognitoPushNotification",
    object:nil)
```

...you can act on the notification like this:

```
func didReceivePushSync(notification: NSNotification) {
    if let data = (notification.object as! [String: AnyObject])["data"] as? [String: AnyObject] {
        let identityId = data["identityId"] as! String
        let datasetName = data["datasetName"] as! String

        if self.dataset.name == datasetName && self.identityId == identityId {
            dataset.synchronize().continueWithBlock { (task) -> AnyObject! in
                if task.error == nil {
                    print("Successfully synced dataset")
                }
                return nil
            }
        }
    }
}
```

The following keys are available in the push notification payload:

- **source**: cognito-sync. This can serve as a differentiating factor between notifications.
- **identityPoolId**: The identity pool ID. This can be used for validation or additional information, though it's not integral from the receiver's point of view.
- **identityId**: The identity ID within the pool.
- **datasetName**: The name of the dataset that was updated. This is available for the sake of the `openOrCreateDataset` call.
- **syncCount**: The sync count for the remote dataset. You can use this as a way to make sure that the local dataset is out of date, and that the incoming synchronization is new.

Amazon Cognito Streams

If you're new to Amazon Cognito Sync, use [AWS AppSync](#). Like Amazon Cognito Sync, AWS AppSync is a service for synchronizing application data across devices.

It enables user data like app preferences or game state to be synchronized. It also extends these capabilities by allowing multiple users to synchronize and collaborate in real time on shared data.

Amazon Cognito Streams gives developers control and insight into their data stored in Amazon Cognito. Developers can now configure a Kinesis stream to receive events as data is updated and synchronized. Amazon Cognito can push each dataset change to a Kinesis stream you own in real time.

Using Amazon Cognito Streams, you can move all of your Sync data to Kinesis, which can then be streamed to a data warehouse tool such as Amazon Redshift for further analysis. To learn more about Kinesis, see [Getting Started Using Amazon Kinesis](#).

Configuring streams

You can set up Amazon Cognito Streams in the Amazon Cognito console. To enable Amazon Cognito Streams in the Amazon Cognito console, you need to select the Kinesis stream to publish to and an IAM role that grants Amazon Cognito permission to put events in the selected stream.

From the [console home page](#):

1. Click the name of the identity pool for which you want to set up Amazon Cognito Streams. The **Dashboard** page for your identity pool appears.
2. In the top-right corner of the **Dashboard** page, click **Manage Identity Pools**. The Manage Federated Identities page appears.

3. Scroll down and click **Cognito Streams** to expand it.
4. In the **Stream name** dropdown menu, select the name of an existing Kinesis stream. Alternatively, click **Create stream** to create one, entering a stream name and the number of shards. To learn about shards and for help on estimating the number of shards needed for your stream, see the [Kinesis Developer Guide](#).
5. In the **Publish role** dropdown menu, select the IAM role that grants Amazon Cognito permission to publish your stream. Click **Create role** to create or modify the roles associated with your identity pool in the [AWS IAM Console](#).
6. In the **Stream status** dropdown menu, select **Enabled** to enable the stream updates. Click **Save Changes**.

After you've successfully configured Amazon Cognito streams, all subsequent updates to datasets in this identity pool will be sent to the stream.

Stream contents

Each record sent to the stream represents a single synchronization. Here is an example of a record sent to the stream:

```
{  
    "identityPoolId": "Pool Id",  
    "identityId": "Identity Id",  
    "dataSetName": "Dataset Name",  
    "operation": "(replace|remove)",  
    "kinesisSyncRecords": [  
        {  
            "key": "Key",  
            "value": "Value",  
            "syncCount": 1,  
            "lastModifiedDate": 1424801824343,  
            "deviceLastModifiedDate": 1424801824343,  
            "op": "(replace|remove)"  
        },  
        ...  
    ],  
    "lastModifiedDate": 1424801824343,  
    "kinesisSyncRecordsURL": "S3Url",  
    "payloadType": "(S3Url|Inline)",  
    "syncCount": 1  
}
```

For updates that are larger than the Kinesis maximum payload size of 1 MB, Amazon Cognito includes a presigned Amazon S3 URL that contains the full contents of the update.

After you have configured Amazon Cognito streams, if you delete the Kinesis stream or change the role trust permission so that Amazon Cognito Sync can no longer assume the role, you turn off the Amazon Cognito streams. You must either recreate the Kinesis stream or fix the role, and then you must turn on the stream again.

Bulk publishing

Once you have configured Amazon Cognito streams, you will be able to execute a bulk publish operation for the existing data in your identity pool. After you initiate a bulk publish operation, either via the console or directly via the API, Amazon Cognito will start publishing this data to the same stream that is receiving your updates.

Amazon Cognito does not guarantee uniqueness of data sent to the stream when using the bulk publish operation. You may receive the same update both as an update as well as part of a bulk publish. Keep this in mind when processing the records from your stream.

To bulk publish all of your streams, follow steps 1-6 under Configuring Streams and then click Start bulk publish. You are limited to one ongoing bulk publish operation at any given time and to one successful bulk publish request every 24 hours.

Amazon Cognito Events

If you're new to Amazon Cognito Sync, use [AWS AppSync](#). Like Amazon Cognito Sync, AWS AppSync is a service for synchronizing application data across devices.

It enables user data like app preferences or game state to be synchronized. It also extends these capabilities by allowing multiple users to synchronize and collaborate in real time on shared data.

Amazon Cognito Events allows you to execute an AWS Lambda function in response to important events in Amazon Cognito. Amazon Cognito raises the Sync Trigger event when a dataset is synchronized. You can use the Sync Trigger event to take an action when a user updates data. The function can evaluate and optionally manipulate the data before it is stored in the cloud and synchronized to the user's other devices. This is useful to validate data coming from the device before it is synchronized to the user's other devices, or to update other values in the dataset based on incoming data such as issuing an award when a player reaches a new level.

The steps below will guide you through setting up a Lambda function that executes each time a Amazon Cognito Dataset is synchronized.

Note

When using Amazon Cognito events, you can only use the credentials obtained from Amazon Cognito Identity. If you have an associated Lambda function, but you call `UpdateRecords` with AWS account credentials (developer credentials), your Lambda function will not be invoked.

Creating a function in AWS Lambda

To integrate Lambda with Amazon Cognito, you first need to create a function in Lambda. To do so:

Selecting the Lambda Function in Amazon Cognito

1. Open the Lambda console.
2. Click Create a Lambda function.
3. On the Select blueprint screen, search for and select "cognito-sync-trigger."
4. On the Configure event sources screen, leave the Event source type set to "Cognito Sync Trigger" and select your identity pool. Click Next.

Note

When configuring a Amazon Cognito Sync trigger outside of the console, you must add Lambda resource-based permissions to allow Amazon Cognito to invoke the function. You can add this permission from the Lambda console (see [Using resource-based policies for AWS Lambda](#)) or by using the Lambda `AddPermission` operation.

Example Lambda Resource-Based Policy

The following AWS Lambda resource-based policy grants Amazon Cognito a limited ability to invoke a Lambda function. Amazon Cognito can only invoke the function on behalf of the identity pool in the `aws:SourceArn` condition and the account in the `aws:SourceAccount` condition.

```
{  
    "Version": "2012-10-17",  
    "Id": "default",  
    "Statement": [  
        {  
            "Sid": "lambda-allow-cognito-my-function",  
            "Effect": "Allow",  
            "Principal": "cognito-identity.amazonaws.com",  
            "Action": "lambda:InvokeFunction",  
            "Resource": "arn:aws:lambda:us-east-1:123456789012:function:my-function"  
        }  
    ]  
}
```

```

        "Effect": "Allow",
        "Principal": {
            "Service": "cognito-sync.amazonaws.com"
        },
        "Action": "lambda:InvokeFunction",
        "Resource": "<your Lambda function ARN>",
        "Condition": {
            "StringEquals": {
                "AWS:SourceAccount": "<your account number>"
            },
            "ArnLike": {
                "AWS:SourceArn": "<your identity pool ARN>"
            }
        }
    ]
}

```

5. On the Configure function screen, enter a name and description for your function. Leave Runtime set to "Node.js." Leave the code unchanged for our example. The default example makes no changes to the data being synced. It only logs the fact that the Amazon Cognito Sync Trigger event occurred. Leave Handler name set to "index.handler." For Role, select an IAM role that grants your code permission to access AWS Lambda. To modify roles, see the IAM console. Leave Advanced settings unchanged. Click Next.
6. On the Review screen, review the details and click Create function. The next page displays your new Lambda function.

Now that you have an appropriate function written in Lambda, you need to choose that function as the handler for the Amazon Cognito Sync Trigger event. The steps below walk you through this process.

From the console home page:

1. Click the name of the identity pool for which you want to set up Amazon Cognito Events. The Dashboard page for your identity pool appears.
2. In the top-right corner of the Dashboard page, click Manage Federated Identities. The Manage Federated Identities page appears.
3. Scroll down and click Cognito Events to expand it.
4. In the Sync Trigger dropdown menu, select the Lambda function that you want to trigger when a Sync event occurs.
5. Click Save Changes.

Now, your Lambda function will be executed each time a dataset is synchronized. The next section explains how you can read and modify the data in your function as it is being synchronized.

Writing a Lambda function for sync triggers

Sync triggers follow the programming pattern that service provider interfaces use. Amazon Cognito provides input to your Lambda function in the following JSON format.

```
{
    "version": 2,
    "eventType": "SyncTrigger",
    "region": "us-east-1",
    "identityPoolId": "identityPoolId",
    "identityId": "identityId",
    "datasetName": "datasetName",
    "datasetRecords": {
        "SampleKey1": {
            "oldValue": "oldValue1",
            "newValue": "newValue1"
        }
    }
}
```

```

        "newValue": "newValue1",
        "op": "replace"
    },
    "SampleKey2": {
        "oldValue": "oldValue2",
        "newValue": "newValue2",
        "op": "replace"
    }...
}
}

```

Amazon Cognito expects the return value of the function to have the same format as the input.

When you write functions for the Sync Trigger event, observe the following:

- When Amazon Cognito calls your Lambda function during `UpdateRecords`, the function must respond within 5 seconds. If it doesn't, the Amazon Cognito Sync service generates a `LambdaSocketTimeoutException` exception. You can't increase this timeout value.
- If you get a `LambdaThrottledException` exception, try the sync operation again to update the records.
- Amazon Cognito provides all the records present in the dataset as input to the function.
- Records that the app user updates have the `op` field set as `replace`. The deleted records have `op` field set as `remove`.
- You can modify any record, even if the app user doesn't update the record.
- All the fields except the `datasetRecords` are read-only. Do not change them. If you change these fields, you can't update the records.
- To modify the value of a record, update the value and set the `op` to `replace`.
- To remove a record, either set the `op` to `remove`, or set the value to null.
- To add a record, add a new record to the `datasetRecords` array.
- Amazon Cognito ignores any omitted record in the response when Amazon Cognito updates the record.

Sample Lambda function

The following sample Lambda function shows how to access, modify and remove the data.

```

console.log('Loading function');

exports.handler = function(event, context) {
    console.log(JSON.stringify(event, null, 2));

    //Check for the event type
    if (event.eventType === 'SyncTrigger') {

        //Modify value for a key
        if('SampleKey1' in event.datasetRecords){
            event.datasetRecords.SampleKey1.newValue = 'ModifyValue1';
            event.datasetRecords.SampleKey1.op = 'replace';
        }

        //Remove a key
        if('SampleKey2' in event.datasetRecords){
            event.datasetRecords.SampleKey2.op = 'remove';
        }

        //Add a key
        if(!( 'SampleKey3' in event.datasetRecords)){
            event.datasetRecords.SampleKey3={'newValue':'ModifyValue3', 'op' : 'replace'};
        }
    }
}

```

```
    }
    context.done(null, event);
};
```

Security in Amazon Cognito

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The [shared responsibility model](#) describes this as security of the cloud and security in the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the [AWS Compliance Programs](#). To learn about the compliance programs that apply to Amazon Cognito, see [AWS Services in Scope by Compliance Program](#).
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations

This documentation helps you understand how to apply the shared responsibility model when using Amazon Cognito. It shows you how to configure Amazon Cognito to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Amazon Cognito resources.

Contents

- [Data protection in Amazon Cognito \(p. 344\)](#)
- [Identity and access management for Amazon Cognito \(p. 345\)](#)
- [Logging and monitoring in Amazon Cognito \(p. 369\)](#)
- [Compliance validation for Amazon Cognito \(p. 379\)](#)
- [Resilience in Amazon Cognito \(p. 380\)](#)
- [Infrastructure security in Amazon Cognito \(p. 380\)](#)
- [Configuration and vulnerability analysis in Amazon Cognito user pools \(p. 381\)](#)
- [Security best practices for Amazon Cognito user pools \(p. 381\)](#)
- [AWS managed policies for Amazon Cognito \(p. 398\)](#)

Data protection in Amazon Cognito

The AWS [shared responsibility model](#) applies to data protection in Amazon Cognito (Amazon Cognito). As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the [Data Privacy FAQ](#).

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM). That way each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.

- Use SSL/TLS to communicate with AWS resources.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with Amazon Cognito or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into Amazon Cognito or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

Data-encryption

Data encryption typically falls into two categories: encryption at rest and encryption in transit.

Encryption at rest

Data within Amazon Cognito is encrypted at rest in accordance with industry standards.

Encryption in transit

All requests to Amazon Cognito must be made over the Transport Layer Security protocol (TLS). Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Note

Amazon Cognito encrypts customer content internally and doesn't support customer provided keys.

Identity and access management for Amazon Cognito

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Cognito resources. IAM is an AWS service that you can use with no additional charge.

Topics

- [Audience \(p. 346\)](#)
- [Authenticating with identities \(p. 346\)](#)
- [Managing access using policies \(p. 348\)](#)
- [How Amazon Cognito works with IAM \(p. 349\)](#)
- [Identity-based policy examples for Amazon Cognito \(p. 355\)](#)
- [Troubleshooting Amazon Cognito identity and access \(p. 358\)](#)
- [Using service-linked roles for Amazon Cognito \(p. 360\)](#)

- [Authentication with a user pool \(p. 362\)](#)

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in Cognito.

Service user – If you use the Cognito service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Cognito features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Cognito, see [Troubleshooting Amazon Cognito identity and access \(p. 358\)](#).

Service administrator – If you're in charge of Cognito resources at your company, you probably have full access to Cognito. It's your job to determine which Cognito features and resources your employees should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Cognito, see [How Amazon Cognito works with IAM \(p. 349\)](#).

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Cognito. To view example Cognito identity-based policies that you can use in IAM, see [Identity-based policy examples for Amazon Cognito \(p. 355\)](#).

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. For more information about signing in using the AWS Management Console, see [Signing in to the AWS Management Console as an IAM user or root user](#) in the *IAM User Guide*.

You must be *authenticated* (signed in to AWS) as the AWS account root user, an IAM user, or by assuming an IAM role. You can also use your company's single sign-on authentication or even sign in using Google or Facebook. In these cases, your administrator previously set up identity federation using IAM roles. When you access AWS using credentials from another company, you are assuming a role indirectly.

To sign in directly to the [AWS Management Console](#), use your password with your root user email address or your IAM user name. You can access AWS programmatically using your root user or IAM users access keys. AWS provides SDK and command line tools to cryptographically sign your request using your credentials. If you don't use AWS tools, you must sign the request yourself. Do this using *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see [Signature Version 4 signing process](#) in the *AWS General Reference*.

Regardless of the authentication method that you use, you might also be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see [Using multi-factor authentication \(MFA\) in AWS](#) in the *IAM User Guide*.

AWS account root user

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the [best practice of using the root user only to create your first IAM user](#). Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

IAM users and groups

An [IAM user](#) is an identity within your AWS account that has specific permissions for a single person or application. An IAM user can have long-term credentials such as a user name and password or a set of access keys. To learn how to generate access keys, see [Managing access keys for IAM users](#) in the *IAM User Guide*. When you generate access keys for an IAM user, make sure you view and securely save the key pair. You cannot recover the secret access key in the future. Instead, you must generate a new access key pair.

An [IAM group](#) is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see [When to create an IAM user \(instead of a role\)](#) in the *IAM User Guide*.

IAM roles

An [IAM role](#) is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by [switching roles](#). You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see [Using IAM roles](#) in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Temporary IAM user permissions** – An IAM user can assume an IAM role to temporarily take on different permissions for a specific task.
- **Federated user access** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an *identity provider*. For more information about federated users, see [Federated users and roles](#) in the *IAM User Guide*.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see [How IAM roles differ from resource-based policies](#) in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
 - **Principal permissions** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see [Actions, resources, and condition keys for Amazon Cognito](#) in the *Service Authorization Reference*.
 - **Service role** – A service role is an [IAM role](#) that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see [Creating a role to delegate permissions to an AWS service](#) in the *IAM User Guide*.
 - **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear

in your IAM account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see [Using an IAM role to grant permissions to applications running on Amazon EC2 instances](#) in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see [When to create an IAM role \(instead of a user\)](#) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to IAM identities or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. You can sign in as the root user or an IAM user, or you can assume an IAM role. When you then make a request, AWS evaluates the related identity-based or resource-based policies. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see [Overview of JSON policies](#) in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

Every IAM entity (user or role) starts with no permissions. In other words, by default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see [Creating IAM policies](#) in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see [Choosing between managed policies and inline policies](#) in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must *specify a principal* in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see [Access control list \(ACL\) overview](#) in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the Principal field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see [Permissions boundaries for IAM entities](#) in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see [How SCPs work](#) in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see [Session policies](#) in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see [Policy evaluation logic](#) in the *IAM User Guide*.

How Amazon Cognito works with IAM

Before you use IAM to manage access to Cognito, learn what IAM features are available to use with Cognito.

IAM features you can use with Amazon Cognito

IAM feature	Cognito support
Identity-based policies (p. 350)	Yes

IAM feature	Cognito support
Resource-based policies (p. 350)	No
Policy actions (p. 351)	Yes
Policy resources (p. 352)	Yes
Policy condition keys (p. 353)	Yes
ACLs (p. 354)	No
ABAC (tags in policies) (p. 354)	Partial
Temporary credentials (p. 354)	Yes
Principal permissions (p. 355)	No
Service roles (p. 355)	Yes
Service-linked roles (p. 355)	Yes

To get a high-level view of how Cognito and other AWS services work with most IAM features, see [AWS services that work with IAM](#) in the *IAM User Guide*.

Identity-based policies for Cognito

Supports identity-based policies	Yes
----------------------------------	-----

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see [Creating IAM policies](#) in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see [IAM JSON policy elements reference](#) in the *IAM User Guide*.

Identity-based policy examples for Cognito

To view examples of Cognito identity-based policies, see [Identity-based policy examples for Amazon Cognito \(p. 355\)](#).

Resource-based policies within Cognito

Supports resource-based policies	No
----------------------------------	----

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must [specify a principal](#) in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see [How IAM roles differ from resource-based policies](#) in the *IAM User Guide*.

Policy actions for Cognito

Supports policy actions	Yes
-------------------------	-----

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The **Action** element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Cognito actions, see [Actions defined by Amazon Cognito](#) in the *Service Authorization Reference*.

Policy actions in Cognito use the following prefix before the action:

cognito-identity

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
    "cognito-identity:action1",
    "cognito-identity:action2"
]
```

Signed versus unsigned APIs

APIs that are signed with AWS credentials are capable of being restricted via an IAM policy. The following Cognito APIs are unsigned, and therefore cannot be restricted via an IAM policy:

Amazon Cognito federated identities

- `GetId`
- `GetOpenIdToken`
- `GetCredentialsForIdentity`
- `UnlinkIdentity`

Amazon Cognito your user pools

- `ChangePassword`
- `ConfirmDevice`

- ConfirmForgotPassword
- ConfirmSignUp
- DeleteUser
- DeleteUserAttributes
- ForgetDevice
- ForgotPassword
- GetDevice
- GetUser
- GetUserAttributeVerificationCode
- GlobalSignOut
- InitiateAuth
- ListDevices
- ResendConfirmationCode
- RespondToAuthChallenge
- SetUserSettings
- SignUp
- UpdateDeviceStatus
- UpdateUserAttributes
- VerifyUserAttribute

To view examples of Cognito identity-based policies, see [Identity-based policy examples for Amazon Cognito \(p. 355\)](#).

Policy resources for Cognito

Supports policy resources	Yes
---------------------------	-----

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The Resource JSON policy element specifies the object or objects to which the action applies. Statements must include either a Resource or a NotResource element. As a best practice, specify a resource using its [Amazon Resource Name \(ARN\)](#). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

Amazon resource names (ARNs)

ARNs for Amazon Cognito federated identities

In Amazon Cognito identity pools (federated identities), it is possible to restrict an IAM user's access to a specific identity pool, using the Amazon Resource Name (ARN) format, as in the following example. For more information about ARNs, see [IAM identifiers](#).

```
arn:aws:cognito-identity:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID
```

ARNs for Amazon Cognito Sync

In Amazon Cognito Sync, customers can also restrict access by the identity pool ID, identity ID, and dataset name.

For APIs that operate on an identity pool, the identity pool ARN format is the same as for Amazon Cognito Federated Identities, except that the service name is `cognito-sync` instead of `cognito-identity`:

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID
```

For APIs that operate on a single identity, such as `RegisterDevice`, you can refer to the individual identity by the following ARN format:

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID/identity/IDENTITY_ID
```

For APIs that operate on datasets, such as `UpdateRecords` and `ListRecords`, you can refer to the individual dataset using the following ARN format:

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID/identity/IDENTITY_ID/dataset/DATASET_NAME
```

ARNs for Amazon Cognito user pools

For Amazon Cognito Your User Pools, it is possible to restrict an IAM user's access to a specific user pool, using the following ARN format:

```
arn:aws:cognito-idp:REGION:ACCOUNT_ID:userpool/USER_POOL_ID
```

To see a list of Cognito resource types and their ARNs, see [Resources defined by Amazon Cognito](#) in the [Service Authorization Reference](#). To learn with which actions you can specify the ARN of each resource, see [Actions defined by Amazon Cognito](#).

To view examples of Cognito identity-based policies, see [Identity-based policy examples for Amazon Cognito \(p. 355\)](#).

Policy condition keys for Cognito

Supports service-specific policy condition keys	Yes
---	-----

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or *Condition block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use [condition operators](#), such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical AND operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical OR operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see [IAM policy elements: variables and tags](#) in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see [AWS global condition context keys](#) in the *IAM User Guide*.

To see a list of Cognito condition keys, see [Condition keys for Amazon Cognito](#) in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see [Actions defined by Amazon Cognito](#).

To view examples of Cognito identity-based policies, see [Identity-based policy examples for Amazon Cognito](#) (p. 355).

Access control lists (ACLs) in Cognito

Supports ACLs	No
---------------	----

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Attribute-based access control (ABAC) with Cognito

Supports ABAC (tags in policies)	Partial
----------------------------------	---------

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the [condition element](#) of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

For more information about ABAC, see [What is ABAC?](#) in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see [Use attribute-based access control \(ABAC\)](#) in the *IAM User Guide*.

Using temporary credentials with Cognito

Supports temporary credentials	Yes
--------------------------------	-----

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see [AWS services that work with IAM](#) in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see [Switching to a role \(console\)](#) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary

credentials instead of using long-term access keys. For more information, see [Temporary security credentials in IAM](#).

Cross-service principal permissions for Cognito

Supports principal permissions	No
--------------------------------	----

When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see [Actions, resources, and condition keys for Amazon Cognito](#) in the *Service Authorization Reference*.

Service roles for Cognito

Supports service roles	Yes
------------------------	-----

A service role is an [IAM role](#) that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see [Creating a role to delegate permissions to an AWS service](#) in the *IAM User Guide*.

For details about Cognito service roles, see [Enable push synchronization \(p. 248\)](#) and [Push sync \(p. 332\)](#).

Warning

Changing the permissions for a service role might break Cognito functionality. Edit service roles only when Cognito provides guidance to do so.

Service-linked roles for Cognito

Supports service-linked roles	Yes
-------------------------------	-----

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing Cognito service-linked roles, see [Using service-linked roles for Amazon Cognito \(p. 360\)](#).

Identity-based policy examples for Amazon Cognito

By default, IAM users and roles don't have permission to create or modify Cognito resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform actions on the resources that they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see [Creating IAM policies](#) in the *IAM User Guide*.

Topics

- [Policy best practices \(p. 356\)](#)

- [Using the Cognito console \(p. 356\)](#)
- [Allow users to view their own permissions \(p. 356\)](#)
- [Restricting console access to a specific identity pool \(p. 357\)](#)
- [Allowing access to specific dataset for all identities in a pool \(p. 357\)](#)

Policy best practices

Identity-based policies are very powerful. They determine whether someone can create, access, or delete Cognito resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started using AWS managed policies** – To start using Cognito quickly, use AWS managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and updated by AWS. For more information, see [Get started using permissions with AWS managed policies](#) in the *IAM User Guide*.
- **Grant least privilege** – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see [Grant least privilege](#) in the *IAM User Guide*.
- **Enable MFA for sensitive operations** – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see [Using multi-factor authentication \(MFA\) in AWS](#) in the *IAM User Guide*.
- **Use policy conditions for extra security** – To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see [IAM JSON policy elements: Condition](#) in the *IAM User Guide*.

Using the Cognito console

To access the Amazon Cognito console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Cognito resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (IAM users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that you're trying to perform.

To ensure that users and roles can still use the Cognito console, also attach the `Cognito_ConsoleAccess` or `ReadOnly` AWS managed policy to the entities. For more information, see [Adding permissions to a user](#) in the *IAM User Guide*.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "ViewOwnUserInfo",  
            "Effect": "Allow",  
            "Action": "cognito-identity:DescribeIdentity",  
            "Resource": "arn:aws:cognito-identity:us-east-1:  
                [REDACTED]:identity/[REDACTED]",  
            "Condition": {}  
        }  
    ]  
}
```

```

    "Effect": "Allow",
    "Action": [
        "iam:GetUserPolicy",
        "iam>ListGroupsForUser",
        "iam>ListAttachedUserPolicies",
        "iam>ListUserPolicies",
        "iam GetUser"
    ],
    "Resource": ["arn:aws:iam::*:user/${aws:username}"]
},
{
    "Sid": "NavigateInConsole",
    "Effect": "Allow",
    "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam GetPolicy",
        "iam>ListAttachedGroupPolicies",
        "iam>ListGroupPolicies",
        "iam>ListPolicyVersions",
        "iam>ListPolicies",
        "iam>ListUsers"
    ],
    "Resource": "*"
}
]
}

```

Restricting console access to a specific identity pool

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cognito-identity>ListIdentityPools"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cognito-identity:/*"
            ],
            "Resource": "arn:aws:cognito-identity:us-east-1:0123456789:identitypool/us-
east-1:1a1a1a1a-ffff-1111-9999-12345678"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cognito-sync:/*"
            ],
            "Resource": "arn:aws:cognito-sync:us-east-1:0123456789:identitypool/us-
east-1:1a1a1a1a-ffff-1111-9999-12345678"
        }
    ]
}

```

Allowing access to specific dataset for all identities in a pool

```
{
}
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cognito-sync>ListRecords",
      "cognito-sync>UpdateRecords"
    ],
    "Resource": "arn:aws:cognito-sync:us-east-1:0123456789:identitypool/us-east-1:1alalala-ffff-1111-9999-12345678/identity/*/dataset/UserProfile"
  }
]
```

Troubleshooting Amazon Cognito identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Cognito and IAM.

Topics

- [I am not authorized to perform an action in Cognito \(p. 358\)](#)
- [I am not authorized to perform iam:PassRole \(p. 358\)](#)
- [I want to view my access keys \(p. 359\)](#)
- [I'm an administrator and want to allow others to access Cognito \(p. 359\)](#)
- [I want to allow people outside of my AWS account to access my Cognito resources \(p. 359\)](#)

I am not authorized to perform an action in Cognito

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional `my-example-widget` resource but does not have the fictional `cognito-identity:GetWidget` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: cognito-identity:GetWidget on resource: my-example-widget
```

In this case, Mateo asks his administrator to update his policies to allow him to access the `my-example-widget` resource using the `cognito-identity:GetWidget` action.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password. Ask that person to update your policies to allow you to pass a role to Cognito.

Some AWS services allow you to pass an existing role to that service, instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Cognito. However, the action requires the service to have permissions granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

In this case, Mary asks her administrator to update her policies to allow her to perform the `iam:PassRole` action.

I want to view my access keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, `AKIAIOSFODNN7EXAMPLE`) and a secret access key (for example, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

Important

Do not provide your access keys to a third party, even to help [find your canonical user ID](#). By doing this, you might give someone permanent access to your account.

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see [Managing access keys](#) in the *IAM User Guide*.

I'm an administrator and want to allow others to access Cognito

To allow others to access Cognito, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in Cognito.

To get started right away, see [Creating your first IAM delegated user and group](#) in the *IAM User Guide*.

I want to allow people outside of my AWS account to access my Cognito resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Cognito supports these features, see [How Amazon Cognito works with IAM \(p. 349\)](#).
- To learn how to provide access to your resources across AWS accounts that you own, see [Providing access to an IAM user in another AWS account that you own](#) in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see [Providing access to AWS accounts owned by third parties](#) in the *IAM User Guide*.
- To learn how to provide access through identity federation, see [Providing access to externally authenticated users \(identity federation\)](#) in the *IAM User Guide*.

- To learn the difference between using roles and resource-based policies for cross-account access, see [How IAM roles differ from resource-based policies](#) in the *IAM User Guide*.

Using service-linked roles for Amazon Cognito

Amazon Cognito uses AWS Identity and Access Management (IAM) [service-linked roles](#). A service-linked role is a unique type of IAM role that is linked directly to Amazon Cognito. Service-linked roles are predefined by Amazon Cognito and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up Amazon Cognito easier because you don't have to manually add the necessary permissions. Amazon Cognito defines the permissions of its service-linked roles, and unless defined otherwise, only Amazon Cognito can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting their related resources. This protects your Amazon Cognito resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see [AWS Services That Work with IAM](#) and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-linked role permissions for Amazon Cognito

Amazon Cognito uses the following service-linked roles:

- **AWSServiceRoleForAmazonCognitoIdpEmailService** – Allows Amazon Cognito user pools service to use your Amazon SES identities for sending email.
- **AWSServiceRoleForAmazonCognitoIdp** – Allows Amazon Cognito user pools to publish events and configure endpoints for your Amazon Pinpoint projects.

AWSServiceRoleForAmazonCognitoIdpEmailService

The **AWSServiceRoleForAmazonCognitoIdpEmailService** service-linked role trusts the following services to assume the role:

- `email.cognito-idp.amazonaws.com`

The role permissions policy allows Amazon Cognito to complete the following actions on the specified resources:

Allowed Actions for AWSServiceRoleForAmazonCognitoIdpEmailService:

- Action: `ses:SendEmail` and `ses:SendRawEmail`
- Resource: *

The policy denies Amazon Cognito the ability to complete the following actions on the specified resources:

Denied Actions

- Action: `ses>List*`
- Resource: *

With these permissions, Amazon Cognito can use your verified email addresses in Amazon SES only to email your users. Amazon Cognito emails your users when they perform certain actions in the client app for a user pool, such as signing up or resetting a password.

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see [Service-linked role permissions](#) in the *IAM User Guide*.

AWSServiceRoleForAmazonCognitoldp

The AWSServiceRoleForAmazonCognitoldp service-linked role trusts the following services to assume the role:

- `email.cognito-idp.amazonaws.com`

The role permissions policy allows Amazon Cognito to complete the following actions on the specified resources:

Allowed Actions for AWSServiceRoleForAmazonCognitoldp

- Action: `cognito-idp:Describe`
- Resource: *

With this permission, Amazon Cognito can call `Describe` Amazon Cognito API operations for you.

Note

When you integrate Amazon Cognito with Amazon Pinpoint using `createUserPoolClient` and `updateUserPoolClient`, resource permissions will be added to the SLR as an inline policy. The inline policy will provide `mobiletargeting:UpdateEndpoint` and `mobiletargeting:PutEvents` permissions. These permissions allow Amazon Cognito to publish events and configure endpoints for Pinpoint projects you integrate with Cognito.

Creating a service-linked role for Amazon Cognito

You don't need to manually create a service-linked role. When you configure a user pool to use your Amazon SES configuration to handle email delivery in the AWS Management Console, the AWS CLI, or the Amazon Cognito API, Amazon Cognito creates the service-linked role for you.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you configure a user pool to use your Amazon SES configuration to handle email delivery, Amazon Cognito creates the service-linked role for you again.

Before Amazon Cognito can create this role, the IAM permissions that you use to set up your user pool must include the `iam:CreateServiceLinkedRole` action. For more information about updating permissions in IAM, see [Changing Permissions for an IAM User](#) in the *IAM User Guide*.

Editing a service-linked role for Amazon Cognito

You can't edit the `AmazonCognitoldp` or `AmazonCognitoldpEmailService` service-linked roles in AWS Identity and Access Management. After you create a service-linked role, you can't change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see [Editing a service-linked role](#) in the *IAM User Guide*.

Deleting a service-linked role for Amazon Cognito

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. If you delete the role, you only retain entities that Amazon Cognito actively

monitors or maintains. Before you can delete AmazonCognitoldp or AmazonCognitoldpEmailService service-linked roles, you must do one of the following for each user pool that uses the role:

- Delete the user pool.
- Update the email settings in the user pool to use the default email functionality. The default setting doesn't use the service-linked role.

Remember to perform the action in each AWS Region with a user pool that uses the role.

Note

If the Amazon Cognito service is using the role when you try to delete the resources, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

To delete an Amazon Cognito user pool

1. Sign in to the AWS Management Console and open the Amazon Cognito console at <https://console.aws.amazon.com/cognito>.
2. Choose **Manage User Pools**.
3. On the **Your User Pools** page, choose the user pool that you want to delete.
4. Choose **Delete pool**.
5. In the **Delete user pool** window, type **delete**, and choose **Delete pool**.

To update an Amazon Cognito user pool to use the default email functionality

1. Sign in to the AWS Management Console and open the Amazon Cognito console at <https://console.aws.amazon.com/cognito>.
2. Choose **Manage User Pools**.
3. On the **Your User Pools** page, choose the user pool that you want to update.
4. In the navigation menu on the left, choose **Message customizations**.
5. Under **Do you want to send emails through your Amazon SES Configuration?**, choose **No - Use Cognito (Default)**.
6. When you finish setting your email account options, choose **Save changes**.

To manually delete the service-linked role using IAM

Use the IAM console, the AWS CLI, or the AWS API to delete AmazonCognitoldp or AmazonCognitoldpEmailService service-linked roles. For more information, see [Deleting a service-linked role](#) in the *IAM User Guide*.

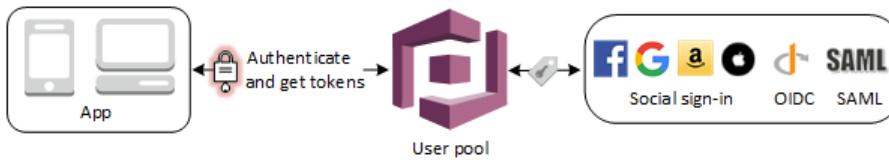
Supported Regions for Amazon Cognito service-linked roles

Amazon Cognito supports service-linked roles in all AWS Regions where the service is available. For more information, see [AWS Regions and Endpoints](#).

Authentication with a user pool

Your app users can sign in either directly through a user pool, or federate through a third-party identity provider (IdP). The user pool manages the overhead of handling the tokens that are returned from social sign-in through Facebook, Google, Amazon, and Apple, and from OpenID Connect (OIDC) and SAML IdPs.

After successful authentication, Amazon Cognito returns user pool tokens to your app. You can use the tokens to grant your users access to your own server-side resources, or to the Amazon API Gateway. Or, you can exchange them for AWS credentials to access other AWS services.



User pool token handling and management for your web or mobile app is provided on the client side through Amazon Cognito SDKs. Likewise, the Mobile SDK for iOS and the Mobile SDK for Android automatically refresh your ID and access tokens if there is a valid (non-expired) refresh token present, and the ID and access tokens have a minimum remaining validity of 5 minutes. For information on the SDKs, and sample code for JavaScript, Android, and iOS see [Amazon Cognito user pool SDKs](#).

After your app user successfully signs in, Amazon Cognito creates a session and returns an ID, access, and refresh token for the authenticated user.

JavaScript

```

// Amazon Cognito creates a session which includes the id, access, and refresh tokens
// of an authenticated user.

var authenticationData = {
    Username : 'username',
    Password : 'password',
};

var authenticationDetails = new
AmazonCognitoIdentity.AuthenticationDetails(authenticationData);
var poolData = { UserPoolId : 'us-east-1_ExaMPle',
    ClientId : 'lexample23456789'
};
var userPool = new AmazonCognitoIdentity.CognitoUserPool(poolData);
var userData = {
    Username : 'username',
    Pool : userPool
};
var cognitoUser = new AmazonCognitoIdentity.CognitoUser(userData);
cognitoUser.authenticateUser(authenticationDetails, {
    onSuccess: function (result) {
        var accessToken = result.getAccessToken().getJwtToken();

        /* Use the idToken for Logins Map when Federating User Pools with identity
        pools or when passing through an Authorization Header to an API Gateway Authorizer */
        var idToken = result.idToken.jwtToken;
    },
    onFailure: function(err) {
        alert(err);
    },
});

```

Android

```

// Session is an object of the type CognitoUserSession, and includes the id, access,
// and refresh tokens for a user.

String idToken = session.getIdToken().getJWTToken();
String accessToken = session.getAccessToken().getJWT();

```

iOS - swift

```
// AWSIdentityUserSession includes id, access, and refresh tokens for a user.
```

```
- (AWSTask<AWSCognitoIdentityUserSession *> *)getSession;
```

iOS - objective-C

```
// AWSCognitoIdentityUserSession includes the id, access, and refresh tokens for a user.

[[user getSession:@"username" password:@"password" validationData:nil scopes:nil]
 continueWithSuccessBlock:^id _Nullable(AWSTask<AWSCognitoIdentityUserSession *> * _Nonnull task) {
    // success, task.result has user session
    return nil;
}];
```

Topics

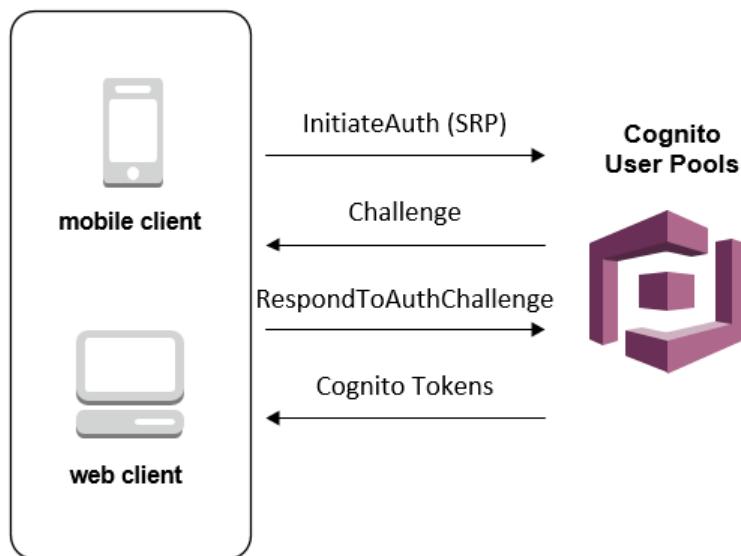
- [User pool authentication flow \(p. 364\)](#)

User pool authentication flow

To verify the identity of users, modern authentication flows incorporate new challenge types, in addition to passwords. Amazon Cognito authentication typically requires that you implement two API operations in the following order:

1. `InitiateAuth`
2. `RespondToAuthChallenge`

A user authenticates by answering successive challenges until authentication either fails or Amazon Cognito issues tokens to the user. You can repeat these steps with Amazon Cognito, in a process that includes different challenges, to support any custom authentication flow.



You can use AWS Lambda triggers to customize the way users authenticate. These triggers issue and verify their own challenges as part of the authentication flow.

You can also use the admin authentication flow for secure backend servers. You can use the user migration authentication flow to make user migration possible without the requirement that your users to reset their passwords.

Note

Users can attempt but fail to sign in correctly five times before Amazon Cognito temporarily locks them out. Lockout time starts at one second and increases exponentially, doubling after each subsequent failed attempt, up to about 15 minutes. Amazon Cognito ignores attempts to log in during a temporary lockout period, and these attempts don't initiate a new lockout period. After a user waits 15 minutes, Amazon Cognito resets the temporary lockout. This behavior is subject to change.

Topics

- [Client-side authentication flow \(p. 365\)](#)
- [Server-side authentication flow \(p. 365\)](#)
- [Custom authentication flow \(p. 366\)](#)
- [Built-in authentication flow and challenges \(p. 366\)](#)
- [Custom authentication flow and challenges \(p. 367\)](#)
- [Use SRP password verification in custom authentication flow \(p. 367\)](#)
- [Admin authentication flow \(p. 368\)](#)
- [User migration authentication flow \(p. 368\)](#)

Client-side authentication flow

The following process works for user client-side apps that you create with the [AWS Mobile SDK for Android](#), [AWS Mobile SDK for iOS](#), or [AWS SDK for JavaScript](#):

1. The user enters their user name and password into the app.
2. The app calls the `InitiateAuth` operation with the user's user name and Secure Remote Password (SRP) details.

This API operation returns the authentication parameters.

Note

The app generates SRP details with the Amazon Cognito SRP features that are built in to AWS SDKs.

3. The app calls the `RespondToAuthChallenge` operation. If the call succeeds, Amazon Cognito returns the user's tokens, and the authentication flow is complete.

If Amazon Cognito requires another challenge, the call to `RespondToAuthChallenge` returns no tokens. Instead, the call returns a session.

4. If `RespondToAuthChallenge` returns a session, the app calls `RespondToAuthChallenge` again, this time with the session and the challenge response (for example, MFA code).

Server-side authentication flow

If you don't have a user app, but instead you use a Java, Ruby, or Node.js secure backend or server-side app, you can use the authenticated server-side API for Amazon Cognito user pools.

For server-side apps, user pool authentication is similar to authentication for client-side apps, except for the following:

- The server-side app calls the `AdminInitiateAuth` API operation (instead of `InitiateAuth`). This operation requires AWS credentials with permissions that include cognito-

`idp:AdminInitiateAuth` and `cognito-idp:AdminRespondToAuthChallenge`. The operation returns the required authentication parameters.

- After the server-side app has the authentication parameters, it calls the `AdminRespondToAuthChallenge` API operation (instead of `RespondToAuthChallenge`). The `AdminRespondToAuthChallenge` API operation only succeeds when you provide AWS credentials.

For more information about signing Amazon Cognito API requests with AWS credentials, see [Signature Version 4 signing process](#) in the *AWS General Reference*.

The `AdminInitiateAuth` and `AdminRespondToAuthChallenge` API operations can't accept username-and-password user credentials for admin sign-in, unless you explicitly enable them to do so in one of the following ways:

- Include `ALLOW_ADMIN_USER_PASSWORD_AUTH` (formerly known as `ADMIN_NO_SR_P_AUTH`) in the `ExplicitAuthFlow` parameter when you call `CreateUserPoolClient` or `UpdateUserPoolClient`.
- Add `ALLOW_ADMIN_USER_PASSWORD_AUTH` to the list of **Authentication flows** for your app client. Configure app clients on the **App integration** tab in your user pool, under **App clients and analytics**. For more information, see [Configuring a user pool app client \(p. 223\)](#).

Custom authentication flow

Amazon Cognito user pools also make it possible to use custom authentication flows, which can help you create a challenge/response-based authentication model using AWS Lambda triggers.

Note

You can't use advanced security features with custom authentication flows. For more information, see [Adding advanced security to a user pool \(p. 387\)](#).

The custom authentication flow makes possible customized challenge and response cycles to meet different requirements. The flow starts with a call to the `InitiateAuth` API operation that indicates the type of authentication to use and provides any initial authentication parameters. Amazon Cognito responds to the `InitiateAuth` call with one of the following types of information:

- A challenge for the user, along with a session and parameters.
- An error if the user fails to authenticate.
- ID, access, and refresh tokens if the supplied parameters in the `InitiateAuth` call are sufficient to sign the user in. (Typically the user or app must first answer a challenge, but your custom code must determine this.)

If Amazon Cognito responds to the `InitiateAuth` call with a challenge, the app gathers more input and calls the `RespondToAuthChallenge` operation. This call provides the challenge responses and passes it back the session. Amazon Cognito responds to the `RespondToAuthChallenge` call similarly to the `InitiateAuth` call. If the user has signed in, Amazon Cognito provides tokens, or if the user isn't signed in, Amazon Cognito provides another challenge, or an error. If Amazon Cognito returns another challenge, the sequence repeats and the app calls `RespondToAuthChallenge` until the user successfully signs in or an error is returned. For more details about the `InitiateAuth` and `RespondToAuthChallenge` API operations, see the [API documentation](#).

Built-in authentication flow and challenges

Amazon Cognito contains built-in `AuthFlow` and `ChallengeName` values so that a standard authentication flow can validate a user name and password through the Secure Remote Password (SRP) protocol. The AWS SDKs have built-in support for these flows with Amazon Cognito.

The flow starts by sending `USER_SR_P_AUTH` as the `AuthFlow` to `InitiateAuth`. You also send `USERNAME` and `SRP_A` values in `AuthParameters`. If the `InitiateAuth` call is successful, the response has included `PASSWORD_VERIFIER` as the `ChallengeName` and `SRP_B` in the challenge parameters. The app then calls `RespondToAuthChallenge` with the `PASSWORD_VERIFIER` `ChallengeName` and the necessary parameters in `ChallengeResponses`. If the call to `RespondToAuthChallenge` is successful and the user signs in, Amazon Cognito issues tokens. If you activated multi-factor authentication (MFA) for the user, Amazon Cognito returns the `ChallengeName` of `SMS_MFA`. The app can provide the necessary code through another call to `RespondToAuthChallenge`.

Custom authentication flow and challenges

An app can initiate a custom authentication flow by calling `InitiateAuth` with `CUSTOM_AUTH` as the `Authflow`. With a custom authentication flow, three Lambda triggers control challenges and verification of the responses.

- The `DefineAuthChallenge` Lambda trigger uses a session array of previous challenges and responses as input. It then generates the next challenge name and Booleans that indicate whether the user is authenticated and can be granted tokens. This Lambda trigger is a state machine that controls the user's path through the challenges.
- The `CreateAuthChallenge` Lambda trigger takes a challenge name as input and generates the challenge and parameters to evaluate the response. When `DefineAuthChallenge` returns `CUSTOM_CHALLENGE` as the next challenge, the authentication flow calls `CreateAuthChallenge`. The `CreateAuthChallenge` Lambda trigger passes the next type of challenge in the challenge metadata parameter.
- The `VerifyAuthChallengeResponse` Lambda function evaluates the response and returns a Boolean to indicate if the response was valid.

A custom authentication flow can also use a combination of built-in challenges, such as SRP password verification and MFA through SMS. It can use custom challenges such as CAPTCHA or secret questions.

Use SRP password verification in custom authentication flow

If you want to include SRP in a custom authentication flow, you must begin with SRP.

- To initiate SRP password verification in a custom flow, the app calls `InitiateAuth` with `CUSTOM_AUTH` as the `Authflow`. In the `AuthParameters` map, the request from your app includes `SRP_A`: (the SRP A value) and `CHALLENGE_NAME`: `SRP_A`.
- The `CUSTOM_AUTH` flow invokes the `DefineAuthChallenge` Lambda trigger with an initial session of `challengeName`: `SRP_A` and `challengeResult`: `true`. Your Lambda function responds with `challengeName`: `PASSWORD_VERIFIER`, `issueTokens`: `false`, and `failAuthentication`: `false`.
- The app next must call `RespondToAuthChallenge` with `challengeName`: `PASSWORD_VERIFIER` and the other parameters required for SRP in the `challengeResponses` map.
- If Amazon Cognito verifies the password, `RespondToAuthChallenge` invokes the `DefineAuthChallenge` Lambda trigger with a second session of `challengeName`: `PASSWORD_VERIFIER` and `challengeResult`: `true`. At that point, the `DefineAuthChallenge` Lambda trigger responds with `challengeName`: `CUSTOM_CHALLENGE` to start the custom challenge.
- If MFA is enabled for a user, after Amazon Cognito verifies the password, your user is then challenged to set up or sign in with MFA.

Note

The Amazon Cognito hosted sign-in webpage can't activate [Custom authentication challenge Lambda triggers \(p. 114\)](#).

For more information about the Lambda triggers, including sample code, see [Customizing user pool workflows with Lambda triggers \(p. 92\)](#).

Admin authentication flow

Best practice for authentication is to use the API operations described in [Custom authentication flow \(p. 366\)](#) with SRP for password verification. The AWS SDKs use that approach, and this approach helps them to use SRP. However, if you want to avoid SRP calculations, an alternative set of admin API operations is available for secure backend servers. For these backend admin implementations, use AdminInitiateAuth in place of InitiateAuth. Also, use AdminRespondToAuthChallenge in place of RespondToAuthChallenge. Because you can submit the password as plaintext, you do not have to do SRP calculations when you use these operations. Here is an example:

```
AdminInitiateAuth Request {
    "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
    "AuthParameters": {
        "USERNAME": "<username>",
        "PASSWORD": "<password>"
    },
    "ClientId": "<clientId>",
    "UserPoolId": "<userPoolId>"
}
```

These admin authentication operations require developer credentials and use the AWS Signature Version 4 (SigV4) signing process. These operations are available in standard AWS SDKs, including Node.js, which is convenient for Lambda functions. To use these operations and have them accept passwords in plaintext, you must activate them for the app in the console. Alternatively, you can pass ADMIN_USER_PASSWORD_AUTH for the ExplicitAuthFlow parameter in calls to CreateUserPoolClient or UpdateUserPoolClient. The InitiateAuth and RespondToAuthChallenge operations do not accept the ADMIN_USER_PASSWORD_AUTH AuthFlow.

In the AdminInitiateAuth response ChallengeParameters, the USER_ID_FOR_SRP attribute, if present, contains the user's actual user name, not an alias (such as email address or phone number). In your call to AdminRespondToAuthChallenge, in the ChallengeResponses, you must pass this user name in the USERNAME parameter.

Note

Because backend admin implementations use the admin authentication flow, the flow doesn't support device tracking. When you have turned on device tracking, admin authentication succeeds, but any call to refresh the access token fails.

User migration authentication flow

A user migration Lambda trigger helps migrate users from a legacy user management system into your user pool. If you choose the USER_PASSWORD_AUTH authentication flow, users don't have to reset their passwords during user migration. This flow sends your users' passwords to the service over an encrypted SSL connection during authentication.

When you have migrated all your users, switch flows to the more secure SRP flow. The SRP flow doesn't send any passwords over the network.

To learn more about Lambda triggers, see [Customizing user pool workflows with Lambda triggers \(p. 92\)](#).

For more information about migrating users with a Lambda trigger, see [Importing users into user pools with a user migration Lambda trigger \(p. 170\)](#).

Logging and monitoring in Amazon Cognito

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon Cognito and your other AWS solutions. Amazon Cognito currently supports the following two AWS services so that you can monitor your organization and the activity that happens within it.

- **AWS CloudTrail** – With CloudTrail you can capture API calls from the Amazon Cognito console and from code calls to the Amazon Cognito API operations. For example, when a user authenticates, CloudTrail can record details such as the IP address in the request, who made the request, and when it was made.
- **Amazon CloudWatch Metrics** – With CloudWatch metrics you can monitor, report and take automatic actions in case of an event in near real time. For example, you can create CloudWatch dashboards on the provided metrics to monitor your Amazon Cognito user pools, or you can create CloudWatch alarms on the provided metrics to notify you on breach of a set threshold.
- **Amazon CloudWatch Logs Insights** - With CloudWatch Logs Insights, you can configure CloudTrail to send events to CloudWatch for monitoring Amazon Cognito CloudTrail log files.

Topics

- [Tracking quotas and usage in CloudWatch and Service Quotas \(p. 369\)](#)
- [Metrics for Amazon Cognito user pools \(p. 370\)](#)
- [Dimensions for Amazon Cognito user pools \(p. 374\)](#)
- [Use the Service Quotas console to track metrics \(p. 375\)](#)
- [Use the CloudWatch console to track metrics \(p. 375\)](#)
- [Create a CloudWatch alarm for a quota \(p. 376\)](#)
- [Logging Amazon Cognito API calls with AWS CloudTrail \(p. 376\)](#)
- [Analyzing Amazon Cognito CloudTrail events with Amazon CloudWatch Logs Insights \(p. 378\)](#)

Tracking quotas and usage in CloudWatch and Service Quotas

You can monitor Amazon Cognito user pools using Amazon CloudWatch or using Service Quotas. CloudWatch collects raw data and processes it into readable, near-real-time metrics. In CloudWatch, you can set alarms that watch for certain thresholds and send notifications or take actions when those thresholds are met. To create a CloudWatch alarm for a service quota, see [Create a CloudWatch alarm](#). Amazon Cognito metrics are available at five minute intervals. For more information about retention periods in CloudWatch, visit the [Amazon CloudWatch FAQ page](#).

You can use Service Quotas to view and manage your Amazon Cognito user pools quota usage. The Service Quotas console has three features, view service quotas, request a service quota increase, and view current utilization. You can use the first feature to view quotas and see whether the quota is adjustable. You can use the second feature to request a Service Quotas increase. You can use the last feature to view quota utilization. This feature is only available after your account has been active for awhile. For more information on viewing quotas in the Service Quotas console, see [Viewing Service Quotas](#).

Note

Amazon Cognito metrics are available at 5 minute intervals. For more information about retention periods in CloudWatch, visit the [Amazon CloudWatch FAQ page](#).

Metrics for Amazon Cognito user pools

The following table lists the metrics available for Amazon Cognito user pools.

Note

Metrics that haven't had any new data points in the past two weeks don't appear in the console. They also don't appear when you enter their metric name or dimension names in the search box in the **All metrics** tab in the console. In addition, they are not returned in the results of a `list-metrics` command. The best way to retrieve these metrics is with the `get-metric-data` or `get-metric-statistics` commands in the AWS CLI.

Metric	Description
<code>SignUpSuccesses</code>	<p>Provides the total number of successful user registration requests made to the Amazon Cognito user pool. A successful user registration request produces a value of 1, whereas an unsuccessful request produces a value of 0. A throttled request is also considered as an unsuccessful request, and hence a throttled request will also produce a count of 0.</p> <p>To find the percentage of successful user registration requests, use the <code>Average</code> statistic on this metric. To count the total number of user registration requests, use the <code>Sample Count</code> statistic on this metric. To count the total number of successful user registration requests, use the <code>Sum</code> statistic on this metric. To count the total number of failed user registration requests, use the CloudWatch Math expression and subtract the <code>Sum</code> statistic from the <code>Sample Count</code> statistic.</p> <p>This metric is published for each user pool for each user pool client. In case when the user registration is performed by an admin, the metric is published with the user pool client as <code>Admin</code>.</p> <p>Note that this metric is not emitted for User import and User migration cases.</p> <p>Metric dimension: <code>UserPool</code>, <code>UserPoolClient</code></p> <p>Units: Count</p>
<code>SignUpThrottles</code>	<p>Provides the total number of throttled user registration requests made to the Amazon Cognito user pool. A count of 1 is published whenever a user registration request is throttled.</p> <p>To count the total number of throttled user registration requests, use the <code>Sum</code> statistic for this metric.</p> <p>This metric is published for each user pool for each client. In case when the request that was throttled was made by an administrator, the metric is published with user pool client as <code>Admin</code>.</p>

Metric	Description
	<p>Metric dimension: <code>UserPool</code>, <code>UserPoolClient</code> Units: Count</p>
<code>SignInSuccesses</code>	<p>Provides the total number of successful user authentication requests made to the Amazon Cognito user pool. A user authentication is considered successful when authentication token is issued to the user. A successful authentication produces a value of 1, whereas an unsuccessful request produces a value of 0. A throttled request is also considered as an unsuccessful request, and hence a throttled request will also produce a count of 0.</p> <p>To find the percentage of successful user authentication requests, use the <code>Average</code> statistic on this metric. To count the total number of user authentication requests, use the <code>Sample Count</code> statistic on this metric. To count the total number of successful user authentication requests, use the <code>Sum</code> statistic on this metric. To count the total number of failed user authentication requests, use the CloudWatch Math expression and subtract the <code>Sum</code> statistic from the <code>Sample Count</code> statistic.</p> <p>This metric is published for each user pool for each client. In case an invalid user pool client is provided with a request, the corresponding user pool client value in the metric contains a fixed value <code>Invalid</code> instead of the actual invalid value sent in the request.</p> <p>Note that requests to refresh the Amazon Cognito token is not included in this metric. There is a separate metric for providing <code>Refresh token</code> statistics.</p> <p>Metric dimension: <code>UserPool</code>, <code>UserPoolClient</code> Units: Count</p>

Metric	Description
<code>SignInThrottles</code>	<p>Provides the total number of throttled user authentication requests made to the Amazon Cognito user pool. A count of 1 is published whenever an authentication request is throttled.</p> <p>To count the total number of throttled user authentication requests, use the <code>Sum</code> statistic for this metric.</p> <p>This metric is published for each user pool for each client. In case an invalid user pool client is provided with a request, the corresponding user pool client value in the metric contains a fixed value <code>Invalid</code> instead of the actual invalid value sent in the request.</p> <p>Requests to refresh Amazon Cognito token is not included in this metric. There is a separate metric for providing Refresh token statistics.</p> <p>Metric dimension: <code>UserPool</code>, <code>UserPoolClient</code></p> <p>Units: Count</p>
<code>TokenRefreshSuccesses</code>	<p>Provides the total number of successful requests to refresh an Amazon Cognito token that were made to the Amazon Cognito user pool. A successful refresh Amazon Cognito token request produces a value of 1, whereas an unsuccessful request produces a value of 0. A throttled request is also considered as an unsuccessful request, and hence a throttled request will also produce a count of 0.</p> <p>To find the percentage of successful requests to refresh an Amazon Cognito token, use the <code>Average</code> statistic on this metric. To count the total number of requests to refresh an Amazon Cognito token, use the <code>Sample Count</code> statistic on this metric. To count the total number of successful requests to refresh an Amazon Cognito token, use the <code>Sum</code> statistic on this metric. To count the total number of failed requests to refresh an Amazon Cognito token, use the CloudWatch Math expression and subtract the <code>Sum</code> statistic from the <code>Sample Count</code> statistic.</p> <p>This metric is published per each user pool client. If an invalid user pool client is in a request, the user pool client value contains a fixed value of <code>Invalid</code>.</p> <p>Metric dimension: <code>UserPool</code>, <code>UserPoolClient</code></p> <p>Units: Count</p>

Metric	Description
<code>TokenRefreshThrottles</code>	<p>Provides the total number of throttled requests to refresh an Amazon Cognito token that were made to the Amazon Cognito user pool. A count of 1 is published whenever a refresh Amazon Cognito token request is throttled.</p> <p>To count the total number of throttled requests to refresh an Amazon Cognito token, use the <code>Sum</code> statistic for this metric.</p> <p>This metric is published for each user pool for each client. In case an invalid user pool client is provided with a request, corresponding user pool client value in the metric contains a fixed value <code>Invalid</code> instead of the actual invalid value sent in the request.</p> <p>Metric dimension: <code>UserPool</code>, <code>UserPoolClient</code></p> <p>Units: Count</p>
<code>FederationSuccesses</code>	<p>Provides the total number of successful identity federation requests to the Amazon Cognito user pool. A successful identity federation request produces a value of 1, whereas an unsuccessful request produces a value of 0. A throttled request is also considered as an unsuccessful request, and hence a throttled request will also produce a count of 0.</p> <p>To find the percentage of successful identity federation requests, use the <code>Average</code> statistic on this metric. To count the total number of identity federation requests, use the <code>Sample Count</code> statistic on this metric. To count the total number of successful identity federation requests, use the <code>Sum</code> statistic on this metric. To count the total number of failed identity federation requests, use the CloudWatch Math expression and subtract the <code>Sum</code> statistic from the <code>Sample Count</code> statistic.</p> <p>Metric dimension: <code>UserPool</code>, <code>UserPoolClient</code>, <code>IdentityProvider</code></p> <p>Units: Count</p>

Metric	Description
FederationThrottles	<p>Provides the total number of throttled identity federation requests to the Amazon Cognito user pool. A count of 1 is published whenever an identity federation request is throttled.</p> <p>To count the total number of throttled identity federation requests, use the <code>Sum</code> statistic for this metric.</p> <p>Metric dimension: <code>UserPool</code>, <code>UserPoolClient</code>, <code>IdentityProvider</code></p> <p>Units: Count</p>
CallCount	<p>Provides the total number of calls customers made related to a category. This metric includes all the calls, such as throttled calls, failed calls, and successful calls.</p> <p>This metric is available in the Usage nameSpace.</p> <p>The category quota is enforced for each AWS account across all user pools in an account and Region.</p> <p>You can count the total number of calls in a category using the <code>Sum</code> statistic for this metric.</p> <p>Metric dimension: Service, Type, Resource, Class</p> <p>Units: Count</p>
ThrottleCount	<p>Provides the total number of throttled calls related to a category.</p> <p>This metric is available in the Usage nameSpace.</p> <p>This metric is published at the account level.</p> <p>You can count the total number of calls in a category, using the <code>Sum</code> statistic for this metric.</p> <p>Metric dimension: Service, Type, Resource, Class</p> <p>Units: Count</p>

Dimensions for Amazon Cognito user pools

The following dimensions are used to refine the usage metrics that are published by Amazon Cognito. The dimensions only apply to `CallCount` and `ThrottleCount` metrics.

Dimension	Description
Service	The name of the AWS service containing the resource. For Amazon Cognito usage metrics, the value for this dimension is <code>Cognito user pool</code> .
Type	The type of entity that is being reported. The only valid value for Amazon Cognito usage metrics is <code>API</code> .
Resource	The type of resource that is running. The only valid value is category name.
Class	The class of resource being tracked. Amazon Cognito doesn't use the class dimension.

Use the Service Quotas console to track metrics

Amazon Cognito user pools is integrated with Service Quotas, which is a service that enables you to view and manage your quotas from a central location. You can use the Service Quotas console to view details about a specific quota, monitor quota utilization, and request a quota increase. For some quota types, you can create a CloudWatch alarm to track your quota utilization. To learn more about what Amazon Cognito metrics you can track, see [Track quota usage \(p. 412\)](#).

To view Amazon Cognito user pools service quotas utilization, complete the following steps.

1. Open the [Service Quotas console](#).
2. In the navigation pane, choose **AWS services**.
3. From the **AWS services** list, enter Amazon Cognito user pools in the search field. The service quota page appears.
4. Select a quota that supports CloudWatch monitoring. For example, choose `Rate of UserAuthentication requests`.
5. Scroll down to **Monitoring**. This section appears only for quotas that support CloudWatch monitoring.
6. In **Monitoring** you can view current service quota utilization in the graph.
7. In **Monitoring** select either one hour, three hours, twelve hours, one day, three days, or one week.
8. Select any area inside of the graph to view the service quota utilization percentage. From here, you can add the graph to your dashboard or use the action menu to select **View in metrics**, which will take you to the related metrics in the CloudWatch console.

Use the CloudWatch console to track metrics

You can track and collect Amazon Cognito user pools metrics using CloudWatch. The CloudWatch dashboard will display metrics about every AWS service you use. You can use CloudWatch to create metric alarms. The alarms can be setup to send you notifications or make a change to a specific resource that you are monitoring. To view service quota metrics in CloudWatch, complete the following steps.

1. Open the [CloudWatch console](#).
2. In the navigation pane, choose **Metrics**.
3. In **All metrics** select a metric and a dimension.
4. Select the check box next to a metric. The metrics will appear in the graph.

Note

Metrics that haven't had any new data points in the past two weeks don't appear in the console. They also don't appear when you enter their metric name or dimension names in the search box in the All metrics tab in the console, and they are not returned in the results of a list-metrics command. The best way to retrieve these metrics is with the get-metric-data or get-metric-statistics commands in the AWS CLI.

Create a CloudWatch alarm for a quota

Amazon Cognito provides CloudWatch usage metrics that correspond to the AWS service quotas for `CallCount` and `ThrottleCount` APIs. For more information about tracking usage in CloudWatch, see [Track quota usage \(p. 412\)](#).

In the Service Quotas console, you can create alarms that alert you when your usage approaches a service quota. To learn how to set up a CloudWatch alarm using the Service Quotas console, see [Service Quotas and CloudWatch alarms](#).

Logging Amazon Cognito API calls with AWS CloudTrail

Amazon Cognito is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Amazon Cognito. CloudTrail captures a subset of API calls for Amazon Cognito as events, including calls from the Amazon Cognito console and from code calls to the Amazon Cognito API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon Cognito. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to Amazon Cognito, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, including how to configure and enable it, see the [AWS CloudTrail User Guide](#).

You can also create Amazon CloudWatch alarms for specific CloudTrail events. For example, you can set up CloudWatch to trigger an alarm if an identity pool configuration is changed. For more information, see [Creating CloudWatch alarms for CloudTrail events: Examples](#).

Amazon Cognito information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When supported event activity occurs in Amazon Cognito, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see [Viewing events with CloudTrail event history](#).

For an ongoing record of events in your AWS account, including events for Amazon Cognito, create a trail. A trail enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see:

- [Overview for creating a trail](#)
- [CloudTrail supported services and integrations](#)
- [Configuring amazon SNS notifications for CloudTrail](#)
- [Receiving CloudTrail log files from multiple regions](#) and [Receiving CloudTrail log files from multiple accounts](#)

Amazon Cognito User Pools

Amazon Cognito supports logging for all of the actions listed on the [User pool actions](#) page as events in CloudTrail log files. Amazon Cognito records `UserSub` but not `UserName` in CloudTrail logs for requests that are specific to a user. You can find a user for a given `UserSub` by calling the `ListUsers` API, and using a filter for sub.

Note

Hosted UI and federation calls are currently not included in CloudTrail.

Amazon Cognito Federated Identities

- [CreateIdentityPool](#)
- [DeleteIdentityPool](#)
- [DescribeIdentityPool](#)
- [GetIdentityPoolRoles](#)
- [ListIdentityPools](#)
- [SetIdentityPoolRoles](#)
- [UpdateIdentityPool](#)

Amazon Cognito Sync

Amazon Cognito supports logging for all of the actions listed on the [Amazon Cognito Sync actions](#) page as events in CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or IAM user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the [CloudTrail userIdentity element](#).

Example: Amazon Cognito log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files are not an ordered stack trace of the public API calls, so they do not appear in any specific order.

The following example is a log entry for a request for the `CreateIdentityPool` action. The request was made by an IAM user named Alice.

```
{  
    "eventVersion": "1.03",  
    "userIdentity": {  
        "type": "IAMUser",  
        "principalId": "PRINCIPAL_ID",  
        "arn": "arn:aws:iam::123456789012:user/Alice",  
        "accountId": "123456789012",  
        "accessKeyId": "[ 'EXAMPLE_KEY_ID' ]",  
        "userName": "Alice"  
    }  
}
```

```
        },
        "eventTime": "2016-01-07T02:04:30Z",
        "eventSource": "cognito-identity.amazonaws.com",
        "eventName": "CreateIdentityPool",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "USER_AGENT",
        "requestParameters": {
            "identityPoolName": "TestPool",
            "allowUnauthenticatedIdentities": true,
            "supportedLoginProviders": {
                "graph.facebook.com": "0000000000000000"
            }
        },
        "responseElements": {
            "identityPoolName": "TestPool",
            "identityPoolId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE",
            "allowUnauthenticatedIdentities": true,
            "supportedLoginProviders": {
                "graph.facebook.com": "0000000000000000"
            }
        },
        "requestID": "15cc73a1-0780-460c-91e8-e12ef034e116",
        "eventID": "f1d47f93-c708-495b-bff1-cb935a6064b2",
        "eventType": "AwsApiCall",
        "recipientAccountId": "123456789012"
    }
}
```

Analyzing Amazon Cognito CloudTrail events with Amazon CloudWatch Logs Insights

Amazon CloudWatch Logs Insights enables you to interactively search and analyze your Amazon Cognito CloudTrail events data. When you configure your trail to send events to CloudWatch Logs, CloudTrail sends only the events that match your trail settings.

To query or research your Amazon Cognito CloudTrail events, in the CloudTrail console, make sure that you select the **Management events** option in your trail settings so that you can monitor the management operations performed on your AWS resources. You can optionally select the **Insights events** option in your trail settings when you want to identify errors, unusual activity or unusual user behavior in your account.

Sample Amazon Cognito queries

You can use the following queries in the Amazon CloudWatch console.

General queries

Find the 25 most recently added log events.

```
fields @timestamp, @message | sort @timestamp desc | limit 25
| filter eventSource = "cognito-idp.amazonaws.com"
```

Get a list of the 25 most recently added log events that include exceptions.

```
fields @timestamp, @message | sort @timestamp desc | limit 25
| filter eventSource = "cognito-idp.amazonaws.com" and @message like /Exception/
```

Exception and Error Queries

Find the 25 most recently added log events with error code `NotAuthorizedException` along with Amazon Cognito user pool sub.

```
fields @timestamp, additionalEventData.sub as user | sort @timestamp desc | limit 25  
| filter eventSource = "cognito-idp.amazonaws.com" and errorCode= "NotAuthorizedException"
```

Find the number of records with `sourceIPAddress` and corresponding `eventName`.

```
filter eventSource = "cognito-idp.amazonaws.com"  
| stats count(*) by sourceIPAddress, eventName
```

Find the top 25 IP addresses that triggered a `NotAuthorizedException` error.

```
filter eventSource = "cognito-idp.amazonaws.com" and errorCode= "NotAuthorizedException"  
| stats count(*) as count by sourceIPAddress, eventName  
| sort count desc | limit 25
```

Find the top 25 IP addresses that called the `ForgotPassword` API.

```
filter eventSource = "cognito-idp.amazonaws.com" and eventName = 'ForgotPassword'  
| stats count(*) as count by sourceIPAddress  
| sort count desc | limit 25
```

Compliance validation for Amazon Cognito

Third-party auditors assess the security and compliance of Amazon Cognito as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others.

For a list of AWS services in scope of specific compliance programs, see [AWS services in scope by compliance program](#). For general information, see [AWS compliance programs](#).

You can download third-party audit reports using AWS Artifact. For more information, see [Downloading reports in AWS Artifact](#).

Your compliance responsibility when using Amazon Cognito is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- [Security and compliance quick start guides](#) – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- [Architecting for HIPAA security and compliance whitepaper](#) – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- [AWS compliance resources](#) – This collection of workbooks and guides might apply to your industry and location.
- [Evaluating resources with rules](#) in the *AWS Config Developer Guide* – AWS Config; assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- [AWS Security Hub](#) – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience in Amazon Cognito

The AWS global infrastructure is built around AWS Regions and Availability Zones. Regions provide multiple physically separated and isolated Availability Zones, which are connected through low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see [AWS global infrastructure](#).

Topics

- [Regional data considerations \(p. 380\)](#)

Regional data considerations

Amazon Cognito user pools are each created in one AWS Region, and they store the user profile data only in that region. User pools can send user data to a different AWS Region, depending on how optional features are configured.

- If the default `no-reply@verificationemail.com` email address setting is used for routing verification of emails addresses with Amazon Cognito user pools, emails are routed through the same region as the associated user pool.
- If a different email address is used to configure Amazon Simple Email Service (Amazon SES) with Amazon Cognito user pools, that email address is routed through the AWS Region associated with the email address in Amazon SES.
- SMS messages from Amazon Cognito user pools are routed through the same region Amazon SNS unless noted otherwise on [Configuring email or phone verification](#).
- If Amazon Pinpoint analytics are used with Amazon Cognito user pools, the event data is routed to the US East (N. Virginia) Region.

Note

Amazon Pinpoint is available in several AWS Regions in North America, Europe, Asia, and Oceania. Amazon Pinpoint regions include the Amazon Pinpoint API. If a Amazon Pinpoint region is supported by Amazon Cognito, then Amazon Cognito will send events to Amazon Pinpoint projects within the *same* Amazon Pinpoint region. If a region *isn't* supported by Amazon Pinpoint, then Amazon Cognito will *only* support sending events in us-east-1. For Amazon Pinpoint detailed region information, see [Amazon Pinpoint endpoints and quotas](#) and [Using Amazon Pinpoint analytics with amazon cognito user pools](#).

Infrastructure security in Amazon Cognito

As a managed service, Amazon Cognito is protected by the AWS global network security procedures that are described in the [Amazon Web Services: Overview of security processes](#) whitepaper.

You use AWS published API calls to access Amazon Cognito through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the [AWS Security Token Service](#) (AWS STS) to generate temporary security credentials to sign requests.

Configuration and vulnerability analysis in Amazon Cognito user pools

AWS handles basic security tasks like guest operating system (OS) and database patching, firewall configuration, and disaster recovery. These procedures have been reviewed and certified by the appropriate third parties. For more details, see the following resources:

- [Compliance validation for Amazon Cognito \(p. 379\)](#)
- [Shared Responsibility Model](#)

Security best practices for Amazon Cognito user pools

You can add multi-factor authentication (MFA) to a user pool to protect the identity of your users. MFA adds a second authentication factor so that your user pool doesn't rely solely on user name and password. You can use either SMS text messages or time-based one-time passwords (TOTP) as second factors to sign in your users. You can also use adaptive authentication with its risk-based model to predict when you might need another authentication factor. User pool *advanced security* features include adaptive authentication and protections against compromised credentials.

Topics

- [Adding MFA to a user pool \(p. 381\)](#)
- [Adding advanced security to a user pool \(p. 387\)](#)
- [User pool case sensitivity \(p. 398\)](#)

Adding MFA to a user pool

Multi-factor authentication (MFA) increases security for your app. It adds a *something you have* authentication factor to the *something you know* factor of user name and password. You can choose SMS text messages or time-based one-time passwords (TOTP) as second factors to sign in your users.

Note

The first time that a new user signs in to your app, Amazon Cognito issues OAuth 2.0 tokens, even if your user pool requires MFA. The second authentication factor when your user signs in for the first time is their confirmation of the verification message that Amazon Cognito sends to them. If your user pool requires MFA, Amazon Cognito prompts your user to register an additional sign-in factor to use during each sign-in attempt after the first.

With adaptive authentication, you can configure your user pool to require second factor authentication in response to an increased risk level. To add adaptive authentication to your user pool, see [Adding advanced security to a user pool \(p. 387\)](#).

When you set MFA to `required` for a user pool, all users must complete MFA to sign in. To sign in, each user must set up at least one MFA factor, such as SMS or TOTP. When you set MFA to `required`, you must include the MFA setup in user onboarding so that your user pool permits them to sign in.

If you activate SMS as an MFA factor, you can require that users provide phone numbers and have your users verify them during sign-up. If you have set MFA to `required` and only support SMS as a factor, users will need to provide phone numbers. Users without phone numbers need your support to add a

phone number to their profile before they can sign in. You can use unverified phone numbers for SMS MFA. These numbers will receive verified status after MFA succeeds.

If you have set MFA to **required** and you activated SMS and TOTP as supported verification methods, Amazon Cognito prompts new users without phone numbers to set up TOTP MFA. If you have set MFA to **required** and the only MFA method you activated is TOTP, Amazon Cognito prompts all new users to set up TOTP MFA the second time they sign in. Amazon Cognito generates a challenge to set up TOTP MFA in response to [InitiateAuth](#) and [AdminInitiateAuth](#) API operations.

Topics

- [Prerequisites \(p. 382\)](#)
- [Configuring multi-factor authentication \(p. 382\)](#)
- [SMS text message MFA \(p. 384\)](#)
- [TOTP software token MFA \(p. 384\)](#)

Prerequisites

Before you set up MFA, consider the following:

- In the legacy Amazon Cognito console, you can only set MFA as **Required** when you initially create a user pool. Switch to the new console or use the [SetUserPoolMfaConfig](#) API operation to set MFA to **required** for existing user pools.
- When you activate MFA in your user pool and choose **SMS text message** as a second factor, you can send SMS messages to a phone number attribute that you haven't verified in Amazon Cognito. After your user completes SMS MFA, Amazon Cognito sets their `phone_number_verified` attribute to `true`.
- If your account is in the SMS sandbox in the AWS Region that contains the Amazon Simple Notification Service (Amazon SNS) resources for your user pool, you must verify phone numbers in Amazon SNS before you can send an SMS message. For more information, see [SMS message settings for Amazon Cognito user pools \(p. 187\)](#).
- Advanced security features require that you activate MFA and set it as optional in the Amazon Cognito user pool console. For more information, see [Adding advanced security to a user pool \(p. 387\)](#).

Configuring multi-factor authentication

You can configure MFA in the Amazon Cognito console.

Original console

To configure MFA in the Amazon Cognito console

1. From the left navigation bar, choose **MFA and verifications**.
2. Choose whether MFA is **Off**, **Optional**, or **Required**.

Do you want to enable Multi-Factor Authentication?

Multi-Factor Authentication (MFA) increases security for your end users. If you choose 'optional', individual users can have MFA enabled or disabled at any time. If you choose 'required', all users must use MFA. Phone numbers must be verified if MFA is enabled. You can configure adaptive authentication based on risk scoring of user sign in attempts. [Learn more about multi-factor authentication.](#)

Note: separate charges apply for sending text messages.

Off Optional Required

Which second factors do you want to enable?

Your users will be able to configured and choose any of the factors you enabled. You can change these settings later.

SMS text message

Important: In order to send SMS messages to users to verify phone numbers or for MFA you must request a specific permission in your IAM role.

Note: separate charges may apply for sending SMS text messages. See the [Worldwide SMS pricing page](#) for more details.

Time-based One-time Password

3. Choose **Optional** to activate MFA on a per-user basis or if you are using the risk-based adaptive authentication. For more information about adaptive authentication, see [Adding advanced security to a user pool \(p. 387\)](#).
4. Choose which second factors to support in your app. Your users can use **SMS text message** or **Time-based One-time Password** as a second factor. We recommend that you use TOTP so that you can use SMS as a password recovery mechanism rather than as an authentication factor.
5. If you use SMS text messages as a second factor and you don't have an IAM role defined with this permission, then you can create one in the console. Choose **Create role** to create an IAM role that allows Amazon Cognito to send SMS messages to your users for you. For more information, see [IAM Roles](#).
6. Choose **Save changes**.

New console

To configure MFA in the Amazon Cognito console

1. Sign in to the [Amazon Cognito console](#).
2. Choose **User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. Choose the **Sign-in experience** tab. Locate **Multi-factor authentication** and choose **Edit**.
5. Choose the **MFA enforcement** method that you want to use with your user pool.
 - a. **Require MFA**. All users in your user pool must sign in with an additional SMS code or time-based one-time password (TOTP) factor.
 - b. **Optional MFA** - You can give your users the option to register an additional sign-in factor but still permit users who haven't configured MFA to sign in. If you use adaptive authentication, choose this option. For more information about adaptive authentication, see [Adding advanced security to a user pool \(p. 387\)](#).
 - c. **No MFA**. Your users can't register an additional sign-in factor.

6. Choose the **MFA methods** that you support in your app. You can set **SMS message** or TOTP-generating **Authenticator apps** as a second factor. We recommend that you implement TOTP-based MFA so that account recovery can use SMS messages.
7. If you use SMS text messages as a second factor and you haven't configured an IAM role to use with Amazon Simple Notification Service (Amazon SNS) for SMS messages, create one in the console. In the **Messaging** tab for your user pool, locate **SMS** and choose **Edit**. You can also use an existing role that allows Amazon Cognito to send SMS messages to your users for you. For more information, see [IAM Roles](#).
8. Choose **Save changes**.

SMS text message MFA

When a user signs in with MFA enabled, they first enter and submit their user name and password. The client app receives a `getMFA` response that indicates where the authorization code was sent. The client app should indicate to the user where to look for the code (such as which phone number the code was sent to). Next, it provides a form for entering the code. Finally, the client app submits the code to complete the sign-in process. The destination is masked, which hides all but the last four digits of the phone number. If an app is using the Amazon Cognito hosted UI, it shows a page for the user to enter the MFA code.

The SMS text message authorization code is valid for 3 minutes.

If a user no longer has access to their device where the SMS text message MFA codes are sent, they must request help from your customer service office. An administrator with necessary AWS account permissions can change the user's phone number, but only through the AWS CLI or the API.

When a user successfully goes through the SMS text message MFA flow, their phone number is also marked as verified.

Note

SMS for MFA is charged separately. (There is no charge for sending verification codes to email addresses.) For information about Amazon SNS pricing, see [Worldwide SMS Pricing](#). For the current list of countries where SMS messaging is available, see [Supported Regions and Countries](#).

Important

To ensure that SMS messages are sent to verify phone numbers and for SMS text message MFA, you must request an increased spend limit from Amazon SNS.

Amazon Cognito uses Amazon SNS for sending SMS messages to users. The number of SMS messages Amazon SNS delivers is subject to spend limits. Spend limits can be specified for an AWS account and for individual messages, and the limits apply only to the cost of sending SMS messages.

The default spend limit per account (if not specified) is 1.00 USD per month. If you want to raise the limit, submit an [SNS Limit Increase case](#) in the AWS Support Center. For **New limit value**, enter your desired monthly spend limit. In the **Use Case Description** field, explain that you're requesting an SMS monthly spend limit increase.

To add MFA to your user pool, see [Adding MFA to a user pool \(p. 381\)](#).

TOTP software token MFA

Important

Your users can't set up time-based one-time password (TOTP) multi-factor authentication (MFA) in the Amazon Cognito hosted UI. Your app must use the Amazon Cognito API to follow the setup steps described in [Associate the TOTP software token \(p. 385\)](#) and [Verify the TOTP token \(p. 386\)](#).

When you set up TOTP software token MFA in your user pool, your user signs in with a user name and password, then uses a TOTP to complete authentication. After your user sets and verifies a user name

and password, they can activate a TOTP software token for MFA. If your app uses the Amazon Cognito hosted UI to sign in users, your user submits their user name and password, and then submits the TOTP password on an additional sign-in page.

You can activate TOTP MFA for your user pool in the Amazon Cognito console, or you can use Amazon Cognito API operations. At the user pool level, you can call [SetUserPoolMfaConfig](#) to configure MFA and enable TOTP MFA.

Note

If you haven't activated TOTP software token MFA for the user pool, Amazon Cognito can't use the token to associate or verify users. In this case, users receive a `SoftwareTokenMFANotFoundException` exception with the description `Software Token MFA has not been enabled by the userPool`. If you deactivate software token MFA for the user pool later, users who previously associated and verified a TOTP token can continue to use it for MFA.

Configuring TOTP for your user is a multi-step process where your user receives a secret code that they validate by entering a one-time password. Next, you can enable TOTP MFA for your user or set TOTP as the preferred MFA method for your user.

To add MFA to your user pool, see [Adding MFA to a user pool \(p. 381\)](#).

TOTP MFA considerations and limitations

1. The Amazon Cognito hosted UI currently doesn't support self-service TOTP setup. After your app associates and verifies a TOTP software token, your user can provide their TOTP in the hosted UI.
2. Amazon Cognito supports software token MFA through an authenticator app that generates TOTP codes. Amazon Cognito doesn't support hardware-based MFA.
3. When your user pool requires TOTP for a user who has not configured it, your user receives a one-time access token that your app can use to activate TOTP MFA for the user. Subsequent sign-in attempts fail until your user has registered an additional TOTP sign-in factor.
 - A user who signs up in your user pool with the `SignUp` API operation or through the hosted UI receives one-time tokens when the user completes sign-up.
 - After you create a user, and the user sets their initial password, Amazon Cognito issues one-time tokens from the hosted UI to the user. If you set a permanent password for the user, Amazon Cognito issues one-time tokens when the user first signs in.
 - Amazon Cognito doesn't issue one-time tokens to an administrator-created user who signs in with the `InitiateAuth` or `AdminInitiateAuth` API operations. After your user succeeds in the challenge to set their initial password, or if you set a permanent password for the user, Amazon Cognito immediately challenges the user to set up MFA.
4. If a user in a user pool that requires MFA has already received a one-time access token but hasn't set up TOTP MFA, the user can't sign in with the hosted UI until they have set up MFA. Instead of the access token, you can use the session response value from an `MFA_SETUP` challenge to `InitiateAuth` or `AdminInitiateAuth` in an `AssociateSoftwareToken` request.
5. If your users have set up TOTP, they can use it for MFA, even if you deactivate TOTP for the user pool later.

When a user first signs in, your app uses their one-time access token to generate the TOTP private key and present it to your user in text or QR code format. Your user configures their authenticator app and provides a TOTP for subsequent sign-in attempts. Your app or the hosted UI presents the TOTP to Amazon Cognito in MFA challenge responses.

Associate the TOTP software token

To associate the TOTP token, send your user a secret code that they must validate with a one-time password. Associating the token requires three steps.

1. When your user chooses TOTP software token MFA, call [AssociateSoftwareToken](#) to return a unique generated shared secret key code for the user account. You can authorize AssociateSoftwareToken with either an access token or a session string.
2. Your app presents the user with the private key, or a QR code that you generate from the private key. Your user must enter the key into a TOTP-generating app such as Google Authenticator. You can use [libqrencode](#) to generate a QR code.
3. Your user enters the key, or scans the QR code into a authenticator app such as Google Authenticator, and the app begins generating codes.

Verify the TOTP token

Next, verify the TOTP token. Request sample codes from your user and provide them to the Amazon Cognito service to confirm that the user is successfully generating TOTP codes, as follows.

1. Your app prompts your user for a code to demonstrate that they have set up their authenticator app properly.
2. The user's authenticator app displays a temporary password. The authenticator app bases the password on the secret key you gave to the user.
3. Your user enters their temporary password. Your app passes the temporary password to Amazon Cognito in a [VerifySoftwareToken](#) API request.
4. Amazon Cognito has retained the secret key associated with the user, and generates a TOTP and compares it with the one that your user provided. If they match, VerifySoftwareToken returns a SUCCESS response.
5. Amazon Cognito associates the TOTP factor with the user.
6. If the VerifySoftwareToken operation returns an ERROR response, make sure that the user's clock is correct and that they have not exceeded the maximum number of retries. Amazon Cognito accepts TOTP tokens that are within 30 seconds before or after the attempt, to account for minor clock skew. When you have resolved the issue, try the VerifySoftwareToken operation again.

Sign in with TOTP MFA

At this point, your user signs in with the time-based one-time password. The process is as follows.

1. Your user enters their user name and password to sign in to your client app.
2. The TOTP MFA challenge is invoked, and your user is prompted by your app to enter a temporary password.
3. Your user gets the temporary password from an associated TOTP-generating app.
4. Your user enters the TOTP code into your client app. Your app notifies the Amazon Cognito service to verify it. For each sign-in, [RespondToAuthChallenge](#) should be called to get a response to the new TOTP authentication challenge.
5. If the token is verified by Amazon Cognito, the sign-in is successful and your user continues with the authentication flow.

Remove the TOTP token

Finally, your app should allow your user to remove the TOTP token:

1. Your client app should ask your user to enter their password.
2. If the password is correct, remove the TOTP token.

Note

A delete TOTP software token operation is not currently available in the API. This functionality is planned for a future release. Use [SetUserMFAPreference](#) to disable TOTP MFA for an individual user.

Adding advanced security to a user pool

After you create your user pool, you have access to **Advanced security** on the navigation bar in the Amazon Cognito console. You can turn the user pool advanced security features on, and customize the actions that are taken in response to different risks. Or you can use audit mode to gather metrics on detected risks without applying any security mitigations. In audit mode, the advanced security features publish metrics to Amazon CloudWatch. See [Viewing advanced security metrics \(p. 395\)](#).

Topics

- [Considerations and limitations \(p. 387\)](#)
- [Prerequisites \(p. 387\)](#)
- [Configuring advanced security features \(p. 388\)](#)
- [Checking for compromised credentials \(p. 390\)](#)
- [Using adaptive authentication \(p. 391\)](#)
- [Viewing advanced security metrics \(p. 395\)](#)
- [Activating user pool advanced security from your app \(p. 397\)](#)

Considerations and limitations

- Additional pricing applies for Amazon Cognito advanced security features. See the [Amazon Cognito pricing page](#).
- Amazon Cognito supports advanced security features with the following standard authentication flows: `USER_PASSWORD_AUTH`, `ADMIN_USER_PASSWORD_AUTH`, `USER_SRP_AUTH`, and `ADMIN_USER_SRP_AUTH`. You can't use advanced security with a `CUSTOM_AUTH` flow and [Custom authentication challenge Lambda triggers \(p. 114\)](#), or with federated sign-in.
- With Amazon Cognito advanced security features in **Full function** mode, you can create IP-address **Always block** and **Always allow** exceptions. A session from an IP address on the **Always block** exception list isn't assigned a risk level by adaptive authentication, and can't sign in to your user pool.
- Blocked requests from IP addresses on an **Always block** exception list in your user pool contribute to the [request rate quotas](#) for your user pools. Amazon Cognito advanced security features don't prevent distributed denial of service (DDoS) attacks. See [Protect public clients for Amazon Cognito by using an Amazon CloudFront proxy](#) in the [AWS Security blog](#) for information about how to prevent unwanted traffic to your Amazon Cognito endpoints.

Prerequisites

Before you begin, you need the following:

- A user pool with an app client. For more information, see [Getting started with user pools \(p. 23\)](#).
- Set multi-factor authentication (MFA) to **Optional** in the Amazon Cognito console to use the risk-based adaptive authentication feature. For more information, see [Adding MFA to a user pool \(p. 381\)](#).
- If you're using email notifications, go to the [Amazon SES console](#) to configure and verify an email address or domain to use with your email notifications. For more information about Amazon SES, see [Verifying Identities in Amazon SES](#).

Configuring advanced security features

You can configure Amazon Cognito advanced security features in the AWS Management Console.

Original console

To configure advanced security for a user pool

1. From the left navigation bar, choose **Advanced security**.
2. For **Do you want to enable advanced security features for this user pool?**, choose **Yes** to turn on advanced security. Or choose **Audit only** to gather information, and send user pool data to CloudWatch.

We recommend keeping the advanced security features in audit mode for two weeks before enabling actions. During this time, Amazon Cognito can learn the usage patterns of your app users.

3. From the drop-down list, choose **What app client do you want to customize settings for?**. The default is to keep your settings as global for all app clients.
4. For **Which action do you want to take with the compromised credentials?**, choose **Allow** or **Block use**.
5. Choose **Customize when compromised credentials are blocked** to select which events should initiate compromised credentials checks:
 - Sign-in
 - Sign-up
 - Password change
6. Choose how to respond to malicious sign-in attempts under **How do you want to use adaptive authentication for sign-in attempts rated as low, medium and high risk?**. You can allow or block the sign-in attempt, or require additional challenges before allowing the sign-in.

To send email notifications when anomalous sign-in attempts are detected, choose **Notify users**

How do you want to use adaptive authentication for sign-in attempts rated as low, medium and high risk?

You can use adaptive authentication to add protections against sign-in attempts that are rated as higher risk such as coming from untrusted devices. Learn more about [adaptive authentication](#).

	Allow	Optional MFA	Require MFA	Block	Notify users
Low risk	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="checkbox"/>
Medium risk	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="checkbox"/>
High risk	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="checkbox"/>

Choosing allow will still log all attempted sign-ins rated as higher risk to [AWS Cloudwatch](#).

7. If you chose **Notify users** in the previous step, then you can customize the email notification messages by using the **Notification message customization** form:

Notification message customization
Customize the email user notification messages sent to users after sign-in attempts with risks detected. Enter your settings below (SES).

SES Region
US East (Virginia)

Source ARN
arn:aws:ses:us-east-1:123456789012:identity/janedoe@example.com
You must verify your email address with Amazon SES before you can select it. [Verify an SES identity](#).

FROM email address
example@example.com

REPLY-TO email address
example@example.com

Adaptive authentication notification messages [Customize...](#)

8. Choose **Customize** to customize adaptive authentication notifications with both HTML and plaintext versions of email messages. To learn more about email message templates, see [Message templates \(p. 218\)](#).
9. Type any IP addresses that you want to **Always allow**, or **Always block**, regardless of the advanced security risk assessment. Specify the IP address ranges in [CIDR notation](#) (such as 192.168.100.0/24).
10. Choose **Save changes**.

New console

To configure advanced security for a user pool

1. Go to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. Choose the **App integration** tab. Locate **Advanced security** and choose **Enable**. If you enabled advanced security earlier, choose **Edit**.
5. Select **Full function** to configure advanced security responses to compromised credentials and adaptive authentication. Select **Audit only** to gather information and send user pool data to CloudWatch. Advanced security pricing applies in both **Audit only** and **Full function** mode. For more information, see [Amazon Cognito Pricing](#).

We recommend keeping the advanced security features in audit mode for two weeks before enabling actions. During this time, Amazon Cognito can learn the usage patterns of your app users.

6. If you selected **Audit only**, choose **Save changes**. If you selected **Full function**:

- a. Select whether you will take **Custom** action or use or **Cognito defaults** to respond to suspected **Compromised credentials**. **Cognito defaults** are:
 - i. Detect compromised credentials on **Sign-in**, **Sign-up**, and **Password change**.
 - ii. Respond to compromised credentials with the action **Block sign-in**.
 - b. If you selected **Custom** actions for **Compromised credentials**, choose the user pool actions that Amazon Cognito will use for **Event detection** and the **Compromised credentials responses** that you would like Amazon Cognito to take. You can **Block sign-in** or **Allow sign-in** with suspected compromised credentials.
 - c. Choose how to respond to malicious sign-in attempts under **Adaptive authentication**. Select whether you will take **Custom** action or use or **Cognito defaults** to respond to suspected malicious activity. When you select **Cognito defaults**, Amazon Cognito blocks sign-in at all risk levels and does not notify the user.
 - d. If you selected **Custom** actions for **Adaptive authentication**, choose the **Automatic risk response** actions that Amazon Cognito will take in response to detected risks based on severity level. When you assign a response to a level of risk, you can't assign a less-restrictive response to a higher level of risk. You can assign the following responses to risk levels:
 - i. **Allow sign-in** - Take no preventative action.
 - ii. **Optional MFA** - If the user has MFA configured, Amazon Cognito will always require the user to provide an additional SMS or time-based one-time password (TOTP) factor when they sign in. If the user does not have MFA configured, they can continue signing in normally.
 - iii. **Require MFA** - If the user has MFA configured, Amazon Cognito will always require the user to provide an additional SMS or TOTP factor when they sign in. If the user does not have MFA configured, Amazon Cognito will prompt them to set up MFA. Before you automatically require MFA for your users, configure a mechanism in your app to capture phone numbers for SMS MFA, or to register authenticator apps for TOTP MFA.
 - iv. **Block sign-in** - Prevent the user from signing in.
 - v. **Notify user** - Send an email message to the user with information about the risk that Amazon Cognito detected and the response you have taken. You can customize email message templates for the messages you send.
7. If you chose **Notify user** in the previous step, you can customize your email delivery settings and email message templates for adaptive authentication.
- a. Under **Email configuration**, choose the **SES Region**, **FROM email address**, **FROM sender name**, and **REPLY-TO email address** that you want to use with adaptive authentication. For more information about integrating your user pool email messages with Amazon Simple Email Service, see [Email settings for Amazon Cognito user pools](#).
 - b. Expand **Email templates** to customize adaptive authentication notifications with both HTML and plaintext versions of email messages. To learn more about email message templates, see [Message templates \(p. 218\)](#).
8. Expand **IP address exceptions** to create an **Always-allow** or an **Always-block** list of IPv4 or IPv6 address ranges that will always be allowed or blocked, regardless of the advanced security risk assessment. Specify the IP address ranges in [CIDR notation](#) (such as 192.168.100.0/24).
9. Choose **Save changes**.

Checking for compromised credentials

Amazon Cognito can detect if a user's user name and password have been compromised elsewhere. This can happen when users reuse credentials at more than one site, or when they use insecure passwords. Amazon Cognito checks *native users* who sign in with user name and password, in the hosted UI and

with the Amazon Cognito API. Native users are users who you created or who signed up in the Amazon Cognito directory without a federated identity provider (IdP).

From **Advanced security** in the **App integration** tab of the Amazon Cognito console, you can configure **Compromised credentials**. Configure **Event detection** to choose the user events that you want to monitor for compromised credentials. Configure **Compromised credentials responses** to choose whether to allow or block the user if compromised credentials are detected. Amazon Cognito can check for compromised credentials during sign-in, sign-up, and password changes.

When you choose **Allow sign-in**, you can review Amazon CloudWatch Logs to monitor the evaluations that Amazon Cognito makes on user events. For more information, see [Viewing advanced security metrics \(p. 395\)](#). When you choose **Block sign-in**, Amazon Cognito prevents sign-in by users who use compromised credentials. When Amazon Cognito blocks sign-in for a user, it sets the user's `UserStatus` to `RESET_REQUIRED`. A user with a `RESET_REQUIRED` status must change their password before they can sign in again.

Note

Currently, Amazon Cognito doesn't check for compromised credentials for sign-in operations with Secure Remote Password (SRP) flow, which doesn't send the password during sign-in. Amazon Cognito checks sign-ins that use the `AdminInitiateAuth` API with `ADMIN_USER_PASSWORD_AUTH` flow, and the `InitiateAuth` API with `USER_PASSWORD_AUTH` flow, for compromised credentials.

To add compromised credentials protections to your user pool, see [Adding advanced security to a user pool \(p. 387\)](#).

Using adaptive authentication

With adaptive authentication, you can configure your user pool to block suspicious sign-ins or add second factor authentication in response to an increased risk level. For each sign-in attempt, Amazon Cognito generates a risk score for how likely the sign-in request is to be from a compromised source. This risk score is based on many factors, including whether it detects a new device, user location, or IP address. Adaptive Authentication adds MFA based on risk level for users who don't have an MFA type enabled at the user level. When an MFA type is enabled at the user level, those users will always receive the second factor challenge during authentication regardless of how you configured adaptive authentication.

Amazon Cognito publishes sign-in attempts, their risk levels, and failed challenges to Amazon CloudWatch. For more information, see [Viewing advanced security metrics \(p. 395\)](#).

To add adaptive authentication to your user pool, see [Adding advanced security to a user pool \(p. 387\)](#).

Topics

- [Adaptive authentication overview \(p. 391\)](#)
- [Adding user device and session data to API requests \(p. 392\)](#)
- [Viewing user event history \(p. 394\)](#)
- [Providing event feedback \(p. 394\)](#)
- [Sending notification messages \(p. 395\)](#)

Adaptive authentication overview

From the **Advanced security** page in the Amazon Cognito console, you can choose settings for adaptive authentication, including what actions to take at different risk levels and customization of notification messages to users.

For each risk level, you can choose from the following options:

Option	Action
Allow	Users can sign in without an additional factor.
Optional MFA	Users who have a second factor configured must complete a second factor challenge to sign in. A phone number for SMS and a TOTP software token are the available second factors. Users without a second factor configured can sign in with only one set of credentials.
Require MFA	Users who have a second factor configured must complete a second factor challenge to sign in. Amazon Cognito blocks sign-in for users who don't have a second factor configured.
Block	Amazon Cognito blocks all sign-in attempts at the designated risk level.

Note

You don't have to verify phone numbers to use them for SMS as a second authentication factor.

Adding user device and session data to API requests

You can collect and pass information about your user's session to Amazon Cognito advanced security when you use the API to sign them up, sign them in, and reset their password. This information includes your user's IP address and a unique device identifier.

You might have an intermediate network device between your users and Amazon Cognito, like a proxy service or an application server. You can collect users' context data and pass it to Amazon Cognito so that adaptive authentication calculates your risk based on the characteristics of the user endpoint, instead of your server or proxy. If your client-side app calls Amazon Cognito API operations directly, adaptive authentication automatically records the source IP address. However, it does not record other device information like the `user-agent` unless you also collect a device fingerprint.

Generate this data with the Amazon Cognito context data collection library and submit it to Amazon Cognito advanced security with the `ContextData` and `UserContextData` parameters. The context data collection library is included in the AWS SDKs. For more information, see [Integrating Amazon Cognito with web and mobile apps](#). You can submit `ContextData` if you have activated advanced security features in your user pool. For more information, see [Configuring advanced security features](#).

When you call the following Amazon Cognito authenticated API operations from your app server, pass the IP of the user's device in the `ContextData` parameter. In addition, pass your server name, server path, and encoded device-fingerprinting data.

- [AdminInitiateAuth](#)
- [AdminRespondToAuthChallenge](#)

When you call Amazon Cognito unauthenticated API operations, you can submit `UserContextData` to Amazon Cognito advanced security features. This data includes a device fingerprint in the `EncodedData` parameter. You can also submit an `IpAddress` parameter in your `UserContextData` if you meet the following conditions:

- You have activated advanced security features in your user pool. For more information, see [Configuring advanced security features](#).

- Your app client has a client secret. For more information, see [Configuring a user pool app client](#).
- You have activated **Accept additional user context data** in your app client. For more information, see [Accepting additional user context data \(AWS console\) \(p. 393\)](#).

Your app can populate the `UserContextData` parameter with encoded device-fingerprinting data and the IP address of the user's device in the following Amazon Cognito unauthenticated API operations.

- [InitiateAuth](#)
- [RespondToAuthChallenge](#)
- [SignUp](#)
- [ConfirmSignUp](#)
- [ForgotPassword](#)
- [ConfirmForgotPassword](#)
- [ResendConfirmationCode](#)

[Accepting additional user context data \(AWS console\)](#)

Your user pool accepts an IP address in a `UserContextData` parameter after you activate the **Accept additional user context data** feature. You don't need to activate this feature if:

- Your users only sign in with authenticated API operations like [AdminInitiateAuth](#), and you use the `ContextData` parameter.
- You only want your unauthenticated API operations to send a device fingerprint, but not an IP address, to Amazon Cognito advanced security features.

Update your app client as follows in the Amazon Cognito console to add support for additional user context data.

1. Sign in to the [Amazon Cognito console](#).
2. In the navigation pane, choose **Manage your User Pools**, and choose the user pool you want to edit.
3. Choose the **App integration** tab.
4. Under **App clients and analytics**, choose or create an app client. For more information, see [Configuring a user pool app client](#).
5. Choose **Edit** from the **App client information** container.
6. In the **Advanced authentication settings** for your app client, choose **Accept additional user context data**.
7. Choose **Save changes**.

To configure your app client to accept user context data in the Amazon Cognito API, set `EnablePropagateAdditionalUserContextData` to `true` in a [CreateUserPoolClient](#) or [UpdateUserPoolClient](#) request. For information about how to activate advanced security from your web or mobile app, see [Activating user pool advanced security from your app](#). When your app calls Amazon Cognito from your server, collect user context data from the client side. The following is an example that uses the JavaScript SDK method `getData`.

```
var encodedData =  
  AmazonCognitoAdvancedSecurityData.getData(username, userPoolId, clientId);
```

When you design your app to use adaptive authentication, we recommend that you incorporate the latest Amazon Cognito SDK into your app.. The latest version of the SDK collects device fingerprinting

information like device ID, model, and time zone. For more information about Amazon Cognito SDKs, see [Install a user pool SDK](#). Amazon Cognito advanced security only saves and assigns a risk score to events that your app submits in the correct format. If Amazon Cognito returns an error response, check that your request includes a valid secret hash and that the `IPaddress` parameter is a valid IPv4 or IPv6 address.

Viewing user event history

Note

In the new Amazon Cognito console, you can view user event history in the **Users** tab.

To see the sign-in history for a user, you can choose the user from **Users and groups** in the Amazon Cognito console. Amazon Cognito retains user event history for two years.

Date (UTC)	Event	Result	Risk level	Risk decision	Challenge	IP	Device
Jan 23, 2018 11:43:05 PM	Sign In	Pass	-	No Risk	Password:Success	52.94.36.11	Chrome, Windows 10
Jan 23, 2018 11:42:14 PM	Sign In	Pass	-	No Risk	Password:Success	52.94.36.11	Chrome, Windows 10
Jan 18, 2018 9:21:21 PM	Sign In	Fail	High	Account Takeover	Password:Success	67.132.130.174	Mobile, Android Mobile
Jan 18, 2018 9:20:28 PM	Sign In	In Progress	High	Account Takeover	Password:Success	67.132.130.174	Mobile, Android Mobile
Jan 18, 2018 9:18:18 PM	Sign In	Pass	-	No Risk	Password:Success	67.132.130.174	Mobile, Android Mobile

Each sign-in event has an event ID. The event also has corresponding context data, such as location, device details, and risk detection results. You can query user event history with the Amazon Cognito API operation [AdminListUserAuthEvents](#) or with the AWS Command Line Interface (AWS CLI) with [admin-list-user-auth-events](#).

You can also correlate the event ID with the token that Amazon Cognito issued at the time that it recorded the event. The ID and access tokens include this event ID in their payload. Amazon Cognito also correlates refresh token use to the original event ID. You can trace the original event ID back to the event ID of the sign-in event that resulted in issuing the Amazon Cognito tokens. You can trace token usage within your system to a particular authentication event. For more information, see [Using tokens with user pools \(p. 191\)](#).

Providing event feedback

Event feedback affects risk evaluation in real time and improves the risk evaluation algorithm over time. You can provide feedback on the validity of sign-in attempts through the Amazon Cognito console and API operations.

The console lists the sign-in history on the **Users and groups** tab. If you select an entry, you can mark the event as valid or not valid. You can also provide feedback through the user pool API operation [AdminUpdateAuthEventFeedback](#), and through the AWS CLI command [admin-update-auth-event-feedback](#).

Sending notification messages

With advanced security protections, Amazon Cognito can notify your users of sign-in attempts. Amazon Cognito can also prompt users to select links to indicate if the sign-in was valid or not valid. Amazon Cognito uses this feedback to improve the risk detection accuracy for your user pool.

In the **How do you want to use adaptive authentication for sign-in attempts rated as low, medium and high risk?** section choose **Notify Users** for the low, medium, and high-risk cases.

How do you want to use adaptive authentication for sign-in attempts rated as low, medium and high risk?

You can use adaptive authentication to add protections against sign-in attempts that are rated as higher risk such as coming from an untrusted device or browser. Choose **Notify users** for the low, medium, and high-risk cases.

	Allow	Optional MFA	Require MFA	Block	Notify users
Low risk	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="checkbox"/>
Medium risk	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="checkbox"/>
High risk	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="checkbox"/>

Choosing allow will still log all attempted sign-ins rated as higher risk to [AWS Cloudwatch](#).

You can customize notification email messages, and provide both plaintext and HTML versions of these messages. Choose **Customize** from **Adaptive authentication notification messages** to customize your email notifications. To learn more about email templates, see [Message templates \(p. 218\)](#).

Viewing advanced security metrics

Amazon Cognito publishes metrics for advanced security features to your account in Amazon CloudWatch. Amazon Cognito groups the advanced security metrics together by risk level and also by request level.

To view metrics in the CloudWatch console

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Metrics**.
3. Choose Amazon Cognito.
4. Choose a group of aggregated metrics, such as **By Risk Classification**.
5. The **All metrics** tab displays all metrics for that choice. You can do the following:
 - To sort the table, use the column heading.
 - To graph a metric, select the check box next to the metric. To select all metrics, select the check box in the heading row of the table.
 - To filter by resource, choose the resource ID, and then choose **Add to search**.
 - To filter by metric, choose the metric name, and then choose **Add to search**.

Metric	Description	Metric Dimensions
CompromisedCredentialRisk	Requests where Amazon Cognito detected compromised credentials.	Operation: The type of operation. PasswordChange, SignIn, or SignUp are the only dimensions. UserPoolId: The identifier of the user pool. RiskLevel: high (default), medium, or low.
AccountTakeOverRisk	Requests where Amazon Cognito detected account take-over risk.	Operation: The type of operation. PasswordChange, SignIn, or SignUp are the only dimensions. UserPoolId: The identifier of the user pool. RiskLevel: high, medium, or low.
OverrideBlock	Requests that Amazon Cognito blocked because of the configuration provided by the developer.	Operation: The type of operation. PasswordChange, SignIn, or SignUp are the only dimensions. UserPoolId: The identifier of the user pool. RiskLevel: high, medium, or low.
Risk	Requests that Amazon Cognito marked as risky.	Operation: The type of operation, such as PasswordChange, SignIn, or SignUp. UserPoolId: The identifier of the user pool.
NoRisk	Requests where Amazon Cognito did not identify any risk.	Operation: The type of operation, such as PasswordChange, SignIn, or SignUp. UserPoolId: The identifier of the user pool.

Amazon Cognito offers you two predefined groups of metrics for ready analysis in CloudWatch. **By Risk Classification** identifies the granularity of the risk level for requests that Amazon Cognito identifies as risky. **By Request Classification** reflects metrics aggregated by request level.

Aggregated Metrics Group	Description
By Risk Classification	Requests that Amazon Cognito identifies as risky.

Aggregated Metrics Group	Description
By Request Classification	Metrics aggregated by request.

Activating user pool advanced security from your app

After you configure the advanced security features for your user pool, you must activate them in your web or mobile app.

To use advanced security with JavaScript

1. You might need to update your Amazon Cognito SDK to the latest version. For more information about Amazon Cognito SDKs, see [Install a user pool SDK](#).
2. To use the Auth SDK to activate the hosted UI, see the [CognitoAuth JavaScript sample app](#).
3. Set `AdvancedSecurityDataCollectionFlag` to `true`. Also, set `UserPoolId` to your user pool ID.
4. In your application replace `<region>` with your AWS Region such as US East (N. Virginia), and add this source reference to your JavaScript file:

```
<script src="https://amazon-cognito-assets.<region>.amazoncognito.com/amazon-cognito-advanced-security-data.min.js"></script>
```

For more information, see [Sample for Amazon Cognito Auth SDK for JavaScript](#).

To use advanced security with Android

1. You might need to update your Amazon Cognito SDK to the latest version. For more information about Amazon Cognito SDKs, see [Install a user pool SDK](#).
2. To use the Auth SDK to activate the hosted UI, see the [CognitoAuth Android sample app](#).
3. When you import `aws-android-sdk-cognitoidentityprovider-asf` through Maven in Gradle, use `{ transitive = true; }`.

Include the following dependency in your `build.gradle` file:

```
compile "com.amazonaws:aws-android-sdk-cognitoidentityprovider-asf:1.0.0"
```

For more information, see [AWS SDK for Android - Amazon Cognito Identity Provider ASF](#).

To use advanced security with iOS

1. You might need to update your Amazon Cognito SDK to the latest version. For more information about Amazon Cognito SDKs, see [Install a user pool SDK](#).
2. To use the Auth SDK to activate the hosted UI, see the [CognitoAuth iOS sample app](#).
3. To configure the Auth SDK with `Info.plist`, add the `PoolIdForEnablingASF` key to your Amazon Cognito user pool configuration. Set the key to your user pool ID.

To configure the Auth SDK with `AWSCognitoAuthConfiguration`, use [this initializer](#). Specify your user pool ID as `userPoolIdForEnablingASF`.

For more information, see [AWSCognitoIdentityProviderASF](#).

User pool case sensitivity

Amazon Cognito user pools that you create in the AWS Management Console are case insensitive by default. When a user pool is case insensitive, `user@example.com` and `User@example.com` refer to the same user. When user names in a user pool are case insensitive, the `preferred_username` and `email` attributes also are case insensitive.

To account for user pool case sensitivity settings, identify users in your app code based on an alternative user attribute. Because the case of a user name, preferred user name, or email address attribute can vary in different user profiles, refer instead to the `sub` attribute. You can also create an immutable custom attribute in your user pool, and assign your own unique identifier value to the attribute in each new user profile. When you first create a user, you can write a value to the immutable custom attribute that you created.

Note

Regardless of the case sensitivity settings of your user pool, Amazon Cognito requires that a federated user from a SAML or OIDC identity provider (IdP) pass a unique and case-sensitive `NameId` or `sub` claim. For more information about unique identifier case sensitivity and SAML IdPs, see [SAML user pool IdP authentication flow \(p. 70\)](#).

Creating a case-sensitive user pool

If you create resources with the AWS Command Line Interface (AWS CLI) and API operations such as [CreateUserPool](#), you must set the Boolean `CaseSensitive` parameter to `false`. This setting creates a case-insensitive user pool. If you do not specify a value, `CaseSensitive` defaults to `true`. This default is the opposite of the default behavior for user pools that you create in the AWS Management Console. Before February 12, 2020, user pools defaulted to case sensitive regardless of platform.

You can use the **Sign-in experience** tab of the AWS Management Console or the [DescribeUserPool](#) API operation to review the case sensitivity settings for each user pool in your account.

Migrating to a new user pool

Because of potential conflicts between user profiles, you can't change an Amazon Cognito user pool from case-sensitive to case-insensitive. Instead, migrate your users to a new user pool. You must build migration code to resolve case-related conflicts. This code must either return a unique new user or reject the sign-in attempt when it detects a conflict. In a new case-insensitive user pool, assign a [Migrate user Lambda trigger \(p. 127\)](#). The AWS Lambda function can create users in the new case-insensitive user pool. When the user fails sign-in with the case-insensitive user pool, the Lambda function finds and duplicates the user from the case-sensitive user pool. You can also activate a migrate user Lambda trigger on [ForgotPassword](#) events. Amazon Cognito passes user information and event metadata from the sign-in or password-recovery action to your Lambda function. You can use event data to manage conflicts between user names and email addresses when your function creates the new user in your case-insensitive user pool. These conflicts are between user names and email addresses that would be unique in a case-insensitive user pool, but identical in a case-sensitive user pool.

For more information about how to use a migrate user Lambda trigger between Amazon Cognito user pools, see [Migrating Users to Amazon Cognito user pools](#) in the AWS blog.

AWS managed policies for Amazon Cognito

To add permissions to users, groups, and roles, it is easier to use AWS managed policies than to write policies yourself. It takes time and expertise to [create IAM customer managed policies](#) that provide your team with only the permissions they need. To get started quickly, you can use our AWS managed policies. These policies cover common use cases and are available in your AWS account. For more information about AWS managed policies, see [AWS managed policies](#) in the *IAM User Guide*.

AWS services maintain and update AWS managed policies. You can't change the permissions in AWS managed policies. Services occasionally add additional permissions to an AWS managed policy to support new features. This type of update affects all identities (users, groups, and roles) where the policy is attached. Services are most likely to update an AWS managed policy when a new feature is launched or when new operations become available. Services do not remove permissions from an AWS managed policy, so policy updates won't break your existing permissions.

Additionally, AWS supports managed policies for job functions that span multiple services. For example, the **ReadOnlyAccess** AWS managed policy provides read-only access to all AWS services and resources. When a service launches a new feature, AWS adds read-only permissions for new operations and resources. For a list and descriptions of job function policies, see [AWS managed policies for job functions](#) in the *IAM User Guide*.

A number of policies are available via the IAM Console that customers can use to grant access to Amazon Cognito:

- **AmazonCognitoPowerUser** – Permissions for accessing and managing all aspects of your identity pools and user pools. To view the permissions for this policy, see [AmazonCognitoPowerUser](#).
- **AmazonCognitoReadOnly** – Permissions for read-only access to your identity pools and user pools. To view the permissions for this policy, see [AmazonCognitoReadOnly](#).
- **AmazonCognitoDeveloperAuthenticatedIdentities** – Permissions for your authentication system to integrate with Amazon Cognito. To view the permissions for this policy, see [AmazonCognitoDeveloperAuthenticatedIdentities](#).

These policies are maintained by the Amazon Cognito team, so even as new APIs are added your IAM users will continue to have the same level of access.

Note

Because creating a new identity pool also requires creating IAM roles, any IAM user you want to be able to create new identity pools with must have the admin policy applied as well.

Cognito updates to AWS managed policies

View details about updates to AWS managed policies for Cognito since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the Cognito [Document history](#) page.

Change	Description	Date
AmazonCognitoPowerUser – Update to an existing policy	Added a new permission to allow Amazon Cognito to call Amazon Simple Email Service <code>PutIdentityPolicy</code> and	November 17, 2021

Change	Description	Date
	<p><code>ListConfigurationSets</code> operations.</p> <p>This change allows Amazon Cognito user pools to update Amazon SES sending authorization policies and to apply Amazon SES configuration sets when you configure email sending in your user pool.</p>	
<code>AmazonCognitoPowerUser</code> – Update to an existing policy	<p>Added a new permission to allow Amazon Cognito to call Amazon Simple Notification Service's <code>GetSMSSandboxAccountStatus</code> operation.</p> <p>This change allows Amazon Cognito user pools to decide if you need to graduate out of the Amazon Simple Notification Service sandbox in order to send messages to all end users through user pools.</p>	June 1, 2021
Cognito started tracking changes	Cognito started tracking changes for its AWS managed policies.	March 1, 2021

Tagging Amazon Cognito resources

A *tag* is a metadata label that you or AWS assigns to an AWS resource. Each tag consists of a *key* and a *value*. For tags that you assign, you define the key and value. For example, you might define the key as `stage` and the value for one resource as `test`.

Tags help you do the following:

- Identify and organize your AWS resources. Many AWS services support tagging, so that you can assign the same tag to resources from different services. This helps you indicate which resources are related. For example, you could assign the same tag to an Amazon Cognito user pool that you assign to an Amazon DynamoDB table.
- Track your AWS costs. You can activate these tags on the AWS Billing and Cost Management dashboard. AWS uses cost allocation tags to categorize your costs and deliver a monthly cost allocation report to you. For more information, see [Use cost allocation tags](#) in the *AWS Billing User Guide*.
- Control access to your resources based on the tags that are assigned to them. You can control access by specifying tag keys and values in the conditions for an AWS Identity and Access Management (IAM) policy. For example, you could allow an IAM user to update a user pool, but only if the user pool has an `owner` tag with a value of that user's name. For more information, see [Controlling access using tags](#) in the *IAM User Guide*.

You can use the AWS Command Line Interface or the Amazon Cognito API to add, edit, or delete tags for both user and identity pools. You can also manage tags for user pools by using the Amazon Cognito console.

For tips on using tags, see the [AWS tagging strategies](#) post on the AWS Answers blog.

The following sections provide more information about tags for Amazon Cognito.

Supported resources in Amazon Cognito

The following resources in Amazon Cognito support tagging:

- User pools
- Identity pools

Tag restrictions

The following restrictions apply to tags on Amazon Cognito resources:

- Maximum number of tags that you can assign to a resource – 50
- Maximum key length – 128 Unicode characters
- Maximum value length – 256 Unicode characters
- Valid characters for keys and values – a-z, A-Z, 0-9, space, and the following characters: `_ . : / = + - @`
- Keys and values are case sensitive
- Don't use `aws :` as a prefix for keys; it's reserved for AWS use

Managing tags using the Amazon Cognito console

You can use the Amazon Cognito console to manage the tags that are assigned to your user pools.

The console does not include tagging features for identity pools at this time. You must manage tags for your identity pools programmatically, such as by using the AWS CLI.

Original console

To add tags to a user pool

1. Open the Amazon Cognito console at <https://console.aws.amazon.com/cognito>. If prompted, enter your AWS credentials.
2. Choose **Manage User Pools**.
3. On the **Your User Pools** page, choose the user pool to which you want to add tags.
4. In the navigation menu, choose **Tags**.
5. Choose **Add tag** to add your first tag. If you have previously assigned tags to this user pool, choose **Add another tag**.
6. Enter values for **Tag Key** and **Tag Value**.
7. For each additional tag that you want to add, choose **Add another tag**.
8. When you are finished adding tags, choose **Save changes**.

On the **Tags** page, you can also edit the keys and values of any existing tags. To remove a tag, choose the **x** in the top-right corner of the tag.

New console

To add tags to a user pool

1. Navigate to the [Amazon Cognito console](#). If prompted, enter your AWS credentials.
2. Choose **User Pools**.
3. Choose an existing user pool from the list, or [create a user pool](#).
4. Choose the **User pool properties** tab and locate **Tags**.
5. Choose **Add tags** to add your first tag. If you have previously assigned tags to this user pool, in **Manage tags**, chose **Add another**.
6. Specify values for **Tag Key** and **Tag Value**.
7. For each additional tag that you want to add, choose **Add another**.
8. When you are finished adding tags, choose **Save changes**.

On the **Manage tags** page, you can also edit the keys and values of any existing tags. To remove a tag, choose **Remove**.

AWS CLI examples

The AWS CLI provides commands that help you manage the tags that you assign to your Amazon Cognito user pools and identity pools.

Assigning tags

Use the following commands to assign tags to your existing user pools and identity pools.

Example tag-resource Command for user pools

Assign tags to a user pool by using [tag-resource](#) within the `cognito-idp` set of commands:

```
$ aws cognito-idp tag-resource \
> --resource-arn user-pool-arn \
> --tags Stage=Test
```

This command includes the following parameters:

- **resource-arn** – The Amazon Resource Name (ARN) of the user pool that you are applying tags to. To look up the ARN, choose the user pool in the Amazon Cognito console, and view the **Pool ARN** value on the **General settings** tab.
- **tags** – The key-value pairs of the tags, in the format `key=value`.

To assign multiple tags at once, specify them in a comma-separated list:

```
$ aws cognito-idp tag-resource \
> --resource-arn user-pool-arn \
> --tags Stage=Test,CostCenter=80432,Owner=SysEng
```

Example tag-resource Command for identity pools

Assign tags to an identity pool by using [tag-resource](#) within the `cognito-identity` set of commands:

```
$ aws cognito-identity tag-resource \
> --resource-arn identity-pool-arn \
> --tags Stage=Test
```

This command includes the following parameters:

- **resource-arn** – The Amazon Resource Name (ARN) of the identity pool that you are applying tags to. To look up the ARN, choose the identity pool in the Amazon Cognito console, and choose **Edit identity pool**. Then, at **Identity pool ID**, choose **Show ARN**.
- **tags** – The key-value pairs of the tags, in the format `key=value`.

To assign multiple tags at once, specify them in a comma-separated list:

```
$ aws cognito-identity tag-resource \
> --resource-arn identity-pool-arn \
> --tags Stage=Test,CostCenter=80432,Owner=SysEng
```

Viewing tags

Use the following commands to view the tags that you have assigned to your user pools and identity pools.

Example list-tags-for-resource Command for user pools

View the tags that are assigned to a user pool by using [list-tags-for-resource](#) within the `cognito-idp` set of commands:

```
$ aws cognito-idp list-tags-for-resource --resource-arn user-pool-arn
```

Example `list-tags-for-resource` Command for identity pools

View the tags that are assigned to an identity pool by using `list-tags-for-resource` within the `cognito-identity` set of commands:

```
$ aws cognito-identity list-tags-for-resource --resource-arn identity-pool-arn
```

Removing tags

Use the following commands to remove tags from your user pools and identity pools.

Example `untag-resource` Command for user pools

Remove tags from a user pool by using `untag-resource` within the `cognito-idp` set of commands:

```
$ aws cognito-idp untag-resource \  
> --resource-arn user-pool-arn \  
> --tag-keys Stage CostCenter Owner
```

For the `--tag-keys` parameter, specify one or more tag keys. Don't include the tag values. Separate keys with spaces.

Example `untag-resource` Command for identity pools

Remove tags from an identity pool by using `untag-resource` within the `cognito-identity` set of commands:

```
$ aws cognito-identity untag-resource \  
> --resource-arn identity-pool-arn \  
> --tag-keys Stage CostCenter Owner
```

For the `--tag-keys` parameter, specify one or more tag keys. Don't include the tag values.

Important

After you delete a user or identity pool, tags related to the deleted pool can still appear in the console or API calls for up to 30 days after deletion.

Applying tags when you create resources

Use the following commands to assign tags at the moment you create a user pool or identity pool.

Example `create-user-pool` Command with tags

When you create a user pool by using the `create-user-pool` command, you can specify tags with the `--user-pool-tags` parameter:

```
$ aws cognito-idp create-user-pool \  
> --pool-name user-pool-name \  
> --user-pool-tags Stage=Test,CostCenter=80432,Owner=SysEng
```

Key-value pairs for tags must be in the format `key=value`. If you are adding multiple tags, specify them in a comma-separated list.

Example `create-identity-pool` Command with tags

When you create an identity pool by using the `create-identity-pool` command, you can specify tags with the `--identity-pool-tags` parameter:

```
$ aws cognito-identity create-identity-pool \
> --identity-pool-name identity-pool-name \
> --allow-unauthenticated-identities \
> --identity-pool-tags Stage=Test,CostCenter=80432,Owner=SysEng
```

Key-value pairs for tags must be in the format *key=value*. If you are adding multiple tags, specify them in a comma-separated list.

Managing tags using the Amazon Cognito API

You can use the following actions in the Amazon Cognito API to manage the tags for your user pools and identity pools.

API actions for user pool tags

Use the following API actions to assign, view, and remove tags for user pools.

- [TagResource](#)
- [ListTagsForResource](#)
- [UntagResource](#)
- [CreateUserPool](#)

API actions for identity pool tags

Use the following API actions to assign, view, and remove tags for identity pools.

- [TagResource](#)
- [ListTagsForResource](#)
- [UntagResource](#)
- [CreateIdentityPool](#)

Quotas in Amazon Cognito

Amazon Cognito has default quotas, formerly referred to as *limits*, for the maximum number of operations that you can perform in your account. Amazon Cognito also has quotas for the maximum number and size of Amazon Cognito resources.

[Operation quotas \(p. 406\)](#)

- [Quota categorization \(p. 406\)](#)
- [Amazon Cognito user pools API operations with special request rate handling \(p. 406\)](#)
- [Amazon Cognito user pools API operation categories and request rate quotas \(p. 407\)](#)
- [Track quota usage \(p. 412\)](#)
- [Optimize quotas \(p. 413\)](#)
- [Identify quota requirements \(p. 413\)](#)
- [Requesting a quota increase \(p. 414\)](#)

[Resource quotas \(p. 415\)](#)

Operation quotas

Quota categorization

Amazon Cognito has quotas for the maximum number of operations, such as `InitiateAuth` or `RespondToAuthChallenge`, that you can use to perform certain user actions in your applications. These operations are grouped into categories of common use cases, such as `UserAuthentication` or `UserCreation`. For a list of categorized operations, see [Amazon Cognito user pools API operation categories and request rate quotas \(p. 407\)](#). You can track your quota usage, and request increases, by category in the [Service Quotas console](#).

Operation quotas are defined as the maximum number of requests per second (RPS) for all operations within a category. The Amazon Cognito user pools service applies quotas to all operations in each category. For example, the category `UserCreation` includes four operations: `SignUp`, `ConfirmSignUp`, `AdminCreateUser`, and `AdminConfirmSignUp`. It's allocated with a combined quota of 50 RPS. If multiple operations take place at the same time, each operation within this category can call up to 50 RPS separately or combined.

Note

The category quota is enforced for each AWS account across all user pools in an account and Region.

Amazon Cognito user pools API operations with special request rate handling

Operation quotas are measured and enforced for the combined total requests at the category level, except for the `AdminRespondToAuthChallenge` and `RespondToAuthChallenge` operations, where special handling rules are applied.

The `UserAuthentication` category includes four operations: `AdminInitiateAuth`, `InitiateAuth`, `AdminRespondToAuthChallenge`, and `RespondToAuthChallenge`. The `InitiateAuth` and `AdminInitiateAuth` operations are measured and enforced per category quota. The matching

operations `RespondToAuthChallenge` and `AdminRespondToAuthChallenge` are subject to a separate quota that is three times the `UserAuthentication` category limit. This elevated quota accommodates multiple authentication challenges set up in developers' apps. The quota is sufficient to cover the large majority of use cases. Beyond the three-times-per-authentication-call threshold, the excess rate counts towards the `UserAuthentication` category quota.

For example, if your quota for the `UserAuthentication` category is 80 RPS, you can call `RespondToAuthChallenge` or `AdminRespondToAuthChallenge` up to 240 RPS (80 RPS x 3). If the app is set up to have four rounds of challenge per authentication call and you have 70 sign-ins per second, then the total `RespondToAuthChallenge` is 280 RPS (70 x 4), which is 40 RPS above the quota. The extra 40 RPS is added to 70 `InitiateAuth` calls, making the total usage of `UserAuthentication` category 110 RPS (40 + 70). This value exceeds the category quota set at 80 RPS by 30 RPS, so this app is throttled.

Monthly active users

Monthly active users (MAUs) are the benchmark by which Amazon Cognito user pools billing is calculated. MAUs are also used to determine if your user pool is eligible for a quota increase. A user is counted as a MAU if, within a calendar month, there is an identity operation related to that user, such as sign-up, sign-in, token refresh, password change, or a user account attribute is updated. Amazon Cognito also counts a user as active in the month that you take an action that deactivates or deletes the user or its attributes.

Amazon Cognito user pools API operation categories and request rate quotas

Mappings between operations and their respective categories appear in the following table. You can only request to increase adjustable category quotas. For more information, see [Requesting a quota increase \(p. 414\)](#). Adjustable quotas are applied at the account level. Some category operations¹ are constrained at the user pool level to 5 RPS, while the **Default quota** applies to all user pools in an account.

Note

The quota for each category is measured in Monthly Active Users (MAUs). The default quota is intended for accounts with fewer than two million MAUs. If you have less than one million MAUs, you should not request a quota increase.

Category operation quotas are applied across all users in a user pool. Per-user quotas also apply. Your app can request up to 10 *read* operations on each user every second, including signing in or getting profile or device information. Your app can also request up to 10 *write* operations on each user every second, including updating profile information or multi-factor authentication (MFA) settings. You can't change per-user quotas.

Category	Description	Default quota (RPS) ²	Adjustable	Quota increase eligibility ³
<code>UserAuthentication</code>	<ul style="list-style-type: none"> • <code>InitiateAuth</code> • <code>RespondToAuthChallenge</code> • <code>AdminInitiateAuth</code> • <code>AdminRespondToAuthChallenge</code> Operations that authenticate (sign in) a user. These operations are subject to Amazon Cognito user pools API operations with special request	120	Yes	40 RPS for each additional 1 million MAUs

Category	Description	Default quota (RPS) ²	Adjustable	Quota increase eligibility ³
	rate handling (p. 406).			
UserCreation <ul style="list-style-type: none">• SignUp• ConfirmSignUp• AdminCreateUser• AdminConfirmSignUp	Operations that create or confirm an Amazon Cognito <i>native user</i> . This is a user that is created and verified directly by your Amazon Cognito user pools.	50	Yes	10 RPS for each additional 1 million MAUs
UserFederation Amazon Cognito managed operations to federate a user through a third-party IdP into Amazon Cognito.	Operations that federate (authenticate) users with a third-party identity provider into your Amazon Cognito user pools.	25	Yes	10 RPS for each additional 1 million MAUs
UserAccountRecovery <ul style="list-style-type: none">• ChangePassword• ConfirmForgotPassword• ForgotPassword• AdminResetUserPassword• AdminSetUserPassword	Operations that recover a user's account, or change or update a user's password.	30	No	N/A
UserRead <ul style="list-style-type: none">• Admin GetUser• GetUser	Operations that retrieve a user from your user pools.	120	Yes	40 RPS for each additional 1 million MAUs

Category	Description	Default quota (RPS) ²	Adjustable	Quota increase eligibility ³
UserUpdate	Operations that you use to manage <ul style="list-style-type: none"> • AdminAddUserToGroup • AdminDeleteUserAttributes. • AdminUpdateUserAttributes • AdminDeleteUser • AdminDisableUser • AdminEnableUser • AdminLinkProviderForUser • AdminDisableProviderForUser • VerifyUserAttribute • DeleteUser • DeleteUserAttributes • UpdateUserAttributes • AdminUserGlobalSignOut • GlobalSignOut • AdminRemoveUserFromGroup 	25	No	N/A
UserToken	Operations for token management <ul style="list-style-type: none"> • RevokeToken 	120	Yes	40 RPS for each additional 1 million MAUs
UserResourceRead	Operations that retrieve user resource information from Amazon Cognito, such as a remembered device or a group membership <ul style="list-style-type: none"> • AdminGetDevice • AdminListGroupsForUser • AdminListDevices • GetDevice • ListDevices • GetUserAttributeVerifications • ResendConfirmationCode • AdminListUserAuthEvents 	50	Yes	20 RPS for each additional 1 million MAUs

Category	Description	Default quota (RPS) ²	Adjustable	Quota increase eligibility ³
UserResourceUpdate	Operations that update resource information for <ul style="list-style-type: none"> AdminForgetDevice AdminUpdateAuthEventFeedback AdminSetUserMFAPreference AdminSetUserSettings AdminUpdateDeviceStatus UpdateDeviceStatus UpdateAuthEventFeedback ConfirmDevice SetUserMFAPreference SetUserSettings VerifySoftwareToken AssociateSoftwareToken ForgetDevice 	25	No	N/A
UserList	Operations that return a list of users. <ul style="list-style-type: none"> ListUsers ListUsersInGroup 	30	No	N/A
UserPoolRead	Operations that read your user pools. <ul style="list-style-type: none"> DescribeUserPool ListUserPools 	15	No	N/A
UserPoolUpdate	Operations that create, update, or delete your user pools. <ul style="list-style-type: none"> CreateUserPool UpdateUserPool DeleteUserPool 	15	No	N/A

Category	Description	Default quota (RPS) ²	Adjustable	Quota increase eligibility ³
UserPoolResource	Operations that retrieve information about resources, such as groups or resource servers, from a user pool. <ul style="list-style-type: none"> • DescribeIdentityProvider • DescribeResourceServer • DescribeUserImportJob • DescribeUserPoolDomain • GetCSVHeader • GetGroup • GetSigningCertificate • GetIdentityProviderByIdentifier • GetUserPoolMfaConfig • ListGroups • ListIdentityProviders • ListResourceServers • ListTagsForResource • ListUserImportJobs • DescribeRiskConfiguration • GetUICustomization 	20	No	N/A
UserPoolResource	Operations that modify resources, such as groups or resource servers, in a user pool. ¹ <ul style="list-style-type: none"> • AddCustomAttributes • CreateGroup • CreateIdentityProvider • CreateResourceServer • CreateUserImportJob • CreateUserPoolDomain • DeleteGroup • DeleteIdentityProvider • DeleteResourceServer • DeleteUserPoolDomain • SetUserPoolMfaConfig • StartUserImportJob • StopUserImportJob • UpdateGroup • UpdateIdentityProvider • UpdateResourceServer • UpdateUserPoolDomain • SetRiskConfiguration • SetUICustomization • TagResource • UntagResource 	15	No	N/A

Category	Description	Default quota (RPS) ²	Adjustable	Quota increase eligibility ³
UserPoolClientRead	Operations that retrieve <ul style="list-style-type: none"> DescribeUserPoolClient information about your user pool clients.¹ ListUserPoolClients 	15	No	N/A
UserPoolClientUpdate	Operations that create, update, and delete your user pool clients. ¹ <ul style="list-style-type: none"> CreateUserPoolClient DeleteUserPoolClient UpdateUserPoolClient 	15	No	N/A
ClientAuthentication	Operations that generate credentials to be used in authorizing machine-to-machine requests <ul style="list-style-type: none"> client_credentials grant type requests to the TOKEN endpoint. 	150	No	N/A

¹ Any individual operation in this category has a constraint that prevents the operation from being called at a rate higher than 5 RPS for a single user pool.

² Request rate for all user pools with fewer than 2 million MAUs.

³ Request rate increases will be considered when you have between 1 million and 10 million MAUs. Quota increase requests are evaluated based on current utilization, existing or expected MAUs, and your implementation of optimization best practices. Amazon Cognito reserves the right to deny quota increase requests to prevent misuse and protect service availability for all customers. If you have more than 10 million MAUs, we recommend that you work with an AWS solutions architect to design your solution in a way that is optimized for Amazon Cognito quotas. Please contact your AWS account team for help.

Track quota usage

Amazon Cognito generates `CallCount` and `ThrottleCount` metrics in Amazon CloudWatch for each API operation category at the account level. You can use `CallCount` to track the total number of calls customers made related to a category. You can use `ThrottleCount` to track the total number of throttled calls related to a category. You can use the `CallCount` and `ThrottleCount` metrics with the `Sum` statistic to count the total number of calls in a category. For more information, see [CloudWatch usage metrics](#).

When monitoring service quotas, *utilization* is the percentage of a service quota in use. For example, if the quota value is 200 resources, and 150 resources are in use, the utilization is 75%. *Usage* is the number of resources or operations in use for a service quota.

Tracking usage through CloudWatch metrics

You can track and collect Amazon Cognito user pools utilization metrics using CloudWatch. The CloudWatch dashboard displays metrics about every AWS service that you use. With CloudWatch, you can create metric alarms to notify you or change a specific resource that you are monitoring. For more information about CloudWatch metrics, see [Track your CloudWatch usage metrics \(p. 369\)](#).

Tracking utilization through Service Quotas metrics

Amazon Cognito user pools are integrated with Service Quotas, which is a browser-based interface that you can use to view and manage your service quota usage. In the Service Quotas console, you can look up the value of a specific quota, view monitoring information, request a quota increase, or set up CloudWatch alarms. After your account has been active a while, you can view a graph of your resource utilization.

For more information on viewing quotas in the Service Quotas console, see [Viewing Service Quotas](#).

Identify quota requirements

Important

If you increase Amazon Cognito quotas for categories such as `UserAuthentication`, `UserCreation`, or `AccountRecovery`, you may need to increase quotas for other services. For example, messages that Amazon Cognito sends with Amazon Simple Notification Service (Amazon SNS) or Amazon Simple Email Service (Amazon SES) can fail if request rate quotas are insufficient in those services.

To calculate quota requirements, determine how many active users will interact with your application in a specific time period. For example, if you expect your application to sign in an average of one million active users within an eight-hour period, then you must be able to authenticate an average of 35 users per second.

In addition, if you assume that the average user session is two hours, and you configure tokens to expire after an hour, each user must refresh their tokens once during their session. The required average quota for the `UserAuthentication` category to support this load is 70 RPS.

If you assume a peak-to-average ratio of 3:1 by accounting for the variance of user sign-in frequency during the eight-hour period, then you need the desired `UserAuthentication` quota of 200 RPS.

Note

If you call multiple operations for each user action, you must sum up the individual operation call rates at the category level.

Optimize quotas

Follow one of the following methods to handle a peak call rate.

Retry the attempt after a back-off waiting period

You can catch errors with each API call, and then re-try the attempt after a back-off period. You can adjust the back-off algorithm according to business needs and load. Amazon SDKs have built-in retry logic. For more information, see [Tools to Build on AWS](#).

Use an external database for frequently updated attributes

If your application requires several calls to a user pool to read or write custom attributes, use external storage. You can use your preferred database to store custom attributes or use a cache layer to load a user profile during sign-in. You can reference this profile from the cache when needed, instead of reloading the user profile from a user pool.

Validate JWT tokens on the client side

Applications must validate JWT tokens before trusting them. You can verify the signature and validity of tokens on the client side without sending API calls to a user pool. After the token is validated, you can trust claims in the token and use the claims instead of making more `getUser` API calls. For more information, see [Verifying a JSON Web Token](#).

Throttle traffic to your web application with a waiting room

If you expect traffic from a large number of users signing in during a time-bound event, such as taking an exam or attending a live event, you can optimize request traffic with self-throttling

mechanisms. You can, for example, set up a waiting room where users can stand by until a session is available, allowing you to process requests when you have available capacity. See the [AWS Virtual Waiting Room solution](#) for a reference architecture of a waiting room.

Requesting a quota increase

Amazon Cognito has a quota for the maximum number of user pool operations that you can perform in your account. You can request an increase to the adjustable API request rate quotas in Amazon Cognito. To request a quota increase, use the Service Quotas console, the Service limit increase form, or the `RequestServiceQuotaIncrease` or `ListAWSDefaultServiceQuotas` API operations.

- To request a quota increase using the Service Quotas console, see [Requesting a API quota increase](#) in the *Service Quotas User Guide*.
- If the quota isn't available in Service Quotas, use the [Service limit increase form](#).

Important

Only adjustable quotas can be increased. Quota increase requests are evaluated based on current utilization, existing or expected MAUs, and your implementation of optimization best practices. When you submit a quota increase request, provide as much information about your current usage, projected usage, and optimization methods as you are able to. For quota increases that increase your request rate, your current average request rate should already be close to the maximum rate at your current quota. See [Amazon Cognito user pools API operation categories and request rate quotas \(p. 407\)](#) to learn more about adjustable quotas.

Consider an example scenario where your application is growing to 5 million MAUs. Your current average utilization in the UserAuthentication category operations is high: 60%, or 72 requests per second against your current quota of 120. You can request a UserAuthentication quota increase for 40 requests per second for each of the 3 million MAUs above the first two million, bringing your quota to a maximum of 240 requests per second.

Amazon Cognito identity pools (federated identities) API operation request rate quotas

Operation	Description	Quota in requests per second	Adjustable	Quota increase eligibility
GetId	Retrieve an identity ID from an identity pool.	25	Yes	Contact your account team.
GetOpenIdToken	Retrieve an OpenID token from an identity pool in the classic workflow.	200	Yes	Contact your account team.
GetCredentialsForIdentity	Retrieve AWS credentials from an identity pool in the enhanced workflow.	200	Yes	Contact your account team.
GetOpenIdTokenForDeveloperIdentity	Retrieve a temporary OpenID token	120	Yes	Contact your account team.

Operation	Description	Quota in requests per second	Adjustable	Quota increase eligibility
	from an identity pool in the developer workflow.			

Resource quotas

Resource quotas define the maximum number and size of your resources. You can request to increase adjustable resource quotas in Amazon Cognito. To request a quota increase, use the Service Quotas console or the [Service limit increase form](#). To request a quota from the Service Quotas console, see [Requesting a quota increase](#) in the *Service Quotas User Guide*. If the quota isn't available in Service Quotas, use the [Service limit increase form](#).

Amazon Cognito user pools resource quotas

Resource	Quota	Adjustable	Maximum quota
App clients per user pool	1,000	Yes	10,000
User pools per account	1,000	Yes	10,000
User import jobs per user pool	1,000	No	N/A
Identity providers per user pool	300	Yes	1,000
Resource servers per user pool	25	Yes	300
Users per user pool	40,000,000	Yes	Contact your account team.
Custom attributes per user pool	50	No	N/A
Characters per attribute	2,048 bytes	No	N/A
Characters in custom attribute name	20	No	N/A
Required minimum password characters in password policy	6–99	No	N/A
Email messages sent daily per AWS account ¹	50	No	N/A
Characters in email subject	140	No	N/A
Characters in email message	20,000	No	N/A

Resource	Quota	Adjustable	Maximum quota
Characters in SMS verification message	140	No	N/A
Characters in password	256	No	N/A
Characters in identity provider name	40	No	N/A
Identifiers per identity provider	50	No	N/A
Identities linked to a user	5	No	N/A
Callback URLs per app client	100	No	N/A
Logout URLs per app client	100	No	N/A
Scopes per resource server	100	No	N/A
Scopes per app client	50	No	N/A
Custom domains per account	4	No	N/A
Groups to which each user can belong	100	No	N/A
Groups per user pool	10,000	No	N/A

¹This quota applies only if you are using the default email feature for an Amazon Cognito user pool. To enable a higher email delivery volume, configure your user pool to use your Amazon SES email configuration. For more information, see [Email settings for Amazon Cognito user pools \(p. 181\)](#).

Amazon Cognito user pools session validity parameters

Token	Quota
ID token	5 minutes – 1 day
Refresh token	1 hour – 3,650 days
Access token	5 minutes – 1 day
Hosted UI session cookie	1 hour

User pools code security resource quotas (non-adjustable)

Resource	Quota
Sign-up confirmation code validity period	24 hours
User attribute verification code validity period	24 hours

Resource	Quota
Multi-factor authentication (MFA) code validity period	3 minutes
Forgot password code validity period	1 hour
Maximum number of <code>ConfirmForgotPassword</code> calls per user	5–20 attempts per hour, depending on risk score
Maximum number of <code>ResendConfirmationCode</code> calls per user	5 attempts per hour
Maximum number of <code>ConfirmUser</code> calls per user	15 attempts per hour
Maximum number of <code>ChangePassword</code> calls per user	5 attempts per hour
Maximum number of <code>GetUserAttributeVerificationCode</code> calls per user	5 attempts per hour
Maximum number of <code>VerifyUserAttribute</code> calls per user	15 attempts per hour

Amazon Cognito identity pools (federated identities) resource quotas

Resource	Quota	Adjustable	Maximum quota
Identity pools per account	1,000	No	N/A
Amazon Cognito user pool providers per identity pool	50	Yes	1000
Character length of an identity pool name	128 bytes	No	N/A
Character length of a login provider name	2,048 bytes	No	N/A
Identities per identity pool	Unlimited	No	N/A
Identity providers for which role mappings can be specified	10	No	N/A
Results from a single list or lookup call	60	No	N/A
Role-based access control (RBAC) rules	25	No	N/A

Amazon Cognito Sync resource quotas

Resource	Quota	Adjustable	Maximum quota
Datasets per identity	20	Yes	Contact your account team.
Records per dataset	1,024	Yes	Contact your account team.
Size of a single dataset	1 MB	Yes	Contact your account team.
Characters in dataset name	128 bytes	No	N/A
Waiting time for a bulk publish after a successful request	24 hours	No	N/A

Amazon Cognito API references

Use the following links to go to the related API reference pages.

- [Amazon Cognito user pools API reference](#)
- [Amazon Cognito identity pools \(federated identities\) API reference](#)
- [Amazon Cognito Sync API reference](#)
- [Amazon Cognito user pools Auth API reference](#)

User pool authentication and authorization endpoints reference

Amazon Cognito creates the public webpages listed here when you set up domains for your app clients. They include the hosted UI, where your users can sign up and sign in (the [Login endpoint \(p. 430\)](#)), and sign out (the [Logout endpoint \(p. 432\)](#)). For more information about these resources, see [Using the Amazon Cognito hosted UI for sign-up and sign-in \(p. 38\)](#).

These pages also include the public web resources that allow your user pool to communicate with third-party SAML, OpenID Connect (OIDC) and OAuth 2.0 identity providers (IdPs). To sign in a user with a federated identity provider, your users must initiate a request to the [Login endpoint \(p. 430\)](#) or the [Authorize endpoint \(p. 420\)](#). Your app can also sign in *native users* with the [Amazon Cognito API](#). Native users are those that you created or who signed up in your user pool.

In addition to the OAuth 2.0 REST API endpoints, Amazon Cognito also integrates with SDKs for Android, iOS, JavaScript, and more. The SDKs provide tools to interact both with your user pool API endpoints and Amazon Cognito API service endpoints. For more information about service endpoints, see [Amazon Cognito Identity endpoints and quotas](#).

Along with the endpoints in this guide, Amazon Cognito creates the following endpoints when you assign a domain to your user pool.

Additional user pool endpoints

Endpoint URL	Description
<code>https://cognito-idp.<i>Region</i>.amazonaws.com/<i>your user pool ID</i>.well-known/openid-configuration</code>	A directory of the OIDC architecture of your user pool.
<code>https://cognito-idp.<i>Region</i>.amazonaws.com/<i>your user pool ID</i>.well-known/jwks.json</code>	Public keys that you can use to validate Amazon Cognito tokens.
<code>https://<i>Your user pool domain</i>/oauth2/idpresponse</code>	Social identity providers must redirect your users to this endpoint with an authorization code. Amazon Cognito redeems the code for a token when it authenticates your federated user.

Endpoint URL	Description
<code>https://<i>Your user pool domain</i>/saml2/idpresponse</code>	To authenticate your SAML 2.0 federated user, your identity provider must redirect your users to this endpoint with a SAML response.

For more information on the OpenID and OAuth standards, see [OpenID Connect 1.0](#) and [OAuth 2.0](#).

Topics

- [Authorize endpoint \(p. 420\)](#)
- [Token endpoint \(p. 425\)](#)
- [UserInfo endpoint \(p. 429\)](#)
- [Login endpoint \(p. 430\)](#)
- [Logout endpoint \(p. 432\)](#)
- [Revoke endpoint \(p. 433\)](#)

Authorize endpoint

The `/oauth2/authorize` endpoint signs in the user. Its function is similar to the [Login endpoint \(p. 430\)](#).

GET `/oauth2/authorize`

The `/oauth2/authorize` endpoint only supports `HTTPS GET`. The user pool client typically makes this request through a browser.

The authorization server requires HTTPS. For more information about the OpenID Connect specification, see [Authorization Endpoint](#).

Request parameters

response_type

The response type. Must be `code` or `token`. Indicates whether the client wants an authorization code for the user (authorization code grant flow), or directly issues tokens for the user (implicit flow).

Required.

client_id

The Client ID.

Must be a client that you already registered in the user pool and that you qualified for federation.

Required.

redirect_uri

The URL where the authentication server redirects the browser after Amazon Cognito authorizes the user.

A redirect uniform resource identifier (URI) must have the following attributes:

- It must be an absolute URI.
- You must have pre-registered the URI with a client.

- It can't include a fragment component.

See [OAuth 2.0 - Redirection Endpoint](#).

Amazon Cognito requires that your redirect URI use HTTPS, except for `http://localhost`, which you can set as a callback URL for testing purposes.

Amazon Cognito also supports app callback URLs such as `myapp://example`.

Required.

state

When your app adds a *state* parameter to a request, Amazon Cognito returns its value to your app when the `/oauth2/authorize` endpoint redirects your user.

Add this value to your requests to guard against [CSRF](#) attacks.

You can't set the value of a *state* parameter to a URL-encoded JSON string. To pass a string that matches this format in a *state* parameter, encode the string to Base64, then decode it in your app.

Optional but recommended.

identity_provider

Add this parameter to bypass the hosted UI and redirect your user to a provider sign-in page. The value of the *identity_provider* parameter is the name of the identity provider (IdP) as it appears in your user pool.

- For social providers, you can use the *identity_provider* values **Facebook**, **Google**, **LoginWithAmazon**, and **SignInWithApple**.
- For Amazon Cognito user pools, use the value **COGNITO**.
- For SAML 2.0 and OpenID Connect (OIDC) identity providers (IdPs), use the name that you assigned to the IdP in your user pool.

Optional.

idp_identifier

Add this parameter to redirect to a provider with an alternative name for the *identity_provider* name. You can enter identifiers for your SAML 2.0 and OIDC IdPs from the **Sign-in experience** tab of the Amazon Cognito console.

Optional.

scope

Can be a combination of any system-reserved scopes or custom scopes that are associated with a client. Scopes must be separated by spaces. System reserved scopes are `openid`, `email`, `phone`, `profile`, and `aws.cognito.signin.user.admin`. Any scope used must be associated with the client, or it will be ignored at runtime.

If the client doesn't request any scopes, the authentication server uses all scopes that are associated with the client.

An ID token is only returned if `openid` scope is requested. The access token can be only used against Amazon Cognito user pools if `aws.cognito.signin.user.admin` scope is requested. The `phone`, `email`, and `profile` scopes can only be requested if `openid` scope is also requested. These scopes dictate the claims that go inside the ID token.

Optional.

code_challenge_method

The method that you used to generate the challenge. The [PKCE RFC](#) defines two methods, S256 and plain; however, Amazon Cognito authentication server supports only S256.

Optional.

code_challenge

The challenge that you generated from the `code_verifier`.

Required only when you specify a `code_challenge_method` parameter.

nonce

A random value that you can add to the request. The nonce value that you provide is included in the ID token that Amazon Cognito issues. You can use a nonce value to guard against replay attacks.

Examples requests with positive responses

Authorization code grant

Sample Request

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
    response_type=code&
    client_id=ad398u21ijw3s9w3939&
    redirect_uri=https://YOUR_APP/redirect_uri&
    state=STATE&
    scope=openid+profile+aws.cognito.signin.user.admin
```

Sample response

The Amazon Cognito authentication server redirects back to your app with the authorization code and state. The code and state must be returned in the query string parameters and not in the fragment. A query string is the part of a web request that appears after a '?' character; the string can contain one or more parameters separated by '&' characters. A fragment is the part of a web request that appears after a '#' character to specify a subsection of a document.

Note

The response returns a one time use code that is valid for five minutes.

```
HTTP/1.1 302 Found
Location: https://YOUR_APP/redirect_uri?
code=AUTHORIZATION_CODE&state=STATE
```

Authorization code grant with PKCE

Sample request

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
    response_type=code&
    client_id=ad398u21ijw3s9w3939&
    redirect_uri=https://YOUR_APP/redirect_uri&
    state=STATE&
    scope=aws.cognito.signin.user.admin&
```

```
code_challenge_method=S256&
code_challenge=CODE_CHALLENGE
```

Sample response

The authentication server redirects back to your app with the authorization code and state. The code and state must be returned in the query string parameters and not in the fragment.

```
HTTP/1.1 302 Found
Location: https://YOUR_APP/redirect_uri?
code=AUTHORIZATION_CODE&state=STATE
```

Token grant without openid scope

Sample request

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
    response_type=token&
    client_id=ad398u21ijw3s9w3939&
    redirect_uri=https://YOUR_APP/redirect_uri&
    state=STATE&
    scope=aws.cognito.signin.user.admin
```

Sample response

The Amazon Cognito authorization server redirects back to your app with access token. Because openid scope was not requested, Amazon Cognito doesn't return an ID token. Also, Amazon Cognito doesn't return a refresh token in this flow. Amazon Cognito returns the access token and state in the fragment and not in the query string.

```
HTTP/1.1 302 Found
Location: https://YOUR_APP/
redirect_uri#access_token=ACCESS_TOKEN&token_type=bearer&expires_in=3600&state=STATE
```

Token grant with openid scope

Sample request

```
GET
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/
authorize?
    response_type=token&
    client_id=ad398u21ijw3s9w3939&
    redirect_uri=https://YOUR_APP/redirect_uri&
    state=STATE&
    scope=aws.cognito.signin.user.admin+openid+profile
```

Sample response

The authorization server redirects back to your app with access token and ID token (because openid scope was included).

```
HTTP/1.1 302 Found
Location: https://YOUR_APP/
redirect_uri#id_token=ID_TOKEN&access_token=ACCESS_TOKEN&token_type=bearer&expires_in=3600&state=STATE
```

Examples of negative responses

The following are examples of negative responses:

- If `client_id` and `redirect_uri` are valid, but the request parameters aren't formatted correctly, the authentication server redirects the error to client's `redirect_uri` and appends an error message in a URL parameter. Examples of incorrect formatting are a request doesn't include a `response_type` parameter, if the response provides `code_challenge` but not `code_challenge_method`, or that `code_challenge_method` is not 'S256'.

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?
error=invalid_request
```

- If the client requests `code` or `token` in `response_type` but doesn't have permission for these requests, the Amazon Cognito authorization server returns `unauthorized_client` to client's `redirect_uri`, as follows:

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?
error=unauthorized_client
```

- If the client requests scope that is unknown, malformed, or not valid, the Amazon Cognito authorization server returns `invalid_scope` to the client's `redirect_uri`, as follows:

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?error=invalid_scope
```

- If there is any unexpected error in the server, the authentication server returns `server_error` to client's `redirect_uri`. Because the HTTP 500 error doesn't get sent to the client, don't display the error to the user in the browser. The following error should result:

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?error=server_error
```

- When Amazon Cognito authenticates through federation to third-party IdPs, Amazon Cognito might experience connection issues such as the following:
 - If a connection timeout occurs while requesting token from the IdP, the authentication server redirects the error to the client's `redirect_uri` as follows:

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?
error=invalid_request&error_description=Timeout+occurred+in+calling+IdP
+token+endpoint
```

- If a connection timeout occurs while calling the `jwks` endpoint for `id_token` validation, the authentication server redirects the error to the client's `redirect_uri` as follows:

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?
error=invalid_request&error_description=error_description=Timeout+in+calling
+jwks+uri
```

- When authenticating by federating to third-party IdPs, the providers may return error responses due to configuration errors or otherwise such as the following:
 - If an error response is received from other providers, the authentication server redirects the error to the client's `redirect_uri` as follows:

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?
error=invalid_request&error_description=[IdP name]+Error++-[status
code]+error getting token
```

- If an error response is received from Google, the authentication server redirects the error to the client's `redirect_uri` as follows: `HTTP 1.1 302 Found Location: https:////`

```
client_redirect_uri?error=invalid_request&error_description=Google+Error+-+[status code]+[Google provided error code]
```

- When Amazon Cognito encounters an exception in the communication protocol while it makes a connection to an external IdP, the authentication server redirects the error to the client's `redirect_uri` with either of the following messages:
 - HTTP 1.1 302 Found Location: `https://client_redirect_uri?error=invalid_request&error_description=Connection+reset`
 - HTTP 1.1 302 Found Location: `https://client_redirect_uri?error=invalid_request&error_description=Read+timed+out`

Token endpoint

The `/oauth2/token` endpoint gets the user's tokens.

POST /oauth2/token

The `/oauth2/token` endpoint only supports `HTTPS POST`. Your app makes requests to this endpoint directly, not through the user's browser.

For more information about the token endpoint from the OpenID Connect specification, see [Token Endpoint](#).

Request parameters in header

Authorization

If the client was issued a secret, the client must pass its `client_id` and `client_secret` in the authorization header through Basic HTTP authorization. The authorization header string is `Basic Base64Encode(client_id:client_secret)`. The following example is an authorization header for app client `djc98u3jiedmi283eu928` with client secret `abcdef01234567890`, using the Base64-encoded version of the string `djc98u3jiedmi283eu928:abcdef01234567890`.

```
Authorization: Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjMONTY3ODkw
```

Content-Type

Must always be '`application/x-www-form-urlencoded`'.

Request parameters in body

grant_type

Grant type.

Must be `authorization_code` or `refresh_token` or `client_credentials`. You can request an access token for a custom scope from the token endpoint when, in the app client, the requested scope is enabled, you have configured a client secret, and you have allowed `client_credentials` grants.

Required.

client_id

The client ID

Must be a preregistered client in the user pool. The client must be enabled for Amazon Cognito federation.

Required if the client is public and does not have a secret.

scope

Can be a combination of any custom scopes associated with an app client. Any scope that you request must be activated for the app client, or Amazon Cognito will ignore it. If the client doesn't request any scopes, the authentication server uses all custom scopes associated with the client.

Optional. Only used if the `grant_type` is `client_credentials`.

redirect_uri

Must be the same `redirect_uri` that was used to get `authorization_code` in `/oauth2/authorize`.

Required only if `grant_type` is `authorization_code`.

refresh_token

The refresh token.

Note

The token endpoint returns `refresh_token` only when the `grant_type` is `authorization_code`.

code

Required if `grant_type` is `authorization_code`.

code_verifier

The proof key.

Required if `grant_type` is `authorization_code` and the authorization code was requested with PKCE.

Examples requests with positive responses

Exchanging an authorization code for tokens

Sample request

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token&
      Content-Type='application/x-www-form-urlencoded'&
      Authorization=Basic aSdxsd892iujendek328uedj

      grant_type=authorization_code&
      client_id=djc98u3jiedmi283eu928&
      code=AUTHORIZATION_CODE&
      redirect_uri=com.myclientapp://myclient/redirect
```

Sample response

```
HTTP/1.1 200 OK

Content-Type: application/json

{
  "access_token": "eyJz9sdfsdfsdfsd",
  "token_type": "Bearer",
  "expires_in": 3600,
  "refresh_token": "aSdxsd892iujendek328uedj"
}
```

```
"id_token": "dmcxd329ujdmkemd349r",
"token_type": "Bearer",
"expires_in": 3600
}
```

Note

The token endpoint returns `refresh_token` only when the `grant_type` is `authorization_code`.

Exchanging client credentials for an access token

Sample request

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token >
Content-Type='application/x-www-form-urlencoded' &
Authorization=Basic aSdx892iujendek328uedj

grant_type=client_credentials&
scope={resourceServerIdentifier1}/{scope1}
{resourceServerIdentifier2}/{scope2}
```

Sample response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "access_token": "eyJz9sdfsdfsdfsd",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

Exchanging an authorization code grant with PKCE for tokens

Sample request

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token
Content-Type='application/x-www-form-urlencoded' &
Authorization=Basic aSdx892iujendek328uedj

grant_type=authorization_code&
client_id=djc98u3jiedmi283eu928&
code=AUTHORIZATION_CODE&
code_verifier=CODE_VERIFIER&
redirect_uri=com.myclientapp://myclient/redirect
```

Sample response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "access_token": "eyJz9sdfsdfsdfsd",
  "refresh_token": "dn43ud8uj32nk2je",
  "id_token": "dmcxd329ujdmkemd349r",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

}

Note

The token endpoint returns `refresh_token` only when the `grant_type` is `authorization_code`.

[Exchanging a refresh token for tokens](#)

Sample request

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token >
Content-Type='application/x-www-form-urlencoded'&
Authorization=Basic aSdx892iujendek328uedj

grant_type=refresh_token&
client_id=djc98u3jiedmi283eu928&
refresh_token=REFRESH_TOKEN
```

Sample response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "id_token": "eyJz9sdfsdfsdfsd",
  "access_token": "dmcxd329ujdmkemd349r",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

Note

The token endpoint returns `refresh_token` only when the `grant_type` is `authorization_code`.

[Examples of negative responses](#)

Sample error response

```
HTTP/1.1 400 Bad Request
Content-Type: application/json; charset=UTF-8

{
  "error": "invalid_request|invalid_client|invalid_grant|
unauthorized_client|unsupported_grant_type|"
}
```

invalid_request

The request is missing a required parameter, includes an unsupported parameter value (other than `unsupported_grant_type`), or is otherwise malformed. For example, `grant_type` is `refresh_token` but `refresh_token` is not included.

invalid_client

Client authentication failed. For example, when the client includes `client_id` and `client_secret` in the authorization header, but there's no such client with that `client_id` and `client_secret`.

invalid_grant

Refresh token has been revoked.

Authorization code has been consumed already or does not exist.

unauthorized_client

Client is not allowed for code grant flow or for refreshing tokens.

unsupported_grant_type

Returned if `grant_type` is anything other than `authorization_code` or `refresh_token` or `client_credentials`.

UserInfo endpoint

The `/oauth2/userInfo` endpoint returns information about the authenticated user.

GET /oauth2/userInfo

The user pool client makes requests to this endpoint directly and not through a browser.

For more information, see [UserInfo Endpoint](#) in the OpenID Connect (OIDC) specification.

Topics

- [Request parameters in header \(p. 429\)](#)
- [Sample request \(p. 429\)](#)
- [Sample positive response \(p. 429\)](#)
- [Sample negative responses \(p. 430\)](#)

Request parameters in header

Authorization: Bearer <access_token>

Pass the access token using the authorization header field.

Required.

Sample request

```
GET https://<your-user-pool-domain>/oauth2/userInfo
    Authorization: Bearer <access_token>
```

Sample positive response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
{
    "sub": "248289761001",
    "name": "Jane Doe",
    "given_name": "Jane",
    "family_name": "Doe",
    "preferred_username": "j.doe",
```

```
        "email": "janedoe@example.com"  
    }
```

For a list of OIDC claims, see [Standard Claims](#).

Sample negative responses

Invalid request

```
HTTP/1.1 400 Bad Request  
    WWW-Authenticate: error="invalid_request",  
                    error_description="Bad OAuth2 request at UserInfo Endpoint"
```

invalid_request

The request is missing a required parameter, includes an unsupported parameter value, or is otherwise malformed.

Invalid token

```
HTTP/1.1 401 Unauthorized  
    WWW-Authenticate: error="invalid_token",  
                    error_description="Access token is expired, disabled, or deleted, or the user has globally signed out."
```

invalid_token

The access token is expired, revoked, malformed, or invalid.

Login endpoint

The `/login` endpoint signs the user in. It loads the login page and presents the authentication options configured for the client to the user.

GET /login

The `/login` endpoint only supports `HTTPS GET`. The user pool client makes this request through a system browser. System browsers for JavaScript include Chrome or Firefox. Android browsers include Custom Chrome Tab. iOS browsers include Safari View Control.

Request parameters

client_id

The app client ID for your app. To obtain an app client ID, register the app in the user pool. For more information, see [Configuring a user pool app client \(p. 223\)](#).

Required.

redirect_uri

The URI where the user is redirected after a successful authentication. It should be configured on `response_type` of the specified `client_id`.

Required.

response_type

The OAuth response type, which can be `code` for code grant flow and `token` for implicit flow.

Required.

state

When your app adds a `state` parameter to a request, Amazon Cognito returns its value to your app when the `/oauth2/login` endpoint redirects your user.

Add this value to your requests to guard against [CSRF](#) attacks.

You can't set the value of a `state` parameter to a URL-encoded JSON string. To pass a string that matches this format in a `state` parameter, encode the string to Base64, then decode it in your app.

Optional but recommended.

scope

Can be a combination of any system-reserved scopes or custom scopes associated with a client. Scopes must be separated by spaces. System reserved scopes are `openid`, `email`, `phone`, `profile`, and `aws.cognito.signin.user.admin`. Any scope that you request must be activated for the app client, or Amazon Cognito will ignore it.

If the client doesn't request any scopes, the authentication server uses all scopes associated with the client.

An ID token is only returned if an `openid` scope is requested. The access token can only be used against Amazon Cognito user pools if an `aws.cognito.signin.user.admin` scope is requested. The `phone`, `email`, and `profile` scopes can only be requested if an `openid` scope is also requested. These scopes dictate the claims that go inside the ID token.

Optional.

code_challenge_method

The method used to generate the challenge. The [PKCE RFC](#) defines two methods, S256 and plain; however, Amazon Cognito authentication server supports only S256.

Optional.

code_challenge

The generated challenge from the `code_verifier`.

Required only when the `code_challenge_method` is specified.

Sample request: Prompt the user to sign in

This example displays the login screen.

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/login?
  response_type=code&
  client_id=ad398u21ijw3s9w3939&
  redirect_uri=https://YOUR_APP/redirect_uri&
  state=STATE&
  scope=openid+profile+aws.cognito.signin.user.admin
```

Sample Response

The authentication server redirects to your app with the authorization code and state. The server must return the code and state in the query string parameters and not in the fragment.

```
HTTP/1.1 302 Found
Location: https://YOUR_APP/redirect_uri?
code=AUTHORIZATION_CODE&state=STATE
```

For more example requests, and examples of positive and negative responses, see [Authorize endpoint \(p. 420\)](#).

Logout endpoint

The `/logout` endpoint signs the user out and redirects either to an authorized sign-out URL for your app client, or to the `/login` endpoint.

GET /logout

The `/logout` endpoint only supports `HTTPS GET`. The user pool client typically makes this request through the system browser. The browser is typically Custom Chrome Tab in Android or Safari View Control in iOS.

Request parameters

client_id

The app client ID for your app. To get an app client ID, you must register the app in the user pool. For more information, see [Configuring a user pool app client \(p. 223\)](#).

Required.

logout_uri

Redirect your user to a custom sign-out page with a `logout_uri` parameter. Set its value to the app client **sign-out URL** where you want to redirect your user after they sign out. Use `logout_uri` only with a `client_id` parameter. For more information, see [Configuring a user pool app client \(p. 42\)](#).

You can also use the `logout_uri` parameter to redirect your user to the sign-in page for another app client. Set the sign-in page for the other app client as an **Allowed callback URL** in your app client. In your request to the `/logout` endpoint, set the value of the `logout_uri` parameter to the URL-encoded sign-in page.

Amazon Cognito requires either a `logout_uri` or a `redirect_uri` parameter in your request to the `/logout` endpoint. A `logout_uri` parameter redirects your user to another website.

redirect_uri

Redirect your user to your sign-in page to authenticate with a `redirect_uri` parameter. Set its value to the app client **Allowed callback URL** where you want to redirect your user after they sign in again. Add `client_id`, `scope`, `state`, and `response_type` parameters that you want to pass to your `/login` endpoint.

Amazon Cognito requires either a `logout_uri` or a `redirect_uri` parameter in your request to the `/logout` endpoint. When you want to redirect your user to your `/login` endpoint to reauthenticate and pass tokens to your app, add a `redirect_uri` parameter.

response_type

The OAuth 2.0 response that you want to receive from Amazon Cognito after your user signs in. `code` and `token` are the valid values for the `response_type` parameter.

Required if you use a *redirect_uri* parameter.

state

When your app adds a *state* parameter to a request, Amazon Cognito returns its value to your app when the /oauth2/logout endpoint redirects your user.

Add this value to your requests to guard against [CSRF](#) attacks.

You can't set the value of a *state* parameter to a URL-encoded JSON string. To pass a string that matches this format in a *state* parameter, encode the string to Base64, then decode it in your app.

Strongly recommended if you use a *redirect_uri* parameter.

scope

The OAuth 2.0 scopes that you want to request from Amazon Cognito after you sign them out with a *redirect_uri* parameter. Amazon Cognito redirects your user to the /login endpoint with the *scope* parameter in your request to the /logout endpoint.

Optional if you use a *redirect_uri* parameter. If you don't include a *scope* parameter, Amazon Cognito redirects your user to the /login endpoint with a *scope* parameter. When Amazon Cognito redirects your user and automatically populates scope, the parameter includes all authorized scopes for your app client.

Sample requests

Example 1: Log out and redirect user to client

This example clears the existing session and redirects the user to the client. An example request like this one, with a *logout_uri* parameter, also requires a *client_id* parameter.

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/logout?  
client_id=ad398u21ijw3s9w3939&  
logout_uri=https://myclient/logout
```

Example 2: Log out and prompt the user to sign in as another user

This example clears the existing session and shows the login screen. The example uses the same parameters as you would use in a request to the [Authorize endpoint \(p. 420\)](#).

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/logout?  
response_type=code&  
client_id=ad398u21ijw3s9w3939&  
redirect_uri=https://YOUR_APP/redirect_uri&  
state=STATE&  
scope=openid+profile+aws.cognito.signin.user.admin
```

Revoke endpoint

The /oauth2/revoke endpoint revokes all of the access tokens that the specified refresh token generated. After the endpoint revokes the tokens, you can't use the revoked tokens to access APIs that Amazon Cognito tokens authenticate.

POST /oauth2/revoke

The /oauth2/revoke endpoint only supports `HTTPS POST`. The user pool client makes requests to this endpoint directly and not through the system browser.

Request parameters in header

Authorization

If your app client has a client secret, the app must pass its `client_id` and `client_secret` in the authorization header through Basic HTTP authorization. The secret is `Basic Base64Encode(client_id:client_secret)`.

Content-Type

Must always be '`application/x-www-form-urlencoded`'.

Request parameters in body

token

The refresh token that the client wants to revoke. The request also revokes all access tokens that Amazon Cognito issued from this refresh token.

Required.

client_id

The app client ID for the token that you want to revoke.

Required if the client is public and doesn't have a secret.

Revocation request examples

Example 1: Revoke a token for an app client without a client secret

```
POST /oauth2/revoke HTTP/1.1
Host: https://mydomain.auth.us-east-1.amazoncognito.com
Accept: application/json
Content-Type: application/x-www-form-urlencoded
token=2YotnFZFEjr1zCsicMWpAA&
client_id=djc98u3jiedmi283eu928
```

Example 2: Revoke a token for an app client with a client secret

```
POST /oauth2/revoke HTTP/1.1
Host: https://mydomain.auth.us-east-1.amazoncognito.com
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czzCaGRSa3F0MzpnWDFmQmF0M2JW
token=2YotnFZFEjr1zCsicMWpAA
```

Revocation error response

A successful response contains an empty body. The error response is a JSON object with an `error` field and, in some cases, an `error_description` field.

Endpoint errors

- If the token isn't present in the request or if the feature is disabled for the app client, you receive an HTTP 400 and error `invalid_request`.
- If the token that Amazon Cognito sent in the revocation request isn't a refresh token, you receive an HTTP 400 and error `unsupported_token_type`.
- If the client credentials aren't valid, you receive an HTTP 401 and error `invalid_client`.
- If the token has been revoked or if the client submitted a token that isn't valid, you receive an HTTP 200 OK.

Amazon Cognito user pools API reference

With Amazon Cognito user pools, you can enable your web and mobile app users to sign up and sign in. You can change passwords for authenticated users and initiate forgotten password flows for unauthenticated users. For more information, see [User pool authentication flow \(p. 364\)](#) and [Using tokens with user pools \(p. 191\)](#).

For a complete user pool API reference see [Amazon Cognito user pools API Reference](#).

Amazon Cognito identity pools (federated identities) API reference

With an Amazon Cognito identity pool, your web and mobile app users can obtain temporary, limited-privilege AWS credentials enabling them to access other AWS services.

For a complete identity pools (federated identities) API reference see [Amazon Cognito API Reference](#).

Amazon Cognito sync API reference

Amazon Cognito Sync is an AWS service and client library that enables cross-device syncing of application-related user data.

For more information on Amazon Cognito Sync API Reference, see [Amazon Cognito Sync API Reference](#).

Document history for Amazon Cognito

The following table describes the documentation for this release of Amazon Cognito.

- **Original API versions:**

Amazon Cognito User Pools: 2021-1-15

Amazon Cognito Federated Identities: 2014-06-30

Amazon Cognito Sync: 2014-06-30

Change	Description	Date
Sign in federated users without interaction with the hosted UI	Added a new page about how to Bookmark Amazon Cognito apps in an enterprise dashboard (p. 78) so that Amazon Cognito silently directs users to federated sign-in.	March 29, 2022
In-Region SMS and email messaging for Amazon Cognito user pools	You can now use Amazon Simple Notification Service for SMS messages, and Amazon Simple Email Service for email messages, in the same AWS Region as your user pool.	March 14, 2022
Updates to quotas page	Added and clarified resource and request-rate quotas to Quotas in Amazon Cognito (p. 406) .	January 10, 2022
New Amazon Cognito user pools console experience	Updated instructions to create and manage user pools in the updated Amazon Cognito console.	November 18, 2021
RevokeToken API and Revocation Endpoint	You can use the RevokeToken operation to revoke a refresh token (p. 197) for a user.	June 10, 2021
Publication of guide markdown to GitHub	As with all of AWS documentation, this guide now has markdown available to review and comment on at https://github.com/awsdocs/amazon-cognito-developer-guide .	March 23, 2021
Multi-tenant best practices	Best practices for multi-tenant applications were added to the documentation.	March 4, 2021

Change	Description	Date
Attributes for access control	Amazon Cognito Identity Pools provide attributes for access control (AFAC) as a way for customers to grant users access to AWS resources. Authorization can be done based on users' attributes from the identity provider which they used to federate with Amazon Cognito.	January 15, 2021
Custom SMS Sender Lambda Trigger and Custom Email Sender Lambda Trigger	The Custom SMS Sender Lambda Trigger and Custom Email Sender Lambda Trigger allow you to enable a third-party provider to send email and SMS notifications to your users from within your Lambda function code.	November 2020
Amazon Cognito token updates	Updated expiration information was added to Access, ID, and Refresh tokens.	October 29, 2020
Amazon Cognito Service Quotas	Service Quotas are available for Amazon Cognito category quotas. You can use the Service Quotas console to view quota usage, request a quota increase, and create CloudWatch alarms to monitor your quota usage. As part of this change the Available CloudWatch Metrics for Amazon Cognito User Pools section was updated to reflect the new information. The new section name is: Tracking quotas and usage in CloudWatch and Service Quotas	October 29, 2020
Amazon Cognito quota categorization	Quota categories are available to help you monitor quota usage and request an increase. The quotas are grouped into categories based on common use cases.	August 17, 2020
Amazon Cognito Pinpoint document updates	New service-linked role was added. Instructions were updated on "Using Amazon Pinpoint Analytics with Amazon Cognito User Pools".	May 13, 2020
Amazon Cognito supported in US AWS GovCloud	Amazon Cognito is now supported in the AWS GovCloud (US) Region.	May 13, 2020

Change	Description	Date
New Amazon Cognito dedicated security chapter	The Security chapter can help your organization get in-depth information about both the built-in and the configurable security of AWS services. Our new chapters provide information about the security of the cloud and in the cloud.	April 30, 2020
Amazon Cognito Identity Pools now supports Sign in with Apple	Sign in with Apple is available in all regions where Amazon Cognito operates, except cn-north-1 region.	April 7, 2020
New Facebook API Versioning	Added version selection to Facebook API.	April 3, 2020
Username case insensitivity update	Added recommendation about enabling username case insensitivity (p. 23) before creating a user pool.	February 11, 2020
New information about AWS Amplify	Added information about integrating Amazon Cognito (p. 17) with your web or mobile app by using AWS Amplify SDKs and libraries. Removed information about using the Amazon Cognito SDKs that preceded AWS Amplify.	November 22, 2019
New attribute for user pool triggers	Amazon Cognito now includes a <code>clientMetadata</code> parameter in the event information that it passes to the AWS Lambda functions for most user pool triggers. You can use this parameter to enhance your custom authentication workflow with additional data.	October 4, 2019
Updated limit	The throttling limit for the <code>ListUsers</code> API action is updated. For more information, see Quotas in Amazon Cognito (p. 406) .	June 25, 2019
New limit	The soft limits for user pools now include a limit for the number of users. For more information, see Quotas in Amazon Cognito (p. 406) .	June 17, 2019

Change	Description	Date
Amazon SES email settings for Amazon Cognito user pools	You can configure a user pool so that Amazon Cognito emails your users by using your Amazon SES configuration. This setting allows Amazon Cognito to send email with a higher delivery volume than is otherwise possible. For more information, see Email settings for Amazon Cognito user pools (p. 181) .	April 8, 2019
Tagging support	Added information about tagging Amazon Cognito resources (p. 401) .	March 26, 2019
Change the certificate for a custom domain	If you use a custom domain to host the Amazon Cognito hosted UI, you can change the SSL certificate for this domain as needed. For more information, see Changing the SSL certificate for your custom domain (p. 52) .	December 19, 2018
New limit	A new limit is added for the maximum number of groups that each user can belong to. For more information, see Quotas in Amazon Cognito (p. 406) .	December 14, 2018
Updated limits	The soft limits for user pools are updated. For more information, see Quotas in Amazon Cognito (p. 406) .	December 11, 2018
Documentation update for verifying email addresses and phone numbers	Added information about configuring your user pool to require email or phone verification when a user signs up in your app. For more information, see Verifying contact information at sign-up (p. 149) .	November 20, 2018
Documentation update for testing emails	Added guidance for initiating emails from Amazon Cognito while you test your app. For more information, see Sending emails while testing your app (p. 156) .	November 13, 2018

Change	Description	Date
Amazon Cognito Advanced Security	Added new security features to enable developers to protect their apps and users from malicious bots, secure user accounts against compromised credentials, and automatically adjust the challenges required to sign in based on the calculated risk of the sign in attempt.	June 14, 2018
Custom Domains for Amazon Cognito Hosted UI	Allow developers to use their own fully custom domain for the hosted UI in Amazon Cognito User Pools.	June 4, 2018
Amazon Cognito User Pools OIDC Identity Provider	Added user pool sign-in through an OpenID Connect (OIDC) identity provider such as Salesforce or Ping Identity.	May 17, 2018
Amazon Cognito Developer Guide Update	Added top level "What is Amazon Cognito" and "Getting Started with Amazon Cognito". Also added common scenarios and reorganized the user pools TOC. Added a new "Getting Started with Amazon Cognito user pools" section.	April 6, 2018
Amazon Cognito Lambda Migration Trigger	Added pages covering the Lambda Migration Trigger feature	February 8, 2018
Amazon Cognito Advanced Security Beta	Added new security features to enable developers to protect their apps and users from malicious bots, secure user accounts against credentials in the wild that have been compromised elsewhere on the internet, and automatically adjust the challenges required to sign in based on the calculated risk of the sign in attempt.	November 28, 2017
Amazon Pinpoint integration	Added the ability to use Amazon Pinpoint to provide analytics for your Amazon Cognito User Pools apps and to enrich the user data for Amazon Pinpoint campaigns. For more information, see Using Amazon Pinpoint analytics with Amazon Cognito user pools (p. 145) .	September 26, 2017

Change	Description	Date
Federation and built-in app UI features of Amazon Cognito User Pools	Added the ability to allow your users to sign in to your user pool through Facebook, Google, Login with Amazon, or a SAML identity provider. Added a customizable built-in app UI and OAuth 2.0 support with custom claims.	August 10, 2017
HIPAA and PCI compliance-related feature changes	Added the ability to allow your users to use a phone number or email address as their user name.	July 6, 2017
User groups and role-based access control features	Added administrative capability to create and manage user groups. Administrators can assign IAM roles to users based on group membership and administrator-created rules. For more information, see Adding groups to a user pool (p. 162) and Role-based access control (p. 264) .	December 15, 2016
Documentation update	Updated iOS code examples in Developer authenticated identities (identity pools) (p. 298) .	November 18, 2016
Documentation update	Added information about confirmation flow for user accounts. For more information, see Signing up and confirming user accounts (p. 148) .	November 9, 2016
Create user accounts feature	Added administrative capability to create user accounts through the Amazon Cognito console and the API. For more information, see Creating user accounts as administrator (p. 157) .	October 6, 2016
Documentation update	Updated examples that show how to use AWS Lambda triggers with user pools. For more information, see Customizing user pool workflows with Lambda triggers (p. 92) .	September 27, 2016

Change	Description	Date
User import feature	Added bulk import capability for Cognito User Pools. Use this feature to migrate users from your existing identity provider to an Amazon Cognito user pool. For more information, see Importing users into user pools from a CSV file (p. 171) .	September 1, 2016
General availability of Cognito User Pools	Added the Cognito User Pools feature. Use this feature to create and maintain a user directory and add sign-up and sign-in to your mobile app or web application using user pools. For more information, see Amazon Cognito user pools (p. 22) .	July 28, 2016
SAML support	Added support for authentication with identity providers through Security Assertion Markup Language 2.0 (SAML 2.0). For more information, see SAML identity providers (identity pools) (p. 297) .	June 23, 2016
CloudTrail integration	Added integration with AWS CloudTrail. For more information, see Logging Amazon Cognito API calls with AWS CloudTrail (p. 376) .	February 18, 2016
Integration of events with Lambda	Enables you to execute an AWS Lambda function in response to important events in Amazon Cognito. For more information, see Amazon Cognito Events (p. 340) .	April 9, 2015
Data stream to Amazon Kinesis	Provides control and insight into your data streams. For more information, see Amazon Cognito Streams (p. 338) .	March 4, 2015
Push synchronization	Enables support for silent push synchronization. For more information, see Amazon Cognito Sync (p. 312) .	November 6, 2014
OpenID Connect support	Enables support for OpenID Connect providers. For more information, see Identity pools (federated identities) external identity providers (p. 276) .	October 23, 2014

Change	Description	Date
Developer-authenticated identities support added	Enables developers who own their own authentication and identity management systems to be treated as an identity provider in Amazon Cognito. For more information, see Developer authenticated identities (identity pools) (p. 298) .	September 29, 2014
Amazon Cognito general availability		July 10, 2014