
AWS Single Sign-On

User Guide



AWS Single Sign-On: User 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 AWS Single Sign-On? | 1 |
| AWS SSO features | 1 |
| Getting started | 3 |
| AWS SSO prerequisites | 3 |
| Step 1: Enable AWS SSO | 4 |
| Step 2: Choose your identity source (optional) | 4 |
| Step 3: Set up SSO to your AWS accounts (optional) | 5 |
| Step 4: Set up SSO to your applications (optional) | 5 |
| Key AWS SSO concepts | 6 |
| Users, groups, and provisioning | 6 |
| User name and email address uniqueness | 6 |
| Groups | 6 |
| User and group provisioning | 6 |
| AWS SSO-integrated application enablement | 7 |
| Considerations for sharing identity information in AWS accounts | 7 |
| Enable AWS SSO-integrated applications in AWS accounts | 7 |
| SAML federation | 8 |
| User authentications | 8 |
| Authentication sessions | 8 |
| Permission sets | 9 |
| Delegating permission set administration | 9 |
| Use cases | 11 |
| Enable SSO access to your AWS applications | 11 |
| Configure my AWS application in a standalone AWS account | 11 |
| AWS SSO is currently not configured in my organization | 11 |
| AWS SSO is currently configured in my organization | 12 |
| Manage your identity source | 13 |
| Considerations for changing your identity source | 13 |
| Changing between AWS SSO and Active Directory | 13 |
| Changing between AWS SSO and an external identity provider (IdP) | 13 |
| Changing from one external IdP to another external IdP | 14 |
| Changing between Microsoft AD and an external IdP | 14 |
| Change your identity source | 14 |
| Manage identities in AWS SSO | 15 |
| Provisioning when users are in AWS SSO | 15 |
| Add users | 15 |
| Add groups | 16 |
| Add users to groups | 16 |
| Edit user properties | 17 |
| Disable a user | 17 |
| Reset a user password | 17 |
| Password requirements | 18 |
| Supported user and group attributes | 18 |
| Using predefined attributes for access control | 19 |
| Connect to your Microsoft AD directory | 20 |
| Provisioning when users come from Active Directory | 20 |
| Connect AWS SSO to an AWS Managed Microsoft AD directory | 21 |
| Connect AWS SSO to a self-managed Active Directory | 21 |
| Attribute mappings | 22 |
| Connect to your external identity provider | 25 |
| Provisioning when users come from an external identity provider | 26 |
| How to connect to an external identity provider | 26 |
| SCIM profile and SAML 2.0 implementation | 27 |
| Supported identity providers | 32 |

| | |
|---|----|
| Other identity providers | 49 |
| Manage SSO to your AWS accounts | 51 |
| Single sign-on access | 51 |
| Assign user access | 52 |
| Remove user access | 53 |
| Delegate who can assign SSO access to users in the management account | 53 |
| Permission sets | 53 |
| Create a permission set | 53 |
| Configure permission set properties | 54 |
| Delete permission sets | 56 |
| Attribute-based access control | 57 |
| Benefits | 57 |
| Checklist: Configuring ABAC in AWS using AWS SSO | 57 |
| Attributes for access control | 59 |
| IAM identity provider | 64 |
| Repair the IAM identity provider | 64 |
| Remove the IAM identity provider | 64 |
| Service-linked roles | 64 |
| Manage SSO to your applications | 66 |
| AWS SSO-integrated applications | 66 |
| Constraining AWS SSO-integrated application use in AWS accounts | 66 |
| Add and configure an AWS SSO-integrated application | 67 |
| Disable or enable an AWS SSO-integrated application | 67 |
| Remove an AWS SSO-integrated application | 67 |
| Cloud applications | 68 |
| Supported applications | 68 |
| Add and configure a cloud application | 70 |
| Custom SAML 2.0 applications | 71 |
| Add and configure a custom SAML 2.0 application | 71 |
| Manage certificates | 72 |
| Considerations before rotating a certificate | 72 |
| Rotate an AWS SSO certificate | 72 |
| Certificate expiration status indicators | 74 |
| Application properties | 74 |
| Application start URL | 74 |
| Relay state | 75 |
| Session duration | 75 |
| Assign user access | 76 |
| Remove user access | 76 |
| Map attributes in your application to AWS SSO attributes | 77 |
| Using the user portal | 78 |
| Tips for using the portal | 78 |
| How to accept the invitation to join AWS SSO | 78 |
| How to sign in to the user portal | 79 |
| Trusted devices | 79 |
| How to sign out of the user portal | 79 |
| How to search for an AWS account or application | 80 |
| How to reset your password | 80 |
| How to get credentials of an IAM role for use with CLI access to an AWS account | 80 |
| How to bookmark an IAM role for quick access to your management console | 81 |
| How to register a device for use with multi-factor authentication | 81 |
| Multi-factor authentication | 83 |
| Getting started | 83 |
| Considerations before using MFA in AWS SSO | 83 |
| Enable MFA | 84 |
| MFA types | 84 |
| Authenticator apps | 85 |

| | |
|---|-----|
| Security keys and built-in authenticators | 85 |
| RADIUS MFA | 86 |
| How to manage MFA | 86 |
| Configure MFA types | 86 |
| Configure MFA device enforcement | 87 |
| Register an MFA device | 88 |
| Manage a user's MFA device | 89 |
| Allow users to register their own MFA devices | 89 |
| Disable MFA | 89 |
| Security | 91 |
| Identity and access management for AWS SSO | 91 |
| Authentication | 92 |
| Access control | 93 |
| Overview of managing access | 93 |
| Using identity-based policies (IAM policies) | 96 |
| Using service-linked roles | 103 |
| AWS SSO console and API authorization | 106 |
| Logging and monitoring | 107 |
| Logging AWS SSO API calls with AWS CloudTrail | 107 |
| Amazon CloudWatch Events | 123 |
| Compliance validation | 123 |
| Resilience | 124 |
| Infrastructure security | 124 |
| Tagging resources | 125 |
| Tag restrictions | 125 |
| Managing tags with the console | 125 |
| AWS CLI examples | 126 |
| Assigning tags | 126 |
| Viewing tags | 126 |
| Removing tags | 127 |
| Applying tags when you create a permission set | 127 |
| API actions | 127 |
| API actions for AWS SSO instance tags | 127 |
| Integrating AWS CLI with AWS SSO | 128 |
| How to integrate AWS CLI with AWS SSO | 128 |
| Region availability | 129 |
| AWS SSO Region data | 129 |
| Delete your AWS SSO configuration | 129 |
| Quotas | 131 |
| Application quotas | 131 |
| AWS account quotas | 131 |
| Active Directory quotas | 131 |
| AWS SSO identity store quotas | 132 |
| AWS SSO throttle limits | 132 |
| Additional quotas | 132 |
| Troubleshooting | 133 |
| Issues regarding contents of SAML assertions created by AWS SSO | 133 |
| Specific users fail to synchronize into AWS SSO from an external SCIM provider | 133 |
| Users can't sign in when their user name is in UPN format | 134 |
| I get a 'Cannot perform the operation on the protected role' error when modifying an IAM role | 134 |
| Directory users cannot reset their password | 135 |
| My user is referenced in a permission set but can't access the assigned accounts or applications | 135 |
| I cannot get my cloud application configured correctly | 135 |
| Error 'An unexpected error has occurred' when a user tries to sign in using an external identity provider | 136 |
| Error 'Attributes for access control failed to enable' | 136 |
| I get a 'Browser not supported' message when I attempt to register a device for MFA | 136 |

| | |
|---|-----|
| Active Directory “Domain Users” group does not properly sync into AWS SSO | 137 |
| Invalid MFA credentials error | 137 |
| I get a 'An unexpected error has occurred' message when I attempt to register or sign in using an authenticator app | 137 |
| Document history | 138 |
| AWS glossary | 140 |

What is AWS Single Sign-On?

AWS Single Sign-On is a cloud-based single sign-on (SSO) service that makes it easy to centrally manage SSO access to all of your AWS accounts and cloud applications. Specifically, it helps you manage SSO access and user permissions across all your AWS accounts in AWS Organizations. AWS SSO also helps you manage access and permissions to commonly used third-party software as a service (SaaS) applications, AWS SSO-integrated applications as well as custom applications that support Security Assertion Markup Language (SAML) 2.0. AWS SSO includes a user portal where your end-users can find and access all their assigned AWS accounts, cloud applications, and custom applications in one place.

AWS SSO features

AWS SSO provides the following features:

Integration with AWS Organizations

AWS SSO is integrated deeply with AWS Organizations and AWS API operations, unlike other cloud native SSO solutions. AWS SSO natively integrates with AWS Organizations and enumerates all your AWS accounts. If you have organized your accounts under organizational units (OUs) you will see them displayed that way within the AWS SSO console. That way you can quickly discover your AWS accounts, deploy common sets of permissions, and manage access from a central location.

SSO access to your AWS accounts and cloud applications

AWS SSO makes it simple for you to manage SSO across all your AWS accounts, cloud applications, AWS SSO-integrated applications, and custom SAML 2.0–based applications, without custom scripts or third-party SSO solutions. Use the AWS SSO console to quickly assign which users should have one-click access to only the applications that you've authorized for their personalized end-user portal.

Create and manage users and groups in AWS SSO

When you enable the service for the first time, we create a default store for you in AWS SSO. You can use this store to manage your users and groups directly in the console. Or, if you prefer, you can connect to an existing AWS Managed Microsoft AD directory and manage your users with standard Active Directory management tools provided in Windows Server. You can also provision users and groups from an external identity provider into AWS SSO and manage access permissions in the AWS SSO console. If you choose to manage your users in AWS SSO, you can quickly create users and then easily organize them into groups, all within the console.

Leverage your existing corporate identities

AWS SSO is integrated with Microsoft AD through the AWS Directory Service. That means your employees can sign in to your AWS SSO user portal using their corporate Active Directory credentials. To grant Active Directory users access to accounts and applications, you simply add them to the appropriate Active Directory groups. For example, you can grant the DevOps group SSO access to your production AWS accounts. Users added to the DevOps group are then granted SSO access to these AWS accounts automatically. This automation makes it easy to onboard new users and give existing users access to new accounts and applications quickly.

Compatible with commonly used cloud applications

AWS SSO supports commonly used cloud applications such as Salesforce, Box, and Office 365. This cuts the time needed to set up these applications for SSO by providing application integration

instructions. These instructions act as guard rails to help administrators set up and troubleshoot these SSO configurations. This eliminates the need for administrators to learn the configuration nuances of each cloud application.

Easy to set up and monitor usage

With AWS SSO, you can enable a highly available SSO service with just a few clicks. There is no additional infrastructure to deploy or AWS account to set up. AWS SSO is a highly available and a completely secure infrastructure that scales to your needs and does not require software or hardware to manage. AWS SSO records all sign-in activity in AWS CloudTrail, giving you the visibility to monitor and audit SSO activity in one place.

Co-exists with existing IAM users, roles, and policies

Enabling AWS SSO, including enabling AWS Organizations, has no impact on the users, roles, or policies that you're already managing in IAM. You can continue to use your existing access management processes and tools as your organization adopts AWS SSO.

No-cost identity management

You can add any AWS account managed using AWS Organizations to AWS SSO. Both AWS SSO and AWS Organizations are available at no additional cost.

Getting started

In this getting started exercise, you enable AWS Single Sign-On, and then can optionally connect your directory, set up SSO to your AWS accounts, and/or set up SSO to your cloud applications. Although not required, we recommend that you review [Understanding key AWS Single Sign-On concepts \(p. 6\)](#) before you begin using the console so that you are familiar with the core features and terminology.

Topics

- [AWS SSO prerequisites \(p. 3\)](#)
- [Enable AWS SSO \(p. 4\)](#)
- [Choose your identity source \(p. 4\)](#)
- [Set up SSO to your AWS accounts \(p. 5\)](#)
- [Set up SSO to your applications \(p. 5\)](#)

AWS SSO prerequisites

Before you can set up AWS SSO, you must:

- Have first set up the AWS Organizations service and have **All features** set to enabled. For more information about this setting, see [Enabling All Features in Your Organization](#) in the *AWS Organizations User Guide*.
- Sign in with the AWS Organizations management account credentials before you begin setting up AWS SSO. These credentials are required to enable AWS SSO. For more information, see [Creating and Managing an AWS Organization](#) in the *AWS Organizations User Guide*. You cannot set up AWS SSO while signed in with credentials from an Organization's member account.
- Have chosen an identity source to determine which pool of users has SSO access to the user portal. If you choose to use the default AWS SSO identity source for your user store, no prerequisite tasks are required. The AWS SSO store is created by default once you enable AWS SSO and is immediately ready for use. There is no cost for using this store. Alternatively, you can choose to [Connect to your external identity provider \(p. 25\)](#) using Azure Active Directory. If you choose to connect to an existing Active Directory for your user store, you must have the following:
 - An existing AD Connector or AWS Managed Microsoft AD directory set up in AWS Directory Service, and it must reside within your organization's management account. You can connect only one AWS Managed Microsoft AD directory at a time. However, you can change it to a different AWS Managed Microsoft AD directory or change it back to an AWS SSO store at any time. For more information, see [Create a AWS Managed Microsoft AD Directory](#) in the *AWS Directory Service Administration Guide*.
 - You must set up AWS SSO in the Region where your AWS Managed Microsoft AD directory is set up. AWS SSO stores the assignment data in the same Region as the directory. To administer AWS SSO, you should switch to the Region where you have setup AWS SSO. Also, note that AWS SSO's user portal uses the same [access URL](#) as your connected directory.
- If you currently filter access to specific Amazon Web Service (AWS) domains or URL endpoints using a web content filtering solution such as next-generation firewalls (NGFW) or secure web gateways (SWG), you must add the following domains and/or URL endpoints to your web-content filtering solution allow-lists in order for AWS SSO to work properly:

Specific DNS domains

- *.awsapps.com (<http://awsapps.com/>)
- *.signin.aws

Specific URL End-points

- [https://\[yourdirectory\].awsapps.com/start](https://[yourdirectory].awsapps.com/start)
- [https://\[yourdirectory\].awsapps.com/login](https://[yourdirectory].awsapps.com/login)
- [https://\[yourregion\].signin.aws/platform/login](https://[yourregion].signin.aws/platform/login)

We highly recommend that before you enable AWS SSO that you first check to see if your AWS account is approaching the quota limit for IAM roles. For more information, see [IAM object quotas](#). If you are nearing the quota limit, you should consider increasing the quota, otherwise you may have issues with AWS SSO as you provision permission sets to accounts that have exceeded the IAM role limit.

Enable AWS SSO

When you open the AWS SSO console for the first time, you are prompted to enable AWS SSO before you can start managing it. If you have already chosen this option, you can skip this step. If not, use the procedure below to enable it now. Once enabled, AWS SSO creates a [service-linked role \(p. 103\)](#) in all accounts within the organization in AWS Organizations. AWS SSO also creates the same service-linked role in every account that is subsequently added to your organization. This role allows AWS SSO to access each account's resources on your behalf.

Note

You might need to grant users or groups permissions to operate in the AWS Organizations management account. Because it is a highly privileged account, additional security restrictions require you to have the [IAMFullAccess](#) policy or equivalent permissions before you can set this up. These additional security restrictions are not required for any of the member accounts in your AWS organization.

To enable AWS SSO

1. Sign in to the AWS Management Console with your AWS Organizations management account credentials.
2. Open the [AWS SSO console](#).
3. Choose **Enable AWS SSO**.
4. If you have not yet set up AWS Organizations, you will be prompted to create an organization. Choose **Create AWS organization** to complete this process.

Choose your identity source

Choosing an identity source determines where AWS SSO looks for users and groups that need SSO access. By default, you get an AWS SSO store for quick and easy user management. Optionally, you can also connect an external identity provider or connect an AWS Managed Microsoft AD directory with your self-managed Active Directory.

AWS SSO provides users in this identity source with a personalized user portal from which they can easily launch multiple AWS accounts or cloud applications. Users sign in to the portal using their corporate credentials or with credentials they set up in AWS SSO. Once they sign in, they have one-click access to all applications and AWS accounts that you have previously authorized.

Depending on which identity source type you are trying to set up, review the topics below for guidance:

- [Manage identities in AWS SSO \(p. 15\)](#)
- [Connect to your Microsoft AD directory \(p. 20\)](#)

- [Connect to your external identity provider \(p. 25\)](#)

For more information about supported identity source types, see [Manage your identity source \(p. 13\)](#).

Set up SSO to your AWS accounts

In this step, you can grant users in your directory with SSO access to one or more AWS consoles for specific AWS accounts in your organization in AWS Organizations. When you do, AWS SSO uses the [service-linked role \(p. 103\)](#) that was created during enablement to create IAM roles. Your end users can access their AWS accounts using these new roles.

Users within these accounts see only the AWS account icon (for example, Development) that they've been assigned from within their user portal. When they choose the icon, they can then choose which IAM role they want to use when signing in to the AWS Management Console for that AWS account.

To get started assigning SSO access to your AWS accounts, see [Assign user access \(p. 52\)](#).

Set up SSO to your applications

With AWS SSO, you can use AWS applications that are integrated with AWS SSO, cloud-applications for which AWS provides preintegration, and custom SAML 2.0 applications. Depending on which application type you are trying to set up, review the topics below:

- [Add and configure an AWS SSO-integrated application \(p. 67\)](#)
- [Add and configure a cloud application \(p. 70\)](#)
- [Add and configure a custom SAML 2.0 application \(p. 71\)](#)

For more information about supported application types, see [Manage SSO to your applications \(p. 66\)](#).

After you follow the guidance in the topic, you will have successfully configured AWS SSO and set up a trust with your service provider. Your users can now access these applications from within their user portal based on the permissions you assigned.

Understanding key AWS Single Sign-On concepts

You'll get more out of AWS Single Sign-On if you become familiar with key concepts relating to SAML federation, user authentication, and IAM permissions.

Topics

- [Users, groups, and provisioning \(p. 6\)](#)
- [AWS SSO-integrated application enablement \(p. 7\)](#)
- [SAML federation \(p. 8\)](#)
- [User authentications \(p. 8\)](#)
- [Permission sets \(p. 9\)](#)

Users, groups, and provisioning

AWS SSO manages access to all your AWS Organizations accounts, AWS SSO-integrated applications, and other business applications that support the Security Assertion Markup Language (SAML) 2.0 standard.

User name and email address uniqueness

When working in AWS SSO, users must be uniquely identifiable. AWS SSO implements a user name that is the primary identifier for your users. Although most people set the user name equal to a user's email address, AWS SSO and the SAML standard do not require this. However, a large percentage of SAML-based applications use an email address as the unique identifier for users. They obtain this from assertions that a SAML identity provider sends during authentication. Such applications depend upon the uniqueness of email addresses for each user. As such, AWS SSO allows you to specify something other than an email address for user sign-in. AWS SSO requires that all user names and email addresses for your users are non-NULL and unique.

Groups

Groups are a logical combination of users that you define. You can create groups and add users to the groups. AWS SSO does not support adding a group to a group (nested groups). Groups are useful when assigning access to AWS accounts and applications. Rather than assign each user individually, you give permissions to a group. Later, as you add or remove users from a group, the user dynamically gets or loses access to accounts and applications that you assigned to the group.

User and group provisioning

You can create users and groups directly in AWS SSO, or work with users and groups you have in Active Directory or an external identity provider. In order for AWS SSO to assign users and groups for permissions in an AWS SSO account, AWS SSO must first be aware of the users and groups. Similarly, AWS SSO-integrated applications can work with users and groups for which AWS SSO is aware. Provisioning is the process of making user and group information available for use by AWS SSO and AWS SSO-integrated applications.

Provisioning in AWS SSO varies based on the identity source you use. For more information, see [Manage your identity source \(p. 13\)](#).

AWS SSO-integrated application enablement

AWS SSO provides support for integration by other AWS applications and services. These applications can use AWS SSO to perform authentication and can access information about users and groups. For example, a user might sign into an application that generates performance dashboards for resources that the user controls. The user might then share the dashboard by looking up a group in AWS SSO.

To enable this capability, AWS SSO provides an identity store which contains user and group attributes, excluding sign-in credentials.

You can use either of the following methods to keep the users and groups in your AWS SSO identity store updated:

- Use the AWS SSO identity store as your main identity source. If you choose this method, you manage your users and groups from within the AWS SSO console or AWS CLI.
- Set up provisioning (synchronization) of users and groups coming from either of the following identity sources to your AWS SSO identity store:
 - **Active Directory** - For more information, see [Connect to your Microsoft AD directory \(p. 20\)](#).
 - **External identity provider** - For more information, see [Connect to your external identity provider \(p. 25\)](#).

If you choose the provisioning method, you continue managing your users and groups from within your identity source and those changes would get synced to the AWS SSO identity store.

Regardless of which identity source you choose, AWS SSO has the ability to share the user and group information with AWS SSO integrated applications. This capability makes it possible to connect an identity source to AWS SSO once and then share identity information with multiple applications in the AWS Cloud. This eliminates the need to set up federation and identity provisioning with each application independently. This sharing feature also makes it easy to give your workforce (employees) access to many applications in different AWS accounts.

Considerations for sharing identity information in AWS accounts

The attributes contained in AWS SSO are the basic attributes commonly used across applications. These attributes include information such as first and last name, phone number, email address, address, and preferred language. You might want to consider which applications and which accounts can use this personally identifiable information.

To control access to this information, you have two options. First, you can choose to enable access in only the AWS Organizations management account or in all AWS Organizations accounts. Second, you can use service control policies (SCPs) to control which applications can access the information in which AWS Organizations accounts. For example, if you enable access in the AWS Organizations management account only, then applications in member accounts have no access to the information. However, if you enable access in all accounts, you can use SCPs to disallow access by all applications except those you want to permit.

Enable AWS SSO-integrated applications in AWS accounts

When you enable AWS SSO for the first time, AWS enables use of integrated applications automatically in all AWS Organizations accounts. To constrain applications, you must implement SCPs.

If you enabled AWS SSO prior to November 25, 2019, AWS SSO disables the use of integrated applications in all AWS Organizations accounts. To use AWS SSO-integrated applications, you must enable them in the management account and optionally enable them in member accounts. If you enable them in the management account only, you can enable them in member accounts in the future. To enable these applications, use the **Enable access** option in the AWS SSO **Settings** page in the AWS SSO-integrated applications section.

SAML federation

AWS SSO supports identity federation with [SAML \(Security Assertion Markup Language\) 2.0](#). SAML 2.0 is an industry standard used for securely exchanging SAML assertions that pass information about a user between a SAML authority (called an identity provider or IdP), and a SAML consumer (called a service provider or SP). AWS SSO service uses this information to provide federated single sign-on (SSO) for those users who are authorized to use applications within the AWS SSO user portal.

AWS SSO adds SAML IdP capabilities to either your AWS Managed Microsoft AD or your AWS SSO store. Users can then SSO into services that support SAML, including the AWS Management Console and third-party applications such as Office 365, SAP Concur, and Salesforce. At this time, AWS SSO does not support other directory types or IdPs.

User authentications

A user signs in to the user portal using their user name. When they do, AWS SSO redirects the request to the AWS SSO authentication service based on the directory associated with the user email address. Once authenticated, users have SSO access to any of the AWS accounts and third-party software-as-a-service (SaaS) applications that show up in the portal without additional sign-in prompts. This means that users no longer need to keep track of multiple account credentials for the various assigned AWS applications that they use on a daily basis.

Authentication sessions

There are two types of authentication sessions maintained by AWS SSO: one to represent the users' sign in to AWS SSO, and another to represent the users' access to AWS SSO-integrated applications, such as Amazon SageMaker Studio or Amazon Managed Service for Grafana. Each time a user signs in to AWS SSO, a sign in session is created with an 8-hour lifetime. Each time the user accesses an AWS SSO-enabled application, the AWS SSO sign in session is used to obtain an AWS SSO application session for that application. AWS SSO application sessions have a refreshable 1-hour lifetime – that is, AWS SSO application sessions are automatically refreshed every hour as long as the AWS SSO sign in session from which they were obtained is still valid. When the user uses AWS SSO to access the AWS Management Console or CLI, the AWS SSO sign in session is used to obtain an IAM session, as specified in the corresponding AWS SSO permission set (more specifically, AWS SSO assumes an IAM role, which AWS SSO manages, in the target account).

When you disable or delete a user in AWS SSO, that user will immediately be prevented from signing in to create new AWS SSO sign in sessions. AWS SSO sign in sessions are cached for one hour, which means that when you disable or delete a user while they have an active AWS SSO sign in session, their existing SSO sign in session will continue for up to an hour, depending on when the sign in session was last refreshed. During this time, the user can initiate new AWS SSO application and IAM role sessions.

After the AWS SSO sign in session expires, the user can no longer initiate new AWS SSO application or IAM role sessions. However, AWS SSO application sessions can also be cached for up to an hour, such that the user may retain access to an AWS SSO-integrated application for up to an hour after the AWS SSO sign in session has expired. Any existing IAM role sessions will continue based on the duration configured in the AWS SSO permission set (admin-configurable, up to 12 hours).

The table below summarizes these behaviors:

| User experience / system behavior | Time after SSO user is disabled / deleted |
|---|---|
| User can no longer sign in to AWS SSO; user cannot obtain a new AWS SSO sign in session | None (effective immediately) |
| User can no longer start new application or IAM role sessions via AWS SSO | Up to 1 hour |
| User can no longer access any AWS SSO-integrated applications (all app sessions are terminated) | Up to 2 hours (up to 1 hour for AWS SSO sign in session expiry, plus up to 1 hour for AWS SSO app session expiry) |
| User can no longer access any AWS accounts through AWS SSO | Up to 13 hours (up to 1 hour for AWS SSO sign in session expiry, plus up to 12 hours for administrator-configured IAM role session expiry per the AWS SSO session duration settings for the permission set) |

For more information about sessions, see [Set session duration \(p. 55\)](#).

Permission sets

A permission set is a collection of administrator-defined policies that AWS SSO uses to determine a user's effective permissions to access a given AWS account. Permission sets can contain either [AWS managed policies](#) or custom policies that are stored in AWS SSO. Policies are essentially documents that act as containers for one or more permission statements. These statements represent individual access controls (allow or deny) for various tasks that determine what tasks users can or cannot perform within the AWS account.

Permission sets are stored in AWS SSO and are only used for AWS accounts. They are not used to manage access to cloud applications. Permission sets ultimately get created as [IAM roles](#) in a given AWS account, with trust policies that allow users to assume the role through AWS SSO.

Delegating permission set administration

AWS SSO enables you to delegate management of permission sets and assignments in accounts by creating [IAM policies](#) that reference the [Amazon Resource Names \(ARNs\)](#) of AWS SSO resources. For example, you can create policies that enable different administrators to manage assignments in specified accounts for permission sets with specific tags.

You can use any of the following three methods to create these kinds of policies.

- (Recommended) Create permission sets in AWS SSO, each with a different policy, and assign the permission sets to different users or groups. This enables you to manage administrative permissions for users that sign in using your chosen [AWS SSO identity source](#).
- Create [custom policies](#) in IAM, and then attach them to IAM roles that your administrators assume. This enables [IAM users](#) or [IAM federated users](#) to [assume the role](#) to get their assigned AWS SSO administrative permissions.
- Create [custom policies](#) in IAM, and then attach them to IAM users that you use for AWS SSO administrator purposes. This enables you to give individual IAM users specific AWS SSO administrative permissions.

Important

AWS SSO resource ARNs are case sensitive.

The following shows the proper case for referencing the AWS SSO permission set and account resource types.

| Resource Types | ARN | Context Keys |
|----------------|--|----------------------------|
| PermissionSet | arn: \${Partition}:sso:::permissionSet/ \${InstanceId}/ \${PermissionSetId} | aws:ResourceTag/\${TagKey} |
| Account | arn:\${Partition}:sso:::account/ \${AccountId} | Not Applicable |

Use cases

Review the following use case to determine how best to use AWS SSO for your business needs.

Topics

- [Enable SSO access to your AWS applications \(Application admin role\) \(p. 11\)](#)

Enable SSO access to your AWS applications (Application admin role)

This use case provides guidance for application administrators who manage [AWS SSO-integrated applications \(p. 66\)](#) such as Amazon SageMaker or AWS IoT SiteWise, and need to provide SSO access to their end users.

Before you get started, consider the following questions:

- Should you create a test or production environment in your own separate AWS organization?
- Is AWS SSO already enabled in your AWS organization? Do you have permissions to enable AWS SSO in the management account of AWS Organizations?

Depending on your specific business needs, review the guidance below.

Configure my AWS application in a standalone AWS account

If you need to provide SSO access to an AWS application and know that your IT department does not yet use AWS SSO, you may need to create a standalone AWS account so that you can get started. By default, when you create your own AWS account, you'll have the necessary permissions, such as Root user, to create and manage your own AWS organization. This permission is required to enable AWS SSO. For more information about the permissions necessary to enable AWS SSO, see [Identity and access management for AWS SSO \(p. 91\)](#).

If they are not already enabled, AWS SSO and AWS Organizations can be enabled automatically during the process of setting up some AWS applications (for example Amazon Managed Service for Grafana does this). If your AWS application does not provide the option to enable them, you'll need to first setup AWS Organizations and AWS SSO before you can provide SSO access to your application.

AWS SSO is currently not configured in my organization

In your role as an application admin, you may not be able to enable AWS SSO as it requires specific permissions in the AWS Organizations management account. In this case, you will need to coordinate with your IT or cloud administrators to get AWS SSO enabled in the Organizations management account.

If you do have sufficient permissions to enable AWS SSO within your AWS organization, do so first, then proceed with the application setup. For more information, see [Enable AWS SSO \(p. 4\)](#).

AWS SSO is currently configured in my organization

In this scenario, you can continue to deploy your AWS application without any further steps or requirements.

Note

If your organization enabled AWS SSO in the management account before November 25th, 2019, you must also enable AWS SSO-integrated applications in the management account and optionally in the member accounts. If you enable them in the management account only, you can enable them in member accounts later. To enable these applications, choose **Enable access** in the AWS SSO console's **Settings** page in the AWS SSO-integrated applications section. For more information, see [AWS SSO-integrated application enablement \(p. 7\)](#).

Manage your identity source

Your identity source in AWS SSO defines where your users and groups are managed. Once configured, you can then look up users or groups in your store to grant them single sign-on access to AWS accounts, cloud applications, or both.

You can have only one identity source per AWS organization. You can choose one of the following as your identity source:

- **AWS SSO identity store** – When you enable AWS SSO for the first time, it is automatically configured with an AWS SSO identity store as your default identity source. This is where you create your users and groups, and assign their level of access to your AWS accounts and applications.
- **Active Directory** – Choose this option if you want to continue managing users in either your self-managed Active Directory (AD) or your AWS Managed Microsoft AD directory using AWS Directory Service.
- **External identity provider** – Choose this option if you prefer to manage users in an external identity provider (IdP) such as Okta or Azure Active Directory.

Note

AWS SSO does not support SAMBA4-based Simple AD as an identity source.

Topics

- [Considerations for changing your identity source \(p. 13\)](#)
- [Change your identity source \(p. 14\)](#)
- [Manage identities in AWS SSO \(p. 15\)](#)
- [Connect to your Microsoft AD directory \(p. 20\)](#)
- [Connect to your external identity provider \(p. 25\)](#)

Considerations for changing your identity source

Before you proceed with the steps in [Change your identity source \(p. 14\)](#), we recommend that you first review the following important information. This information can help you understand how the process of switching your identity source might affect your current deployment.

Changing between AWS SSO and Active Directory

Whether you switch from Microsoft AD to AWS SSO or from AWS SSO to Microsoft AD, the conversion deletes all users, groups, and assignments (entitlements). If you switch to AWS SSO, you must create your users, groups, and entitlements. If you switch to Microsoft AD, you must create entitlements with the users and groups that are in your Microsoft AD.

For information about how AWS SSO provisions users and groups, see [Connect to your Microsoft AD directory \(p. 20\)](#).

Changing between AWS SSO and an external identity provider (IdP)

If you switch to an external IdP, AWS SSO preserves all entitlements you had. The user and group entitlements will work if the AWS SSO user names and groups match those that you have in the external

IdP. Any unmatched users and groups are unusable. If you switch from an external IdP to AWS SSO, AWS SSO preserves all users, groups, and entitlements. If any of the users previously had passwords in AWS SSO, then those users can continue signing in with their old passwords. The administrator must force a password reset for users that came from the external IdP and that did not previously exist in AWS SSO.

For information about how AWS SSO provisions users and groups, see [Connect to your external identity provider \(p. 25\)](#).

Changing from one external IdP to another external IdP

If you are already using an external IdP in AWS SSO and switch to a different external IdP, AWS SSO preserves all entitlements you had. The user entitlements, group entitlements, and group memberships will continue to work as long as the new IdP sends the correct assertions (e.g., SAML nameIDs). These assertions must match the usernames in AWS SSO when your users authenticate through the new external IdP. Note that if you are using SCIM for provisioning into AWS SSO, you will want to review the IdP-specific information in this guide and/or the IdP's documentation to ensure that the new provider will match users and groups correctly when SCIM is enabled.

For information about how AWS SSO provisions users and groups, see [Connect to your external identity provider \(p. 25\)](#).

Changing between Microsoft AD and an external IdP

Whether you switch from an external IdP to Microsoft AD, or from Microsoft AD to an external IdP, the conversion deletes all users, groups, and entitlements in AWS SSO. No user or group information is affected in either the external IdP or Microsoft AD. If you switch to an external IdP, you must configure AWS SSO to provision your users. Or you must manually provision your external IdP users and groups before you can configure entitlements. If you switch to Microsoft AD, you must create entitlements with the users and groups that are in your Microsoft AD.

For information about how AWS SSO provisions users and groups, see [Connect to your Microsoft AD directory \(p. 20\)](#).

Change your identity source

You can change where you store users at any time. Use the following procedure to switch from a directory that AWS SSO provides (the default) to an external identity provider, AWS Managed Microsoft AD directory or vice versa. Make sure to review identity source considerations before proceeding. For more information, see [Considerations for changing your identity source \(p. 13\)](#).

To change your identity source

1. Open the [AWS SSO console](#).
2. Choose **Settings**.
3. On the **Settings** page, under **Identity source**, choose **Change**.
4. On the **Change identity source** page, select the source you want to switch to, and then choose **Next**. If you are switching to a Microsoft AD directory, you must choose the available directory from the provided menu.

Important

Changing your source to or from Active Directory removes all existing user and group assignments. You must manually reapply assignments after you have successfully changed your source.

5. Choose **Next: Review**.

6. Once you have read the disclaimer and are ready to proceed, type **CONFIRM**.
7. Choose **Finish**.

Manage identities in AWS SSO

AWS Single Sign-On provides you with a default store where you can store your users and groups. If you choose to store them in AWS SSO, all you need to do is the following:

- Create your users and groups.
- Add your users as members to the groups.
- Assign the groups with the desired level of access to your AWS accounts and applications.

If you prefer to manage users in AWS Managed Microsoft AD, you can discontinue use of your AWS SSO store at any time and instead connect AWS SSO to your Microsoft AD using AWS Directory Service. For more information, see [Connect to your Microsoft AD directory \(p. 20\)](#).

If you prefer to manage users in an external identity provider (IdP), you can connect AWS SSO to your IdP and enable automatic provisioning. For more information, see [Connect to your external identity provider \(p. 25\)](#).

Note

When identities are deleted in the AWS SSO store, corresponding assignments also get deleted in AWS SSO. However in Microsoft AD, when identities are deleted (either in AD or the synced in identities), corresponding assignments are not deleted.

Provisioning when users are in AWS SSO

When you create users and groups directly in AWS SSO, provisioning is automatic. These identities are immediately available for use in making assignments and for use by AWS SSO-integrated applications. For more information, see [User and group provisioning \(p. 6\)](#).

Topics

- [Add users \(p. 15\)](#)
- [Add groups \(p. 16\)](#)
- [Add users to groups \(p. 16\)](#)
- [Edit user properties \(p. 17\)](#)
- [Disable a user \(p. 17\)](#)
- [Reset a user password \(p. 17\)](#)
- [Password requirements for the AWS SSO identity store \(p. 18\)](#)
- [Supported user and group attributes \(p. 18\)](#)
- [Using predefined attributes from the AWS SSO identity store for access control in AWS \(p. 19\)](#)

Add users

Users and groups that you create in your AWS SSO store are available in AWS SSO only. Use the following procedure to add users to your AWS SSO store.

To add a user

1. Open the [AWS SSO console](#).
2. Choose **Users**.

3. Choose **Add user** and provide the following required information:
 - a. **Username** – This user name will be required to sign in to the user portal and cannot be changed later.
 - b. **Password** – Choose from one of the following choices to send the user's password.
 - i. **Send an email to the user with password setup instructions** – This option automatically sends the user an email addressed from Amazon Web Services. The email invites the user on behalf of your company to access the AWS SSO user portal.
 - ii. **Generate a one-time password that you can share with the user** – This option provides you with the user portal URL and password details that you can manually send to the user from your email address.
 - c. **Email address** – The value you provide here must be unique.
 - d. **Confirm email address**
 - e. **First name** – You must enter a name here for automatic provisioning to work. For more information, see [Automatic provisioning \(p. 27\)](#).
 - f. **Last name** – You must enter a name here for automatic provisioning to work.
 - g. **Display name**

Note
(Optional) You can provide additional attributes such as **Employee ID** and **Office 365 Immutable ID** to help map the user's identity in AWS SSO with certain business applications that the user needs to use.
4. Choose **Next: Groups**.
5. Select one or more groups that you want the user to be a member of. Then choose **Add user**.

Add groups

Use the following procedure to add groups to your AWS SSO store.

To add a group

1. Open the [AWS SSO console](#).
2. Choose **Groups**.
3. Choose **Create group**.
4. In the **Create group** dialog box, enter a **Group name** and **Description**. The description should provide details on what permissions have been or will be assigned to the group.
5. Choose **Create**.

Add users to groups

Use the following procedure to add users as members of a group that you previously created in your AWS SSO store.

To add a user as a member of a group

1. Open the [AWS SSO console](#).
2. Choose **Groups**.
3. Choose the group from the list.
4. On the group **Details** page, under **Group members**, choose **Add users**.
5. On the **Add users to group** page, locate the users you want to add as members. Then select the check box next to each of them.

6. Choose **Add user**.

Edit user properties

Use the following procedure to edit the properties of a user in your AWS SSO store.

To edit user properties

1. Open the [AWS SSO console](#).
2. Choose **Users**.
3. Choose the user that you want to edit.
4. On the user **Details** page, choose **Edit user**.
5. On the **Edit user details** page, make the updates to the properties as needed. Then choose **Save changes**.

Note

(Optional) You can modify additional attributes such as **Employee ID** and **Office 365 Immutable ID** to help map the user's identity in AWS SSO with certain business applications that users need to use.

Note

The **Email address** attribute is an editable field and the value you provide must be unique.

Disable a user

When you disable a user, you cannot edit their user details, reset their password, add the user to a group, or view their group membership. Use the following procedure to disable a user in your AWS SSO store.

Note

When you disable or delete a user in AWS SSO, that user will immediately be prevented from signing in to the user portal and will not be able to create new AWS SSO sign in sessions. For more information, see [Authentication sessions](#) (p. 8).

To disable a user

1. Open the [AWS SSO console](#).
2. Choose **Users**.
3. Choose the user you want to disable.
4. On the user details page, choose **Disable user**.
5. On the **Disable user** dialog, choose **Disable user**.

Reset a user password

Use the following procedure to reset the password for a user in your AWS SSO store.

To reset a user password

1. Open the [AWS SSO console](#).
2. Choose **Users**.
3. Choose the user whose password you want to reset.
4. On the user **Profile** page, choose **Reset password**.
5. In the **Reset password** dialog box, select one of the following choices, and then choose **Reset password**:

- a. **Send an email to the user with instructions to reset the password** – This option automatically sends the user an email addressed from Amazon Web Services that walks them through how to reset their password.

Warning

As a security best practice, verify that the email address for this user is correct prior to selecting this option. If this password reset email were to be sent to an incorrect or misconfigured email address, a malicious recipient could use it to gain unauthorized access to your AWS environment.

- b. **Generate a one-time password and share the password with the user** – This option provides you with the password details that you can manually send to the user from your email address.

Password requirements for the AWS SSO identity store

When you create an AWS SSO identity store, a default password policy is created and applied to the store. Users who sign in to the user portal must adhere to the requirements in this policy when they set or change their password. The default password policy includes the following requirements:

- Passwords are case-sensitive
- Passwords must be between 8 and 64 characters in length
- Passwords must contain at least one character from each of the following four categories:
 - Lowercase letters (a-z)
 - Uppercase letters (A-Z)
 - Numbers (0-9)
 - Non-alphanumeric characters (~!@#\$%^&*_-+=`\'()\{\}[];:'"<>.,?/)
- The last three passwords cannot be reused

Note

These requirements apply only to users created in the AWS SSO identity store. If you have configured an identity source other than AWS SSO for authentication, such as Active Directory or an external identity provider, the password policies for your users are defined and enforced in those systems, not in AWS SSO.

Supported user and group attributes

Attributes are pieces of information that help you define and identify individual user or group objects, such as name, email, or members. AWS SSO supports most commonly used attributes regardless if they are entered manually during user creation or when automatically provisioned using a synchronization engine such as defined in the System for Cross-Domain Identity Management (SCIM) specification. For more information about this specification, see <https://tools.ietf.org/html/rfc7642>. For more information about manual and automatic provisioning, see [Provisioning when users come from an external identity provider](#) (p. 26).

Because AWS SSO supports SCIM for automatic provisioning use cases, the AWS SSO identity store supports all of the same user and group attributes that are listed in the SCIM specification, with a few exceptions. The following sections describe which attributes AWS SSO does not support.

User objects

All attributes from the SCIM user schema (<https://tools.ietf.org/html/rfc7643#section-8.3>) are supported in the AWS SSO identity store EXCEPT the following:

- password
- ims
- photos
- entitlements
- x509Certificates

All sub-attributes for users are supported EXCEPT the following:

- 'display' sub-attribute of any multi-valued attribute (For example, emails or phoneNumbers)
- 'version' sub-attribute of 'meta' attribute

Group objects

All attributes from the SCIM group schema (<https://tools.ietf.org/html/rfc7643#section-8.4>) are supported.

All sub-attributes for groups are supported EXCEPT the following:

- 'display' sub-attribute of any multi-valued attribute (For example, members).

Using predefined attributes from the AWS SSO identity store for access control in AWS

Each user in the AWS SSO Identity Store is assigned a unique `UserId`. You can view the `UserId` for your users using the AWS SSO console by navigating to each user or calling the [DescribeUser](#) API. AWS SSO allows you to use this `UserId` in permissions sets or resource based policies for making access control decisions in AWS. For example, the bucket policy below allows only the user with `<UserId>` `s3:GetObject` permission to *mybucket*. This permission allows the user to read the object data from *mybucket*.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::mybucket/*"
      ],
      "Condition": {
        "StringEquals": {
          "identitystore:UserId": [
            "<UserId>"
          ]
        },
        "Null": {
          "identitystore:UserId": "false"
        }
      }
    }
  ]
}
```

}

Connect to your Microsoft AD directory

With AWS Single Sign-On, administrators can connect their self-managed Active Directory (AD) or their AWS Managed Microsoft AD directory using AWS Directory Service. This Microsoft AD directory defines the pool of identities that administrators can pull from when using the AWS SSO console to assign single sign-on (SSO) access. After connecting their corporate directory to AWS SSO, administrators can then grant their AD users or groups access to AWS accounts, cloud applications, or both.

AWS Directory Service helps you to set up and run a standalone AWS Managed Microsoft AD directory hosted in the AWS Cloud. You can also use AWS Directory Service to connect your AWS resources with an existing self-managed AD. To configure AWS Directory Service to work with your self-managed AD, you must first set up trust relationships to extend authentication to the cloud.

AWS SSO uses the connection provided by AWS Directory Service to perform pass-through authentication to the source AD instance, leveraging the Kerberos protocol. When you use AWS Managed Microsoft AD as your identity source, AWS SSO can work with users from AWS Managed Microsoft AD or from any domain connected through an AD trust. If you want to locate your users in four or more domains, users must use the `DOMAIN\user` syntax as their user name when performing sign-ins to AWS SSO.

Notes

- As a prerequisite step, make sure your AD Connector or AWS Managed Microsoft AD directory in AWS Directory Service resides within your AWS Organizations management account. For more information, see [AWS SSO prerequisites \(p. 3\)](#).
- AWS SSO does not support SAMBA 4-based Simple AD as a connected directory.

Provisioning when users come from Active Directory

AWS SSO uses the connection provided by the AWS Directory Service to synchronize user, group, and membership information from your source AD to the AWS SSO identity store. No password information is synchronized to AWS SSO, since user authentication takes place directly from the source AD. This identity data is used by AWS SSO integrated applications to facilitate in-app lookup, authorization and collaboration scenarios without passing LDAP activity all the way back to the source AD.

How It Works

AWS SSO refreshes the AD-based identity data in the identity store using the following process.

Creation

When users or groups are assigned to AWS resources or applications using the AWS console or entitlement API calls, the users, groups and membership information are periodically synchronized into the AWS SSO identity store. Users or groups added to AWS SSO assignments will usually appear in the AWS identity store within two hours, but may take longer based on the amount of data being synchronized. Only users and groups that are directly assigned access, or are members of a group that is assigned access, are synchronized.

Groups that are members of assigned groups (called nested groups) are also written to the identity store. Users that are members of nested groups are “flattened,” meaning they are added to the parent group in the AWS SSO identity store. This allows AWS SSO administrators to use only the parent group for authorization.

If a user accesses AWS SSO before their user object has been synchronized for the first time, that user's identity store object is created on demand using just-in-time (JIT) provisioning. Users created by JIT provisioning are not synchronized unless they have directly assigned or group-based AWS SSO entitlements. Group memberships for JIT-provisioned users are unavailable until after synchronization.

Update

The identity data in the AWS SSO identity store stays fresh by periodically reading data from the source AD. Identity data changed in AD will usually appear in the AWS identity store within four hours, but may take longer based on the amount of data being synchronized.

User and group objects and their memberships are created or updated in AWS SSO to map to the corresponding objects in the source AD. For user attributes, only the subset of attributes listed in the Attribute Mapping section of the AWS SSO console are updated in AWS SSO. In addition, user attributes are updated with each user authentication event.

Deletion

Users and groups are deleted from the AWS SSO identity store when the corresponding user or group objects are deleted from the source AD.

For more information about provisioning, see [User and group provisioning \(p. 6\)](#).

Topics

- [Connect AWS SSO to an AWS Managed Microsoft AD directory \(p. 21\)](#)
- [Connect AWS SSO to a self-managed Active Directory \(p. 21\)](#)
- [Attribute mappings \(p. 22\)](#)

Connect AWS SSO to an AWS Managed Microsoft AD directory

Use the following procedure to connect an AWS Managed Microsoft AD directory that is managed by AWS Directory Service to AWS SSO.

To connect AWS SSO to AWS Managed Microsoft AD

1. Open the [AWS SSO console](#).

Note

Make sure that the AWS SSO console is using one of the Regions where your AWS Managed Microsoft AD directory is located before you move to the next step.

2. Choose **Settings**.
3. Under **Identity source**, choose **Change**.
4. On the **Change identity source** page, choose **Active Directory**, choose the AWS Managed Microsoft AD directory from the list, and then choose **Next: Review**.
5. On the **Review and confirm** page, review the information and type **CONFIRM**.
6. Choose **Change identity source**.

Connect AWS SSO to a self-managed Active Directory

Users in your self-managed Active Directory (AD) can also have SSO access to AWS accounts and cloud applications in the AWS SSO user portal. To do that, AWS Directory Service has the following two options available:

- **Create a two-way trust relationship** – When two-way trust relationships are created between AWS Managed Microsoft AD and a self-managed AD, users in your self-managed AD can sign in with their corporate credentials to various AWS services and business applications. One-way trusts do not work with AWS SSO. For more information about setting up a two-way trust, see [When to Create a Trust Relationship](#) in the *AWS Directory Service Administration Guide*.
- **Create an AD Connector** – AD Connector is a directory gateway that can redirect directory requests to your self-managed AD without caching any information in the cloud. For more information, see [Connect to a Directory](#) in the *AWS Directory Service Administration Guide*.

Note

If you are connecting AWS SSO to an AD Connector directory, any future user password resets must be done from within AD. This means that users will not be able to reset their passwords from the user portal.

Note

AWS SSO does not work with SAMBA4-based Simple AD directories.

Attribute mappings

Attribute mappings are used to map attribute types that exist in AWS SSO with like attributes in an AWS Managed Microsoft AD directory. AWS SSO retrieves user attributes from your Microsoft AD directory and maps them to AWS SSO user attributes. These AWS SSO user attribute mappings are also used for generating SAML assertions for your cloud applications. Each cloud application determines the list of SAML attributes it needs for successful single sign-on.

AWS SSO prefills a set of attributes for you under the **Attribute mappings** tab found on your application's configuration page. AWS SSO uses these user attributes to populate SAML assertions (as SAML attributes) that are sent to the cloud application. These user attributes are in turn retrieved from your Microsoft AD directory. For more information, see [Map attributes in your application to AWS SSO attributes](#) (p. 77).

AWS SSO also manages a set of attributes for you under the **Attribute mappings** section of your directory configuration page. For more information, see [Map attributes in AWS SSO to attributes in your AWS Managed Microsoft AD directory](#) (p. 25).

Supported directory attributes

The following table lists all AWS Managed Microsoft AD directory attributes that are supported and that can be mapped to user attributes in AWS SSO.

| Supported attributes in your Microsoft AD directory |
|---|
| <code>\${dir:email}</code> |
| <code>\${dir:displayname}</code> |
| <code>\${dir:distinguishedName}</code> |
| <code>\${dir:firstname}</code> |
| <code>\${dir:guid}</code> |
| <code>\${dir:initials}</code> |
| <code>\${dir:lastname}</code> |
| <code>\${dir:proxyAddresses}</code> |

| Supported attributes in your Microsoft AD directory |
|---|
| <code>\${dir:proxyAddresses:smtp}</code> |
| <code>\${dir:proxyAddresses:SMTP}</code> |
| <code>\${dir:windowsUpn}</code> |

You can specify any combination of supported Microsoft AD directory attributes to map to a single attribute in AWS SSO. For example, you could choose the `preferredUsername` attribute under the **User attribute in AWS SSO** column. Then map it to either `${dir:displayname}` or `${dir:lastname}${dir:firstname}` or any single supported attribute or any arbitrary combination of supported attributes.

Supported AWS SSO attributes

The following table lists all AWS SSO attributes that are supported and that can be mapped to user attributes in your AWS Managed Microsoft AD directory. Later, after you set up your application attribute mappings, you can use these same AWS SSO attributes to map to actual attributes used by that application.

| Supported attributes in AWS SSO |
|---|
| <code>\${user:AD_GUID}</code> |
| <code>\${user:email}</code> |
| <code>\${user:familyName}</code> |
| <code>\${user:givenName}</code> |
| <code>\${user:middleName}</code> |
| <code>\${user:name}</code> |
| <code>\${user:preferredUsername}</code> |
| <code>\${user:subject}</code> |
| <code>\${user:groups}</code> |

Supported external identity provider attributes

The following table lists all external identity provider (IdP) attributes that are supported and that can be mapped to attributes you can use when configuring [Attributes for access control \(p. 59\)](#) in AWS SSO. When using SAML assertions, you can use whichever attributes your IdP supports.

| Supported attributes in your IdP |
|---------------------------------------|
| <code>\${path:userName}</code> |
| <code>\${path:name.familyName}</code> |
| <code>\${path:name.givenName}</code> |
| <code>\${path:displayName}</code> |

| Supported attributes in your IdP |
|---|
| <code>\${path:nickName}</code> |
| <code>\${path:emails[primary eq true].value}</code> |
| <code>\${path:addresses[type eq "work"].streetAddress}</code> |
| <code>\${path:addresses[type eq "work"].locality}</code> |
| <code>\${path:addresses[type eq "work"].region}</code> |
| <code>\${path:addresses[type eq "work"].postalCode}</code> |
| <code>\${path:addresses[type eq "work"].country}</code> |
| <code>\${path:addresses[type eq "work"].formatted}</code> |
| <code>\${path:phoneNumbers[type eq "work"].value}</code> |
| <code>\${path:userType}</code> |
| <code>\${path:title}</code> |
| <code>\${path:locale}</code> |
| <code>\${path:timezone}</code> |
| <code>\${path:enterprise.employeeNumber}</code> |
| <code>\${path:enterprise.costCenter}</code> |
| <code>\${path:enterprise.organization}</code> |
| <code>\${path:enterprise.division}</code> |
| <code>\${path:enterprise.department}</code> |
| <code>\${path:enterprise.manager.value}</code> |

Default mappings

The following table shows the default mappings for user attributes in AWS SSO to the user attributes in your AWS Managed Microsoft AD directory. At this time, AWS SSO only supports the list of attributes shown in the **User attribute in AWS SSO** column.

| User attribute in AWS SSO | Maps to this attribute in your Microsoft AD directory |
|---------------------------|---|
| AD_GUID | <code>\${dir:guid}</code> |
| email * | <code>\${dir:windowsUpn}</code> |
| familyName | <code>\${dir:lastname}</code> |
| givenName | <code>\${dir:firstname}</code> |
| middleName | <code>\${dir:initials}</code> |
| name | <code>\${dir:displayname}</code> |

| User attribute in AWS SSO | Maps to this attribute in your Microsoft AD directory |
|---------------------------|---|
| preferredUsername | \${dir:displayname} |
| subject | \${dir:windowsUpn} |

* The email attribute in AWS SSO must be unique within the directory. Otherwise, the JIT login process could fail.

You can change the default mappings or add more attributes to the SAML assertion based on your requirements. For example, assume that your cloud application requires the users email in the `User.Email` SAML attribute. In addition, assume that email messages are stored in the `windowsUpn` attribute in your Microsoft AD directory. To achieve this mapping, you must make changes in the following two places in the AWS SSO console:

1. On the **Directory** page, under the **Attribute mappings** section, you would need to map the user attribute **email** to the `${dir:windowsUpn}` attribute (in the **Maps to this attribute in your directory** column)
2. On the **Applications** page, choose the application from the table. Choose the **Attribute mappings** tab. Then map the `User.Email` attribute to the `${user:email}` attribute (in the **Maps to this string value or user attribute in AWS SSO** column).

Please note that you must supply each directory attribute in the form `${dir:AttributeName}`. For example, the `firstname` attribute in your Microsoft AD directory becomes `${dir:firstname}`. It is important that every directory attribute have an actual value assigned. Attributes missing a value after `${dir:}` will cause user sign-in issues.

Map attributes in AWS SSO to attributes in your AWS Managed Microsoft AD directory

You can use the following procedure to specify how your user attributes in AWS SSO should map to corresponding attributes in your Microsoft AD directory.

To map attributes in AWS SSO to attributes in your directory

1. Open the [AWS SSO console](#).
2. Choose **Settings**.
3. On the **Settings** page, under **Attribute mappings**, choose **Edit attribute mappings**.
4. On the **Edit attribute mappings** page, find the attribute in AWS SSO that you want to map and then type a value in the text box. For example, you might want to map the AWS SSO user attribute **email** to the Microsoft AD directory attribute `${dir:windowsUpn}`.
5. Choose **Save changes**.

Connect to your external identity provider

You can use AWS Single Sign-On (AWS SSO) to authenticate identities from external identity providers (IdPs) through the Security Assertion Markup Language (SAML) 2.0 standard. This enables your users to sign in to the AWS SSO user portal with their corporate credentials. They can then navigate to their assigned accounts, roles, and applications hosted in external identity providers.

For example, you can connect an external IdP such as Okta or Azure Active Directory (AD), to AWS SSO. Your users can then sign in to the AWS SSO user portal with their existing Okta or Azure credentials. In addition, you can assign access permissions centrally for your users across all the accounts and applications in your AWS organization. In addition, developers can simply sign in to the AWS Command Line Interface (AWS CLI) using their existing credentials, and benefit from automatic short-term credential generation and rotation.

The SAML protocol does not provide a way to query the IdP to learn about users and groups. Therefore, you must make AWS SSO aware of those users and groups by provisioning them into AWS SSO.

Provisioning when users come from an external identity provider

When using an external IdP, you must provision all users and groups into AWS SSO before you can make any assignments to AWS accounts or applications. In this case you have two options: You can configure [Automatic provisioning \(p. 27\)](#), or you can configure [Manual provisioning \(p. 30\)](#) of your users and groups. Regardless of how you provision users, AWS SSO redirects the AWS Management Console, command line interface, and application authentication to your external IdP. AWS SSO then grants access to those resources based on policies you create in AWS SSO. For more information about provisioning, see [User and group provisioning \(p. 6\)](#).

How to connect to an external identity provider

Use the following procedure to connect to an external identity provider from the AWS SSO console.

To connect to an external identity provider

1. Open the [AWS SSO console](#).
2. Choose **Settings**.
3. On the **Settings** page, under **Identity source**, choose **Change**.
4. On the **Change identity source** page, choose **External identity provider**. Then do the following:
 - a. Under **Service provider metadata**, for **AWS SSO SAML metadata**, choose **Download metadata file** to download the metadata file and save it on your system. The AWS SSO SAML metadata file is required by your external identity provider.
 - b. Under **Identity provider metadata**, choose **Browse**, and locate the metadata file that you downloaded from your external identity provider. Then upload the file. This metadata file contains the necessary public x509 certificate used to trust messages that are sent from the IdP.
 - c. Choose **Next: Review**.

Important

Changing your source to or from Active Directory removes all existing user and group assignments. You must manually reapply assignments after you have successfully changed your source.

5. After you read the disclaimer and are ready to proceed, enter **ACCEPT**.
6. Choose **Change identity source**.

Topics

- [SCIM profile and SAML 2.0 implementation \(p. 27\)](#)
- [Supported identity providers \(p. 32\)](#)
- [Other identity providers \(p. 49\)](#)

SCIM profile and SAML 2.0 implementation

Both SCIM and SAML are important considerations for configuring AWS SSO.

SAML 2.0 implementation

AWS SSO supports identity federation with [SAML \(Security Assertion Markup Language\) 2.0](#). This allows AWS SSO to authenticate identities from external identity providers (IdPs). SAML 2.0 is an open standard used for securely exchanging SAML assertions. SAML 2.0 passes information about a user between a SAML authority (called an identity provider or IdP), and a SAML consumer (called a service provider or SP). The AWS SSO service uses this information to provide federated single sign-on (SSO). Single sign-on allows users to access AWS accounts and configured applications based on their existing identity provider credentials (such as a user name and password).

AWS SSO adds SAML IdP capabilities to your AWS SSO store, AWS Managed Microsoft AD, or to an external identity provider. Users can then SSO into services that support SAML, including the AWS Management Console and third-party applications such as Microsoft 365, Concur, and Salesforce.

The SAML protocol however does not provide a way to query the IdP to learn about users and groups. Therefore, you must make AWS SSO aware of those users and groups by provisioning them into AWS SSO.

SCIM profile

AWS SSO provides support for the System for Cross-domain Identity Management (SCIM) v2.0 standard. SCIM keeps your AWS SSO identities in sync with identities from your IdP. This includes any provisioning, updates, and deprovisioning of users between your IdP and AWS SSO.

For more information about how to implement SCIM, see [Automatic provisioning \(p. 27\)](#). For additional details about AWS SSO's SCIM implementation, see the [AWS SSO SCIM Implementation Developer Guide](#).

Topics

- [Automatic provisioning \(p. 27\)](#)
- [Manual provisioning \(p. 30\)](#)
- [Manage SAML 2.0 certificates \(p. 30\)](#)

Automatic provisioning

AWS SSO supports automatic provisioning (synchronization) of user and group information from your identity provider (IdP) into AWS SSO using the System for Cross-domain Identity Management (SCIM) v2.0 protocol. When you configure SCIM synchronization, you create a mapping of your identity provider (IdP) user attributes to the named attributes in AWS SSO. This causes the expected attributes to match between AWS SSO and your IdP. You configure this connection in your IdP using your SCIM endpoint for AWS SSO and a bearer token that you create in AWS SSO.

Topics

- [Considerations for using automatic provisioning \(p. 28\)](#)
- [How to enable automatic provisioning \(p. 28\)](#)
- [How to disable automatic provisioning \(p. 29\)](#)
- [How to generate a new access token \(p. 29\)](#)
- [How to delete an access token \(p. 29\)](#)
- [How to rotate an access token \(p. 30\)](#)

Considerations for using automatic provisioning

Before you begin deploying SCIM, we recommend that you first review the following important considerations about how it works with AWS SSO. For additional provisioning considerations applicable to your IdP, see [Supported identity providers \(p. 32\)](#).

- If you are provisioning a primary email address, this attribute value must be unique for each user. In some IdPs, the primary email address might not be a real email address. For example, it might be a Universal Principal Name (UPN) that only looks like an email. These IdPs may have a secondary or “other” email address that contains the user’s real email address. You must configure SCIM in your IdP to map the non-Null unique email address to the AWS SSO primary email address attribute. And you must map the user’s non-Null unique sign-in identifier to the AWS SSO user name attribute. Check to see whether your IdP has a single value that is both the sign-in identifier and the user’s email name. If so, you can map that IdP field to both the AWS SSO primary email and the AWS SSO user name.
- For SCIM synchronization to work, every user must have a **First name**, **Last name**, **Username** and **Display name** value specified. If any of these values are missing from a user, that user will not be provisioned.
- The following special characters are invalid when used in attributes that are synchronized with SCIM:
<>;: %
- If you need to use third-party applications, you will first need to map the outbound SAML subject attribute to the user name attribute. If the third-party application needs a routable email address, you must provide the email attribute to your IdP.
- SCIM provisioning and update intervals are controlled by your identity provider. Changes to users and groups in your identity provider are only reflected in AWS SSO after your identity provider sends those changes to AWS SSO. Check with your identity provider for details on the frequency of user and group updates.
- Currently, multivalue attributes (such as multiple emails or phone numbers for a given user) are not provisioned with SCIM. Attempts to synchronize multivalue attributes into AWS SSO with SCIM will fail. To avoid failures, ensure that only a single value is passed for each attribute. If you have users with multivalue attributes, remove or modify the duplicate attribute mappings in SCIM at your IdP for the connection to AWS SSO.
- Verify that the `externalId` SCIM mapping at your IdP corresponds to a value that is unique, always present, and least likely to change for your users. For example, your IdP might provide a guaranteed `objectId` or other identifier that’s not affected by changes to user attributes like name and email. If so, you can map that value to the SCIM `externalId` field. This ensures that your users won’t lose AWS entitlements, assignments, or permissions if you need to change their name or email.
- Users who have not yet been assigned to an application or AWS account cannot be provisioned into AWS SSO. To synchronize users and groups, make sure that they are assigned to the application or other setup that represents your IdP’s connection to AWS SSO.

For more information about AWS SSO’s SCIM implementation, see the [AWS SSO SCIM Implementation Developer Guide](#).

How to enable automatic provisioning

Use the following procedure to enable automatic provisioning of users and groups from your IdP to AWS SSO using the SCIM protocol.

Note

Before you begin this procedure, we recommend that you first review provisioning considerations that are applicable to your IdP. For more information, see [Supported identity providers \(p. 32\)](#).

To enable automatic provisioning in AWS SSO

1. After you have completed the prerequisites, open the [AWS SSO console](#).

2. Choose **Settings** in the left navigation pane.
3. On the **Settings** page, under **Identity source**, next to **Provisioning**, choose **Enable automatic provisioning**. This immediately enables automatic provisioning in AWS SSO and displays the necessary endpoint and access token information.
4. In the **Inbound automatic provisioning** dialog box, copy each of the values for the following options. You will need to paste these in later when you configure provisioning in your IdP.
 - a. **SCIM endpoint**
 - b. **Access token**
5. Choose **Close**.

Once you have completed this procedure, you must configure automatic provisioning in your IdP. For more information, see [Supported identity providers \(p. 32\)](#).

How to disable automatic provisioning

Use the following procedure to disable automatic provisioning in the AWS SSO console.

Important

You must delete the access token before you start this procedure. For more information, see [How to delete an access token \(p. 29\)](#).

To disable automatic provisioning in the AWS SSO console

1. In the [AWS SSO console](#), choose **Settings** in the left navigation pane.
2. On the **Settings** page, under the **Identity source** section, next to **Provisioning**, choose **View details**.
3. On the **Automatic provisioning** page, choose **Disable automatic provisioning**.
4. In the **Disable automatic provisioning** dialog box, review the information, type **DISABLE**, and then choose **Disable automatic provisioning**.

How to generate a new access token

Use the following procedure to generate a new access token in the AWS SSO console.

To generate a new access token

1. In the [AWS SSO console](#), choose **Settings** in the left navigation pane.
2. On the **Settings** page, under the **Identity source** section, next to **Provisioning**, choose **View details**.
3. On the **Automatic provisioning** page, under **Access tokens**, choose **Generate new token**.
4. In the **Generate new access token** dialog box, under **Access token**, choose **Show token**.

How to delete an access token

Use the following procedure to delete an existing access token in the AWS SSO console.

To delete an existing access token

1. In the [AWS SSO console](#), choose **Settings** in the left navigation pane.
2. On the **Settings** page, under the **Identity source** section, next to **Provisioning**, choose **View details**.
3. On the **Automatic provisioning** page, under **Access tokens**, next to the access token you want to delete, choose **Delete**.
4. In the **Delete access token** dialog box, review the information, type **DELETE**, and then choose **Delete access token**.

How to rotate an access token

If your SCIM access token is close to expiring, you can use the following procedure to rotate an existing access token in the AWS SSO console.

To rotate an access token

1. In the [AWS SSO console](#), choose **Settings** in the left navigation pane.
2. On the **Settings** page, under the **Identity source** section, next to **Provisioning**, choose **View details**.
3. On the **Automatic provisioning** page, under **Access tokens**, make a note of the token ID of the token you want to rotate.
4. Follow the steps in [How to generate a new access token \(p. 29\)](#) to create a new token. If you have already created the maximum number of SCIM access tokens, you will first need to delete one of the existing tokens.
5. Go to your identity provider's website and configure the new access token for SCIM provisioning, and then test connectivity to AWS SSO using the new SCIM access token. Once you've confirmed that provisioning is working successfully using the new token, continue to the next step in this procedure.
6. Follow the steps in [How to delete an access token \(p. 29\)](#) to delete the old access token you noted earlier. You can also use the token's creation date as a hint for which token to remove.

Manual provisioning

Some IdPs do not have System for Cross-domain Identity Management (SCIM) support or have an incompatible SCIM implementation. In those cases, you can manually provision users through the AWS SSO console. When you add users to AWS SSO, ensure that you set the user name to be identical to the user name that you have in your IdP. At a minimum, you must have a unique email address and user name. For more information, see [User name and email address uniqueness \(p. 6\)](#).

You must also manage all groups manually in AWS SSO. To do this, you create the groups and add them using the AWS SSO console. For more information, see [Groups \(p. 6\)](#).

Manage SAML 2.0 certificates

AWS SSO uses certificates to set up a SAML trust relationship between AWS SSO and your external identity provider (IdP). When you add an external IdP in AWS SSO, you must also obtain at least one public SAML 2.0 X.509 certificate from the external IdP. That certificate is usually installed automatically during the IdP SAML metadata exchange during trust creation.

As an AWS SSO administrator, you'll occasionally need to replace older IdP certificates with newer ones. For example, you might need to replace an IdP certificate when the expiration date on the certificate approaches. The process of replacing an older certificate with a newer one is referred to as certificate rotation.

Topics

- [Rotate a SAML 2.0 certificate \(p. 30\)](#)
- [Certificate expiration status indicators \(p. 32\)](#)

Rotate a SAML 2.0 certificate

You may need to import certificates periodically in order to rotate invalid or expired certificates issued by your identity provider. This helps to prevent authentication disruption or downtime. All imported certificates are automatically active. Certificates should only be deleted after ensuring that they are no longer in use with the associated identity provider.

You should also consider that some IdPs might not support multiple certificates. In this case, the act of rotating certificates with these IdPs might mean a temporary service disruption for your users. Service is restored when the trust with that IdP has been successfully reestablished. Plan this operation carefully during off peak hours if possible.

Note

As a security best practice, upon any signs of compromise or mishandling of an existing SAML certificate, you should immediately remove and rotate the certificate.

Rotating an AWS SSO certificate is a multistep process that involves the following:

- Obtaining a new certificate from the IdP
- Importing the new certificate into AWS SSO
- Activating the new certificate in the IdP
- Deleting the older certificate

Use all of the following procedures to complete the certificate rotation process while avoiding any authentication downtime.

Step 1: Obtain a new certificate from the IdP

Go to the IdP website and download their SAML 2.0 certificate. Make sure that the certificate file is downloaded in PEM encoded format. Most providers allow you to create multiple SAML 2.0 certificates in the IdP. It is likely that these will be marked as disabled or inactive.

Step 2: Import the new certificate into AWS SSO

Use the following procedure to import the new certificate using the AWS SSO console.

1. In the [AWS SSO console](#), choose **Settings**.
2. On the **Settings** page, under **Identity source**, next to **Authentication**, choose **View details**.
3. On the **SAML 2.0 authentication** page, under **Identity provider metadata**, choose **Manage certificates**.
4. On the **Manage SAML 2.0 certificates** page, choose **Import Certificate**.

At this point, AWS SSO will trust all incoming SAML messages signed from both of the certificates that you have imported.

Step 3: Activate the new certificate in the IdP

Go back to the IdP website and mark the new certificate that you created earlier as primary or active. At this point all SAML messages signed by the IdP should be using the new certificate.

Step 4: Delete the old certificate

Use the following procedure to complete the certificate rotation process for your IdP. There must always be at least one valid certificate listed, and it cannot be removed.

Note

Make sure that your identity provider is no longer signing SAML responses with this certificate before deleting it.

1. On the **Manage SAML 2.0 certificates** page, choose the certificate that you want to delete. Choose **Delete**.
2. In the **Delete SAML 2.0 certificate** dialog box, type **DELETE** to confirm, and then choose **Delete**.
3. Return to the IdP's website and perform the necessary steps to remove the older inactive certificate.

Certificate expiration status indicators

While on the **Manage SAML 2.0 certificates** page, you might notice colored status indicator icons. These icons appear in the **Expires on** column next to each certificate in the list. The following describes the criteria that AWS SSO uses to determine which icon is displayed for each certificate.

- **Red** – Indicates that a certificate is currently expired.
- **Yellow** – Indicates that a certificate will expire in 90 days or less.
- **Green** – Indicates that a certificate is currently valid and will remain valid for at least 90 more days.

To check the current status of a certificate

1. In the [AWS SSO console](#), choose **Settings**.
2. On the **Settings** page, under **Identity source**, next to **Authentication**, choose **View details**.
3. On the **SAML 2.0 authentication** page, under **Identity provider metadata**, choose **Manage certificates**.
4. On the **Manage SAML 2.0 certificates** page, review the status of the certificates in the list as indicated in the **Expires on** column.

Supported identity providers

The following external identity providers have been tested with the AWS SSO SCIM implementation.

Topics

- [Azure AD \(p. 32\)](#)
- [Okta \(p. 35\)](#)
- [OneLogin \(p. 39\)](#)
- [Ping Identity \(p. 42\)](#)

Azure AD

AWS SSO supports automatic provisioning (synchronization) of user and group information from Azure AD into AWS SSO using the System for Cross-domain Identity Management (SCIM) v2.0 protocol. You configure this connection in Azure AD using your SCIM endpoint for AWS SSO and a bearer token that is created automatically by AWS SSO. When you configure SCIM synchronization, you create a mapping of your user attributes in Azure AD to the named attributes in AWS SSO. This causes the expected attributes to match between AWS SSO and your IdP.

The following steps walk you through how to enable automatic provisioning of users and groups from Azure AD to AWS SSO using the AWS Single Sign-On app in the Azure AD Application Gallery and the SCIM protocol.

Note

Before you begin deploying SCIM, we recommend that you first review [Considerations for using automatic provisioning \(p. 28\)](#), and then continue reviewing [Prerequisites \(p. 32\)](#) and [Additional considerations \(p. 33\)](#) in the next sections.

Prerequisites

You will need the following before you can get started:

- An Azure AD tenant. For more information, see [Quickstart: Set up a tenant](#) on Microsoft's website.

- An AWS SSO-enabled account ([free](#)). For more information, see [Enable AWS SSO](#).
- A SAML connection from your Azure AD account to AWS SSO, as described in [Tutorial: Azure Active Directory single sign-on \(SSO\) integration with AWS Single Sign-On](#) on Microsoft's website.

Important

Make sure that all users in Azure AD have filled out **First name**, **Last name**, and **Display name** values in their user properties. Otherwise, automatic provisioning won't work with Azure AD.

Additional considerations

If an attribute is removed from a user in Azure AD, that attribute will not be removed from the corresponding user in AWS SSO. This is a known limitation in Azure AD. If an attribute is changed to a different (non-empty) value on a user, that change will be synchronized to AWS SSO.

Step 1: Set up AWS SSO and configure automatic provisioning

To get started, you'll need to first follow the instructions in [Tutorial: Configure AWS Single Sign-On for automatic user provisioning](#). These instructions walk you through the following:

- Enable AWS SSO.
- Install the AWS Single Sign-On app from the Azure AD Application Gallery.
- Configure automatic provisioning (SCIM) within the Azure portal.

Step 2: (Optional) Configure attribute-based access control

Now that you have configured Azure AD to work with AWS SSO, you can optionally choose to configure attribute-based access control (ABAC). ABAC is an authorization strategy that defines permissions based on attributes.

With Azure AD, you have two different ways to configure ABAC for use with AWS SSO. Choose either of the following methods.

Method 1: Configure ABAC using Azure AD

This method can be used when you need to define which attributes in Azure AD can be used by AWS SSO to manage access to your AWS resources. Once defined, Azure AD sends these attributes to AWS SSO through SAML assertions. You will then need to [Create a permission set \(p. 53\)](#) in AWS SSO to manage access based on the attributes you passed from Azure AD.

Before you begin this procedure, you first need to enable the [Attributes for access control \(p. 59\)](#) feature. For more information about how to do this, see [Enable and configure attributes for access control \(p. 60\)](#).

To configure user attributes in Azure AD for access control in AWS SSO

1. While signed into the Azure portal, navigate to **Azure Active Directory, Enterprise applications**. Search for the name of the application that you created previously to form your SAML connection. Then choose the application.
2. Choose **Single sign-on**.
3. In the **User Attributes & Claims** section, choose **Edit**.
4. On the **User Attributes & Claims** page, do the following:
 - a. Choose **Add new claim**
 - b. For **Name**, enter `AccessControl:AttributeName`. Replace `AttributeName` with the name of the attribute you are expecting in AWS SSO. For example, `AccessControl:Department`.

- c. For **Namespace**, enter **`https://aws.amazon.com/SAML/Attributes`**.
 - d. For **Source**, choose **Attribute**.
 - e. For **Source attribute**, use the drop-down list to choose the Azure AD user attributes. For example, **`user.department`**.
5. Repeat the previous step for each attribute you need to send to AWS SSO in the SAML assertion.
 6. Choose **Save**.

Method 2: Configure ABAC using AWS SSO

With this method, you use the [Attributes for access control \(p. 59\)](#) feature in AWS SSO to pass an **Attribute** element with the **Name** attribute set to `https://aws.amazon.com/SAML/Attributes/AccessControl:{TagKey}`. You can use this element to pass attributes as session tags in the SAML assertion. For more information about session tags, see [Passing session tags in AWS STS](#) in the *IAM User Guide*.

To pass attributes as session tags, include the **AttributeValue** element that specifies the value of the tag. For example, to pass the tag key-value pair `CostCenter = blue`, use the following attribute:

```
<saml:AttributeStatement>
<saml:Attribute Name="https://aws.amazon.com/SAML/Attributes/AccessControl:CostCenter">
<saml:AttributeValue>blue
</saml:AttributeValue>
</saml:Attribute>
</saml:AttributeStatement>
```

If you need to add multiple attributes, include a separate **Attribute** element for each tag.

Troubleshooting

The following can help you troubleshoot some common issues you might encounter while setting up automatic provisioning with Azure AD.

Azure AD users are not synchronizing to AWS SSO

This might be due to a syntax issue that AWS SSO has flagged when a new user is being added to AWS SSO. You can confirm this by checking the Azure audit logs for failed events, such as an **'Export'**. The **Status Reason** for this event will state:

```
{"schema":["urn:ietf:params:scim:api:messages:2.0:Error"],"detail":"Request is unparsable, syntactically incorrect, or violates schema.","status":"400"}
```

You can also check AWS CloudTrail for the failed event. This can be done by searching in the **Event History** console of CloudTrail using the following filter:

```
"eventName": "CreateUser"
```

The error in the CloudTrail event will state the following:

```
"errorCode": "ValidationException",
"errorMessage": "Currently list attributes only allow single item"
```

Ultimately, this exception means that one of the values passed from Azure contained more values than anticipated. The solution here is to review the attributes of the user in Azure AD, ensuring that none

contain duplicate values. One common example of duplicate values is having multiple values present for contact numbers such as **mobile**, **work**, and **fax**. Although separate values, they are all passed to AWS SSO under the single parent attribute **phoneNumbers**.

Okta

AWS SSO supports automatic provisioning (synchronization) of user and group information from Okta into AWS SSO using the System for Cross-domain Identity Management (SCIM) v2.0 protocol. To configure this connection in Okta, you use your SCIM endpoint for AWS SSO and a bearer token that is created automatically by AWS SSO. When you configure SCIM synchronization, you create a mapping of your user attributes in Okta to the named attributes in AWS SSO. This causes the expected attributes to match between AWS SSO and your IdP.

Okta supports the following provisioning features when connected to AWS SSO through SCIM:

- Create users – Users assigned to the AWS SSO application in Okta will be provisioned in AWS SSO.
- Update user attributes – Attribute changes for users who are assigned to the AWS SSO application in Okta will be updated in AWS SSO.
- Deactivate users – Users who are unassigned from the AWS SSO application in Okta will be disabled in AWS SSO.
- Group push – Groups (and their members) in Okta are synchronized to AWS SSO.

The following steps walk you through how to enable automatic provisioning of users and groups from Okta to AWS SSO using the SCIM protocol.

Note

Before you begin deploying SCIM, we recommend that you first review the [Considerations for using automatic provisioning \(p. 28\)](#). Then continue reviewing additional considerations in the next section.

Topics

- [Additional considerations \(p. 35\)](#)
- [Prerequisites \(p. 36\)](#)
- [Step 1: Enable provisioning in AWS SSO \(p. 36\)](#)
- [Step 2: Configure provisioning in Okta \(p. 36\)](#)
- [Step 3: Assign access for users and groups in Okta \(p. 37\)](#)
- [\(Optional\) Step 4: Configure user attributes in Okta for access control in AWS SSO \(p. 37\)](#)
- [\(Optional\) Passing attributes for access control \(p. 38\)](#)
- [Troubleshooting \(p. 38\)](#)

Additional considerations

The following are important considerations about Okta that can affect how you implement provisioning with AWS SSO.

- Using the same Okta group for both assignments and group push is not currently supported. To maintain consistent group memberships between Okta and AWS SSO, you need to create a separate group and configure it to push groups to AWS SSO.
- If you update a user's address you must have **streetAddress**, **city**, **state**, **zipCode** and the **countryCode** value specified. If any of these values are not specified for the Okta user at the time of synchronization, the user or changes to the user will not be provisioned.
- Entitlements and role attributes are not supported and cannot be synced to AWS SSO.

Prerequisites

You will need the following before you can get started:

- An Okta account ([free trial](#)) with Okta's [AWS Single Sign-On application](#) installed. Note also that for paid Okta products, you might need to confirm that your Okta license supports “lifecycle management” or similar capabilities that enable outbound provisioning. These features might be necessary to configure SCIM from Okta to AWS SSO.
- A SAML connection from your Okta account to AWS SSO, as described in [How to Configure SAML 2.0 for AWS Single Sign-On](#).
- An AWS SSO-enabled account ([free](#)). For more information, see [Enable AWS SSO](#).

Step 1: Enable provisioning in AWS SSO

In this first step, you use the AWS SSO console to enable automatic provisioning.

To enable automatic provisioning in AWS SSO

1. After you have completed the prerequisites, open the [AWS SSO console](#).
2. Choose **Settings** in the left navigation pane.
3. On the **Settings** page, under **Identity source**, next to **Provisioning**, choose **Enable automatic provisioning**. This immediately enables automatic provisioning in AWS SSO and displays the necessary endpoint and access token information.
4. In the **Inbound automatic provisioning** dialog box, copy each of the values for the following options. You will need to paste these in later when you configure provisioning in your IdP.
 - a. **SCIM endpoint**
 - b. **Access token**
5. Choose **Close**.

You have set up provisioning in the AWS SSO console. Now you need to do the remaining tasks using the Okta user interface as described in the following procedures.

Step 2: Configure provisioning in Okta

Use the following procedure in the Okta admin portal to enable integration between AWS SSO and the AWS Single Sign-On app.

To configure provisioning in Okta

1. In a separate browser window, log in to the Okta admin portal and navigate to the [AWS Single Sign-On app](#).
2. On the **AWS Single Sign-On app** page, choose the **Provisioning** tab, and then choose **Integration**.
3. Choose **Configure API Integration**, and then select the check box next to **Enable API integration** to enable provisioning.
4. In the previous procedure you copied the **SCIM endpoint** value in AWS SSO. Paste that value into the **Base URL** field in Okta. Make sure that you remove the trailing forward slash at the end of the URL. Also, in the previous procedure you copied the **Access token** value in AWS SSO. Paste that value into the **API Token** field in Okta.
5. Choose **Test API Credentials** to verify the credentials entered are valid.
6. Choose **Save**.
7. Under **Settings**, choose **To App**, choose **Edit**, and then select the **Enable** check box for each of the **Provisioning Features** you want to enable.

8. Choose **Save**.

By default, no users or groups are assigned to your Okta AWS Single Sign-On app. Therefore you must complete the next procedure to begin synchronizing users and groups to AWS SSO.

Step 3: Assign access for users and groups in Okta

Use the following procedures in Okta to assign access to your users and groups. Okta users who belong to groups that you assign here are synchronized automatically to AWS SSO. To minimize administrative overhead in both Okta and AWS SSO, we recommend that you assign and *push* groups instead of individual users.

After you complete this step and the first synchronization with SCIM is completed, the users and groups that you have assigned appear in AWS SSO. Those users are able to access the AWS SSO user portal using their Okta credentials.

To assign access for users in Okta

1. In the **AWS Single Sign-On app** page, choose the **Assignments** tab.
2. In the **Assignments** page, choose **Assign**, and then choose **Assign to People**.
3. Choose the Okta user or users whom you want to assign access to the AWS Single Sign-On app. Choose **Assign**, choose **Save and Go Back**, and then choose **Done**. This starts the process of provisioning the user or users into AWS SSO.

To assign access for groups in Okta

1. On the **AWS Single Sign-On app** page, choose the **Assignments** tab.
2. In the **Assignments** page, choose **Assign**, and then choose **Assign to Groups**.
3. Choose the Okta group or groups that you want to assign access to the AWS Single Sign-On app. Choose **Assign**, choose **Save and Go Back**, and then choose **Done**. This starts the process of provisioning the users in the group into AWS SSO.
4. Choose the **Push Groups** tab, choose the Okta group or groups that you chose in the previous step. Then choose **Save**. The group you choose must be different from the group assigned to the application. The group status changes to **Active** after the group and its members have successfully been pushed to AWS SSO.

To grant your Okta users access to AWS accounts and cloud applications, complete the following applicable procedures from the AWS SSO console:

- To grant access to AWS accounts, see [Assign user access \(p. 52\)](#).
- To grant access to cloud applications, see [Assign user access \(p. 76\)](#).

(Optional) Step 4: Configure user attributes in Okta for access control in AWS SSO

This is an optional procedure for Okta should you choose to configure attributes you will use in AWS SSO to manage access to your AWS resources. The attributes you define in Okta will be passed in a SAML assertion to AWS SSO, you will then create a permission set in AWS SSO to manage access based on the attributes you passed from Okta.

Before you begin this procedure, you first need to enable the [Attributes for access control \(p. 59\)](#) feature. For more information about how to do this, see [Enable and configure attributes for access control \(p. 60\)](#).

To configure user attributes in Okta for access control in AWS SSO

1. In a separate browser window, log in to the Okta admin portal and navigate to the [AWS Single Sign-On app](#).
2. On the **AWS Single Sign-On app** page, choose the **Sign On** tab, and then choose **Edit**.
3. In the **SAML** section, expand **Attributes (Optional)**.
4. In the **Attribute Statements (optional)** section, do the following for each attribute where you will use AWS SSO for access control:
 - a. In the **Name** field, enter `https://aws.amazon.com/SAML/Attributes/AccessControl:AttributeName`, and replace **AttributeName** with the name of the attribute you are expecting in AWS SSO. For example, `https://aws.amazon.com/SAML/Attributes/AccessControl:Department`.
 - b. In the **Name Format** field, choose **URI reference**.
 - c. In the **Value** field, enter `user.AttributeName`, replace **AttributeName** with the Okta default user profile variable name. For example, `user.department`. To find your Okta default user profile variable name, see [View the Okta default user profile](#) on the Okta website.
5. (Optional) Choose **Preview SAML**, to review a sample SAML assertion that includes the new attributes.
6. Choose **Save**.

(Optional) Passing attributes for access control

You can optionally use the [Attributes for access control \(p. 59\)](#) feature in AWS SSO to pass an **Attribute** element with the **Name** attribute set to `https://aws.amazon.com/SAML/Attributes/AccessControl:{TagKey}`. This element allows you to pass attributes as session tags in the SAML assertion. For more information about session tags, see [Passing session tags in AWS STS](#) in the *IAM User Guide*.

To pass attributes as session tags, include the **AttributeValue** element that specifies the value of the tag. For example, to pass the tag key-value pair `CostCenter = blue`, use the following attribute.

```
<saml:AttributeStatement>
<saml:Attribute Name="https://aws.amazon.com/SAML/Attributes/AccessControl:CostCenter">
<saml:AttributeValue>blue
</saml:AttributeValue>
</saml:Attribute>
</saml:AttributeStatement>
```

If you need to add multiple attributes, include a separate **Attribute** element for each tag.

Troubleshooting

The following can help you troubleshoot some common issues you might encounter while setting up automatic provisioning with Okta.

Base URL: Does not match required pattern

The SCIM endpoint URL that you pasted into **Base URL** likely contains a trailing forward slash (/). Remove the forward slash from the SCIM endpoint URL before pasting into **Base URL**. For example, `https://scim.us-east-2.amazonaws.com/xxxxxxxx-xxxx-xxxxx-xxxxxx-xxxx/scim/v2`.

Error during synchronization

After you have started synchronization, you might see the following error:

Automatic profile push of <user> to app AWS Single Sign-On failed: Error while trying to push profile update for <user>@Corp.Example.com: Bad Request. Errors reported by remote server: Request is unparsable, syntactically incorrect, or violates schema.

For SCIM synchronization to work:

- Every user must have a **First name**, **Last name**, **Username**, and **Display name** value specified. If any of these values are missing from a user, that user will not be provisioned.
- Usernames should be mapped to attributes that are unique within your directory in Okta.
- The following special characters must not be used in attributes that are synchronized with SCIM: <> ; : %
- If you update a user's address you must have **streetAddress**, **city**, **state**, **zipCode** and the **countryCode** value specified. If any of these values are not specified for the Okta user at the time of synchronization, the user or changes to the user will not be provisioned.

OneLogin

AWS SSO supports automatic provisioning (synchronization) of user and group information from OneLogin into AWS SSO using the System for Cross-domain Identity Management (SCIM) v2.0 protocol. You configure this connection in OneLogin, using your SCIM endpoint for AWS SSO and a bearer token that is created automatically by AWS SSO. When you configure SCIM synchronization, you create a mapping of your user attributes in OneLogin to the named attributes in AWS SSO. This causes the expected attributes to match between AWS SSO and OneLogin.

The following steps walk you through how to enable automatic provisioning of users and groups from OneLogin to AWS SSO using the SCIM protocol.

Note

Before you begin deploying SCIM, we recommend that you first review the [Considerations for using automatic provisioning \(p. 28\)](#).

Topics

- [Prerequisites \(p. 39\)](#)
- [Step 1: Enable provisioning in AWS SSO \(p. 39\)](#)
- [Step 2: Configure provisioning in OneLogin \(p. 40\)](#)
- [\(Optional\) Step 3: Configure user attributes in OneLogin for access control in AWS SSO \(p. 41\)](#)
- [\(Optional\) Passing attributes for access control \(p. 41\)](#)
- [Troubleshooting \(p. 41\)](#)

Prerequisites

You will need the following before you can get started:

- A OneLogin account. If you do not have an existing account, you may be able to obtain a free trial or developer account from the [OneLogin website](#).
- An AWS SSO-enabled account ([free](#)). For more information, see [Enable AWS SSO](#).
- A SAML connection from your OneLogin account to AWS SSO. For more information, see [Enabling Single Sign-On Between OneLogin and AWS](#) on the AWS Partner Network Blog.

Step 1: Enable provisioning in AWS SSO

In this first step, you use the AWS SSO console to enable automatic provisioning.

To enable automatic provisioning in AWS SSO

1. After you have completed the prerequisites, open the [AWS SSO console](#).
2. Choose **Settings** in the left navigation pane.
3. On the **Settings** page, under **Identity source**, next to **Provisioning**, choose **Enable automatic provisioning**. This immediately enables automatic provisioning in AWS SSO and displays the necessary endpoint and access token information.
4. In the **Inbound automatic provisioning** dialog box, copy each of the values for the following options. You will need to paste these in later when you configure provisioning in your IdP.
 - a. **SCIM endpoint**
 - b. **Access token**
5. Choose **Close**.

You have now set up provisioning in the AWS SSO console. Now you need to do the remaining tasks using the OneLogin admin console as described in the following procedure.

Step 2: Configure provisioning in OneLogin

Use the following procedure in the OneLogin admin console to enable integration between AWS SSO and the AWS Single Sign-On app. This procedure assumes you have already configured the AWS Single Sign-On application in OneLogin for SAML authentication. If you have not yet created this SAML connection, please do so before proceeding and then return here to complete the SCIM provisioning process. For more information about configuring SAML with OneLogin, see [Enabling Single Sign-On Between OneLogin and AWS](#) on the AWS Partner Network Blog.

To configure provisioning in OneLogin

1. Sign in to OneLogin, and then navigate to **Applications > Applications**.
2. On the **Applications** page, search for the application you created previously to form your SAML connection with AWS SSO. Choose it and then choose **Configuration** from the left navigation bar.
3. In the previous procedure, you copied the **SCIM endpoint** value in AWS SSO. Paste that value into the **SCIM Base URL** field in OneLogin. Make sure that you remove the trailing forward slash at the end of the URL. Also, in the previous procedure you copied the **Access token** value in AWS SSO. Paste that value into the **SCIM Bearer Token** field in OneLogin.
4. Next to **API Connection**, click **Enable**, and then click **Save** to complete the configuration.
5. In the left navigation bar, choose **Provisioning**.
6. Select the check boxes for **Enable provisioning**, **Create user**, **Delete user**, and **Update user**, and then choose **Save**.
7. In the left navigation bar, choose **Users**.
8. Click **More Actions** and choose **Sync logins**. You should receive the message *Synchronizing users with AWS Single Sign-on*.
9. Click **More Actions** again, and then choose **Reapply entitlement mappings**. You should receive the message *Mappings are being reapplied*.
10. At this point, the provisioning process should begin. To confirm this, navigate to **Activity > Events**, and monitor the progress. Successful provisioning events, as well as errors, should appear in the event stream.
11. To verify that your users and groups have all been successfully synchronized to AWS SSO, return to the AWS SSO console and choose **Users**. Your synchronized users from OneLogin will appear on the **Users** page. You can also view your synchronized groups on the **Groups** page.
12. To synchronize user changes automatically to AWS SSO, navigate to the **Provisioning** page, locate the **Require admin approval before this action is performed** section, de-select **Create User**, **Delete User**, and/or **Update User**, and click **Save**.

(Optional) Step 3: Configure user attributes in OneLogin for access control in AWS SSO

This is an optional procedure for OneLogin should you choose to configure attributes you will use in AWS SSO to manage access to your AWS resources. The attributes that you define in OneLogin will be passed in a SAML assertion to AWS SSO. You will then create a permission set in AWS SSO to manage access based on the attributes you passed from OneLogin.

Before you begin this procedure, you must first enable the [Attributes for access control \(p. 59\)](#) feature. For more information about how to do this, see [Enable and configure attributes for access control \(p. 60\)](#).

To configure user attributes in OneLogin for access control in AWS SSO

1. Sign in to OneLogin, and then navigate to **Applications > Applications**.
2. On the **Applications** page, search for the application you created previously to form your SAML connection with AWS SSO. Choose it and then choose **Parameters** from the left navigation bar.
3. In the **Required Parameters** section, do the following for each attribute you want to use in AWS SSO:
 - a. Choose **+**.
 - b. In **Field name**, enter `https://aws.amazon.com/SAML/Attributes/AccessControl:AttributeName`, and replace `AttributeName` with the name of the attribute you are expecting in AWS SSO. For example, `https://aws.amazon.com/SAML/Attributes/AccessControl:Department`.
 - c. Under **Flags**, check the box next to **Include in SAML assertion**, and choose **Save**.
 - d. In the **Value** field, use the drop-down list to choose the OneLogin user attributes. For example, **Department**.
4. Choose **Save**.

(Optional) Passing attributes for access control

You can optionally use the [Attributes for access control \(p. 59\)](#) feature in AWS SSO to pass an `Attribute` element with the `Name` attribute set to `https://aws.amazon.com/SAML/Attributes/AccessControl:{TagKey}`. This element allows you to pass attributes as session tags in the SAML assertion. For more information about session tags, see [Passing session tags in AWS STS](#) in the *IAM User Guide*.

To pass attributes as session tags, include the `AttributeValue` element that specifies the value of the tag. For example, to pass the tag key-value pair `CostCenter = blue`, use the following attribute.

```
<saml:AttributeStatement>
<saml:Attribute Name="https://aws.amazon.com/SAML/Attributes/AccessControl:CostCenter">
<saml:AttributeValue>blue
</saml:AttributeValue>
</saml:Attribute>
</saml:AttributeStatement>
```

If you need to add multiple attributes, include a separate `Attribute` element for each tag.

Troubleshooting

The following can help you troubleshoot some common issues you might encounter while setting up automatic provisioning with OneLogin.

Groups are not provisioned to AWS SSO

By default, groups may not be provisioned from OneLogin to AWS SSO. Ensure that you've enabled group provisioning for your AWS SSO application in OneLogin. To do this, sign in to the OneLogin admin console, and check to make sure that the **Include in User Provisioning** option is selected under the properties of the AWS SSO application (**AWS SSO application > Parameters > Groups**). For more details on how to create groups in OneLogin, including how to synchronize OneLogin roles as groups in SCIM, please see the [OneLogin website](#).

Nothing is synchronized from OneLogin to AWS SSO, despite all settings being correct

In addition to the note above regarding admin approval, you will need to **Reapply entitlement mappings** for many configuration changes to take effect. This can be found in **Applications > Applications > AWS SSO application > More Actions**. You can see details and logs for most actions in OneLogin, including synchronization events, under **Activity > Events**.

I've deleted or disabled a group in OneLogin, but it still appears in AWS SSO

OneLogin currently does not support the SCIM DELETE operation for groups, which means that the group will continue to exist in AWS SSO. You must therefore remove the group from AWS SSO directly to ensure that any corresponding permissions in AWS SSO for that group are removed.

I deleted a group in AWS SSO without first deleting it from OneLogin and now I'm having user/group sync issues

To remedy this situation, first ensure that you do not have any redundant group provisioning rules or configurations in OneLogin. For example, a group directly assigned to an application along with a rule that publishes to the same group. Next, delete any undesirable groups in AWS SSO. Finally, in OneLogin, **Refresh** the entitlements (**AWS SSO App > Provisioning > Entitlements**), and then **Reapply entitlement mappings** (**AWS SSO App > More Actions**). To avoid this issue in the future, first make the change to stop provisioning the group in OneLogin, then delete the group from AWS SSO.

Ping Identity

The following Ping Identity products have been tested with AWS SSO.

Topics

- [PingFederate \(p. 42\)](#)
- [PingOne \(p. 46\)](#)

PingFederate

AWS SSO supports automatic provisioning (synchronization) of user and group information from the PingFederate product by Ping Identity (hereafter "Ping") into AWS SSO. This provisioning uses the System for Cross-domain Identity Management (SCIM) v2.0 protocol. You configure this connection in PingFederate using your AWS SSO SCIM endpoint and access token. When you configure SCIM synchronization, you create a mapping of your user attributes in PingFederate to the named attributes in AWS SSO. This causes the expected attributes to match between AWS SSO and PingFederate.

This guide is based on PingFederate version 10.2. Steps for other versions may vary. Contact Ping for more information about how to configure provisioning to AWS SSO for other versions of PingFederate.

The following steps walk you through how to enable automatic provisioning of users and groups from PingFederate to AWS SSO using the SCIM protocol.

Note

Before you begin deploying SCIM, we recommend that you first review the [Considerations for using automatic provisioning \(p. 28\)](#). Then continue reviewing additional considerations in the next section.

Topics

- [Prerequisites \(p. 43\)](#)
- [Additional considerations \(p. 43\)](#)
- [Step 1: Enable provisioning in AWS SSO \(p. 43\)](#)
- [Step 2: Configure provisioning in PingFederate \(p. 44\)](#)
- [\(Optional\) Step 3: Configure user attributes in PingFederate for access control in AWS SSO \(p. 45\)](#)
- [\(Optional\) Passing attributes for access control \(p. 46\)](#)

Prerequisites

You will need the following before you can get started:

- A working PingFederate server. If you do not have an existing PingFederate server, you might be able to obtain a free trial or developer account from the [Ping Identity](#) website. The trial includes licenses and software downloads and associated documentation.
- A copy of the PingFederate AWS Single Sign-On Connector software installed on your PingFederate server. For more information about how to obtain this software, see [AWS Single Sign-On Connector](#) on the Ping Identity website.
- An AWS SSO-enabled account ([free](#)). For more information, see [Enable AWS SSO](#).
- A SAML connection from your PingFederate instance to AWS SSO. For instructions on how to configure this connection, see the PingFederate documentation. In summary, the recommended path is to use the AWS Single Sign-On Connector to configure "Browser SSO" in PingFederate, using the "download" and "import" metadata features on both ends to exchange SAML metadata between PingFederate and AWS SSO.

Additional considerations

The following are important considerations about PingFederate that can affect how you implement provisioning with AWS SSO.

- If an attribute (such as a phone number) is removed from a user in the data store configured in PingFederate, that attribute will not be removed from the corresponding user in AWS SSO. This is a known limitation in PingFederate's provisioner implementation. If an attribute is changed to a different (non-empty) value on a user, that change will be synchronized to AWS SSO.

Step 1: Enable provisioning in AWS SSO

In this first step, you use the AWS SSO console to enable automatic provisioning.

To enable automatic provisioning in AWS SSO

1. After you have completed the prerequisites, open the [AWS SSO console](#).
2. Choose **Settings** in the left navigation pane.
3. On the **Settings** page, under **Identity source**, next to **Provisioning**, choose **Enable automatic provisioning**. This immediately enables automatic provisioning in AWS SSO and displays the necessary endpoint and access token information.
4. In the **Inbound automatic provisioning** dialog box, copy each of the values for the following options. You will need to paste these in later when you configure provisioning in your IdP.
 - a. **SCIM endpoint**
 - b. **Access token**
5. Choose **Close**.

Now that you have set up provisioning in the AWS SSO console, you must complete the remaining tasks using the PingFederate administrative console. The steps are described in the following procedure.

Step 2: Configure provisioning in PingFederate

Use the following procedure in the PingFederate administrative console to enable integration between AWS SSO and the AWS Single Sign-On Connector. This procedure assumes that you have already installed the AWS Single Sign-On Connector software. If you have not yet done so, refer to [Prerequisites \(p. 43\)](#), and then complete this procedure to configure SCIM provisioning.

Important

If your PingFederate server has not been previously configured for outbound SCIM provisioning, you may need to make a configuration file change to enable provisioning. For more information, see Ping documentation. In summary, you must modify the `pf.provisioner.mode` setting in the `pingfederate-<version>/pingfederate/bin/run.properties` file to a value other than `OFF` (which is the default), and restart the server if currently running. For example, you may choose to use `STANDALONE` if you don't currently have a high-availability configuration with PingFederate.

To configure provisioning in PingFederate

1. Sign on to the PingFederate administrative console.
2. Select **Applications** from the top of the page, then click **SP Connections**.
3. Locate the application you created previously to form your SAML connection with AWS SSO, and click on the connection name.
4. Select **Connection Type** from the dark navigation headings near the top of the page. You should see **Browser SSO** already selected from your previous configuration of SAML. If not, you must complete those steps first before you can continue.
5. Select the **Outbound Provisioning** check box, choose **AWS SSO Cloud Connector** as the type, and click **Save**. If **AWS SSO Cloud Connector** does not appear as an option, ensure that you have installed the AWS Single Sign-On Connector and have restarted your PingFederate server.
6. Click **Next** repeatedly until you arrive on the **Outbound Provisioning** page, and then click the **Configure Provisioning** button.
7. In the previous procedure, you copied the **SCIM endpoint** value in AWS SSO. Paste that value into the **SCIM URL** field in the PingFederate console. Make sure that you remove the trailing forward slash at the end of the URL. Also, in the previous procedure you copied the **Access token** value in AWS SSO. Paste that value into the **Access Token** field in the PingFederate console. Click **Save**.
8. On the **Channel Configuration (Configure Channels)** page, click **Create**.
9. Enter a **Channel Name** for this new provisioning channel (such as `AWSSSOchannel1`), and click **Next**.
10. On the **Source** page, choose the **Active Data Store** you want to use for your connection to AWS SSO, and click **Next**.

Note

If you have not yet configured a data source, you must do so now. See the Ping product documentation for information on how to choose and configure a data source in PingFederate.

11. On the **Source Settings** page, confirm all values are correct for your installation, then click **Next**.
12. On the **Source Location** page, enter settings appropriate to your data source, and then click **Next**. For example, if using Active Directory as an LDAP directory:
 - a. Enter the **Base DN** of your AD forest (such as `DC=myforest,DC=mydomain,DC=com`).
 - b. In **Users > Group DN**, specify a single group that contains all of the users that you want to provision to AWS SSO. If no such single group exists, create that group in AD, return to this setting, and then enter the corresponding DN.
 - c. Specify whether to search subgroups (**Nested Search**), and any required LDAP **Filter**.

- d. In **Groups > Group DN**, specify a single group that contains all of the groups that you want to provision to AWS SSO. In many cases, this may be the same DN as you specified in the **Users** section. Enter **Nested Search** and **Filter** values as required.
13. On the **Attribute Mapping** page, ensure the following, and then click **Next**:
 - a. The **userName** field must be mapped to an **Attribute** that is formatted as an email (user@domain.com). It must also match the value that the user will use to log in to Ping. This value in turn is populated in the SAML `nameId` claim during federated authentication and used for matching to the user in AWS SSO. For example, when using Active Directory, you may choose to specify the `UserPrincipalName` as the **userName**.
 - b. Other fields suffixed with a * must be mapped to attributes that are non-null for your users.
14. On the **Activation & Summary** page, set the **Channel Status** to **Active** to cause the synchronization to start immediately after the configuration is saved.
15. Confirm that all configuration values on the page are correct, and click **Done**.
16. On the **Manage Channels** page, click **Save**.
17. At this point, provisioning will start. To confirm activity, you can view the **provisioner.log** file, located by default in the **pingfederate-`<version>`/pingfederate/log** directory on your PingFederate server.
18. To verify that users and groups have been successfully synchronized to AWS SSO, return to the AWS SSO Console and choose **Users**. Synchronized users from PingFederate will appear on the **Users** page. You can also view synchronized groups on the **Groups** page.

(Optional) Step 3: Configure user attributes in PingFederate for access control in AWS SSO

This is an optional procedure for PingFederate should you choose to configure attributes you will use in AWS SSO to manage access to your AWS resources. The attributes that you define in PingFederate will be passed in a SAML assertion to AWS SSO. You will then create a permission set in AWS SSO to manage access based on the attributes you passed from PingFederate.

Before you begin this procedure, you must first enable the [Attributes for access control \(p. 59\)](#) feature. For more information about how to do this, see [Enable and configure attributes for access control \(p. 60\)](#).

To configure user attributes in PingFederate for access control in AWS SSO

1. Sign on to the PingFederate administrative console.
2. Choose **Applications** from the top of the page, then click **SP Connections**.
3. Locate the application you created previously to form your SAML connection with AWS SSO, and click on the connection name.
4. Choose **Browser SSO** from the dark navigation headings near the top of the page. Then click on **Configure Browser SSO**.
5. On the **Configure Browser SSO** page, choose **Assertion Creation**, and then click on **Configure Assertion Creation**.
6. On the **Configure Assertion Creation** page, choose **Attribute Contract**.
7. On the **Attribute Contract** page, under **Extend the Contract** section, add a new attribute by performing the following steps:
 - a. In the text box, enter `https://aws.amazon.com/SAML/Attributes/AccessControl:AttributeName`, replace **AttributeName** with the name of the attribute you are expecting in AWS SSO. For example, `https://aws.amazon.com/SAML/Attributes/AccessControl:Department`.
 - b. For **Attribute Name Format**, choose `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`.
 - c. Choose **Add**, and then choose **Next**.

8. On the **Authentication Source Mapping** page, choose the Adapter Instance configured with your application.
9. On the **Attribute Contract Fulfillment** page, choose the **Source** (*data store*) and **Value** (*data store attribute*) for the **Attribute Contract** `https://aws.amazon.com/SAML/Attributes/AccessControl:Department`.

Note

If you have not yet configured a data source, you will need to do so now. See the Ping product documentation for information on how to choose and configure a data source in PingFederate.

10. Click **Next** repeatedly until you arrive on the **Activation & Summary** page, and then click **Save**.

(Optional) Passing attributes for access control

You can optionally use the [Attributes for access control \(p. 59\)](#) feature in AWS SSO to pass an `Attribute` element with the `Name` attribute set to `https://aws.amazon.com/SAML/Attributes/AccessControl:{TagKey}`. This element allows you to pass attributes as session tags in the SAML assertion. For more information about session tags, see [Passing session tags in AWS STS](#) in the *IAM User Guide*.

To pass attributes as session tags, include the `AttributeValue` element that specifies the value of the tag. For example, to pass the tag key-value pair `CostCenter = blue`, use the following attribute.

```
<saml:AttributeStatement>
<saml:Attribute Name="https://aws.amazon.com/SAML/Attributes/AccessControl:CostCenter">
<saml:AttributeValue>blue
</saml:AttributeValue>
</saml:Attribute>
</saml:AttributeStatement>
```

If you need to add multiple attributes, include a separate `Attribute` element for each tag.

PingOne

AWS SSO supports automatic provisioning (synchronization) of user information from the PingOne product by Ping Identity (hereafter “Ping”) into AWS SSO. This provisioning uses the System for Cross-domain Identity Management (SCIM) v2.0 protocol. You configure this connection in PingOne using your AWS SSO SCIM endpoint and access token. When you configure SCIM synchronization, you create a mapping of your user attributes in PingOne to the named attributes in AWS SSO. This causes the expected attributes to match between AWS SSO and PingOne.

This guide is based on PingOne as of October 2020. Steps for newer versions may vary. Contact Ping for more information about how to configure provisioning to AWS SSO for other versions of PingOne. This guide also contains a few notes regarding configuration of user authentication through SAML.

The following steps walk you through how to enable automatic provisioning of users from PingOne to AWS SSO using the SCIM protocol.

Note

Before you begin deploying SCIM, we recommend that you first review the [Considerations for using automatic provisioning \(p. 28\)](#). Then continue reviewing additional considerations in the next section.

Topics

- [Prerequisites \(p. 47\)](#)
- [Additional considerations \(p. 47\)](#)
- [Step 1: Enable provisioning in AWS SSO \(p. 47\)](#)

- [Step 2: Configure provisioning in PingOne \(p. 48\)](#)
- (Optional) [Step 3: Configure user attributes in PingOne for access control in AWS SSO \(p. 48\)](#)
- (Optional) [Passing attributes for access control \(p. 49\)](#)

Prerequisites

You will need the following before you can get started:

- A PingOne subscription or free trial, with both federated authentication and provisioning capabilities. For more information about how to obtain a free trial, see the [Ping Identity](#) website.
- An AWS SSO-enabled account ([free](#)). For more information, see [Enable AWS SSO](#).
- The PingOne AWS Single Sign-On application added to your PingOne admin portal. You can obtain the PingOne AWS Single Sign-On application from the PingOne Application Catalog. For general information, see [Add an application from the Application Catalog](#) on the Ping Identity website.
- A SAML connection from your PingOne instance to AWS SSO. After the PingOne AWS Single Sign-On application has been added to your PingOne admin portal, you must use it to configure a SAML connection from your PingOne instance to AWS SSO. Use the “download” and “import” metadata feature on both ends to exchange SAML metadata between PingOne and AWS SSO. For instructions on how to configure this connection, see the PingOne documentation.

Additional considerations

The following are important considerations about PingOne that can affect how you implement provisioning with AWS SSO.

- As of October 2020, PingOne does not support provisioning of groups through SCIM. Please contact Ping for the latest information on group support in SCIM for PingOne.
- Users may continue to be provisioned from PingOne after disabling provisioning in the PingOne admin portal. If you need to terminate provisioning immediately, delete the relevant SCIM bearer token, and/or disable [Automatic provisioning \(p. 27\)](#) in AWS SSO.
- If an attribute for a user is removed from the data store configured in PingOne, that attribute will not be removed from the corresponding user in AWS SSO. This is a known limitation in PingOne’s provisioner implementation. If an attribute is modified, the change will be synchronized to AWS SSO.
- The following are important notes regarding your SAML configuration in PingOne:
 - AWS SSO supports only `emailaddress` as a `NameID` format. This means you need to choose a user attribute that is unique within your directory in PingOne, non-null, and formatted as an email/UPN (for example, `user@domain.com`) for your **SAML_SUBJECT** mapping in PingOne. **Email (Work)** is a reasonable value to use for test configurations with the PingOne built-in directory.
 - Users in PingOne with an email address containing a `+` character may be unable to sign in to AWS SSO, with errors such as `'SAML_215'` or `'Invalid input'`. To fix this, in PingOne, choose the **Advanced** option for the **SAML_SUBJECT** mapping in **Attribute Mappings**. Then set **Name ID Format to send to SP** to `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress` in the drop-down menu.

Step 1: Enable provisioning in AWS SSO

In this first step, you use the AWS SSO console to enable automatic provisioning.

To enable automatic provisioning in AWS SSO

1. After you have completed the prerequisites, open the [AWS SSO console](#).
2. Choose **Settings** in the left navigation pane.

3. On the **Settings** page, under **Identity source**, next to **Provisioning**, choose **Enable automatic provisioning**. This immediately enables automatic provisioning in AWS SSO and displays the necessary endpoint and access token information.
4. In the **Inbound automatic provisioning** dialog box, copy each of the values for the following options. You will need to paste these in later when you configure provisioning in your IdP.
 - a. **SCIM endpoint**
 - b. **Access token**
5. Choose **Close**.

Now that you have set up provisioning in the AWS SSO console, you need to complete the remaining tasks using the PingOne AWS Single Sign-On application. These steps are described in the following procedure.

Step 2: Configure provisioning in PingOne

Use the following procedure in the PingOne AWS Single Sign-On application to enable provisioning with AWS SSO. This procedure assumes that you have already added the PingOne AWS Single Sign-On application to your PingOne admin portal. If you have not yet done so, refer to [Prerequisites \(p. 47\)](#), and then complete this procedure to configure SCIM provisioning.

To configure provisioning in PingOne

1. Open the PingOne AWS Single Sign-On application that you installed as part of configuring SAML for PingOne (**Applications > My Applications**). See [Prerequisites \(p. 47\)](#).
2. Scroll to the bottom of the page. Under **User Provisioning**, choose the **complete** link to navigate to the user provisioning configuration of your connection.
3. On the **Provisioning Instructions** page, choose **Continue to Next Step**.
4. In the previous procedure, you copied the **SCIM endpoint** value in AWS SSO. Paste that value into the **SCIM URL** field in the PingOne AWS Single Sign-On application. Make sure that you remove the trailing forward slash at the end of the URL. Also, in the previous procedure you copied the **Access token** value in AWS SSO. Paste that value into the **ACCESS_TOKEN** field in the PingOne AWS Single Sign-On application.
5. For **REMOVE_ACTION**, choose either **Disabled** or **Deleted** (see the description text on the page for more details).
6. On the **Attribute Mapping** page, choose a value to use for the **SAML_SUBJECT** (NameId) assertion, following guidance from [Additional considerations \(p. 47\)](#) earlier on this page. Then choose **Continue to Next Step**.
7. On the **PingOne App Customization - AWS Single Sign-On** page, make any desired customization changes (optional), and click **Continue to Next Step**.
8. On the **Group Access** page, choose the groups containing the users you would like to enable for provisioning and single sign-on to AWS SSO. Choose **Continue to Next Step**.
9. Scroll to the bottom of the page, and choose **Finish** to start provisioning.
10. To verify that users have been successfully synchronized to AWS SSO, return to the AWS SSO console and choose **Users**. Synchronized users from PingOne will appear on the **Users** page. These users can now be assigned to accounts and applications within AWS SSO.

Remember that PingOne does not support provisioning of groups or group memberships through SCIM. Contact Ping for more information.

(Optional) Step 3: Configure user attributes in PingOne for access control in AWS SSO

This is an optional procedure for PingOne should you choose to configure attributes for AWS SSO to manage access to your AWS resources. The attributes that you define in PingOne will be passed in a

SAML assertion to AWS SSO. You then create a permission set in AWS SSO to manage access based on the attributes you passed from PingOne.

Before you begin this procedure, you must first enable the [Attributes for access control \(p. 59\)](#) feature. For more information about how to do this, see [Enable and configure attributes for access control \(p. 60\)](#).

To configure user attributes in PingOne for access control in AWS SSO

1. Open the PingOne AWS Single Sign-On application that you installed as part of configuring SAML for PingOne (**Applications > My Applications**).
2. Choose **Edit**, and then choose **Continue to Next Step** until you get to the **Attribute Mappings** page.
3. On the **Attribute Mappings** page, choose **Add new attribute**, and then do the following. You must perform these steps for each attribute you will add for use in AWS SSO for access control.
 - a. In the **Application Attribute** field, enter `https://aws.amazon.com/SAML/Attributes/AccessControl:AttributeName`. Replace `AttributeName` with the name of the attribute you are expecting in AWS SSO. For example, `https://aws.amazon.com/SAML/Attributes/AccessControl:Email`.
 - b. In the **Identity Bridge Attribute or Literal Value** field, choose user attributes from your PingOne directory. For example, **Email (Work)**.
4. Choose **Next** a few times, and then choose **Finish**.

(Optional) Passing attributes for access control

You can optionally use the [Attributes for access control \(p. 59\)](#) feature in AWS SSO to pass an `Attribute` element with the `Name` attribute set to `https://aws.amazon.com/SAML/Attributes/AccessControl:{TagKey}`. This element allows you to pass attributes as session tags in the SAML assertion. For more information about session tags, see [Passing session tags in AWS STS](#) in the *IAM User Guide*.

To pass attributes as session tags, include the `AttributeValue` element that specifies the value of the tag. For example, to pass the tag key-value pair `CostCenter = blue`, use the following attribute.

```
<saml:AttributeStatement>
<saml:Attribute Name="https://aws.amazon.com/SAML/Attributes/AccessControl:CostCenter">
<saml:AttributeValue>blue
</saml:AttributeValue>
</saml:Attribute>
</saml:AttributeStatement>
```

If you need to add multiple attributes, include a separate `Attribute` element for each tag.

Other identity providers

AWS SSO implements the following standards-based protocols for identity federation:

- SAML 2.0 for user authentication
- SCIM for provisioning

Any identity provider (IdP) that implements these standard protocols is expected to interoperate successfully with AWS SSO, with the following special considerations:

- **SAML**

- AWS SSO requires a SAML nameID format of email address (that is, `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress`)
 - The value of the nameID field in assertions must be an RFC 2822 (<https://tools.ietf.org/html/rfc2822>) addr-spec compliant ("name@domain.com") string (<https://tools.ietf.org/html/rfc2822#section-3.4.1>)
- **SCIM**
- The AWS SSO SCIM implementation is based on SCIM RFCs 7642 (<https://tools.ietf.org/html/rfc7642>), 7643 (<https://tools.ietf.org/html/rfc7643>), and 7644 (<https://tools.ietf.org/html/rfc7644>), and the interoperability requirements laid out in the March 2020 draft of the FastFed Basic SCIM Profile 1.0 (https://openid.net/specs/fastfed-scim-1_0-02.html#rfc.section.4). Any differences between these documents and the current implementation in AWS SSO are described in the [Supported API operations](#) section of the *AWS SSO SCIM Implementation Developer Guide*.

IdPs that do not conform to the standards and considerations mentioned above are not supported. Please contact your IdP for questions or clarifications regarding the conformance of their products to these standards and considerations.

If you have any issues connecting your IdP to AWS SSO, we recommend that you check:

- AWS CloudTrail logs by filtering on the event name **ExternalIdPDirectoryLogin**
- IdP-specific logs and/or debug logs
- [Troubleshooting AWS SSO issues \(p. 133\)](#)

Note

Some IdPs, including the list of [Supported identity providers \(p. 32\)](#), offer a simplified configuration experience for AWS SSO in the form of an "application" or "connector" built specifically for AWS Single Sign-On. If your IdP provides this option, we recommend that you use it, being careful to choose the item that's built specifically for AWS Single Sign-On. Other items called "AWS", "AWS federation", or similar generic "AWS" names may use other federation approaches and/or endpoints, and may not work as expected with AWS SSO.

Manage SSO to your AWS accounts

AWS Single Sign-On is integrated with AWS Organizations so that administrators can pick multiple AWS accounts whose users need single sign-on (SSO) access to the AWS Management Console. These AWS accounts can be either the management account of the AWS Organizations or a member account. A management account is the AWS account that is used to create the organization. The rest of the accounts that belong to an organization are called member accounts. For more information about the different account types, see [AWS Organizations Terminology and Concepts](#) in the *AWS Organizations User Guide*.

Once you assign access from the AWS SSO console, you can use permission sets to further refine what users can do in the AWS Management Console. For more information about permission sets, see [Permission sets \(p. 53\)](#).

Users follow a simple sign-in process:

1. Users use their directory credentials to sign in to the user portal.
2. Users then choose the AWS account name that will give them federated access to the AWS Management Console for that account.
3. Users who are assigned multiple permission sets choose which IAM role to use.

Permission sets are a way to define permissions centrally in AWS SSO so that they can be applied to all of your AWS accounts. These permission sets are provisioned to each AWS account as an IAM role. The user portal gives users the ability to retrieve temporary credentials for the IAM role of a given AWS account so they can use it for short-term access to the AWS CLI. For more information, see [How to get credentials of an IAM role for use with CLI access to an AWS account \(p. 80\)](#).

To use AWS SSO with AWS Organizations, you must first [Enable AWS SSO \(p. 4\)](#), which grants AWS SSO the capability to create [Service-linked roles \(p. 64\)](#) in each account in your AWS organization. These roles are not created until after you [Assign user access \(p. 52\)](#) for a given account.

You can also connect an AWS account that is not part of your organization by setting up the account as a custom SAML application in AWS SSO. In this scenario, you provision and manage the IAM roles and trust relationships that are required to enable SSO access. For more information on how to do this, see [Add and configure a custom SAML 2.0 application \(p. 71\)](#).

Topics

- [Single sign-on access \(p. 51\)](#)
- [Permission sets \(p. 53\)](#)
- [Attribute-based access control \(p. 57\)](#)
- [IAM identity provider \(p. 64\)](#)
- [Service-linked roles \(p. 64\)](#)

Single sign-on access

You can assign users in your connected directory permissions to the management account or member accounts in your AWS Organizations organization based on [common job functions](#). Or you can use custom permissions to meet your specific security requirements. For example, you can grant database administrators broad permissions to Amazon RDS in development accounts but limit their permissions

in production accounts. AWS SSO configures all the necessary user permissions in your AWS accounts automatically.

Note

You might need to grant users or groups permissions to operate in the AWS Organizations management account. Because it is a highly privileged account, additional security restrictions require you to have the [IAMFullAccess](#) policy or equivalent permissions before you can set this up. These additional security restrictions are not required for any of the member accounts in your AWS organization.

Assign user access

Use the following procedure to assign SSO access to users and groups in your connected directory and use permission sets to determine their level of access.

Note

To simplify administration of access permissions, we recommended that you assign access directly to groups rather than to individual users. With groups you can grant or deny permissions to groups of users rather than having to apply those permissions to each individual. If a user moves to a different organization, you simply move that user to a different group and they automatically receive the permissions that are needed for the new organization.

To assign access to users or groups

1. Open the [AWS SSO console](#).

Note

Make sure that the AWS SSO console is using the Region where your AWS Managed Microsoft AD directory is located before you move to the next step.

2. Choose **AWS accounts**.
3. Under the **AWS organization** tab, in the list of AWS accounts, choose one or more accounts to which you want to assign access.

Note

The AWS SSO console supports selecting up to 10 AWS accounts at a time per permission set when assigning user access. If you need to assign more than 10 AWS accounts to the same set of users, repeat this procedure for the additional accounts, selecting the same users and permission set when prompted.

4. Choose **Assign users**.
5. On the **Select users or groups** page, type a user or group name to filter the results. You can specify multiple users or groups by selecting the applicable accounts as they appear in search results, and then choose **Next: Permission sets**.
6. On the **Select permission sets** page, select the permission sets that you want to apply to the users or groups from the table. Then choose **Finish**. You can optionally choose to **Create a new permission set** if none of the permissions in the table meets your needs. For detailed instructions, see [Create a permission set \(p. 53\)](#).
7. Choose **Finish** to begin the process of configuring your AWS account.

Important

The user assignment process may take a few minutes to complete. It is important that you leave this page open until the process successfully completes.

Note

You might need to grant users or groups permissions to operate in the AWS Organizations management account. Because it is a highly privileged account, additional security restrictions require you to have the [IAMFullAccess](#) policy or equivalent permissions before you can set this up. These additional security restrictions are not required for any of the member accounts in your AWS organization.

Remove user access

Use this procedure when you need to remove SSO access to an AWS account for a particular user or group in your connected directory.

To remove user access from an AWS account

1. Open the [AWS SSO console](#).
2. Choose **AWS accounts**.
3. In the table, select the AWS account with the user or group whose access you want to remove.
4. On the **Details** page for the AWS account, under **Assigned users and groups**, locate the user or group in the table. Then choose **Remove access**.
5. In the **Remove access** dialog box, confirm the user or group name. Then choose **Remove access**.

Delegate who can assign SSO access to users in the management account

Assigning single sign-on access to the management account using the AWS SSO console is a privileged action. By default, only an AWS account root user, or a user who has the **AWSSSOMasterAccountAdministrator** and **IAMFullAccess** AWS managed policies attached, can assign SSO access to the management account. The **AWSSSOMasterAccountAdministrator** and **IAMFullAccess** policies manage SSO access to the management account within an AWS Organizations organization.

Use the following steps to delegate permissions to manage SSO access to users in your directory.

To grant permissions to manage SSO access to users in your directory

1. Sign in to the AWS SSO console as a root user of the management account or with another IAM user who has IAM administrator permissions to the management account.
2. Use the procedure [Create a permission set \(p. 53\)](#) to create a permission set. When you get to the **Create new permission set** page, select the **Create a custom permission set** option, choose **Next: Details**, and then select the option **Attach AWS managed policies**. In the list of IAM policies that appear in the table, choose both the **AWSSSOMasterAccountAdministrator** and **IAMFullAccess** AWS managed policies. These policies grant permissions to any user who will be assigned access to this permission set in the future.
3. Use the procedure [Assign user access \(p. 52\)](#) to assign the appropriate users to the permission set that you just created.
4. Communicate the following to the assigned users: When they sign in to the user portal and select the **AWS Account** icon, they must choose the appropriate IAM role name to be authenticated with the permissions that you just delegated.

Permission sets

Permission sets define the level of access that users and groups have to an AWS account. Permission sets are stored in AWS SSO and provisioned to the AWS account as IAM roles. You can assign more than one permission set to a user. Users who have multiple permission sets must choose one when they sign in to the user portal. (Users will see these as IAM roles). For more information, see [Permission sets \(p. 9\)](#).

Create a permission set

Use this procedure to create a permission set based on a custom permissions policy that you create, or on predefined AWS managed policies that exist in IAM, or both.

To create a permission set

1. Open the [AWS SSO console](#).
2. Choose **AWS accounts**.
3. Select the **Permission sets** tab.
4. Choose **Create permission set**.
5. On the **Create new permission set** page, choose from one of the following options, and then follow the instructions provided under that option:
 - **Use an existing job function policy**
 1. Choose **Next: Details**.
 2. Under **Select job function policy**, select one of the default IAM job function policies in the list, and then choose **Next: Tags**. For more information, see [AWS managed policies for job functions](#).
 3. Under **Add tags (optional)**, specify values for **Key** and **Value (optional)**, and then choose **Next: Review**. For more information about tags, see [Tagging AWS Single Sign-On resources \(p. 125\)](#).
 4. Review the selections you made, and then choose **Create**.
 - **Create a custom permission set**
 1. Choose **Next: Details**.
 2. Under **Create a custom permission set**, type a name that will identify this permission set in AWS SSO. This name will also appear as an IAM role in the user portal for any users who have access to it.
 3. (Optional) You can also type a description. This description will only appear in the AWS SSO console and will not be visible to users in the user portal.
 4. (Optional) Specify the value for **Session duration**. This value is used to determine the length of time a user can be logged on before the console logs them out of their session. For more information, see [Set session duration \(p. 55\)](#).
 5. (Optional) Specify the value for **Relay state**. This value is used in the federation process to redirect users within the account. For more information, see [Set relay state \(p. 55\)](#).
 6. Select either **Attach AWS managed policies** or **Create a custom permissions policy**. Or select both if you need to link more than one policy type to this permission set.
 7. If you chose **Attach AWS managed policies**, select up to 10 job-related or service-specific AWS managed policies from the list.
 8. If you chose **Create a custom permissions policy**, under **Create a custom permissions policy**, paste in a policy document with your preferred permissions. For a list of example policies to use for delegating AWS SSO tasks, see [Customer managed policy examples \(p. 96\)](#).

For more information about the access policy language, see [Overview of policies](#) in the *IAM User Guide*. To test the effects of this policy before applying your changes, use the [IAM policy simulator](#).
 9. Choose **Next: Tags**.
 10. Under **Add tags (optional)**, specify values for **Key** and **Value (optional)**, and then choose **Next: Review**. For more information about tags, see [Tagging AWS Single Sign-On resources \(p. 125\)](#).
 11. Review the selections you made, and then choose **Create**.

Configure permission set properties

In AWS SSO you can customize the user experience by configuring the following permission set properties.

Topics

- [Set session duration \(p. 55\)](#)
- [Set relay state \(p. 55\)](#)

Set session duration

For each [permission set](#), you can specify a session duration to control the length of time that a user can be signed in to an AWS account. When the specified duration has elapsed, AWS signs the user out of the session.

When you create a new permission set, the session duration is set to 1 hour (in seconds) by default. The minimum session duration is 1 hour, and can be set to a maximum of 12 hours. AWS SSO automatically creates IAM roles in each assigned account for each permission set, and configures these roles with a maximum session duration of 12 hours.

When end-users federate into their AWS Account's console or when using the AWS Command Line Interface (CLI), AWS SSO uses the session duration setting on the permission set to control the duration of the session. By default, AWS SSO-generated IAM roles for permission sets can only be assumed by SSO users, which ensures that the session duration specified in the SSO permission set is enforced.

Important

As a security best practice, we recommend that you do not set the session duration length longer than is needed to perform the role.

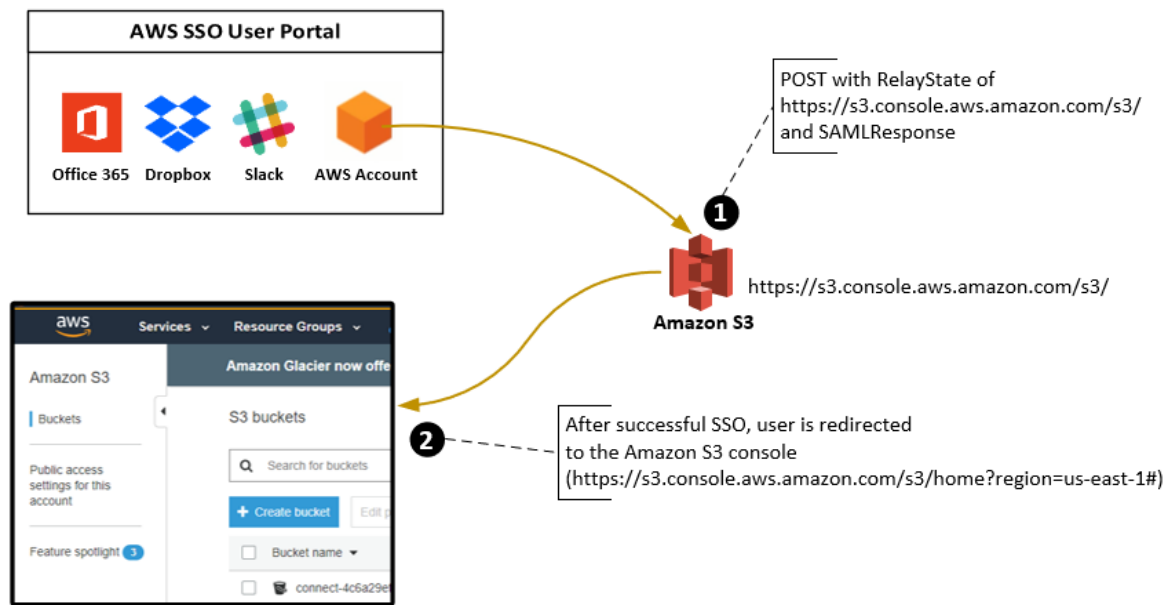
Once a permission set has been created, you can later update it to apply a new session duration. Use the following procedure to modify the session duration length for a given permission set.

To set the session duration

1. Open the [AWS SSO console](#).
2. Choose **AWS accounts**.
3. Choose the **Permission sets** tab.
4. Choose the name of the permission set where you want to change the new session duration time.
5. On the **Permissions** tab, under the **General** section, choose **Edit**.
6. Next to **Session duration**, choose a new session length value, and then choose **Continue**.
7. Select the AWS accounts in the list that you want the new session duration value to apply to, and then choose **Reprovision**.

Set relay state

During the federation authentication process, the relay state redirects users within the AWS Management Console. You can specify a relay state URL to redirect links to any service in the AWS Management Console. For example, the below illustration shows the process for redirecting to the S3 console (<https://s3.console.aws.amazon.com/s3/home?region=us-east-1#>).



Use the following procedure to modify the relay state URL for a given permission set.

To set the relay state

1. Open the [AWS SSO console](#).
2. Choose **AWS accounts**.
3. Choose the **Permission sets** tab.
4. Choose the name of the permission set where you want to change the new relay state URL.
5. On the **Permissions** tab, under the **General** section, choose **Edit**.
6. Next to **Relay state**, type a URL value for any of the AWS services, and then choose **Continue**.
7. Select the AWS accounts in the list that you want the new relay state value to apply to, and then choose **Reprovision**.

Delete permission sets

Use this procedure to delete one or more permission sets so that they can no longer be used by any AWS account in the organization.

Note

All users and groups that have been assigned this permission set, regardless of what AWS account is using it, will no longer be able to sign in.

To delete a permissions set from an AWS account

1. Open the [AWS SSO console](#).
2. Choose **AWS accounts**.
3. Choose the **Permission sets** tab.
4. Select the permission set you want to delete, and then choose **Delete**.
5. In the **Delete permission set** dialog box, choose **Delete**.

Attribute-based access control

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. You can use AWS SSO to manage access to your AWS resources across multiple AWS accounts using user attributes that come from any AWS SSO identity source. In AWS, these attributes are called tags. Using user attributes as tags in AWS helps you simplify the process of creating fine-grained permissions in AWS and ensures that your workforce gets access only to the AWS resources with matching tags.

For example, you can assign developers Bob and Sally, who are from two different teams, to the same permission set in AWS SSO and then select the team name attribute for access control. When Bob and Sally sign in to their AWS accounts, AWS SSO sends their team name attribute in the AWS session so Bob and Sally can access AWS project resources only if their team name attribute matches the team name tag on the project resource. If Bob moves to Sally's team in the future, you can modify his access by simply updating his team name attribute in the corporate directory. When Bob signs in next time, he will automatically get access to the project resources of his new team without requiring any permissions updates in AWS.

This approach also helps in reducing the number of distinct permissions you need to create and manage in AWS SSO as users associated with the same permission sets can now have unique permissions based on their attributes. You can use these user attributes in AWS SSO permission sets and resource-based policies to implement ABAC to AWS resources and simplify permissions management at scale.

Benefits

The following are additional benefits of using ABAC in AWS SSO.

- **ABAC requires fewer permission sets** – Because you don't have to create different policies for different job functions, you create fewer permission sets. This reduces your permissions management complexity.
- **Using ABAC, teams can change and grow quickly** – Permissions for new resources are automatically granted based on attributes when resources are appropriately tagged upon creation.
- **Use employee attributes from your corporate directory with ABAC** – You can use existing employee attributes from any identity source configured in AWS SSO to make access control decisions in AWS.
- **Track who is accessing resources** – Security administrators can easily determine the identity of a session by looking at the user attributes in AWS CloudTrail to track user activity in AWS.

For information about how to configure ABAC using the AWS SSO console, see [Attributes for access control \(p. 59\)](#). For information about how to enable and configure ABAC using the AWS SSO APIs, see [CreateInstanceAccessControlAttributeConfiguration](#) in the AWS SSO API Reference Guide.

Topics

- [Checklist: Configuring ABAC in AWS using AWS SSO \(p. 57\)](#)
- [Attributes for access control \(p. 59\)](#)

Checklist: Configuring ABAC in AWS using AWS SSO

This checklist includes the configuration tasks that are necessary to prepare your AWS resources and to set up AWS SSO for ABAC access. Complete the tasks in this checklist in order. When a reference link takes you to a topic, return back to this topic so that you can proceed with the remaining tasks in this checklist.

| Step | Task | Reference |
|------|--|---|
| 1 | Review how to add tags to all your AWS resources. To implement ABAC in AWS SSO, you'll first need to add tags to all your AWS resources that you want to implement ABAC for. | <ul style="list-style-type: none"> • Tagging AWS resources |
| 2 | Review how to configure your identity source in AWS SSO with the associated user identities and attributes in SSO. AWS SSO lets you use user attributes from any supported AWS SSO identity source for ABAC in AWS. | <ul style="list-style-type: none"> • Manage your identity source (p. 13) |
| 3 | Based on the following criteria, determine which attributes you want to use for making access control decisions in AWS and send them to AWS SSO. | <ul style="list-style-type: none"> • Getting started (p. 59) |
| | <ul style="list-style-type: none"> • If you are using an external IdP, decide whether you want to use attributes passed from the IdP or select attributes from within AWS SSO. | <ul style="list-style-type: none"> • Choosing attributes when using an external identity provider as your identity source (p. 59) |
| | <ul style="list-style-type: none"> • If you choose to have your IdP send attributes, configure your IdP to transmit the attributes in SAML assertions. See the <code>Optional</code> sections in the procedure for your specific IdP. | <ul style="list-style-type: none"> • Supported identity providers (p. 32) |
| | <ul style="list-style-type: none"> • If you use an IdP as your identity source and choose to select attributes in AWS SSO, investigate how to configure SCIM so that the attribute values come from your IdP. If you cannot use SCIM with your IdP, add the users and their attributes using the AWS SSO console's User page. | <ul style="list-style-type: none"> • Automatic provisioning (p. 27) • Supported external identity provider attributes (p. 23) |
| | <ul style="list-style-type: none"> • If you use Active Directory or AWS SSO as your identity source, or you use an IdP and choose to select attributes in AWS SSO, review the available attributes that you can configure. Then jump immediately to step 4 to start configuring your ABAC attributes using the AWS SSO console. | <ul style="list-style-type: none"> • Choosing attributes when using AWS SSO as your identity source (p. 59) • Choosing attributes when using AWS Managed Microsoft AD as your identity source (p. 59) • Default mappings (p. 24) |
| 4 | Select the attributes to use for ABAC using the Attributes for access control page in the AWS SSO console. From this page you can select attributes for access control from the identity source that you configured in step 2. After your identities and their attributes are in AWS SSO, you must create key-value pairs (mappings) which will be passed to your AWS accounts for use in access control decisions. | <ul style="list-style-type: none"> • Enable and configure attributes for access control (p. 60) |
| 5 | Create custom permissions policies within your permission set and use access control attributes to create ABAC rules so that users can only access resources with matching tags. User attributes that you configured in step 4 are used as tags in AWS for access control decisions. You can refer to the access control attributes in the permissions policy using the <code>aws:PrincipalTag/key</code> condition. | <ul style="list-style-type: none"> • Create permission policies for ABAC in AWS SSO (p. 63) |

| Step | Task | Reference |
|------|---|--|
| 6 | In your various AWS accounts, assign users to permissions sets you created in step 5. Doing so ensures that when they federate into their accounts and access AWS resources, they only get access based on matching tags. | <ul style="list-style-type: none">• Assign user access (p. 52) |

Once you have completed this checklist, users that federate into an AWS account using SSO will get access to their AWS resources based on matching attributes.

Attributes for access control

Attributes for access control is the name of the page in the AWS SSO console where you select user attributes that you want to use in policies to control access to resources. Administrators can assign users to workloads in AWS based on existing attributes in the users' identity source.

For example, suppose you want to assign access to S3 buckets based on department names. While on the **Attributes for access control** page, you select the **Department** user attribute for use with attribute-based access control (ABAC). In the AWS SSO permission set, you then write a policy that grants users access only when the **Department** attribute matches the department tag that you assigned to your S3 buckets. AWS SSO passes the user's department attribute to the account being accessed. The attribute is then used to determine access based on the policy. For more information about ABAC, see [Attribute-based access control \(p. 57\)](#).

Getting started

How you get started configuring attributes for access control depends on which identity source you are using. Regardless of the identity source you choose, after you have selected your attributes you need to create or edit permission set policies. These policies must grant user identities access to AWS resources.

Choosing attributes when using AWS SSO as your identity source

When you configure AWS SSO as the identity source, you first add users and configure their attributes. Next, navigate to the **Attributes for access control** page and select the attributes you want to use in policies. Finally, navigate to the **AWS accounts** page to create or edit permission sets to use the attributes for ABAC.

Choosing attributes when using AWS Managed Microsoft AD as your identity source

When you configure AWS SSO with AWS Managed Microsoft AD as your identity source, you first map a set of attributes from Active Directory to user attributes in AWS SSO. Next, navigate to the **Attributes for access control** page. Then choose which attributes to use in your ABAC configuration based on the existing set of SSO attributes mapped from Active Directory. Finally, author ABAC rules using the access control attributes in permission sets to grant user identities access to AWS resources. For a list of the default mappings for user attributes in AWS SSO to the user attributes in your AWS Managed Microsoft AD directory, see [Default mappings \(p. 24\)](#).

Choosing attributes when using an external identity provider as your identity source

When you configure AWS SSO with an external identity provider (IdP) as your identity source, there are two ways to use attributes for ABAC.

- You can configure your IdP to send the attributes through SAML assertions. In this case, AWS SSO passes the attribute name and value from the IdP through for policy evaluation.

Note

Attributes in SAML assertions will not be visible to you on the **Attributes for access control** page. You will have to know these attributes in advance and add them to access control rules when you author policies. If you decide to trust your external IdPs for attributes, then these attributes will always be passed when users federate into AWS accounts. In scenarios where the same attributes are coming to SSO through SAML and SCIM, the SAML attributes value take precedence in access control decisions.

- You can configure which attributes you use from the **Attributes for access control** page in the AWS SSO console. The attributes values that you choose here replace the values for any matching attributes that come from an IdP through an assertion. Depending on whether you are using SCIM, consider the following:
 - If using SCIM, the IdP automatically synchronizes the attribute values into AWS SSO. Additional attributes that are required for access control might not be present in the list of SCIM attributes. In that case, consider collaborating with the IT admin in your IdP to send such attributes to SSO via SAML assertions using the required `https://aws.amazon.com/SAML/Attributes/AccessControl:` prefix. For information about how to configure user attributes for access control in your IdP to send through SAML assertions, see [Supported identity providers \(p. 32\)](#).
 - If you are not using SCIM, you must manually add the users and set their attributes just as if you were using AWS SSO as an identity source. Next, navigate to the **Attributes for access control** page and choose the attributes you want to use in policies.

For a complete list of supported attributes for user attributes in AWS SSO to the user attributes in your external IdPs, see [Supported external identity provider attributes \(p. 23\)](#).

To get started with ABAC in AWS SSO, see the following topics.

Topics

- [Enable and configure attributes for access control \(p. 60\)](#)
- [Create permission policies for ABAC in AWS SSO \(p. 63\)](#)

Enable and configure attributes for access control

To use ABAC in all cases, you must first enable ABAC using the AWS SSO console or the AWS SSO API. If you choose to use AWS SSO to select attributes, you use the **Attributes for access control** page in the AWS SSO console or the AWS SSO API. If you use an external identity provider (IdP) as an identity source and choose to send attributes through the SAML assertions, you configure your IdP to pass the attributes. If a SAML assertion passes any of these attributes, AWS SSO will replace the attribute value with the value from the AWS SSO identity store. Only attributes configured in AWS SSO will be sent over for making access control decisions when users federate into their accounts.

Note

You cannot view attributes configured and sent by an external IdP from the **Attributes for access control** page in the AWS SSO console. If you are passing access control attributes in the SAML assertions from your external IdP, then those attributes are directly sent to the AWS account when users federate in. The attributes won't be available in AWS SSO for mapping.

Enable attributes for access control

Use the following procedure to enable the attributes for access (ABAC) control feature using the AWS SSO console.

Note

If you have existing permission sets and you plan to enable ABAC in your AWS SSO instance, additional security restrictions require you to first have the `iam:UpdateAssumeRolePolicy`

policy. These additional security restrictions are not required if you do not have any permission sets created in your account.

To enable Attributes for access control

1. Open the [AWS SSO console](#).
2. Choose **Settings**
3. On the **Settings** page, under **Identity source**, next to **Attributes for access control**, choose **Enable**.

Select your attributes

Use the following procedure to set up attributes for your ABAC configuration.

To select your attributes using the AWS SSO console

1. Open the [AWS SSO console](#).
2. Choose **Settings**
3. On the **Settings** page, under **Identity source**, next to **Attributes for access control**, choose **View details**.
4. On the **Attributes for access control** page, notice the **Key** and **Value** columns as shown in the screenshot below. This is where you will be mapping the attribute coming from your identity source to an attribute that AWS SSO passes as a session tag.

| Key ⓘ | Value (optional) ⓘ | Remove |
|-----------------------------|---|--------|
| Department | <code>\${path:enterprise.department}</code> | × |
| CostCenter | <code>\${path:enterprise.costCenter}</code> | × |
| Add new key | Add new value | |

Key represents the name you are giving to the attribute for use in policies. This can be any arbitrary name but you need to specify that exact name in the policies you author for access control. For example, let's say that you are using Okta (an external IdP) as your identity source and need to pass your organization's cost center data along as session tags. In **Key**, you would enter a similarly matched name like **CostCenter** as your key name. It's important to note that whichever name you choose here, it must also be named exactly the same in your [aws:PrincipalTag condition key](#) (p. 63) (i.e., `"ec2:ResourceTag/CostCenter": "${aws:PrincipalTag/CostCenter}"`).

Value represents the content of the attribute coming from your configured identity source. Here you can enter any value from the appropriate identity source table listed in [Attribute mappings](#) (p. 22). For example, using the context provided in the above mentioned example, you would review the list of

The following table lists all external identity provider (IdP) attributes that are supported and that can be mapped to attributes you can use when configuring Attributes for access control (p. 59) in AWS SSO. When using SAML assertions, you can use whichever attributes your IdP supports.

| |
|---------------------------------------|
| <code>\${path:userName}</code> |
| <code>\${path:name.familyName}</code> |

`${path:name.givenName}`

`${path:displayName}`

`${path:nickName}`

`${path:emails[primary eq true].value}`

`${path:addresses[type eq "work"].streetAddress}`

`${path:addresses[type eq "work"].locality}`

`${path:addresses[type eq "work"].region}`

`${path:addresses[type eq "work"].postalCode}`

`${path:addresses[type eq "work"].country}`

`${path:addresses[type eq "work"].formatted}`

`${path:phoneNumbers[type eq "work"].value}`

`${path:userType}`

`${path:title}`

`${path:locale}`

`${path:timezone}`

`${path:enterprise.employeeNumber}`

`${path:enterprise.costCenter}`

`${path:enterprise.organization}`

`${path:enterprise.division}`

`${path:enterprise.department}`

`${path:enterprise.manager.value}`

Supported attributes in your IdP

(p. 23) and determine that the closest match of a supported attribute would be **`${path:enterprise.costCenter}`** and you would then enter it in the **Value** field. See the screenshot provided above for reference. Note, that you can't use external IdP attribute values outside of this list for ABAC unless you use the option of passing attributes through the SAML assertion.

5. Choose **Save changes**.

Now that you have configured mapping your access control attributes, you need to complete the ABAC configuration process. To do this, author your ABAC rules and add them to your permission sets and/or resource-based policies. This is required so that you can grant user identities access to AWS resources. For more information, see [Create permission policies for ABAC in AWS SSO \(p. 63\)](#).

Disable attributes for access control

Use the following procedure to disable the ABAC feature and delete all of the attribute mappings that have been configured.

To disable Attributes for access control

1. Open the [AWS SSO console](#).
2. Choose **Settings**
3. On the **Settings** page, under **Identity source**, next to **Attributes for access control**, choose **View details**.
4. On the **Attributes for access control**, choose **Disable**.

Important

This step deletes all attributes that have been configured. Once deleted, any attributes that are received from an identity source and any custom attributes you have previously configured will not be passed.

Create permission policies for ABAC in AWS SSO

You can create permissions policies that determine who can access your AWS resources based on the configured attribute value. When you enable ABAC and specify attributes, AWS SSO passes the attribute value of the authenticated user into IAM for use in policy evaluation.

[aws:PrincipalTag condition key](#)

You can use access control attributes in your permission sets using the `aws:PrincipalTag` condition key for creating access control rules. For example, in the following trust policy you can tag all the resources in your organization with their respective cost centers. You can also use a single permission set that grants developers access to their cost center resources. Now, whenever developers federate into the account using SSO and their cost center attribute, they only get access to the resources in their respective cost centers. As the team adds more developers and resources to their project, you only have to tag resources with the correct cost center. Then you pass cost center information in the AWS session when developers federate into AWS accounts. As a result, as the organization adds new resources and developers to the cost center, developers can manage resources aligned to their cost centers without needing any permission updates.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:StartInstances",
        "ec2:StopInstances"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "ec2:ResourceTag/CostCenter": "${aws:PrincipalTag/CostCenter}"
        }
      }
    }
  ]
}
```

For more information, see [aws:PrincipalTag](#) and [EC2: Start or stop instances based on matching principal and resource tags](#) in the IAM User Guide.

If policies contain invalid attributes in their conditions, then the policy condition will fail and access will be denied. For more information, see [Error 'An unexpected error has occurred' when a user tries to sign in using an external identity provider](#) (p. 136).

IAM identity provider

When you add SSO access to an AWS account, AWS SSO creates an IAM identity provider in each AWS account. An IAM identity provider helps keep your AWS account secure because you don't have to distribute or embed long-term security credentials, such as IAM access keys, in your application.

Repair the IAM identity provider

Use the following procedure to repair your identity provider in case it was deleted or modified.

To repair an identity provider for an AWS account

1. Open the [AWS SSO console](#).
2. Choose **AWS accounts**.
3. In the table, select the AWS account that is associated with the identity provider that you want to repair.
4. On the AWS account details page, under **IAM identity provider**, choose **Repair identity provider**.

Remove the IAM identity provider

Use the following procedure to remove the IAM identity provider from AWS SSO.

To remove the IAM identity provider from AWS SSO

1. Open the [AWS SSO management console](#)
2. Choose **AWS accounts**
3. In the table select the AWS account that is associated with the IAM identity provider that you want to remove.
4. On the **Details** page for the AWS account, under **IAM identity provider**, choose **Remove identity provider**.

Service-linked roles

[Service-linked roles](#) are predefined IAM permissions that allow AWS SSO to delegate and enforce which users have SSO access to specific AWS accounts in your organization in AWS Organizations. The service enables this functionality by provisioning a service-linked role in every AWS account within its organization. The service then allows other AWS services like AWS SSO to leverage those roles to perform service-related tasks. For more information, see [AWS Organizations and service-linked roles](#).

During the process to [Enable AWS SSO](#) (p. 4), AWS SSO creates a service-linked role in all accounts within the organization in AWS Organizations. AWS SSO also creates the same service-linked role in every account that is subsequently added to your organization. This role allows AWS SSO to access each account's resources on your behalf. For more information, see [Manage SSO to your AWS accounts](#) (p. 51).

Service-linked roles that are created in each AWS account are named `AWSServiceRoleForSSO`. For more information, see [Using service-linked roles for AWS SSO \(p. 103\)](#).

Manage SSO to your applications

With AWS Single Sign-On, you can easily control who can have single sign-on (SSO) access to your cloud applications. Users get one-click access to these applications after they use their directory credentials to sign in to their user portal.

AWS SSO securely communicates with these applications through a trusted relationship between AWS SSO and the application's service provider. This trust is created when you add the application from the AWS SSO console and configure it with the appropriate metadata for both AWS SSO and the service provider.

After the application has been successfully added to the AWS SSO console, you can manage which users or groups need permissions to the application. By default, when you add an application, no users are assigned to the application. In other words, newly added applications in the AWS SSO console are inaccessible until you assign users to them. AWS SSO supports the following applications types:

- AWS SSO-integrated applications
- Cloud applications
- Custom Security Assertion Markup Language (SAML 2.0) applications

You can also grant your employees access to the AWS Management Console for a given AWS account in your organization. For more information on how to do this, see [Manage SSO to your AWS accounts](#) (p. 51).

The following sections explain how to configure user access to your AWS applications and third-party software as a service (SaaS) applications. You can also configure any custom applications that support identity federation with SAML 2.0.

Topics

- [AWS SSO-integrated applications](#) (p. 66)
- [Cloud applications](#) (p. 68)
- [Custom SAML 2.0 applications](#) (p. 71)
- [Manage AWS SSO certificates](#) (p. 72)
- [Application properties](#) (p. 74)
- [Assign user access](#) (p. 76)
- [Remove user access](#) (p. 76)
- [Map attributes in your application to AWS SSO attributes](#) (p. 77)

AWS SSO-integrated applications

With [AWS SSO-integrated application enablement](#) (p. 7), AWS enterprise applications such as Amazon SageMaker or AWS IoT SiteWise, can exist in a child account in your organization but still use your AWS SSO identities. This provides your application end users with an easy sign-in experience and allows for delegation of administrator of your applications to operators in a child account.

Constraining AWS SSO-integrated application use in AWS accounts

If you want to constrain which of your AWS Organizations accounts that an integrated application can be used, you can do so using service control policies (SCPs). You can use SCPs to block access to the AWS

SSO user and group information and to prevent the application from being started except in designated accounts.

Add and configure an AWS SSO-integrated application

To use AWS SSO-integrated applications, you must first enable AWS SSO to allow them access. For more information, see [AWS SSO-integrated application enablement \(p. 7\)](#).

After they are enabled, integrated applications can access user and group information directly from AWS SSO. As a result, you won't have to manage access in both AWS SSO and then again inside the application. Instead, AWS SSO delegates application access to the application administrator. To add users to integrated applications, use the console of the application where you created the application.

Disable or enable an AWS SSO-integrated application

If you only want to stop or restart user authentications to your application, use the following procedure to either disable or enable your application.

To disable or enable your application

1. In the AWS SSO console, choose **Applications** in the left navigation pane.
2. Select the application in the list.
3. Choose **Actions**, and then choose either **Disable** or **Enable**.

Remove an AWS SSO-integrated application

To remove an integrated application, visit the AWS Management Console where you manage your application. By removing the app this way, the application can gracefully remove resources that you might otherwise pay for. For emergency purposes only, you can force-remove an application from the AWS SSO console. AWS strongly recommends that you avoid this as such an action is irreversible and you might not be able to recover data from the application.

Before you force-remove, consider the following options:

- You can stop user authentications to this application without removing it by using the **Disable** option instead. For more information, see [Disable or enable an AWS SSO-integrated application \(p. 67\)](#).
- If you want to disconnect the application from AWS SSO permanently, use the AWS Management Console where you created the application and remove it there instead. This helps to avoid unnecessary application-related charges that may otherwise appear if you continue with force-remove. This process also removes the application from AWS SSO.

Warning

Force-remove should only be used as a last resort. This operation deletes all user permissions to this application, disconnects the application from AWS SSO, and renders the application inaccessible.

To force-remove an AWS application

1. In the AWS SSO console, choose **Applications** in the left navigation pane.
2. Choose the application you want to remove in the list.
3. On the application **Details** page, under **Remove Application**, choose **force-remove**.

4. On the **Force-remove application** page, review the warning message. If you agree, enter **remove**, and then choose **Force-remove**.

Cloud applications

You can use the AWS SSO application configuration wizard to include built-in SAML integrations to many popular cloud applications. Examples include Salesforce, Box, and Office 365. For a complete list of applications that you can add from the wizard, see [Supported applications \(p. 68\)](#).

Most cloud applications come with detailed instructions on how to set up the trust between AWS SSO and the application's service provider. You can find these instructions on the cloud applications configuration page during the setup process and after the application has been set up. After the application has been configured, you can assign access to the groups or users that require it.

Supported applications

AWS SSO has built-in support for the following commonly used cloud applications.

Note

AWS Support engineers can assist customers who have Business and Enterprise support plans with some integration tasks that involve third-party software. For a current list of supported platforms and applications, see [Third-Party Software Support](#) on the *AWS Support Features* page.

| | | | | |
|----------------------|-----------------|----------------|-------------------------|-----------------|
| 10000ft | Cybozu Mailwise | Heap | Pivotal Tracker | Squadcast |
| 4me | Cybozu Office | HelloSign | PlanMyLeave | Stackify |
| 7Geese | Cybozu.com | Helpdocs.io | PolicyIQ | Status Hero |
| Abstract | Dashlane | HelpScout | ProcessPlan | StatusCast |
| Accredible | Databricks | HighGear | ProdPad | StatusDashboard |
| Adobe Connect | Datadog | Hightail | Proto.io | StatusHub |
| Adobe Creative Cloud | Declaree | Honey | Proxyclick | Statuspage |
| Adobe Sign | Deputy | Honeycomb.io | PurelyHR | StoriesOnBoard |
| Aha | DeskPro | HostedGraphite | Quip | Stormboard |
| Airbrake | Deskradar | HubSpot | Rapid7 Insight products | SugarCRM |
| AlertOps | Detectify | Humanity | Recognize | SumoLogic |
| AlertSite | Digicert | IdeaScale | Redash.io | SurveyGizmo |
| Amazon Business | Dmarcian | Igloo | Redlock | SurveyMonkey |
| Amazon WorkLink | Docebo | ImageRelay | RescueAssist | Syncplicity |
| Andfrankly | DocuSign | iSpring | RingCentral | Tableau |
| AnswerHub | Dome9 | IT Glue | Roadmunk | Tableau Server |
| AppDynamics | Domo | JamaSoftware | Robin | TalentLMS |

| | | | | |
|----------------|----------------------|-------------------|-----------------------------|---------------------------|
| AppFollow | Drift | Jamf | Rollbar | TargetProcess |
| Area 1 | Dropbox | Jenkins | Room Booking System | TeamSupport |
| Asana | DruvalnSync | JFrog Artifactory | Salesforce | Tenable.io |
| Assembla | Duo | Jira | Salesforce Service Cloud | TextMagic |
| Atlassian | Dynatrace | Jitbit | Samange | ThousandEyes |
| Automox | EduBrite | Jive | SAP BW | TinfoilSecurity |
| BambooHR | Egnyte | join.me | SAP Cloud Platform | TitanFile |
| BenSelect | eLeaP | Kanbanize | SAP CRM ABAP | TOPdesk Operator |
| BitabIZ | Engagedly | Keeper Security | SAP CRM Java | TOPdesk Self Service Desk |
| Bitglass | Envoy | Kentik | SAP Enterprise Portal Java | Trakdesk |
| BlueJeans | Evernote | Kintone | SAP ERP ABAP | Trello |
| BMCRemedyforce | ExpenseIn | Klipfolio | SAP EWM ABAP | Trend Micro Deep Security |
| Bonusly | Expensify | KnowledgeOwl | SAP Fiori ABAP | Uptime.com |
| Box | Expiration Reminder | Kudos | SAP GRC Access Control ABAP | Uptrends |
| Brandfolder | External AWS Account | LiquidFiles | SAP LMS | UserEcho |
| Breezy HR | EZOfficeInventory | LiquidPlanner | SAP Netweaver ABAP | UserVoice |
| Buddy Punch | EZRentOut | Litmos | SAP Netweaver Java | Velpic |
| Bugsee | Fastly | LiveChat | SAP S4 ABAP | Veracode |
| BugSnag | Federated Directory | LogMeInRescue | SAP Solution Manager ABAP | VictorOps |
| Buildkite | FileCloud | Lucidchart | SAP Solution Manager Java | vtiger |
| Bynder | FireHydrant | ManageEngine | SAP SRM ABAP | WayWeDo |
| CakeHR | Fivetran | MangoApps | SAP xMII Java | WeekDone |
| Canvas | Flock | Marketo | ScaleFT | WhosOnLocation |
| Chartio | FogBugz | Metricly | Scalyr | Wordbee |
| Chatwork | Formstack | Miro | ScreenSteps | Workable |

| | | | | |
|------------------|---------------|---------------------|--------------------|-----------------------|
| Circonus | Fossa | MockFlow | Seeit | Workfront |
| Cisco Webex | Freedcamp | Mode Analytics | Sentry.io | Workplace by Facebook |
| CiscoMeraki | Freshdesk | MongoDB | ServiceNow | Workstars |
| CiscoUmbrella | FreshService | Moodle | SimpleMDM | Wrike |
| CitrixShareFile | Front | MuleSoft Anypoint | Skeddly | xMatters |
| Clari | G Suite | MyWebTimeSheets | Skilljar | XperienceHR |
| Clarizen | Genesys Cloud | N2F Expense Reports | Slack | Yodeck |
| ClickTime | GitBook | NewRelic | Slemma | Zendesk |
| Cloud CMS | Github | Nuclino | Sli.do | Zephyr |
| Cloud Conformity | GitLab | Office365 | Small Improvements | Ziflow |
| CloudAMQP | Glasscubes | OnDMARC | Smartsheet | Zillable |
| CloudCheckr | GlassFrog | OpenVoice | SnapEngage | Zoho |
| CloudEndure | GorillaStack | OpsGenie | Snowflake | Zoho One |
| CloudHealth | GoToAssist | Pacific Timesheet | SonarQube | Zoom |
| CloudPassage | GoToMeeting | PagerDuty | SparkPost | |
| CMNTY | GoToTraining | Panopta | Spinnaker | |
| CoderPad | GoToWebinar | Panorama9 | Split.io | |
| Confluence | Grovo | ParkMyCloud | Splunk Cloud | |
| Convo | HackerOne | Peakon | Splunk Enterprise | |
| Coralogix | HackerRank | PhraseApp | Spotinst | |
| Cybozu Garoon | HappyFox | PipeDrive | SproutVideo | |

Add and configure a cloud application

Use this procedure when you need to set up a SAML trust relationship between AWS SSO and your cloud application's service provider. Before you begin this procedure, make sure you have the service provider's metadata exchange file so that you can more efficiently set up the trust. If you do not have this file, you can still use this procedure to configure it manually.

To add and configure a cloud application

1. In the AWS SSO console, choose **Applications** in the left navigation pane. Then choose **Add a new application**.
2. Select the application you want to add from the list. Then choose **Add application**.
3. On the **Configure <application name>** page, under **Details**, enter a **Display name** for the application, such as **Salesforce**.

4. Under **AWS SSO metadata**, do the following:
 - a. Next to **AWS SSO SAML metadatafile**, choose **Download** to download the identity provider metadata.
 - b. Next to **AWS SSO certificate**, choose **Download certificate** to download the identity provider certificate.

Note

You will need these files later when you set up the cloud application from the service provider's website. Follow the instructions from that provider.

5. (Optional) Under **Application properties**, you can specify additional properties for the **Application start URL**, **Relay state**, and **Session duration**. For more information, see [Application properties \(p. 74\)](#).
6. Under **Application metadata**, do one of the following:
 - a. Next to **AWS SSO SAML metadatafile**, choose **Browse** to find and select the metadata file.
 - b. If you do not have a metadata file, choose the link **If you don't have a metadata file, you can manually type your metadata values.**, and then provide the **Application ACS URL** and **Application SAML audience** values.
7. Choose **Save changes** to save the configuration.

Custom SAML 2.0 applications

You can use the AWS SSO application configuration wizard to add support for applications that allow identity federation using Security Assertion Markup Language (SAML) 2.0. In the console, you set these up by choosing **Custom SAML 2.0 application** from the application selector. Most of the steps for configuring a custom SAML application are the same as configuring a cloud application.

However, you also need to provide additional SAML attribute mappings for a custom SAML application. These mappings tell AWS SSO how to populate the SAML assertion correctly for your application. You can provide this additional SAML attribute mapping when you set up the application for the first time. You can also provide SAML attribute mappings on the application detail page that is accessible from the AWS SSO console.

Add and configure a custom SAML 2.0 application

Use this procedure when you need to set up a SAML trust relationship between AWS SSO and your custom application's service provider. Before you begin this procedure, make sure that you have the service provider's certificate and metadata exchange files so that you can finish setting up the trust.

To add and configure a custom SAML application

1. In the AWS SSO console, choose **Applications** in the left navigation pane. Then choose **Add a new application**.
2. In the **Select an application** dialog box, select **Custom SAML 2.0 application** from the list. Then choose **Configure application**.
3. On the **Configure <Custom app name>** page, under **Details**, enter a **Display name** for the application, such as **MyApp**.
4. Under **AWS SSO metadata**, do the following:
 - a. Next to **AWS SSO SAML metadatafile**, choose **Download** to download the identity provider metadata.

- b. Next to **AWS SSO certificate**, choose **Download certificate** to download the identity provider certificate.

Note

You will need these files later when you set up the custom application from the service provider's website.

5. (Optional) Under **Application properties**, you can specify additional properties for the **Application start URL**, **Relay State**, and **Session Duration**. For more information, see [Application properties \(p. 74\)](#).
6. Under **Application metadata**, provide the **Application ACS URL** and **Application SAML audience** values.
7. Choose **Save changes** to save the configuration.

Manage AWS SSO certificates

AWS SSO uses certificates to set up a SAML trust relationship between AWS SSO and your cloud application's service provider. When you add an application in AWS SSO, an AWS SSO certificate is automatically created for use with that application during the setup process. By default, this autogenerated AWS SSO certificate is valid for a period of five years.

As an AWS SSO administrator, you'll occasionally need to replace older certificates with newer ones for a given application. For example, you might need to replace a certificate when the expiration date on the certificate approaches. The process of replacing an older certificate with a newer one is referred to as *certificate rotation*.

Topics

- [Considerations before rotating a certificate \(p. 72\)](#)
- [Rotate an AWS SSO certificate \(p. 72\)](#)
- [Certificate expiration status indicators \(p. 74\)](#)

Considerations before rotating a certificate

Before you start the process of rotating a certificate in AWS SSO, consider the following:

- The certification rotation process requires that you reestablish the trust between AWS SSO and the service provider. To reestablish the trust, use the procedures provided in [Rotate an AWS SSO certificate \(p. 72\)](#).
- Updating the certificate with the service provider may cause a temporary service disruption for your users until the trust has been successfully reestablished. Plan this operation carefully during off peak hours if possible.

Rotate an AWS SSO certificate

Rotating an AWS SSO certificate is a multistep process that involves the following:

- Generating a new certificate
- Adding the new certificate to the service provider's website
- Setting the new certificate to active
- Deleting the inactive certificate

Use all of the following procedures in the following order to complete the certificate rotation process for a given application.

Step 1: Generate a new certificate.

New AWS SSO certificates that you generate can be configured to use the following properties:

- **Validity period** – Specifies the time allotted (in months) before a new AWS SSO certificate expires.
- **Key size** – Determines the number of bits that a key must use with its cryptographic algorithm. You can set this value to either 1024-bit RSA or 2048-bit RSA. For general information about how key sizes work in cryptography, see [Key size](#).
- **Algorithm** – Specifies the algorithm that AWS SSO uses when signing the SAML assertion/response. You can set this value to either SHA-1 or SHA-256. AWS recommends using SHA-256 when possible, unless your service provider requires SHA-1. For general information about how cryptography algorithms work, see [Public-key cryptography](#).

1. Open the [AWS SSO console](#).
2. Choose **Applications**.
3. In the list of applications, choose the application that you want to generate a new certificate for.
4. On the application details page, choose the **Configuration** tab. Under **AWS SSO metadata**, choose **Manage certificate**.
5. On the **AWS SSO certificate** page, choose **Generate new certificate**.
6. In the **Generate new AWS SSO certificate** dialog box, specify the appropriate values for **Validity period**, **Algorithm**, and **Key size**. Then choose **Generate**.

Step 2: Update the service provider's website.

Use the following procedure to reestablish the trust with the application's service provider.

Important

When you upload the new certificate to the service provider, your users might not be able to get authenticated. To correct this situation, set the new certificate as active as described in the next step.

1. In the [AWS SSO console](#), choose the application that you just generated a new certificate for.
2. On the application details page, choose **Edit configuration**.
3. Choose **View instructions**, and then follow the instructions for your specific application service provider's website to add the newly generated certificate.

Step 3: Set the new certificate to active.

An application can have up to two certificates assigned to it. Whichever certificate is set as active, AWS SSO will use it to sign all SAML assertions.

1. Open the [AWS SSO console](#).
2. Choose **Applications**.
3. In the list of applications, choose your application.
4. On the application details page, choose the **Configuration** tab. Under **AWS SSO metadata**, choose **Manage certificate**.
5. On the **AWS SSO certificate** page, select the certificate you want to set to active, choose **Actions**, and then choose **Set as active**.
6. In the **Set the selected certificate as active** dialog, confirm that you understand that setting a certificate to active may require you to re-establish the trust, and then choose **Make active**.

Step 4: Delete the old certificate.

Use the following procedure to complete the certificate rotation process for your application. You can only delete a certificate that is in an **Inactive** state.

1. Open the [AWS SSO console](#).
2. Choose **Applications**.
3. In the list of applications, choose your application.
4. On the application details page, select the **Configuration** tab. Under **AWS SSO metadata**, choose **Manage certificate**.
5. On the **AWS SSO certificate** page, select the certificate you want to delete. Choose **Actions** and then choose **Delete**.
6. In the **Delete certificate** dialog box, choose **Delete**.

Certificate expiration status indicators

While on the **Applications** page in the properties of an application, you may notice colored status indicator icons. These icons appear in the **Expires on** column next to each certificate in the list. The following describes the criteria that AWS SSO uses to determine which icon is displayed for each certificate.

- **Red** – Indicates that a certificate is currently expired.
- **Yellow** – Indicates that a certificate will expire in 90 days or less.
- **Green** – Indicates that a certificate is currently valid and will remain valid for at least 90 more days.

To check the current status of a certificate

1. Open the [AWS SSO console](#).
2. Choose **Applications**.
3. In the list of applications, review the status of the certificates in the list as indicated in the **Expires on** column.

Application properties

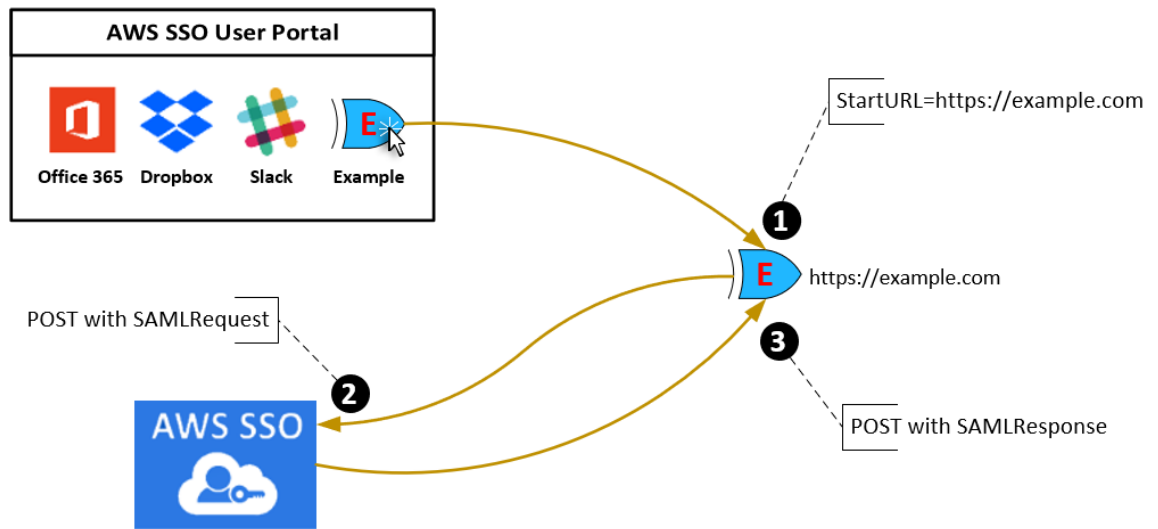
In AWS SSO you can customize the user experience by configuring the following additional application properties.

Application start URL

You use an application start URL to start the federation process with your application. The typical use is for an application that supports only service provider (SP)-initiated binding.

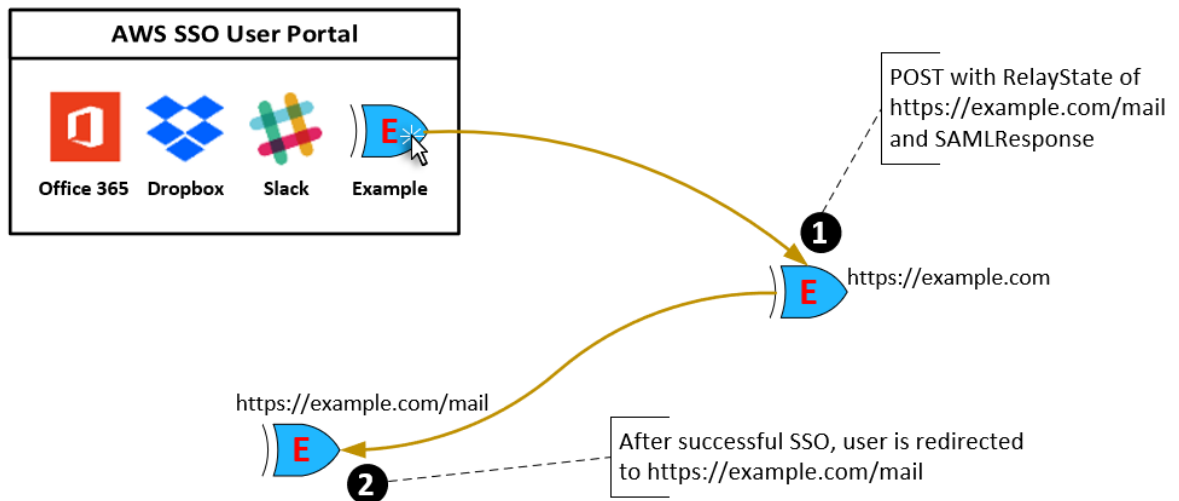
The following steps and diagram illustrate the application start URL authentication workflow when a user chooses an application in the user portal:

1. The user's browser redirects the authentication request using the value for the application start URL (in this case <https://example.com>).
2. The application sends an **HTML POST** with a **SAMLRequest** to AWS SSO.
3. AWS SSO then sends an **HTML POST** with a **SAMLResponse** back to the application.



Relay state

During the federation authentication process, the relay state redirects users within the application. For SAML 2.0, this value is passed, unmodified, to the application. After the application properties are configured, AWS SSO sends the relay state value along with a SAML response to the application.



Session duration

Session duration is the length of time that the application user sessions are valid for. For SAML 2.0, this is used to set the `NotOnOrAfter` date of the SAML assertion's elements; `saml2:SubjectConfirmationData` and `saml2:Conditions`.

Session duration can be interpreted by applications in any of the following ways:

- Applications can use it to determine how long the SAML assertion is valid. Applications do not consider session duration when deciding the time allowed for the user.
- Applications can use it to determine the maximum time that is allowed for the user's session. Applications might generate a user session with a shorter duration. This can happen when the

application only supports user sessions with a duration that is shorter than the configured session length.

- Applications can use it as the exact duration and might not allow administrators to configure the value. This can happen when the application only supports a specific session length.

For more information about how session duration is used, see your specific application's documentation.

Assign user access

Use the following procedure to assign users SSO access to cloud applications or custom SAML 2.0 applications.

Note

- To help simplify administration of access permissions, we recommend that you assign access directly to groups rather than to individual users. With groups you can grant or deny permissions to groups of users, rather than having to apply those permissions to each individual. If a user moves to a different organization, you simply move that user to a different group. The user then automatically receives the permissions that are needed for the new organization.
- When assigning user access to applications, AWS SSO does not currently support users being added to nested groups. If a user is added to a nested group, they may receive a "You do not have any applications" message during sign-in. Assignments must be made against the immediate group the user is a member of.

To assign access to users or groups

1. Open the [AWS SSO console](#).

Note

Make sure that the AWS SSO console is using the Region where your AWS Managed Microsoft AD directory is located before taking the next step.

2. Choose **Applications**.
3. In the list of applications, choose an application to which you want to assign access.
4. On the application details page, choose the **Assigned users** tab. Then choose **Assign users**.
5. In the **Assign users** dialog box, enter a user or group name. Then choose **Search connected directory**. You can specify multiple users or groups by selecting the applicable accounts as they appear in search results.
6. Choose **Assign users**.

Remove user access

Use this procedure to remove user access to cloud applications or custom SAML 2.0 applications.

To remove user access from an application

1. Open the [AWS SSO console](#).
2. Choose **Applications**.
3. In the list of applications, choose an application whose access you want to remove.
4. On the application details page, choose the **Assigned users** tab. Select the user or group that you want to remove and then choose **Remove**.

5. In the **Remove access** dialog box, verify the user or group name. Then choose **Remove access**.

Map attributes in your application to AWS SSO attributes

Some service providers require custom SAML assertions to pass additional data about your user sign-ins. In that case, use the following procedure to specify how your applications user attributes should map to corresponding attributes in AWS SSO.

To map application attributes to attributes in AWS SSO

1. Open the [AWS SSO console](#).
2. Choose **Applications**.
3. In the list of applications, choose the application where you want to map attributes.
4. On the application details page, choose the **Attribute mappings** tab.
5. Choose **Add new attribute mapping**
6. In the first text box, enter the application attribute.
7. In the second text box, enter the attribute in AWS SSO that you want to map to the application attribute. For example, you might want to map the application attribute **Username** to the AWS SSO user attribute **email**. To see the list of allowed user attributes in AWS SSO, see the table in [Attribute mappings \(p. 22\)](#).
8. In the third column of the table, choose the appropriate format for the attribute from the menu.
9. Choose **Save changes**.

Using the user portal

Your user portal provides you with single sign-on access to all your AWS accounts and most commonly used cloud applications such as Office 365, Concur, Salesforce, and many more. From here you can quickly launch multiple applications simply by choosing the AWS account or application icon in the portal. The presence of icons in your portal means that an administrator or designated help desk employee from your company has granted you access to those AWS accounts or applications. It also means that you can access all these accounts or applications from the portal without additional sign-in prompts.

Contact your administrator or help desk to request additional access in the following situations:

- You don't see an AWS account or application that you need access to.
- The access that you have to a given account or application is not what you expected.

Topics

- [Tips for using the portal \(p. 78\)](#)
- [How to accept the invitation to join AWS SSO \(p. 78\)](#)
- [How to sign in to the user portal \(p. 79\)](#)
- [How to sign out of the user portal \(p. 79\)](#)
- [How to search for an AWS account or application \(p. 80\)](#)
- [How to reset your password \(p. 80\)](#)
- [How to get credentials of an IAM role for use with CLI access to an AWS account \(p. 80\)](#)
- [How to bookmark an IAM role for quick access to your management console \(p. 81\)](#)
- [How to register a device for use with multi-factor authentication \(p. 81\)](#)

Tips for using the portal

Like any business tool or application that you use on a daily basis, the user portal might not work as you expected. If that happens, try these tips:

- Occasionally, you may need to sign out and sign back in to the user portal. This might be necessary to access new applications that your administrator recently assigned to you. This is not required, however, because all new applications are refreshed every hour.
- When you sign in to the user portal, you can open any of the applications listed in the portal by choosing the application's icon. After you are done using the application, you can either close the application or sign out of the user portal. Closing the application signs you out of that application only. Any other applications that you have opened from the user portal remain open and running.
- Before you can sign in as a different user, you must first sign out of the user portal. Signing out from the portal completely removes your credentials from the browser session.

How to accept the invitation to join AWS SSO

If this is your first time signing into the user portal, check your email for instructions on how to activate your account.

To activate your account

1. Depending on the email you received from your company, choose one of the following methods to activate your account so that you can start using the user portal.
 - a. If you received an email with the subject **Invitation to join AWS Single Sign-On**, open it and choose **Accept invitation**, which takes you to the **Single Sign-On** page. Here you specify a password, which you use each time you sign in to the portal. Once you have provided a password and have confirmed it, choose **Update User**.
 - b. If you were sent an email from your company's IT support or IT administrator, follow the instructions they provided to activate your account.
2. Once you activate your account by providing a new password, the user portal signs you in automatically. If this does not occur, you can manually sign in to the user portal using the instructions provided in the next step.

How to sign in to the user portal

By this time, you should have been provided a specific sign-in URL to the user portal by an administrator or help desk employee. Once you have this, you can proceed with the following steps to sign in to the portal.

Note

Once you have been signed-in, your user portal session will be valid for 8 hours.

To sign in to the user portal

1. In your browser window, paste in the sign-in URL that you were provided. Then press **Enter**. We recommend that you bookmark this link to the portal now so that you can quickly access it later.
2. Sign in using your standard company user name and password. If you are prompted for a **Verification code**, check your email and then copy and paste the code into the sign-in page.

Note

Verification codes are typically sent through email, but the delivery method can vary. Check with your administrator for details.

3. Once signed in, you can access any AWS account and application that appears in the portal. Simply choose an icon.

Trusted devices

After you choose the option **This is a trusted device** from the sign-in page, AWS SSO will consider all future sign-ins from that device as authorized. This means that AWS SSO will not present an option to enter in an MFA code as long as you are using that trusted device. However, there are some exceptions. These include signing in from a new browser or when your device has been issued an unknown IP address.

How to sign out of the user portal

When you sign out from the portal, your credentials are completely removed from the browser session.

Note

If you want to sign in as a different user, you must first sign out of the user portal.

To sign out of the user portal

- In the user portal, choose **Sign out** from the upper right corner of the portal.

How to search for an AWS account or application

If your list of applications or AWS accounts is too large to find what you need, you can use the **Search** box.

To search for an AWS account or application in the user portal

1. While signed into the portal, choose the **Search** box.
2. Enter the name of the application. Then press **Enter**.

How to reset your password

From time to time you may need to reset your password, depending on your company policies.

To reset your password

1. Open a browser and go to the sign-in page for your user portal.
2. Under the **Sign In** button, choose **Forgot Password?**
3. Provide your **Username** and enter the characters for the provided image to confirm that you are not a robot. Then choose **Recover Password**. This sends an email to you with the subject **AWS Directory Service Reset Password Request**.
4. Once you receive the email, choose **Reset Password**.
5. On the **Single Sign-On** page, need to specify a new password for the portal. Once you have provided a password and have confirmed it, choose **Reset Password**.

How to get credentials of an IAM role for use with CLI access to an AWS account


Use this procedure in the user portal when you need temporary security credentials for short-term access to resources in an AWS account using the AWS CLI. The user portal makes it easy for you to quickly select an AWS account and get the temporary credentials for a given IAM role. You can then copy the necessary CLI syntax (including all necessary credentials) and paste them into your AWS CLI command prompt.

By default, credentials retrieved through the user portal are valid for 1 hour. You can change this value up to 12 hours. Once you have completed this procedure, you can run any AWS CLI command (that your administrator has given you access to) until those temporary credentials have expired.

Note

Before you get started with the steps in this procedure, you must first install the AWS CLI. For instructions, see [Installing the AWS Command Line Interface](#).

To get temporary credentials of an IAM role for CLI access to an AWS account


1. While signed into the portal, choose the **AWS Accounts** icon  to expand the list of accounts.
2. Choose the AWS account from which you want to retrieve access credentials. Then, next to the IAM role name (for example **Administrator**), choose **Command line or programmatic access**.

3. In the **Get credentials** dialog box, choose either **MacOS and Linux** or **Windows**, depending on the operating system where you plan to use the CLI command prompt.
4. Depending on how you want to use the temporary credentials, choose one or more of the following options:
 - If you need to run commands from the AWS CLI in the selected AWS account, under **Option 1: Set AWS environment variables**, pause on the commands. Then choose **Copy**. Paste the commands into the CLI terminal window and press **Enter** to set the necessary environment variables.
 - If you need to run commands from multiple command prompts in the same AWS account, under **Option 2: Add a profile to your AWS credentials file**, pause on the commands. Then choose **Copy**. Paste the commands into your AWS credentials file to set up a newly named profile. For more information, see [Configuration and Credential Files](#) in the *AWS CLI User Guide*. Modifying the credential files in this way enables the `--profile` option in your AWS CLI command so that you can use this credential. This affects all command prompts that use the same credential file.
 - If you need to access AWS resources from an AWS service client, under **Option 3: Use individual values in your AWS service client**, choose **Copy** next to the commands you need to use. For more information, see [Configuration and Credential File Settings](#) in the *AWS CLI User Guide* or see [Tools for Amazon Web Services](#).
5. Continue using the AWS CLI as necessary for your AWS account until the credentials have expired.

How to bookmark an IAM role for quick access to your management console

To make it easier for you to access frequently-used IAM roles from the user portal, you can create a bookmark for a given role associated with a specific AWS account.

To bookmark an IAM role for a specific AWS account

1. While signed into the portal, choose the **AWS Accounts** icon  to expand the list of accounts.
2. Choose the AWS account where you want to create the bookmark.
3. Right-click the **Management console** link, copy the link address, and then use that URL to create your bookmark.

How to register a device for use with multi-factor authentication

Use the following procedure within the user portal to register your new device for multi-factor authentication (MFA).

Note

We recommend that you first download the appropriate Authenticator app onto your device before starting the steps in this procedure. For a list of apps that you can use for MFA devices, see [Authenticator apps](#) (p. 85).

To register your device for use with MFA

1. Go to your user portal.
2. Near the top-right of the page, choose **MFA devices**.
3. On the **Multi-factor authentication (MFA) devices** page, choose **Register device**.

Note

If the **Register MFA device** option is grayed out, you will need to contact your administrator for assistance with registering your device.

4. On the **Register MFA device** page, select one of the following MFA device types, and follow the instructions:

- **Authenticator app**

1. On the **Set up the authenticator app** page, you might notice configuration information for the new MFA device, including a QR code graphic. The graphic is a representation of the secret key that is available for manual entry on devices that do not support QR codes.
2. Using the physical MFA device, do the following:
 - a. Open a compatible MFA authenticator app. For a list of tested apps that you can use with MFA devices, see [Authenticator apps \(p. 85\)](#). If the MFA app supports multiple accounts (multiple MFA devices), choose the option to create a new account (a new MFA device).
 - b. Determine whether the MFA app supports QR codes, and then do one of the following on the **Set up the authenticator app** page:
 - i. Choose **Show QR code**, and then use the app to scan the QR code. For example, you might choose the camera icon or choose an option similar to **Scan code**. Then use the device's camera to scan the code.
 - ii. Choose **show secret key**, and then enter that secret key into your MFA app.

Important

When you configure an MFA device for AWS SSO, we recommend that you save a copy of the QR code or secret key *in a secure place*. This can help if you lose the phone or have to reinstall the MFA authenticator app. If either of those things happen, you can quickly reconfigure the app to use the same MFA configuration.

3. On the **Set up the authenticator app** page, under **Authenticator code**, enter the one-time password that currently appears on the physical MFA device.

Important

Submit your request immediately after generating the code. If you generate the code and then wait too long to submit the request, the MFA device is successfully associated with your user account. But the MFA device is out of sync. This happens because time-based one-time passwords (TOTP) expire after a short period of time. If this happens, you can resync the device.

4. Choose **Assign MFA**. The MFA device can now start generating one-time passwords and is now ready for use with AWS.

- **Security key or Built-in authenticator**

1. On the **Register your user's security key** page, follow the instructions given to you by your browser or platform.

Note

The experience here will vary based on the different operating systems and browsers, so follow the instructions displayed by your browser or platform. After your device has been successfully registered, you will be given the option to associate a friendly display name to your newly enrolled device. If you want to change this, choose **Rename**, enter the new name, and then choose **Save**.

Multi-factor authentication

When you enable multi-factor authentication (MFA), users must sign in to the user portal with their user name and password. This is the first factor, something they know. Users must also sign in with either a code or security key. This is the second factor, something they have or something they are. The second factor could be either an authentication code generated from their mobile device or alternatively by tapping on a security key connected to their computer. Taken together, these multiple factors provide increased security by preventing unauthorized access to your AWS resources unless a valid MFA challenge has been successfully completed.

Important

As a security best practice, we strongly recommend that you enable multi-factor authentication. MFA provides a simple and secure way to add an extra layer of protection on top of the default authentication mechanism of user name and password.

Topics

- [Getting started \(p. 83\)](#)
- [MFA types \(p. 84\)](#)
- [How to manage MFA for AWS SSO \(p. 86\)](#)

Getting started

You can enforce secure access to the user portal, AWS SSO integrated apps, and the AWS CLI by enabling multi-factor authentication (MFA). Review the following topics to get started.

Topics

- [Considerations before using MFA in AWS SSO \(p. 83\)](#)
- [Enable MFA \(p. 84\)](#)

Considerations before using MFA in AWS SSO

Before you enable MFA, consider the following information:

- Users are encouraged to register multiple backup authenticators for all enabled MFA types. This practice can prevent the user's losing access in case of a broken or misplaced MFA device.
- Do not use the option **Require Them to Provide a One-Time Password Sent by Email** if your users must sign in to the user portal to access their email. For example, your users might use Office 365 on the user portal to read their email. In this case, users would not be able to retrieve the verification code and would be unable to sign in to the user portal. For more information, see [Configure MFA device enforcement \(p. 87\)](#).
- If you are already using RADIUS MFA that you configured with AWS Directory Service, then you do not need to enable MFA within AWS SSO. MFA in AWS SSO is an alternative to RADIUS MFA for Microsoft Active Directory users of AWS SSO. For more information, see [RADIUS MFA \(p. 86\)](#).
- You can use AWS SSO's multi-factor authentication capabilities when your identity source is configured with AWS SSO's identity store, AWS Managed Microsoft AD, or AD Connector. MFA in AWS SSO is currently not supported for use by [external identity providers](#).

Enable MFA

Use the following steps to enable MFA using the AWS SSO console. Before you enable MFA, we recommend that you first review details about [MFA types \(p. 84\)](#).

To enable MFA

1. Open the [AWS SSO console](#).
2. In the left navigation pane, choose **Settings**.
3. On the **Settings** page, under **Multi-factor authentication**, choose **Configure**.
4. On the **Configure multi-factor authentication** page, choose one of the following authentication modes based on the level of security that your business needs:

- **Only when their sign-in context changes (context-aware)**

In this mode (the default), AWS SSO analyzes the sign-in context (browser, location, and devices) for each user. AWS SSO then determines whether the user is signing in with a previously trusted context. If a user is signing in from an unknown IP address or is using an unknown device, AWS SSO prompts the user for MFA. This prompt comes in addition to their email address and password credentials.

This mode provides additional protection for users who frequently sign in from their offices. This mode is also easier for those users because they do not need to complete MFA on every sign-in. AWS SSO prompts users with MFA once and permits them to trust their device. Once a user indicates that they want to trust a device, AWS SSO does not challenge the user for MFA for that device. Users are required to provide additional verification only when their sign-in context changes. Such changes include signing in from a new device, a new browser, or an unknown IP address.

- **Every time they sign in (always-on)**

In this mode, AWS SSO requires that users with a registered MFA device will be prompted every time they sign in. You should use this mode if you have organizational or compliance policies that require your users to complete MFA every time they sign in to the user portal. For example, PCI DSS strongly recommends MFA during every sign-in to access applications that support high-risk payment transactions.

- **Never (disabled)**

While in this mode, all users will sign in with their standard user name and password only. Choosing this option disables AWS SSO MFA.

Note

If you are already using RADIUS MFA with AWS Directory Service, and want to continue using it as your default MFA type, then you can leave the authentication mode as disabled to bypass AWS SSO MFA's capabilities. Changing from **Disabled** mode to **Context-aware** or **Always-on** mode will override the existing RADIUS MFA settings. For more information, see [RADIUS MFA \(p. 86\)](#).

5. Choose **Save changes**.

MFA types

MFA types represent the different mechanisms that users will be able to register and provide as secondary means of verifying their identity when challenged. All MFA types are supported for both browser-based console access as well as using the AWS CLI v2 with AWS SSO.

AWS SSO MFA provides support to enable one or both of the following types of client-side authentication types.

- Authenticator apps
- Security keys and built-in authenticators

Alternatively, you can also utilize your own RADIUS implementation connected through AWS Managed Microsoft AD. For more information, see [RADIUS MFA \(p. 86\)](#).

For more information, see [Configure MFA types \(p. 86\)](#).

Topics

- [Authenticator apps \(p. 85\)](#)
- [Security keys and built-in authenticators \(p. 85\)](#)
- [RADIUS MFA \(p. 86\)](#)

Authenticator apps

Authenticator apps are essentially one-time password (OTP)–based third party-authenticators. Users can use an authenticator application installed on their mobile device or tablet as an authorized MFA device. The third-party authenticator application must be compliant with RFC 6238, which is a standards-based TOTP (time-based one-time password) algorithm capable of generating six-digit authentication codes.

When prompted for MFA, users must enter a valid code from their authenticator app within the input box presented. Each MFA device assigned to a user must be unique. Two authenticator apps can be registered for any given user.

Tested authenticator apps

Although any TOTP-compliant application will work with AWS SSO MFA, the following table lists well-known third-party authenticator apps to choose from.

| Operating system | Tested authenticator app |
|------------------|--|
| Android | Authy , Duo Mobile , LastPass Authenticator , Microsoft Authenticator , Google Authenticator |
| iOS | Authy , Duo Mobile , LastPass Authenticator , Microsoft Authenticator , Google Authenticator |

Security keys and built-in authenticators

Web authentication (WebAuthn), enables strong cryptographic authentication using a variety of interoperable authenticators. WebAuthn is a core component of FIDO Alliance's FIDO2 set of specifications. WebAuthn enables the use of any FIDO2 compliant device as well as backward-compatible support for U2F devices.

- **Security keys** – Users can be authenticated using an external USB/BLE/NFC-connected security key such as YubiKey or Fetian devices. Authentication involves simply tapping on a key's sensor when prompted for MFA.
- **Built-In Authenticators** – Some FIDO2-enabled authenticators are built into devices. Examples include TouchID on MacBook or a Windows Hello compatible camera. Users with such a built-in authenticator, can simply provide their fingerprint or facial recognition as a suitable second factor.

FIDO2 and WebAuthn ensures privacy, as cryptographic data generated on devices is unique across sites and biometric data never leaves the device when used. FIDO devices are more secure than traditional forms of MFA, as they are strongly attestable and phishing resistant. For more information about WebAuthn and FIDO2, see [FIDO2: Web Authentication \(WebAuthn\)](#).

RADIUS MFA

[Remote Authentication Dial-In User Service \(RADIUS\)](#) is an industry-standard client-server protocol that provides authentication, authorization, and accounting management so users can connect to network services. AWS Directory Service includes a RADIUS client that connects to the RADIUS server upon which you have implemented your MFA solution. For more information, see [Enable Multi-Factor Authentication for AWS Managed Microsoft AD](#).

You can use either RADIUS MFA or MFA in AWS SSO for user sign-ins to the user portal, but not both. MFA in AWS SSO is an alternative to RADIUS MFA in cases where you want AWS native two-factor authentication for access to the portal.

When you enable MFA in AWS SSO, your users need an MFA device to sign in to the AWS SSO user portal. If you had previously used RADIUS MFA, enabling MFA in AWS SSO effectively overrides RADIUS MFA for users who sign in to the user portal. However, RADIUS MFA continues to challenge users when they sign in to all other applications that work with AWS Directory Service, such as Amazon WorkDocs.

If your MFA is **Disabled** on the AWS SSO console and you have configured RADIUS MFA with AWS Directory Service, RADIUS MFA governs user portal sign-in. This means that AWS SSO falls back to RADIUS MFA configuration if MFA is disabled.

How to manage MFA for AWS SSO

The following topics provide instructions for managing MFA in AWS SSO.

Topics

- [Configure MFA types \(p. 86\)](#)
- [Configure MFA device enforcement \(p. 87\)](#)
- [Register an MFA device \(p. 88\)](#)
- [Manage a user's MFA device \(p. 89\)](#)
- [Allow users to register their own MFA devices \(p. 89\)](#)
- [Disable MFA \(p. 89\)](#)

Configure MFA types

Use the following procedure to configure which device types your users can use when prompted for MFA in the AWS SSO user portal.

To configure MFA types for your users

1. Open the [AWS SSO console](#).
2. In the left navigation pane, choose **Settings**.
3. On the **Settings** page, under **Multi-factor authentication**, choose **Configure**.
4. On the **Configure multi-factor authentication** page, under **Users can authenticate with these MFA types** choose one of the following MFA types based on your business needs. For more information, see [MFA types \(p. 84\)](#).
 - **Security keys and built-in authenticators**

- **Authenticator apps**
5. Choose **Save changes**.

Configure MFA device enforcement

Use the following procedure to determine whether your users must have a registered MFA device when signing in to the AWS SSO user portal.

To configure MFA device enforcement for your users

1. Open the [AWS SSO console](#).
2. In the left navigation pane, choose **Settings**.
3. On the **Settings** page, under **Multi-factor authentication**, choose **Configure**.
4. On the **Configure multi-factor authentication** page, under **If a user does not yet have a registered MFA device** choose one of the following choices based on your business needs:

- **Require them to register an MFA device at sign in**

Use this option when you want to require users who do not yet have a registered MFA device, to self-enroll a device during sign-in following a successful password authentication. This allows you to secure your organization's AWS environments with MFA without having to individually enroll and distribute authentication devices to your users. During self-enrollment, your users can register any device from the available [MFA types \(p. 84\)](#) you've previously enabled. After completing registration, users have the option to give their newly enrolled MFA device a friendly name, after which AWS SSO redirects the user to their original destination. If the user's device is lost or stolen, you can simply remove that device from their account, and AWS SSO will require them to self-enroll a new device during their next sign-in.

- **Require them to provide a one-time password sent by email to sign in**

Use this option when you want to have verification codes sent to users by email. Because email is not bound to a specific device, this option does not meet the bar for industry-standard multi-factor authentication. But it does improve security over having a password alone. Email verification will only be requested if a user has not registered an MFA device. If the **Context-aware** authentication method has been enabled, the user will have the opportunity to mark the device on which they receive the email as trusted. Afterward they will not be required to verify an email code on future logins from that device, browser, and IP address combination.

Note

If you are using Active Directory as your AWS SSO enabled identity source, the email address will always be based on the Active Directory `email` attribute. Custom Active Directory attribute mappings will not override this behavior.

- **Block their sign-in**

Use the **Block Their Sign-In** option when you want to enforce MFA use by every user before they can sign in to AWS.

Important

If your authentication method is set to **Context-aware** a user might select the **This is a trusted device** check box on the sign-in page. In that case, that user will not be prompted for MFA even if you have the **Block their sign in** setting enabled. If you want these users to be prompted, change your authentication method to **Always On**.

- **Allow them to sign in**

The default setting when you first configure AWS SSO MFA. Use this option to indicate that MFA devices are not required in order for your users to sign in to the user portal. Users who chose to register MFA devices will still be prompted for MFA.

5. Choose **Save changes**.

Register an MFA device

Use the following procedure to set up a new MFA device for access by a specific user in the AWS SSO console. You must have physical access to the user's MFA device in order to register it. For example, you might configure MFA for a user who will use an MFA device running on a smartphone. In that case, you must have the smartphone available in order to finish the wizard. For this reason, you might want to let users configure and manage their own MFA devices. For details on how to set this up, see [Allow users to register their own MFA devices \(p. 89\)](#).

To register an MFA device

1. Open the [AWS SSO console](#).
2. In the left navigation pane, choose **Users**. Then choose a user in the list.
3. Choose the **MFA devices** tab, and then choose **Register MFA device**.
4. On the **Register MFA device** page, select one of the following MFA device types, and follow the instructions:
 - **Authenticator app**
 1. On the **Set up the authenticator app** page, AWS SSO displays configuration information for the new MFA device, including a QR code graphic. The graphic is a representation of the secret key that is available for manual entry on devices that do not support QR codes.
 2. Using the physical MFA device, do the following:
 - a. Open a compatible MFA authenticator app. For a list of tested apps that you can use with MFA devices, see [Authenticator apps \(p. 85\)](#). If the MFA app supports multiple accounts (multiple MFA devices), choose the option to create a new account (a new MFA device).
 - b. Determine whether the MFA app supports QR codes, and then do one of the following on the **Set up the authenticator app** page:
 - i. Choose **Show QR code**, and then use the app to scan the QR code. For example, you might choose the camera icon or choose an option similar to **Scan code**. Then use the device's camera to scan the code.
 - ii. Choose **show secret key**, and then type that secret key into your MFA app.

Important

When you configure an MFA device for AWS SSO, we recommend that you save a copy of the QR code or secret key *in a secure place*. This can help if the assigned user loses the phone or has to reinstall the MFA authenticator app. If either of those things happen, you can quickly reconfigure the app to use the same MFA configuration. This avoids the need to create a new MFA device in AWS SSO for the user.

 3. On the **Set up the authenticator app** page, under **Authenticator code**, type the one-time password that currently appears on the physical MFA device.

Important

Submit your request immediately after generating the code. If you generate the code and then wait too long to submit the request, the MFA device is successfully associated with the user. But the MFA device is out of sync. This happens because time-based one-time passwords (TOTP) expire after a short period of time. If this happens, you can resync the device.

 4. Choose **Assign MFA**. The MFA device can now start generating one-time passwords and is now ready for use with AWS.
 - **Security key or Built-in authenticator**

1. On the **Register your user's security key** page, follow the instructions given to you by your browser or platform.

Note

The experience here will vary based on the different operating systems and browsers, so please follow the instructions displayed by your browser or platform. After your user's device has been successfully registered, you will be given the option to associate a friendly display name to your user's newly enrolled device. If you want to change this, choose **Rename**, enter the new name, and then choose **Save**. If you have enabled the option to allow users to manage their own devices, the user will see this friendly name in the user portal.

Manage a user's MFA device

Use the following procedures when you need to rename or delete a user's MFA device.

To rename an MFA device

1. Open the [AWS SSO console](#).
2. In the left navigation pane, choose **Users**. Then choose the user in the list.
3. Choose the **MFA devices** tab, select the device, and then choose **Rename**.
4. When prompted, enter the new name and then choose **Rename**.

To delete an MFA device

1. Open the [AWS SSO console](#).
2. In the left navigation pane, choose **Users**. Then choose the user in the list.
3. Choose the **MFA devices** tab, select the device, and then choose **Delete**.
4. To confirm, type **DELETE**, and then choose **Delete**.

Allow users to register their own MFA devices

Use the following procedure to allow your users to self-register their own MFA devices.

To allow users to register their own MFA devices

1. Open the [AWS SSO console](#).
2. In the left navigation pane, choose **Settings**.
3. On the **Settings** page, under **Multi-factor authentication**, choose **Configure**.
4. On the **Configure multi-factor authentication** page, under **Who can manage MFA devices**, choose **Users can add and manage their own MFA devices**.
5. Choose **Save changes**.

Note

After you set up self-registration for your users, you might want to send them a link to the procedure [How to register a device for use with multi-factor authentication \(p. 81\)](#). This topic provides instructions on how to set up their own MFA devices.

Disable MFA

Use the following procedure to disable MFA in the AWS SSO console.

To disable MFA

1. Open the [AWS SSO console](#).
2. In the left navigation pane, choose **Settings**.
3. On the **Settings** page, under **Multi-factor authentication**, choose **Configure**.
4. On the **Configure multi-factor authentication** page, choose **Never (disabled)**.
5. Choose **Save changes**.

Security in AWS Single Sign-On

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 AWS Single Sign-On, 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 AWS SSO. The following topics show you how to configure AWS SSO to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your AWS SSO resources.

Topics

- [Identity and access management for AWS SSO \(p. 91\)](#)
- [AWS SSO console and API authorization \(p. 106\)](#)
- [Logging and monitoring in AWS Single Sign-On \(p. 107\)](#)
- [Compliance validation for AWS Single Sign-On \(p. 123\)](#)
- [Resilience in AWS Single Sign-On \(p. 124\)](#)
- [Infrastructure security in AWS Single Sign-On \(p. 124\)](#)

Identity and access management for AWS SSO

Access to AWS SSO requires credentials that AWS can use to authenticate your requests. Those credentials must have permissions to access AWS resources, such as an AWS SSO application.

Authentication to the AWS SSO user portal is controlled by the directory that you have connected to AWS SSO. However, authorization to the AWS accounts that are available to end users from within the user portal is determined by two factors:

1. Who has been assigned access to those AWS accounts in the AWS SSO console. For more information, see [Single sign-on access \(p. 51\)](#).
2. What level of permissions have been granted to the end users in the AWS SSO console to allow them the appropriate access to those AWS accounts. For more information, see [Permission sets \(p. 53\)](#).

The following sections explain how you as an administrator can control access to the AWS SSO console or can delegate administrative access for day-to-day tasks from the AWS SSO console.

- [Authentication](#) (p. 92)
- [Access control](#) (p. 93)

Authentication

You can access AWS as any of the following types of identities:

- **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 user** – An [IAM user](#) is an identity within your AWS account that has specific custom permissions (for example, permissions to create a directory in AWS SSO). You can use an IAM user name and password to sign in to secure AWS webpages like the [AWS Management Console](#), [AWS Discussion Forums](#), or the [AWS Support Center](#).

In addition to a user name and password, you can also generate [access keys](#) for each user. You can use these keys when you access AWS services programmatically, either through [one of the several SDKs](#) or by using the [AWS Command Line Interface \(CLI\)](#). The SDK and CLI tools use the access keys to cryptographically sign your request. If you don't use AWS tools, you must sign the request yourself. AWS SSO supports *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*.

- **IAM role** – An [IAM role](#) is an IAM identity that you can create in your account that has specific permissions. An IAM role is similar to an IAM user in that it is an AWS identity with permissions policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have standard long-term credentials such as a password or access keys associated with it. Instead, when you assume a role, it provides you with temporary security credentials for your role session. IAM roles with temporary credentials are useful in the following situations:
- **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*.
- **AWS service access** – A service role is an [IAM role](#) that a service assumes to perform actions on your behalf. Service roles provide access only within your account and cannot be used to grant access to services in other accounts. 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*.
- **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*.

Access control

You can have valid credentials to authenticate your requests, but unless you have permissions you cannot create or access AWS SSO resources. For example, you must have permissions to create an AWS SSO connected directory.

The following sections describe how to manage permissions for AWS SSO. We recommend that you read the overview first.

- [Overview of managing access permissions to your AWS SSO resources](#) (p. 93)
- [Using identity-based policies \(IAM policies\) for AWS SSO](#) (p. 96)
- [Using service-linked roles for AWS SSO](#) (p. 103)

Overview of managing access permissions to your AWS SSO resources

Every AWS resource is owned by an AWS account, and permissions to create or access the resources are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles). Some services (such as AWS Lambda) also support attaching permissions policies to resources.

Note

An *account administrator* (or administrator user) is a user with administrator privileges. For more information, see [IAM best practices](#) in the *IAM User Guide*.

When granting permissions, you decide who is getting the permissions, the resources they get permissions for, and the specific actions that you want to allow on those resources.

Topics

- [AWS SSO resources and operations](#) (p. 93)
- [Understanding resource ownership](#) (p. 93)
- [Managing access to resources](#) (p. 94)
- [Specifying policy elements: actions, effects, resources, and principals](#) (p. 95)
- [Specifying conditions in a policy](#) (p. 95)

AWS SSO resources and operations

In AWS SSO, the primary resources are application instances, profiles, and permission sets.

Understanding resource ownership

A *resource owner* is the AWS account that created a resource. That is, the resource owner is the AWS account of the *principal entity* (the account, an IAM user, or an IAM role) that authenticates the request that creates the resource. The following examples illustrate how this works:

- If the AWS account root user creates an AWS SSO resource, such as an application instance or permission set, your AWS account is the owner of that resource.
- If you create an IAM user in your AWS account and grant that user permissions to create AWS SSO resources, the user can then create AWS SSO resources. However, your AWS account, to which the user belongs, owns the resources.
- If you create an IAM role in your AWS account with permissions to create AWS SSO resources, anyone who can assume the role can create AWS SSO resources. Your AWS account, to which the role belongs, owns the AWS SSO resources.

Managing access to resources

A *permissions policy* describes who has access to what. The following section explains the available options for creating permissions policies.

Note

This section discusses using IAM in the context of AWS SSO. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see [What is IAM?](#) in the *IAM User Guide*. For information about IAM policy syntax and descriptions, see [AWS IAM policy reference](#) in the *IAM User Guide*.

Policies that are attached to an IAM identity are referred to as *identity-based* policies (IAM policies). Policies that are attached to a resource are referred to as *resource-based* policies. AWS SSO supports only identity-based policies (IAM policies).

Topics

- [Identity-based policies \(IAM policies\)](#) (p. 94)
- [Resource-based policies](#) (p. 95)

Identity-based policies (IAM policies)

You can attach policies to IAM identities. For example, you can do the following:

- **Attach a permissions policy to a user or a group in your account** – An account administrator can use a permissions policy that is associated with a particular user to grant permissions for that user to add an AWS SSO resource, such as a new application.
- **Attach a permissions policy to a role (grant cross-account permissions)** – You can attach an identity-based permissions policy to an IAM role to grant cross-account permissions. For example, the administrator in Account A can create a role to grant cross-account permissions to another AWS account (for example, Account B) or an AWS service as follows:
 1. Account A administrator creates an IAM role and attaches a permissions policy to the role that grants permissions to resources in Account A.
 2. Account A administrator attaches a trust policy to the role identifying Account B as the principal who can assume the role.
 3. Account B administrator can then delegate permissions to assume the role to any users in Account B. Doing this allows users in Account B to create or access resources in Account A. The principal in the trust policy can also be an AWS service principal if you want to grant an AWS service permissions to assume the role.

For more information about using IAM to delegate permissions, see [Access management](#) in the *IAM User Guide*.

The following permissions policy grants permissions to a user to run all of the actions that begin with `List`. These actions show information about an AWS SSO resource, such as an application instance or

permissions set. Note that the wildcard character (*) in the `Resource` element indicates that the actions are allowed for all AWS SSO resources that are owned by the account.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sso:List*",
      "Resource": "*"
    }
  ]
}
```

For more information about using identity-based policies with AWS SSO, see [Using identity-based policies \(IAM policies\) for AWS SSO \(p. 96\)](#). For more information about users, groups, roles, and permissions, see [Identities \(users, groups, and roles\)](#) in the *IAM User Guide*.

Resource-based policies

Other services, such as Amazon S3, also support resource-based permissions policies. For example, you can attach a policy to an S3 bucket to manage access permissions to that bucket. AWS SSO doesn't support resource-based policies.

Specifying policy elements: actions, effects, resources, and principals

For each AWS SSO resource (see [AWS SSO resources and operations \(p. 93\)](#)), the service defines a set of API operations. To grant permissions for these API operations, AWS SSO defines a set of actions that you can specify in a policy. Note that performing an API operation can require permissions for more than one action.

The following are the basic policy elements:

- **Resource** – In a policy, you use an Amazon Resource Name (ARN) to identify the resource to which the policy applies. For AWS SSO resources, you always use the wildcard character (*) in IAM policies. For more information, see [AWS SSO resources and operations \(p. 93\)](#).
- **Action** – You use action keywords to identify resource operations that you want to allow or deny. For example, the `sso:DescribePermissionsPolicies` permission allows the user permissions to perform the AWS SSO `DescribePermissionsPolicies` operation.
- **Effect** – You specify the effect when the user requests the specific action—this can be either allow or deny. If you don't explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource, which you might do to make sure that a user cannot access it, even if a different policy grants access.
- **Principal** – In identity-based policies (IAM policies), the user that the policy is attached to is the implicit principal. For resource-based policies, you specify the user, account, service, or other entity that you want to receive permissions (applies to resource-based policies only). AWS SSO doesn't support resource-based policies.

To learn more about IAM policy syntax and descriptions, see [AWS IAM policy reference](#) in the *IAM User Guide*.

Specifying conditions in a policy

When you grant permissions, you can use the access policy language to specify the conditions that are required for a policy to take effect. For example, you might want a policy to be applied only after a

specific date. For more information about specifying conditions in a policy language, see [Condition](#) in the *IAM User Guide*.

To express conditions, you use predefined condition keys. There are no condition keys specific to AWS SSO. However, there are AWS condition keys that you can use as appropriate. For a complete list of AWS keys, see [Available global condition keys](#) in the *IAM User Guide*.

Using identity-based policies (IAM policies) for AWS SSO

This topic provides examples of permissions policies that an account administrator can attach to AWS identities, including IAM users, groups, and roles, and AWS SSO users (as part of a custom permissions policy), for administration of AWS SSO.

Important

We recommend that you first review the introductory topics that explain the basic concepts and options available for you to manage access to your AWS SSO resources. For more information, see [Overview of managing access permissions to your AWS SSO resources \(p. 93\)](#).

The sections in this topic cover the following:

- [AWS managed \(predefined\) policies for AWS SSO \(p. 96\)](#)
- [Customer managed policy examples \(p. 96\)](#)
- [Permissions required to use the AWS SSO console \(p. 102\)](#)

AWS managed (predefined) policies for AWS SSO

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. Managed policies grant necessary permissions for common use cases so you can avoid having to investigate what permissions are needed. For more information, see [AWS Managed Policies](#) in the *IAM User Guide*.

If an existing AWS Managed Policy satisfies your requirements, that is the recommended approach to assigning permissions.

Customer managed policy examples

In this section, you can find examples of common use cases which require a custom IAM policy. The example policies below are identity-based policies, which do not specify the Principal element. This is because with an identity-based policy, you don't specify the principal who gets the permission. Instead, you attach the policy to the principal. When you attach an identity-based permission policy to an IAM role, the principal identified in the role's trust policy gets the permissions. Identity-based policies can be created in IAM and are attached to users, groups, and/or roles, or can be applied to AWS SSO users as part of a custom permissions policy in an AWS SSO permission set.

Note

Use these examples when crafting policies for your environment and make sure to test for both positive ("access granted") and negative ("access denied") test cases prior to deploying in your production deployment. For more information about testing IAM policies, see [Testing IAM policies with the IAM policy simulator](#) in the *IAM User Guide*.

Topics

- [Example 1: Allow a user to view AWS SSO \(p. 97\)](#)
- [Example 2: Allow a user to manage permissions to AWS accounts in AWS SSO \(p. 97\)](#)
- [Example 3: Allow a user to manage applications in AWS SSO \(p. 98\)](#)

- [Example 4: Allow a user to manage users and groups in your AWS SSO directory \(p. 99\)](#)
- [Example 5: Allow a user to administer AWS SSO for specific permission sets, accounts, or OUs \(p. 99\)](#)

Example 1: Allow a user to view AWS SSO

The following permissions policy grants read-only permissions to a user so they can view all of the settings and directory information configured within AWS SSO.

Note

This example policy is provided purely for illustrative purposes. In a production environment, we recommend that you use the AWS Managed Policy for Read-Only access to AWS SSO.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "ds:DescribeDirectories",
        "ds:DescribeTrusts",
        "iam:ListPolicies",
        "organizations:DescribeOrganization",
        "organizations:DescribeAccount",
        "organizations:ListParents",
        "organizations:ListChildren",
        "organizations:ListAccounts",
        "organizations:ListRoots",
        "organizations:ListAccountsForParent",
        "organizations:ListOrganizationalUnitsForParent",
        "sso:ListManagedPoliciesInPermissionSet",
        "sso:ListPermissionSetsProvisionedToAccount",
        "sso:ListAccountAssignments",
        "sso:ListAccountsForProvisionedPermissionSet",
        "sso:ListPermissionSetsProvisionedToAccount",
        "sso:ListPermissionSets",
        "sso:DescribePermissionSet",
        "sso:GetInlinePolicyForPermissionSet",
        "sso-directory:DescribeDirectory",
        "sso-directory:SearchUsers",
        "sso-directory:SearchGroups"
      ],
      "Resource": "*"
    }
  ]
}
```

Example 2: Allow a user to manage permissions to AWS accounts in AWS SSO

The following permissions policy grants permissions to allow a user to create, manage and deploy permission sets for your AWS accounts.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sso:AttachManagedPolicyToPermissionSet",
        "sso:CreateAccountAssignment",

```

```
        "sso:CreatePermissionSet",
        "sso:DeleteAccountAssignment",
        "sso:DeleteInlinePolicyFromPermissionSet",
        "sso:DeletePermissionSet",
        "sso:DetachManagedPolicyFromPermissionSet",
        "sso:ProvisionPermissionSet",
        "sso:PutInlinePolicyToPermissionSet",
        "sso:UpdatePermissionSet"
    ],
    "Resource": "*"
  },
  {
    "Sid": "IAMListPermissions",
    "Effect": "Allow",
    "Action": [
      "iam:ListRoles",
      "iam:ListPolicies"
    ],
    "Resource": "*"
  },
  {
    "Sid": "AccessToSSOProvisionedRoles",
    "Effect": "Allow",
    "Action": [
      "iam:AttachRolePolicy",
      "iam:CreateRole",
      "iam:DeleteRole",
      "iam:DeleteRolePolicy",
      "iam:DetachRolePolicy",
      "iam:GetRole",
      "iam:ListAttachedRolePolicies",
      "iam:ListRolePolicies",
      "iam:PutRolePolicy",
      "iam:UpdateRole",
      "iam:UpdateRoleDescription"
    ],
    "Resource": "arn:aws:iam::*:role/aws-reserved/sso.amazonaws.com/*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "iam:GetSAMLProvider"
    ],
    "Resource": "arn:aws:iam::*:saml-provider/AWSSSO_*_DO_NOT_DELETE"
  }
]
```

Note

The additional permissions listed under the "Sid": "IAMListPermissions", and "Sid": "AccessToSSOProvisionedRoles" sections are required only to enable the user to create assignments in the AWS Organizations management account.

Example 3: Allow a user to manage applications in AWS SSO

The following permissions policy grants permissions to allow a user to view and configure applications in AWS SSO, including pre-integrated SaaS applications from within the AWS SSO catalog.

Note

The `sso:AssociateProfile` operation used in the policy example below is required for management of user and group assignments to applications. It also allows a user to assign users and groups to AWS accounts, using existing permission sets. If a user needs to manage AWS account access within AWS SSO, and requires permissions necessary to manage permission sets, see [Example 2: Allow a user to manage permissions to AWS accounts in AWS SSO \(p. 97\)](#).

As of October 2020, many of these operations are available only through the AWS console. This example policy includes “read” actions such as list, get, and search, which are relevant to the error-free operation of the console for this case.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sso:AssociateProfile",
        "sso:CreateApplicationInstance",
        "sso:ImportApplicationInstanceServiceProviderMetadata",
        "sso:DeleteApplicationInstance",
        "sso:DeleteProfile",
        "sso:DisassociateProfile",
        "sso:GetApplicationTemplate",
        "sso:UpdateApplicationInstanceServiceProviderConfiguration",
        "sso:UpdateApplicationInstanceDisplayData",
        "sso:DeleteManagedApplicationInstance",
        "sso:UpdateApplicationInstanceStatus",
        "sso:GetManagedApplicationInstance",
        "sso:UpdateManagedApplicationInstanceStatus",
        "sso:CreateManagedApplicationInstance",
        "sso:UpdateApplicationInstanceSecurityConfiguration",
        "sso:UpdateApplicationInstanceResponseConfiguration",
        "sso:GetApplicationInstance",
        "sso:CreateApplicationInstanceCertificate",
        "sso:UpdateApplicationInstanceResponseSchemaConfiguration",
        "sso:UpdateApplicationInstanceActiveCertificate",
        "sso:DeleteApplicationInstanceCertificate",
        "sso:ListApplicationInstanceCertificates",
        "sso:ListApplicationTemplates",
        "sso:ListApplications",
        "sso:ListApplicationInstances",
        "sso:ListDirectoryAssociations",
        "sso:ListProfiles",
        "sso:ListProfileAssociations",
        "sso:ListInstances",
        "sso:GetProfile",
        "sso:GetSSOStatus",
        "sso:GetSsoConfiguration",
        "sso-directory:DescribeDirectory",
        "sso-directory:DescribeUsers",
        "sso-directory:ListMembersInGroup",
        "sso-directory:SearchGroups",
        "sso-directory:SearchUsers"
      ],
      "Resource": "*"
    }
  ]
}
```

Example 4: Allow a user to manage users and groups in your AWS SSO directory

The following permissions policy grants permissions to allow a user to create, view, modify, and delete users and groups in AWS SSO.

Note that in some cases direct modifications to users and groups in AWS SSO are restricted. For example, when Active Directory, or an external identity provider with Automatic Provisioning enabled, is selected as the identity source.

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "VisualEditor0",
    "Effect": "Allow",
    "Action": [
      "sso-directory:ListGroupForUser",
      "sso-directory:DisableUser",
      "sso-directory:EnableUser",
      "sso-directory:SearchGroups",
      "sso-directory>DeleteGroup",
      "sso-directory:AddMemberToGroup",
      "sso-directory:DescribeDirectory",
      "sso-directory:UpdateUser",
      "sso-directory:ListMembersInGroup",
      "sso-directory:CreateUser",
      "sso-directory:DescribeGroups",
      "sso-directory:SearchUsers",
      "sso:ListDirectoryAssociations",
      "sso-directory:RemoveMemberFromGroup",
      "sso-directory>DeleteUser",
      "sso-directory:DescribeUsers",
      "sso-directory:UpdateGroup",
      "sso-directory:CreateGroup"
    ],
    "Resource": "*"
  }
]
```

Example 5: Allow a user to administer AWS SSO for specific permission sets, accounts, or OUs

As your team grows, an AWS SSO administrator may want to consider implementing a delegation model that enables AWS account and application administrators to manage their users SSO access to their resources. Using this model and the example policies below, permissions can be delegated for administering AWS accounts, permission sets, or OUs. Once any of these policies have been created, that same policy can be selected anytime the same permissions need to be delegated to other administrative users.

Account-based delegation

The following policy grants permissions that allow an administrator to delegate access for each of the AWS accounts noted under Resource.

```
{
  "Sid": "DelegateAccountsAdminAccess",
  "Effect": "Allow",
  "Action": [
    "sso:ProvisionPermissionSet",
    "sso:CreateAccountAssignment",
    "sso>DeleteInlinePolicyFromPermissionSet",
    "sso:UpdateInstanceAccessControlAttributeConfiguration",
    "sso:PutInlinePolicyToPermissionSet",
    "sso>DeleteAccountAssignment",
    "sso:DetachManagedPolicyFromPermissionSet",
    "sso>DeletePermissionSet",
    "sso:AttachManagedPolicyToPermissionSet",
    "sso:CreatePermissionSet",
    "sso:UpdatePermissionSet",
    "sso:CreateInstanceAccessControlAttributeConfiguration",
    "sso>DeleteInstanceAccessControlAttributeConfiguration"
  ]
}
```

```
],
  "Resource":[
    "arn:aws:sso::account/112233445566",
    "arn:aws:sso::account/223344556677",
    "arn:aws:sso::account/334455667788"
  ]
}
```

Permission-based delegation

The following permissions policy grants permissions that allow an administrator to delegate access for a given AWS SSO instance ID ARN (in this case, `ssoins-1111111111`) or permission set ARN (`ssoins-1111111111/ps-112233abcdef123`).

For more information about finding the ARNs associated with an AWS SSO instance ID or permission set, see [Identify your permission set and AWS SSO Instance IDs](#) on the AWS Security blog.

```
{
  "Sid": "DelegatePermissionsAdminAccess",
  "Effect": "Allow",
  "Action": [
    "sso:ProvisionPermissionSet",
    "sso:CreateAccountAssignment",
    "sso:DeleteInlinePolicyFromPermissionSet",
    "sso:UpdateInstanceAccessControlAttributeConfiguration",
    "sso:PutInlinePolicyToPermissionSet",
    "sso:DeleteAccountAssignment",
    "sso:DetachManagedPolicyFromPermissionSet",
    "sso:DeletePermissionSet",
    "sso:AttachManagedPolicyToPermissionSet",
    "sso:CreatePermissionSet",
    "sso:UpdatePermissionSet",
    "sso:CreateInstanceAccessControlAttributeConfiguration",
    "sso:DeleteInstanceAccessControlAttributeConfiguration",
    "sso:ProvisionApplicationInstanceForAWSAccount"
  ],
  "Resource": [
    "arn:aws:sso::instance/ssoins-1111111111",
    "arn:aws:sso::permissionSet/ssoins-1111111111/ps-112233abcdef123"
  ]
}
```

OU-based delegation

OU-based delegation requires two policy statements within the same permission set and the ability for [Tagging AWS Single Sign-On resources \(p. 125\)](#).

The following permissions policy grants permissions that allow an administrator to delegate access for an OU. In the first policy statement, we filter the permission sets by both the `Environment` and `OU` tags. In the second policy statement, we filter the accounts that are tagged for the `Development` OU.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DelegateOUsAdminAccess",
      "Effect": "Allow",
      "Action": [
        "sso:ProvisionPermissionSet",
        "sso:CreateAccountAssignment",
        "sso:DeleteInlinePolicyFromPermissionSet",
        "sso:UpdateInstanceAccessControlAttributeConfiguration",

```

```
        "sso:PutInlinePolicyToPermissionSet",
        "sso:DeleteAccountAssignment",
        "sso:DetachManagedPolicyFromPermissionSet",
        "sso:DeletePermissionSet",
        "sso:AttachManagedPolicyToPermissionSet",
        "sso:CreatePermissionSet",
        "sso:UpdatePermissionSet",
        "sso:CreateInstanceAccessControlAttributeConfiguration",
        "sso:DeleteInstanceAccessControlAttributeConfiguration",
        "sso:ProvisionApplicationInstanceForAWSAccount"
    ],
    "Resource": "arn:aws:sso:::permissionSet/*/*",
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/Environment": "Development",
            "aws:ResourceTag/OU": "Test"
        }
    }
},
{
    "Sid": "Instance",
    "Effect": "Allow",
    "Action": [
        "sso:ProvisionPermissionSet",
        "sso:CreateAccountAssignment",
        "sso:DeleteInlinePolicyFromPermissionSet",
        "sso:UpdateInstanceAccessControlAttributeConfiguration",
        "sso:PutInlinePolicyToPermissionSet",
        "sso:DeleteAccountAssignment",
        "sso:DetachManagedPolicyFromPermissionSet",
        "sso:DeletePermissionSet",
        "sso:AttachManagedPolicyToPermissionSet",
        "sso:CreatePermissionSet",
        "sso:UpdatePermissionSet",
        "sso:CreateInstanceAccessControlAttributeConfiguration",
        "sso:DeleteInstanceAccessControlAttributeConfiguration",
        "sso:ProvisionApplicationInstanceForAWSAccount"
    ],
    "Resource": [
        "arn:aws:sso:::instance/ssoins-82593a6ed92c8920",
        "arn:aws:sso:::account/112233445566",
        "arn:aws:sso:::account/223344556677",
        "arn:aws:sso:::account/334455667788"
    ]
}
]
```

More information

To see an example walkthrough that covers how to delegate administration of user identities in an IAM environment, see [How to delegate management of identity in AWS Single Sign-On](#) on the AWS Security Blog.

Permissions required to use the AWS SSO console

For a user to work with the AWS SSO console without errors, additional permissions are required. If an IAM policy has been created that is more restrictive than the minimum required permissions, the console won't function as intended for users with that policy. The following example lists the set of permissions that may be needed to ensure error-free operation within the AWS SSO console.

```
{
    "Version": "2012-10-17",
```

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "sso:DescribeAccountAssignmentCreationStatus",
      "sso:DescribeAccountAssignmentDeletionStatus",
      "sso:DescribePermissionSet",
      "sso:DescribePermissionSetProvisioningStatus",
      "sso:DescribePermissionsPolicies",
      "sso:DescribeRegisteredRegions",
      "sso:GetApplicationInstance",
      "sso:GetApplicationTemplate",
      "sso:GetInlinePolicyForPermissionSet",
      "sso:GetManagedApplicationInstance",
      "sso:GetMfaDeviceManagementForDirectory",
      "sso:GetPermissionSet",
      "sso:GetPermissionsPolicy",
      "sso:GetProfile",
      "sso:GetSharedSsoConfiguration",
      "sso:GetSsoConfiguration",
      "sso:GetSSOStatus",
      "sso:GetTrust",
      "sso:ListAccountAssignmentCreationStatus",
      "sso:ListAccountAssignmentDeletionStatus",
      "sso:ListAccountAssignments",
      "sso:ListAccountsForProvisionedPermissionSet",
      "sso:ListApplicationInstanceCertificates",
      "sso:ListApplicationInstances",
      "sso:ListApplications",
      "sso:ListApplicationTemplates",
      "sso:ListDirectoryAssociations",
      "sso:ListInstances",
      "sso:ListManagedPoliciesInPermissionSet",
      "sso:ListPermissionSetProvisioningStatus",
      "sso:ListPermissionSets",
      "sso:ListPermissionSetsProvisionedToAccount",
      "sso:ListProfileAssociations",
      "sso:ListProfiles",
      "sso:ListTagsForResource",
      "sso-directory:DescribeDirectory",
      "sso-directory:DescribeGroups",
      "sso-directory:DescribeUsers",
      "sso-directory:ListGroupsForUser",
      "sso-directory:ListMembersInGroup",
      "sso-directory:SearchGroups",
      "sso-directory:SearchUsers"
    ],
    "Resource": "*"
  }
]
}

```

Using service-linked roles for AWS SSO

AWS Single Sign-On 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 AWS SSO. It is predefined by AWS SSO and includes all the permissions that the service requires to call other AWS services on your behalf. For more information, see [Service-linked roles \(p. 64\)](#).

A service-linked role makes setting up AWS SSO easier because you don't have to manually add the necessary permissions. AWS SSO defines the permissions of its service-linked role, and unless defined otherwise, only AWS SSO can assume its role. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

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 AWS SSO

AWS SSO uses the service-linked role named **AWSServiceRoleForSSO** to grant AWS SSO permissions to manage AWS resources, including IAM roles, policies, and SAML IdP on your behalf.

The AWSServiceRoleForSSO service-linked role trusts the following services to assume the role:

- AWS SSO

The AWSServiceRoleForSSO service-linked role permissions policy allows AWS SSO to complete the following on roles on the path `"/aws-reserved/sso.amazonaws.com/"` and with the name prefix `"AWSReservedSSO_"`:

- `iam:AttachRolePolicy`
- `iam:CreateRole`
- `iam>DeleteRole`
- `iam>DeleteRolePolicy`
- `iam:DetachRolePolicy`
- `iam:GetRole`
- `iam:ListRolePolicies`
- `iam:PutRolePolicy`
- `iam:ListAttachedRolePolicies`

The AWSServiceRoleForSSO service-linked role permissions policy allows AWS SSO to complete the following on SAML providers with name prefix as `"AWSSSO_"`:

- `iam:CreateSAMLProvider`
- `iam:GetSAMLProvider`
- `iam:UpdateSAMLProvider`
- `iam>DeleteSAMLProvider`

The AWSServiceRoleForSSO service-linked role permissions policy allows AWS SSO to complete the following on all organizations:

- `organizations:DescribeAccount`
- `organizations:DescribeOrganization`
- `organizations:ListAccounts`

The AWSServiceRoleForSSO service-linked role permissions policy allows AWS SSO to complete the following on all IAM roles (*):

- `iam:listRoles`

The AWSServiceRoleForSSO service-linked role permissions policy allows AWS SSO to complete the following on `"arn:aws:iam::*:role/aws-service-role/sso.amazonaws.com/AWSServiceRoleForSSO"`:

- `iam:GetServiceLinkedRoleDeletionStatus`
- `iam:DeleteServiceLinkedRole`

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*.

Creating a service-linked role for AWS SSO

You don't need to manually create a service-linked role. Once enabled, AWS SSO creates a service-linked role in all accounts within the organization in AWS Organizations. AWS SSO also creates the same service-linked role in every account that is subsequently added to your organization. This role allows AWS SSO to access each account's resources on your behalf.

Important

If you were using the AWS SSO service before December 7, 2017, when it began supporting service-linked roles, then AWS SSO created the `AWSServiceRoleForSSO` role in your account. To learn more, see [A New Role Appeared in My IAM Account](#).

If you delete this service-link role and then need to create it again, you can use the same process to recreate the role in your account.

Editing a service-linked role for AWS SSO

AWS SSO does not allow you to edit the `AWSServiceRoleForSSO` service-linked role. After you create a service-linked role, you cannot 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 AWS SSO

You don't need to manually delete the `AWSServiceRoleForSSO` role. When an AWS account is removed from an AWS organization, AWS SSO automatically cleans up the resources and deletes the service-linked role from that AWS account.

You can also use the IAM console, the IAM CLI, or the IAM API to manually delete the service-linked role. To do this, you must first manually clean up the resources for your service-linked role and then you can manually delete it.

Note

If the AWS SSO 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 AWS SSO resources used by the `AWSServiceRoleForSSO`

1. [Remove user access \(p. 53\)](#) for all users and groups that have access to the AWS account.
2. [Delete permission sets \(p. 56\)](#) that you have associated with the AWS account.
3. [Remove the IAM identity provider \(p. 64\)](#) to delete the trust between AWS SSO and the AWS account.

To manually delete the service-linked role using IAM

Use the IAM console, the IAM CLI, or the IAM API to delete the `AWSServiceRoleForSSO` service-linked role. For more information, see [Deleting a Service-Linked Role](#) in the *IAM User Guide*.

AWS SSO console and API authorization

Existing AWS SSO console APIs support dual authorization. If you have existing AWS SSO instances that were created prior to October 15th 2020, you can use the following table to determine which API operations now map to newer API operations that were released after that date.

SSO Instances created before October 15th 2020 will honor both old and new API actions as long as there is no explicit deny on any one of the actions. SSO Instances created after October 15th 2020 will use the newer API actions for authorization in the AWS SSO console.

| Operation name | API actions used before October 15th, 2020 | API actions used after October 15th, 2020 |
|---|--|---|
| AssociateProfile | AssociateProfile | CreateAccountAssignment |
| AttachManagedPolicy | PutPermissionsPolicy | AttachManagedPolicyToPermissionSet |
| CreatePermissionSet | CreatePermissionSet | CreatePermissionSet |
| DeleteApplicationInstanceForAWSAccount | DeleteApplicationInstance DeleteTrust | DeleteAccountAssignment |
| DeleteApplicationProfileForAWSAccount | DeleteProfile | DeleteAccountAssignment |
| DeletePermissionsPolicy | DeletePermissionsPolicy | DeleteInlinePolicyFromPermissionSet |
| DeletePermissionSet | DeletePermissionSet | DeletePermissionSet |
| DescribePermissionsPolicies | DescribePermissionsPolicies | ListManagedPoliciesInPermissionSet |
| DetachManagedPolicy | DeletePermissionsPolicy | DetachManagedPolicyFromPermissionSet |
| DisassociateProfile | DisassociateProfile | DeleteAccountAssignment |
| GetApplicationInstanceForAWSAccount | GetApplicationInstance | ListAccountAssignments |
| GetAWSAccountProfileStatus | GetProfile | ListPermissionSetsProvisionedToAccount |
| GetPermissionSet | GetPermissionSet | DescribePermissionSet |
| GetPermissionsPolicy | GetPermissionsPolicy | GetInlinePolicyForPermissionSet |
| ListAccountsWithProvisionedPermissionSets | ListApplicationInstances GetApplicationInstance | ListAccountsForProvisionedPermissionSet |
| ListAWSAccountProfiles | ListProfiles GetProfile | ListPermissionSetsProvisionedToAccount |
| ListPermissionSets | ListPermissionSets | ListPermissionSets |
| ListProfileAssociations | ListProfileAssociations | ListAccountAssignments |
| ProvisionApplicationInstanceForAWSAccount | CreateApplicationInstance SetApplicationInstance | CreateAccountAssignment |
| ProvisionApplicationProfileForAWSAccount | CreateProfile SetProfile UpdateProfile | CreateAccountAssignment |
| ProvisionSAMLProvider | GetTrust CreateTrust UpdateTrust | CreateAccountAssignment |

| Operation name | API actions used before October 15th, 2020 | API actions used after October 15th, 2020 |
|----------------------|--|---|
| PutPermissionsPolicy | PutPermissionsPolicy | PutInlinePolicyToPermissionSet |
| UpdatePermissionSet | UpdatePermissionSet | UpdatePermissionSet |

Logging and monitoring in AWS Single Sign-On

As a best practice, you should monitor your organization to ensure that changes are logged. This helps you to ensure that any unexpected change can be investigated and unwanted changes can be rolled back. AWS Single Sign-On currently supports two AWS services that help you monitor your organization and the activity that happens within it.

Topics

- [Logging AWS SSO API calls with AWS CloudTrail \(p. 107\)](#)
- [Amazon CloudWatch Events \(p. 123\)](#)

Logging AWS SSO API calls with AWS CloudTrail

AWS SSO is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS SSO. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, Amazon CloudWatch Logs, and Amazon CloudWatch Events. Using the information collected by CloudTrail, you can determine the request that was made to AWS SSO, 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, see the [AWS CloudTrail User Guide](#).

Topics

- [AWS SSO information in CloudTrail \(p. 107\)](#)
- [Understanding AWS SSO log file entries \(p. 110\)](#)
- [Understanding AWS SSO sign-in events \(p. 112\)](#)

AWS SSO information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in AWS SSO, 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 AWS SSO, 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 AWS 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 the following:

- [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](#)

When CloudTrail logging is enabled in your AWS account, API calls made to AWS SSO actions are tracked in log files. AWS SSO records are written together with other AWS service records in a log file. CloudTrail determines when to create and write to a new file based on a time period and file size.

The following AWS SSO CloudTrail operations are supported:

| Console API operations | Public API operations |
|--|---|
| AssociateDirectory | AttachManagedPolicyToPermissionSet |
| AssociateProfile | CreateAccountAssignment |
| CreateApplicationInstance | CreateInstanceAccessControlAttributeConfiguration |
| CreateApplicationInstanceCertificate | CreatePermissionSet |
| CreatePermissionSet | DeleteAccountAssignment |
| CreateProfile | DeleteInlinePolicyFromPermissionSet |
| DeleteApplicationInstance | DeleteInstanceAccessControlAttributeConfiguration |
| DeleteApplicationInstanceCertificate | DeletePermissionSet |
| DeletePermissionsPolicy | DescribeAccountAssignmentCreationStatus |
| DeletePermissionSet | DescribeAccountAssignmentDeletionStatus |
| DeleteProfile | DescribeInstanceAccessControlAttributeConfiguration |
| DescribePermissionsPolicies | DescribePermissionSet |
| DisassociateDirectory | DescribePermissionSetProvisioningStatus |
| DisassociateProfile | DetachManagedPolicyFromPermissionSet |
| GetApplicationInstance | GetInlinePolicyForPermissionSet |
| GetApplicationTemplate | ListAccountAssignmentCreationStatus |
| GetMfaDeviceManagementForDirectory | ListAccountAssignmentDeletionStatus |
| GetPermissionSet | ListAccountAssignments |
| GetSSOStatus | ListAccountsForProvisionedPermissionSet |
| ImportApplicationInstanceServiceProvider | ListInstances |
| ListApplicationInstances | ListManagedPoliciesInPermissionSet |
| ListApplicationInstanceCertificates | ListPermissionSetProvisioningStatus |
| ListApplicationTemplates | ListPermissionSets |
| ListDirectoryAssociations | ListPermissionSetsProvisionedToAccount |
| ListPermissionSets | ListTagsForResource |
| ListProfileAssociations | ProvisionPermissionSet |

| Console API operations | Public API operations |
|---|---|
| ListProfiles | PutInlinePolicyToPermissionSet |
| PutMfaDeviceManagementForDirectory | TagResource |
| PutPermissionsPolicy | UntagResource |
| StartSSO | UpdateInstanceAccessControlAttributeConfiguration |
| UpdateApplicationInstanceActiveCertificate | UpdatePermissionSet |
| UpdateApplicationInstanceDisplayData | |
| UpdateApplicationInstanceServiceProviderConfiguration | |
| UpdateApplicationInstanceStatus | |
| UpdateApplicationInstanceResponseConfiguration | |
| UpdateApplicationInstanceResponseSchemaConfiguration | |
| UpdateApplicationInstanceSecurityConfiguration | |
| UpdateDirectoryAssociation | |
| UpdateProfile | |

For more information about AWS SSO's public API operations, see the [AWS Single Sign-On API Reference Guide](#).

The following AWS SSO identity store CloudTrail operations are supported:

- AddMemberToGroup
- CompleteVirtualMfaDeviceRegistration
- CompleteWebAuthnDeviceRegistration
- CreateAlias
- CreateExternalIdPConfigurationForDirectory
- CreateGroup
- CreateUser
- DeleteExternalIdPConfigurationForDirectory
- DeleteGroup
- DeleteMfaDeviceForUser
- DeleteUser
- DescribeDirectory
- DescribeGroups
- DescribeUsers
- DisableExternalIdPConfigurationForDirectory
- DisableUser
- EnableExternalIdPConfigurationForDirectory
- EnableUser
- GetAWSSPConfigurationForDirectory
- ListExternalIdPConfigurationsForDirectory

- `ListGroupsForUser`
- `ListMembersInGroup`
- `ListMfaDevicesForUser`
- `PutMfaDeviceManagementForDirectory`
- `RemoveMemberFromGroup`
- `SearchGroups`
- `SearchUsers`
- `StartVirtualMfaDeviceRegistration`
- `StartWebAuthnDeviceRegistration`
- `UpdateExternalIdPConfigurationForDirectory`
- `UpdateGroup`
- `UpdateMfaDeviceForUser`
- `UpdatePassword`
- `UpdateUser`
- `VerifyEmail`

The following AWS SSO OIDC CloudTrail actions are supported:

- `CreateToken`
- `RegisterClient`
- `StartDeviceAuthorization`

The following AWS SSO Portal CloudTrail actions are supported:

- `Authenticate`
- `Federate`

Every log entry contains information about who generated the request. The identity information in the log helps you determine whether the request was made by an AWS account root user or with IAM user credentials. You can also learn whether the request was made with temporary security credentials for a role or federated user or by another AWS service. For more information, see the [CloudTrail userIdentity Element](#).

You can create a trail and store your log files in your Amazon S3 bucket for as long as you want. You can also define Amazon S3 lifecycle rules to archive or delete log files automatically. By default, your log files are encrypted with Amazon S3 server-side encryption (SSE).

To be notified of log file delivery, configure CloudTrail to publish Amazon SNS notifications when new log files are delivered. For more information, see [Configuring Amazon SNS notifications for CloudTrail](#).

You can also aggregate AWS SSO log files from multiple AWS Regions and multiple AWS accounts into a single Amazon S3 bucket. For more information, see [Receiving CloudTrail log files from multiple Regions](#) and [Receiving CloudTrail log files from multiple accounts](#).

Understanding AWS SSO 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 shows a CloudTrail log entry for an administrator (samadams@example.com) that took place in the AWS SSO console:

```
{
  "Records": [
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAJAIENLMexample",
        "arn": "arn:aws:iam::08966example:user/samadams",
        "accountId": "08966example",
        "accessKeyId": "AKIAIJJM2K4example",
        "userName": "samadams"
      },
      "eventTime": "2017-11-29T22:39:43Z",
      "eventSource": "sso.amazonaws.com",
      "eventName": "DescribePermissionsPolicies",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "203.0.113.0",
      "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202.94 Safari/537.36",
      "requestParameters": {
        "permissionSetId": "ps-79a0dde74b95ed05"
      },
      "responseElements": null,
      "requestID": "319ac6a1-d556-11e7-a34f-69a333106015",
      "eventID": "a93a952b-13dd-4ae5-a156-d3ad6220b071",
      "readOnly": true,
      "resources": [

    ],
      "eventType": "AwsApiCall",
      "recipientAccountId": "08966example"
    }
  ]
}
```

The following example shows a CloudTrail log entry for an end-user (bobsmith@example.com) action that took place in the AWS SSO user portal:

```
{
  "Records": [
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "Unknown",
        "principalId": "example.com/S-1-5-21-1122334455-3652759393-4233131409-1126",
        "accountId": "08966example",
        "userName": "bobsmith@example.com"
      },
      "eventTime": "2017-11-29T18:48:28Z",
      "eventSource": "sso.amazonaws.com",
      "eventName": "https://portal.sso.us-east-1.amazonaws.com/instance/appinstances",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "203.0.113.0",
      "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202.94 Safari/537.36",
      "requestParameters": null,
      "responseElements": null,
      "requestID": "de6c0435-ce4b-49c7-9bcc-bc5ed631ce04",
    }
  ]
}
```

```
        "eventID": "e6e1f3df-9528-4c6d-a877-6b2b895d1f91",  
        "eventType": "AwsApiCall",  
        "recipientAccountId": "08966example"  
    }  
]  
}
```

The following example shows a CloudTrail log entry for an end-user (bobsmith@example.com) action that took place in AWS SSO OIDC:

```
{  
  "eventVersion": "1.05",  
  "userIdentity": {  
    "type": "Unknown",  
    "principalId": "example.com//S-1-5-21-1122334455-3652759393-4233131409-1126",  
    "accountId": "08966example",  
    "userName": "bobsmith@example.com"  
  },  
  "eventTime": "2020-06-16T01:31:15Z",  
  "eventSource": "sso.amazonaws.com",  
  "eventName": "CreateToken",  
  "awsRegion": "us-east-1",  
  "sourceIPAddress": "203.0.113.0",  
  "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36  
(KHTML, like Gecko) Chrome/62.0.3202.94 Safari/537.36",  
  "requestParameters": {  
    "clientId": "clientid1234example",  
    "clientSecret": "HIDDEN_DUE_TO_SECURITY_REASONS",  
    "grantType": "urn:ietf:params:oauth:grant-type:device_code",  
    "deviceCode": "devicecode1234example"  
  },  
  "responseElements": {  
    "accessToken": "HIDDEN_DUE_TO_SECURITY_REASONS",  
    "tokenType": "Bearer",  
    "expiresIn": 28800,  
    "refreshToken": "HIDDEN_DUE_TO_SECURITY_REASONS",  
    "idToken": "HIDDEN_DUE_TO_SECURITY_REASONS"  
  },  
  "eventID": "09a6e1a9-50e5-45c0-9f08-e6ef5089b262",  
  "readOnly": false,  
  "resources": [  
    {  
      "accountId": "08966example",  
      "type": "IdentityStoreId",  
      "ARN": "d-1234example"  
    }  
  ],  
  "eventType": "AwsApiCall",  
  "recipientAccountId": "08966example"  
}
```

Understanding AWS SSO sign-in events

AWS CloudTrail logs successful and unsuccessful sign-in events for all AWS Single Sign-On identity sources. Native SSO and Active Directory (AD Connector and AWS Managed Microsoft AD) sourced identities will include additional sign-in events that are captured each time a user is prompted to solve a specific credential challenge or factor, as well as the status of that particular credential verification request. Only after a user has completed all required credential challenges will the user be signed in, which will result in a `UserAuthentication` event being logged.

The following table captures each of the AWS SSO sign-in CloudTrail event names, their purpose, and applicability to different identity sources.

| Event name | Event purpose | Identity source applicability |
|------------------------|---|--|
| CredentialChallenge | Used to notify that AWS SSO has requested the user to solve a specific credential challenge and specifies the CredentialType that was required (For example, PASSWORD or TOTP). | Native AWS SSO users, AD Connector, and AWS Managed Microsoft AD |
| CredentialVerification | Used to notify that the user has attempted to solve a specific CredentialChallenge request and specifies whether that credential succeeded or failed. | Native AWS SSO users, AD Connector, and AWS Managed Microsoft AD |
| UserAuthentication | Used to notify that all authentication requirements the user was challenged with have been successfully completed and that the user was successfully signed in. <i>Users failing to successfully complete the required credential challenges will result in no UserAuthentication event being logged.</i> | All identity sources |

The following table captures additional useful event data fields contained within specific sign-in CloudTrail events.

| Event name | Event purpose | Sign-in event applicability | Example values |
|--------------------------|---|---|--|
| AuthWorkflowID | Used to correlate all events emitted across an entire sign-in sequence. For each user sign-in, multiple events may be emitted by AWS SSO. | CredentialChallenge, CredentialVerification, UserAuthentication | "AuthWorkflowID": "9cde74b32-8362-4a01-a524-de21df59fd83" |
| CredentialType | Used to specify the credential or factor that was challenged. UserAuthentication events will include all of the CredentialType values that were successfully verified across the user's sign-in sequence. | CredentialChallenge, CredentialVerification, UserAuthentication | "CredentialType": "PASSWORD" or "CredentialType": "PASSWORD,TOTP" (possible values include: PASSWORD, TOTP, WEBAUTHN, EXTERNAL_IDP, RESYNC_TOTP) |
| DeviceEnrollmentRequired | Used to specify that the user was required to register an MFA device | UserAuthentication | "DeviceEnrollmentRequired": "true" |

| Event name | Event purpose | Sign-in event applicability | Example values |
|------------|--|-----------------------------|--|
| | during sign-in, and that the user successfully completed that request. | | |
| LoginTo | Used to specify the redirect location following a successful sign-in sequence. | UserAuthentication | "LoginTo": "https://mydirectory.awsapps.com/start/....." |

Example events for AWS SSO sign-in scenarios

The following examples show the expected sequence of CloudTrail events for different sign-in scenarios.

Topics

- [Successful sign-in when authenticating with only a password \(p. 114\)](#)
- [Successful sign-in when authenticating with an external identity provider \(p. 116\)](#)
- [Successful sign-in when authenticating with a password and a TOTP authenticator app \(p. 117\)](#)
- [Successful sign-in when authenticating with a password and forced MFA registration is required \(p. 120\)](#)
- [Failed sign-in when authenticating with only a password \(p. 122\)](#)

Successful sign-in when authenticating with only a password

The following sequence of events captures an example of a successful password only sign-in.

CredentialChallenge (Password)

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "Unknown",
    "principalId": "111122223333",
    "arn": "",
    "accountId": "111122223333",
    "accessKeyId": "",
    "userName": "user1"
  },
  "eventTime": "2020-12-07T20:33:58Z",
  "eventSource": "signin.amazonaws.com",
  "eventName": "CredentialChallenge",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36",
  "requestParameters": null,
  "responseElements": null,
  "additionalEventData": {
    "AuthWorkflowID": "9de74b32-8362-4a01-a524-de21df59fd83",
    "CredentialType": "PASSWORD"
  },
  "requestID": "5be44ffb-6946-4f47-acaf-1adebd4afead",
  "eventID": "27ea7725-c1fd-4355-bdba-d0e628e0e604",
  "readOnly": false,
  "eventType": "AwsServiceEvent",
  "managementEvent": true,
}
```



```
"eventCategory": "Management",
"recipientAccountId": "111122223333",
"serviceEventDetails": {
  "CredentialChallenge": "Success"
}
}
```

Successful CredentialVerification (Password)

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "Unknown",
    "principalId": "111122223333",
    "arn": "",
    "accountId": "111122223333",
    "accessKeyId": "",
    "userName": "user1"
  },
  "eventTime": "2020-12-07T20:34:09Z",
  "eventSource": "signin.amazonaws.com",
  "eventName": "CredentialVerification",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/87.0.4280.66 Safari/537.36",
  "requestParameters": null,
  "responseElements": null,
  "additionalEventData": {
    "AuthWorkflowID": "9de74b32-8362-4a01-a524-de21df59fd83",
    "CredentialType": "PASSWORD"
  },
  "requestID": "f3cf52ad-fd3d-4889-8c15-f18d1a7c7393",
  "eventID": "c49640f6-0c8a-43d3-a6e0-900e3bb188d4",
  "readOnly": false,
  "eventType": "AwsServiceEvent",
  "managementEvent": true,
  "eventCategory": "Management",
  "recipientAccountId": "111122223333",
  "serviceEventDetails": {
    "CredentialVerification": "Success"
  }
}
```

Successful UserAuthentication (Password Only)

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "Unknown",
    "principalId": "111122223333",
    "arn": "",
    "accountId": "111122223333",
    "accessKeyId": "",
    "userName": "user1"
  },
  "eventTime": "2020-12-07T20:34:09Z",
  "eventSource": "signin.amazonaws.com",
  "eventName": "UserAuthentication",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/87.0.4280.66 Safari/537.36",
}
```

```
    "requestParameters":null,
    "responseElements":null,
    "additionalEventData":{
      "AuthWorkflowID":"9de74b32-8362-4a01-a524-de21df59fd83",
      "LoginTo":"https://d-1234567890.awsapps.com/start/?
state=QVlBQmVGMHFiS0wzWlp1SFgrR25BRnFobU5nQUlnQUJBQk5FWVhSaFVHeGhibVZUZEdGMFpWQmhjbUZ0QUFsUVpYSmxaM0pwY
BshlIc5OBAA6ftz73M6LsfLWDlfOxviO2K3wet9461C30f_iWdilx-
zv__4pSHf7mcUIs&wdc_csrf_token=srAzW1jK4GPYYoR452ruZ38DxEsDY9x81q1tVRSnno5pUjISvP7TqziOLiBLBUSxEjOmQk2X
east-1",
      "CredentialType":"PASSWORD"
    },
    "requestID":"f3cf52ad-fd3d-4889-8c15-f18d1a7c7393",
    "eventID":"e959a95a-2b33-478d-906c-4fe303e8a9f1",
    "readOnly":false,
    "eventType":"AwsServiceEvent",
    "managementEvent":true,
    "eventCategory":"Management",
    "recipientAccountId":"111122223333",
    "serviceEventDetails":{
      "UserAuthentication":"Success"
    }
  }
}
```

Successful sign-in when authenticating with an external identity provider

The following sequence of events captures an example of a successful sign-in when authenticated through the SAML protocol using an external identity provider.

Successful UserAuthentication (External Identity Provider)

```
{
  "eventVersion":"1.08",
  "userIdentity":{
    "type":"Unknown",
    "principalId":"111122223333",
    "arn":"",
    "accountId":"111122223333",
    "accessKeyId":"",
    "userName":"user1"
  },
  "eventTime":"2020-12-07T20:34:09Z",
  "eventSource":"signin.amazonaws.com",
  "eventName":"UserAuthentication",
  "awsRegion":"us-east-1",
  "sourceIPAddress":"203.0.113.0",
  "userAgent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/87.0.4280.66 Safari/537.36",
  "requestParameters":null,
  "responseElements":null,
  "additionalEventData":{
    "AuthWorkflowID":"9de74b32-8362-4a01-a524-de21df59fd83",
    "LoginTo":"https://d-1234567890.awsapps.com/start/?
state=QVlBQmVGMHFiS0wzWlp1SFgrR25BRnFobU5nQUlnQUJBQk5FWVhSaFVHeGhibVZUZEdGMFpWQmhjbUZ0QUFsUVpYSmxaM0pwY
BshlIc5OBAA6ftz73M6LsfLWDlfOxviO2K3wet9461C30f_iWdilx-
zv__4pSHf7mcUIs&wdc_csrf_token=srAzW1jK4GPYYoR452ruZ38DxEsDY9x81q1tVRSnno5pUjISvP7TqziOLiBLBUSxEjOmQk2X
east-1",
    "CredentialType":"EXTERNAL_IDP"
  },
  "requestID":"f3cf52ad-fd3d-4889-8c15-f18d1a7c7393",
  "eventID":"e959a95a-2b33-478d-906c-4fe303e8a9f1",
  "readOnly":false,
  "eventType":"AwsServiceEvent",
  "managementEvent":true,
  "eventCategory":"Management",

```

```
"recipientAccountId":"111122223333",
"serviceEventDetails":{
  "UserAuthentication":"Success"
}
}
```

Successful sign-in when authenticating with a password and a TOTP authenticator app

The following sequence of events captures an example where multi-factor authentication was required during sign-in and the user successfully signed in using a password and a TOTP authenticator app.

CredentialChallenge (Password)

```
{
  "eventVersion":"1.08",
  "userIdentity":{
    "type":"Unknown",
    "principalId":"111122223333",
    "arn":"",
    "accountId":"111122223333",
    "accessKeyId":"",
    "userName":"user1"
  },
  "eventTime":"2020-12-08T20:40:13Z",
  "eventSource":"signin.amazonaws.com",
  "eventName":"CredentialChallenge",
  "awsRegion":"us-east-1",
  "sourceIPAddress":"203.0.113.0",
  "userAgent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/87.0.4280.66 Safari/537.36",
  "requestParameters":null,
  "responseElements":null,
  "additionalEventData":{
    "AuthWorkflowID":"303486b5-fce1-4d59-ba1d-eb3acb790729",
    "CredentialType":"PASSWORD"
  },
  "requestID":"e454ea66-1027-4d00-9912-09c0589649e1",
  "eventID":"d89cc0b5-a23a-4b88-843a-89329aaef2e",
  "readOnly":false,
  "eventType":"AwsServiceEvent",
  "managementEvent":true,
  "eventCategory":"Management",
  "recipientAccountId":"111122223333",
  "serviceEventDetails":{
    "CredentialChallenge":"Success"
  }
}
```

Successful CredentialVerification (Password)

```
{
  "eventVersion":"1.08",
  "userIdentity":{
    "type":"Unknown",
    "principalId":"111122223333",
    "arn":"",
    "accountId":"111122223333",
    "accessKeyId":"",
    "userName":"user1"
  },
  "eventTime":"2020-12-08T20:40:20Z",
  "eventSource":"signin.amazonaws.com",
  "eventName":"CredentialVerification",

```

```
    "awsRegion": "us-east-1",
    "sourceIPAddress": "203.0.113.0",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/87.0.4280.66 Safari/537.36",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData": {
      "AuthWorkflowID": "303486b5-fce1-4d59-ba1d-eb3acb790729",
      "CredentialType": "PASSWORD"
    },
    "requestID": "92c4ac90-0d9b-452d-95d5-728487612f5e",
    "eventID": "4533fd49-6669-4d0b-b272-a0b2139309a8",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "serviceEventDetails": {
      "CredentialVerification": "Success"
    }
  }
}
```

CredentialChallenge (TOTP)

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "Unknown",
    "principalId": "111122223333",
    "arn": "",
    "accountId": "111122223333",
    "accessKeyId": "",
    "userName": "user1"
  },
  "eventTime": "2020-12-08T20:40:20Z",
  "eventSource": "signin.amazonaws.com",
  "eventName": "CredentialChallenge",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/87.0.4280.66 Safari/537.36",
  "requestParameters": null,
  "responseElements": null,
  "additionalEventData": {
    "AuthWorkflowID": "303486b5-fce1-4d59-ba1d-eb3acb790729",
    "CredentialType": "TOTP"
  },
  "requestID": "92c4ac90-0d9b-452d-95d5-728487612f5e",
  "eventID": "29202f08-f240-40cc-b789-c0cea8a27847",
  "readOnly": false,
  "eventType": "AwsServiceEvent",
  "managementEvent": true,
  "eventCategory": "Management",
  "recipientAccountId": "111122223333",
  "serviceEventDetails": {
    "CredentialChallenge": "Success"
  }
}
```

Successful CredentialVerification (TOTP)

```
{
  "eventVersion": "1.08",
```

```
"userIdentity":{
  "type":"Unknown",
  "principalId":"111122223333",
  "arn":"",
  "accountId":"111122223333",
  "accessKeyId":"",
  "userName":"user1"
},
"eventTime":"2020-12-08T20:40:27Z",
"eventSource":"signin.amazonaws.com",
"eventName":"CredentialVerification",
"awsRegion":"us-east-1",
"sourceIPAddress":"203.0.113.0",
"userAgent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/87.0.4280.66 Safari/537.36",
"requestParameters":null,
"responseElements":null,
"additionalEventData":{
  "AuthWorkflowID":"303486b5-fce1-4d59-ba1d-eb3acb790729",
  "CredentialType":"TOTP"
},
"requestID":"c40a691f-eeb1-4352-b286-5e909f96f318",
"eventID":"e889ff1d-fcaf-454f-805d-7132cf2362a4",
"readOnly":false,
"eventType":"AwsServiceEvent",
"managementEvent":true,
"eventCategory":"Management",
"recipientAccountId":"111122223333",
"serviceEventDetails":{
  "CredentialVerification":"Success"
}
}
```

Successful UserAuthentication (Password + TOTP)

```
{
  "eventVersion":"1.08",
  "userIdentity":{
    "type":"Unknown",
    "principalId":"111122223333",
    "arn":"",
    "accountId":"111122223333",
    "accessKeyId":"",
    "userName":"user1"
  },
  "eventTime":"2020-12-08T20:40:27Z",
  "eventSource":"signin.amazonaws.com",
  "eventName":"UserAuthentication",
  "awsRegion":"us-east-1",
  "sourceIPAddress":"203.0.113.0",
  "userAgent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/87.0.4280.66 Safari/537.36",
  "requestParameters":null,
  "responseElements":null,
  "additionalEventData":{
    "AuthWorkflowID":"303486b5-fce1-4d59-ba1d-eb3acb790729",
    "LoginTo":"https://d-1234567890.awsapps.com/start/?state
\u003dQVLBQmVLeFhWeDRmZFJmMmxHcWYwdzhZck5RQUlnQUJBQk5FWVhSaFVHeGhibVZUZEdGMFpWQmhjbUZ0QUFsUVpYSmxaM0pwY
\u0026auth_code
\u003d11FirlmCVJ-4Y5UY6RI10UCXvRePCHd6195xvYglrwo1Pj7B-7UGIGlYUUVe31Nkzd7ihxKn6DMdnFf00108qc3RFR8FUD1w8
Sx-pjBXXG_jUcvBk_UILdGytV4o1u97h42B-
TA_6uwdmJiwlDcCz_Rv44d_BS0PkulW-5LVJyloeP1H0FPPMeheyuk5Uy48d5of9-c\u0026wdc_csrf_token
\u003dNMlui44guoVnxRd0qu2tYJIdyyFPX6SDRNTspIScfMM0AgFbho1nvvCaxPTghHbghHCRIXdfffTzH0sL1ow419BobnmqBsnJN
\u0026organization\u003dd-9067230c03\u0026region\u003dus-east-1",
    "CredentialType":"PASSWORD,TOTP"
  }
}
```

```
    },  
    "requestID": "c40a691f-eeb1-4352-b286-5e909f96f318",  
    "eventID": "7a8c8725-db2f-488d-a43e-788dc6c73a4a",  
    "readOnly": false,  
    "eventType": "AwsServiceEvent",  
    "managementEvent": true,  
    "eventCategory": "Management",  
    "recipientAccountId": "111122223333",  
    "serviceEventDetails": {  
        "UserAuthentication": "Success"  
    }  
}
```

Successful sign-in when authenticating with a password and forced MFA registration is required

The following sequence of events captures an example of a successful password sign in, but the user was required and successfully completed registering an MFA device before completing their sign-in.

CredentialChallenge (Password)

```
{  
  "eventVersion": "1.08",  
  "userIdentity": {  
    "type": "Unknown",  
    "principalId": "111122223333",  
    "arn": "",  
    "accountId": "111122223333",  
    "accessKeyId": "",  
    "userName": "user1"  
  },  
  "eventTime": "2020-12-09T01:24:02Z",  
  "eventSource": "signin.amazonaws.com",  
  "eventName": "CredentialChallenge",  
  "awsRegion": "us-east-1",  
  "sourceIPAddress": "203.0.113.0",  
  "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36",  
  "requestParameters": null,  
  "responseElements": null,  
  "additionalEventData": {  
    "AuthWorkflowID": "76d8a26d-ad9c-41a4-90c3-d607cdd7155c",  
    "CredentialType": "PASSWORD"  
  },  
  "requestID": "321f4b13-42b5-4005-a0f7-826cad26d159",  
  "eventID": "8c707b0f-e45a-4a9c-bee2-ff68638d2f1b",  
  "readOnly": false,  
  "eventType": "AwsServiceEvent",  
  "managementEvent": true,  
  "eventCategory": "Management",  
  "recipientAccountId": "111122223333",  
  "serviceEventDetails": {  
    "CredentialChallenge": "Success"  
  }  
}
```

Successful CredentialVerification (Password)

```
{  
  "eventVersion": "1.08",  
  "userIdentity": {  
    "type": "Unknown",  
    "principalId": "111122223333",  
    "arn": "",
```

```
    "accountId": "111122223333",
    "accessKeyId": "",
    "userName": "user1"
  },
  "eventTime": "2020-12-09T01:24:09Z",
  "eventSource": "signin.amazonaws.com",
  "eventName": "CredentialVerification",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/87.0.4280.66 Safari/537.36",
  "requestParameters": null,
  "responseElements": null,
  "additionalEventData": {
    "AuthWorkflowID": "76d8a26d-ad9c-41a4-90c3-d607cdd7155c",
    "CredentialType": "PASSWORD"
  },
  "requestID": "12b57efa-0a92-4479-91a3-5b6641817c21",
  "eventID": "783b0c89-7142-4942-8b84-6ee0de1b992e",
  "readOnly": false,
  "eventType": "AwsServiceEvent",
  "managementEvent": true,
  "eventCategory": "Management",
  "recipientAccountId": "111122223333",
  "serviceEventDetails": {
    "CredentialVerification": "Success"
  }
}
```

Successful UserAuthentication (Password + MFA Registration Required)

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "Unknown",
    "principalId": "111122223333",
    "arn": "",
    "accountId": "111122223333",
    "accessKeyId": "",
    "userName": "user1"
  },
  "eventTime": "2020-12-09T01:24:14Z",
  "eventSource": "signin.amazonaws.com",
  "eventName": "UserAuthentication",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/87.0.4280.66 Safari/537.36",
  "requestParameters": null,
  "responseElements": null,
  "additionalEventData": {
    "AuthWorkflowID": "76d8a26d-ad9c-41a4-90c3-d607cdd7155c",
    "LoginTo": "https://d-1234567890.awsapps.com/start/?state
\u003dQVLBQmVGQ3VqdHF5aW9CUDdrNXRTVTJUaWNNQUlnQUJBQk5FWVhSaFVhGhibVZUZEdGMFpWQmhjbUZ0QUFsUVpYSmxaM0pwY
\u0026auth_code
\u003d11eZ80S_maUsZ7ABETjeQhyWfvIHyz52rgR28sYAKN1oEk2G07czrwzXvE9HL1N2K9De8LyBEV83SFeDQfrWpkwXfaBc2kNR1
FJyJqkoGrt_w6rm_MpAn0uyrVq8udY_EgU3fhOL3QWvWiquYnDPMyPmmy_qkZgR9rz__BI\u0026wdc_csrf_token
\u003dJih9U62o5LQDtYLNqCK8a6xj0GJg5BRWq2tb175y8vAmwZhAqrgrgbxXat2M646UZGp93krw7WYQdHIgi5OYI9Qsckf4aovhC
\u003dd-9067230c03\u0026region\u003dus-east-1",
    "CredentialType": "PASSWORD",
    "DeviceEnrollmentRequired": "true"
  },
  "requestID": "74d24604-a365-4237-8c4a-350795494b92",
  "eventID": "a15bf257-7f37-46c0-b67c-fea5fa6166be",
  "readOnly": false,
}
```

```
"eventType": "AwsServiceEvent",
"managementEvent": true,
"eventCategory": "Management",
"recipientAccountId": "111122223333",
"serviceEventDetails": {
  "UserAuthentication": "Success"
}
}
```

Failed sign-in when authenticating with only a password

The following sequence of events captures an example of a failed password only sign-in.

CredentialChallenge (Password)

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "Unknown",
    "principalId": "111122223333",
    "arn": "",
    "accountId": "111122223333",
    "accessKeyId": "",
    "userName": "user1"
  },
  "eventTime": "2020-12-08T18:56:15Z",
  "eventSource": "signin.amazonaws.com",
  "eventName": "CredentialChallenge",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/87.0.4280.66 Safari/537.36",
  "requestParameters": null,
  "responseElements": null,
  "additionalEventData": {
    "AuthWorkflowID": "adb67c4-8188-4e2b-8527-fe539e328fa7",
    "CredentialType": "PASSWORD"
  },
  "requestID": "f54848ea-b1aa-402f-bf0d-a54561a2ffcc",
  "eventID": "d96f1d6c-dbd9-4a0b-9a45-6a2b66078c78",
  "readOnly": false,
  "eventType": "AwsServiceEvent",
  "managementEvent": true,
  "eventCategory": "Management",
  "recipientAccountId": "111122223333",
  "serviceEventDetails": {
    "CredentialChallenge": "Success"
  }
}
```

Failed CredentialVerification (Password)

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "Unknown",
    "principalId": "111122223333",
    "arn": "",
    "accountId": "111122223333",
    "accessKeyId": "",
    "userName": "user1"
  },
  "eventTime": "2020-12-08T18:56:21Z",
```



```
"eventSource": "signin.amazonaws.com",
"eventName": "CredentialVerification",
"awsRegion": "us-east-1",
"sourceIPAddress": "203.0.113.0",
"userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/87.0.4280.66 Safari/537.36",
"requestParameters": null,
"responseElements": null,
"additionalEventData": {
  "AuthWorkflowID": "adbf67c4-8188-4e2b-8527-fe539e328fa7",
  "CredentialType": "PASSWORD"
},
"requestID": "04528c82-a678-4a1f-a56d-ea2c6445a72a",
"eventID": "9160fe06-fc2a-474f-9b78-000ee067a09d",
"readOnly": false,
"eventType": "AwsServiceEvent",
"managementEvent": true,
"eventCategory": "Management",
"recipientAccountId": "111122223333",
"serviceEventDetails": {
  "CredentialVerification": "Failure"
}
}
```

Amazon CloudWatch Events

AWS SSO can work with CloudWatch Events to raise events when administrator-specified actions occur in an organization. For example, because of the sensitivity of such actions, most administrators would want to be warned every time someone creates a new account in the organization or when an administrator of a member account attempts to leave the organization. You can configure CloudWatch Events rules that look for these actions and then send the generated events to administrator-defined targets. Targets can be an Amazon SNS topic that emails or text messages its subscribers. You could also create an AWS Lambda function that logs the details of the action for your later review.

To learn more about CloudWatch Events, including how to configure and enable it, see the [Amazon CloudWatch Events User Guide](#).

Compliance validation for AWS Single Sign-On

Third-party auditors assess the security and compliance of AWS services as part of multiple AWS compliance programs, such as SOC, PCI, FedRAMP, and HIPAA.

To learn whether AWS Single Sign-On or other AWS services are 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 AWS services 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 baseline environments on AWS that are security and compliance focused.
- [Architecting for HIPAA Security and Compliance Whitepaper](#) – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.

Note

Not all services are compliant with HIPAA.

- [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* – The AWS Config service 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.
- [AWS Audit Manager](#) – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

Resilience in AWS Single Sign-On

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability 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](#).

Infrastructure security in AWS Single Sign-On

As a managed service, AWS Single Sign-On 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 AWS SSO 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 by 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.

Tagging AWS Single Sign-On resources

A *tag* is a metadata label that you assign or that 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 you can assign the same tag to resources from different services to indicate that the resources are related. For example, you could assign the same tag to a specific permission set in your AWS SSO instance.
- Track your AWS costs. You activate these tags on the AWS Billing and Cost Management dashboard. AWS uses the 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 and Cost Management User Guide*.
- Control access to your resources based on the tags that are assigned to them. You 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 an AWS SSO instance, but only if the AWS SSO instance 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*.

Currently, tags can only be applied to permission sets and cannot be applied to corresponding roles that AWS SSO creates in AWS accounts. You can use the AWS SSO console, AWS CLI or the AWS SSO APIs to add, edit, or delete tags for a given permission set.

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 AWS SSO.

Tag restrictions

The following basic restrictions apply to tags on AWS SSO 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 key and value – a-z, A-Z, 0-9, space, and the following characters: `_` `.` `:` `/` `=` `+` `-` and `@`
- Keys and values are case sensitive
- Don't use `aws:` as a prefix for keys; it's reserved for AWS use

Manage tags using the AWS SSO console

You can use the AWS SSO console to add, edit and remove tags that are associated with your permission sets.

To manage tags for an AWS SSO console

1. Open the [AWS SSO console](#).
2. Choose **AWS accounts**.
3. Choose the **Permission sets** tab.
4. Choose the name of the permission set that has the tags you want to manage.
5. On the **Permissions** tab, under **Tags** do one of the following, and then proceed to the next step:
 - a. If you have existing tags assigned for this permission set, choose **Edit tags**.
 - b. If no tags have been assigned to this permission set, choose **Add tags**.
6. For each new tag, type the values in the **Key** and **Value (optional)** columns. When you are finished, choose **Save changes**.

To remove a tag, choose the **X** in the **Remove** column next to the tag you want to remove.

AWS CLI examples

The AWS CLI provides commands that you can use to manage the tags that you assign to your permission set.

Assigning tags

Use the following commands to assign tags to your permission set.

Example `tag-resource` Command for a permission set

Assign tags to a permission set by using `tag-resource` within the `sso` set of commands:

```
$ aws sso tag-resource \  
> --instance-arn sso-instance-arn \  
> --resource-arn sso-resource-arn \  
> --tags Stage=Test
```

This command includes the following parameters:

- `instance-arn` – The Amazon Resource Name (ARN) of the AWS SSO instance under which the operation will be executed.
- `resource-arn` – The ARN of the resource with the tags to be listed.
- `tags` – The key-value pairs of the tags.

To assign multiple tags at once, specify them in a comma-separated list:

```
$ aws sso tag-resource \  
> --instance-arn sso-instance-arn \  
> --resource-arn sso-resource-arn \  
> --tags Stage=Test, CostCenter=80432, Owner=SysEng
```

Viewing tags

Use the following commands to view the tags that you have assigned to your permission set.

Example `list-tags-for-resource` Command for a permission set

View the tags that are assigned to a permission set by using `list-tags-for-resource` within the `sso` set of commands:

```
$ aws sso list-tags-for-resource --resource-arn sso-resource-arn
```

Removing tags

Use the following commands to remove tags from a permission set.

Example `untag-resource` Command for a permission set

Remove tags from a permission set by using `untag-resource` within the `sso` set of commands:

```
$ aws sso untag-resource \  
> --instance-arn sso-instance-arn \  
> --resource-arn sso-resource-arn \  
> --tag-keys Stage CostCenter Owner
```

For the `--tag-keys` parameter, specify one or more tag keys, and do not include the tag values.

Applying tags when you create a permission set

Use the following commands to assign tags at the moment you create a permission set.

Example `create-permission-set` Command with tags

When you create a permission set by using the `create-permission-set` command, you can specify tags with the `--tags` parameter:

```
$ aws sso create-permission-set \  
> --instance-arn sso-instance-arn \  
> --name permission=set-name \  
> --tags Stage=Test, CostCenter=80432, Owner=SysEng
```

Manage tags using the AWS SSO API

You can use the following actions in the AWS SSO API to manage the tags for your permission set.

API actions for AWS SSO instance tags

Use the following API actions to assign, view, and remove tags for a permission set.

- [TagResource](#)
- [ListTagsForResource](#)
- [UntagResource](#)
- [CreatePermissionSet](#)

Integrating AWS CLI with AWS SSO

AWS Command Line Interface (CLI) version 2 integration with AWS Single Sign-On (AWS SSO) simplifies the sign-in process. Developers can sign in directly to the AWS CLI using the same Active Directory or AWS SSO credentials that they normally use to sign in to AWS SSO, and access their assigned accounts and roles. For example, after an administrator configures AWS SSO to use Active Directory for authentication, a developer can sign into the AWS CLI directly using their Active Directory credentials.

AWS CLI integration with AWS SSO offers the following benefits:

- Enterprises can enable their developers to sign in using credentials from AWS SSO or Active Directory by connecting AWS SSO to their Active Directory using AWS Directory Service.
- Developers can sign in from the CLI for faster access.
- Developers can list and switch between accounts and roles to which they have assigned access.
- Developers can generate and save named role profiles in their CLI configuration automatically and reference them in the CLI to run commands in desired accounts and roles.
- The CLI manages short-term credentials automatically so developers can start in and stay in the CLI securely without interruption, and run long running scripts.

How to integrate AWS CLI with AWS SSO

To use the AWS CLI integration with AWS SSO, you need to download, install, and configure AWS Command Line Interface version 2. For detailed steps on how to download and integrate the AWS CLI with AWS SSO, see [Configuring the AWS CLI to use AWS Single Sign-On \(AWS SSO\)](#) in the *AWS Command Line Interface User Guide*.

AWS SSO Region availability

AWS SSO is available in several commonly used AWS Regions. This availability makes it easier for you to configure user access to multiple AWS accounts and business applications. When your users sign in to the user portal, they can select the AWS account that they have permission to. Then they can access the AWS Management Console. For a full list of the Regions that AWS SSO supports, see [AWS Single Sign-On endpoints and quotas](#).

AWS SSO Region data

When you first enable AWS SSO, all the data that you configure in AWS SSO is stored in the Region where you configured it. This data includes directory configurations, permission sets, application instances, and user assignments to AWS account applications. If you are using the AWS SSO identity store, all users and groups that you create in AWS SSO are also stored in the same Region.

AWS Organizations only supports one AWS SSO Region at a time. If you want to make AWS SSO available in a different Region, you must first delete your current AWS SSO configuration. Switching to a different Region also changes the URL for the user portal.

Delete your AWS SSO configuration

When an AWS SSO configuration is deleted, all the data in that configuration is deleted and cannot be recovered. The following table describes what data is deleted based on the directory type that you have currently configured in AWS SSO.

| What data gets deleted | Connected directory (AWS Managed Microsoft AD or AD Connector) | AWS SSO identity store |
|--|---|------------------------|
| All permission sets you have configured for AWS accounts | X | X |
| All applications you have configured in AWS SSO | X | X |
| All user assignments you have configured for AWS accounts and applications | X | X |
| All users and groups in the directory or store | N/A | X |

Use the following procedure when you need to delete your current AWS SSO configuration.

To delete your AWS SSO configuration

1. Open the [AWS SSO console](#).
2. In the left navigation pane, choose **Settings**.

3. On the **Settings** page, under **Delete AWS SSO configuration**, choose **Delete AWS SSO**.
4. On the **Delete AWS SSO configuration** page, select each of the check boxes to acknowledge you understand the data that will be deleted. Type **DELETE** in the text box, and then choose **Delete AWS SSO**.

AWS Single Sign-On quotas

The following tables describe quotas within AWS SSO.

Application quotas

| Resource | Default quota | Can be increased |
|---|---------------|------------------|
| File size of service provider SAML certificates (in PEM format) | 2 KB | No |

AWS account quotas

| Resource | Default quota | Can be increased |
|---|---------------|------------------|
| Number of permission sets allowed in AWS SSO | 500 | Yes |
| Number of permission sets allowed per AWS account | 50 | Yes |
| Number of inline policies per permission set | 1 | No |
| Maximum size of inline policy per permission set | 10,240 bytes | No |
| Number of IAM roles in the AWS account that can be repaired at a time | 1 | No |

Note

[Permission sets \(p. 9\)](#) are provisioned in AWS accounts as IAM roles and therefore follow IAM quotas. For more information about IAM quotas, see [IAM and STS quotas](#).

Active Directory quotas

| Resource | Default quota | Can be increased |
|---|---------------|------------------|
| Number of unique directory groups that can be assigned * | 2500 | Yes |
| Number of connected directories that you can have at a time | 1 | No |

* Users can belong to many directory groups, and a directory may contain many groups (see [AWS SSO identity store quotas \(p. 132\)](#)). Within AWS SSO, a maximum of 2500 of these groups can be assigned for using accounts and applications.

AWS SSO identity store quotas

| Resource | Default quota |
|--|---------------|
| Number of unique groups that can be assigned * | 100 |
| Number of users supported in AWS SSO | 50000 |
| Number of groups supported in AWS SSO | 10000 |

* Users within an AWS SSO store can have up to 100 of their groups assigned for using applications.

AWS SSO throttle limits

| Resource | Default quota |
|--------------|---|
| AWS SSO APIs | All AWS SSO APIs have a throttle limit maximum of 20 transactions per second (TPS). The CreateAccountAssignment has a maximum rate of 10 outstanding async calls. These quotas cannot be changed. |

Additional quotas

| Resource | Default quota | Can be increased |
|--|---------------|------------------|
| Total number of AWS accounts or applications that can be configured * | 500 | Yes |
| Number of unique groups that can be used to evaluate the permissions for a user ** | 1000 | No |

* Only 500 AWS accounts or applications (total combined) are supported. For example, you might configure 275 accounts and 225 applications, resulting in a total of 500 accounts and applications.

** Before displaying the user's available AWS accounts and application icons in the user portal, AWS SSO evaluates the user's effective permissions by evaluating their group memberships. Only 1000 unique groups can be used to determine a user's effective permissions.

Troubleshooting AWS SSO issues

The following can help you troubleshoot some common issues you might encounter while setting up or using the AWS SSO console.

Issues regarding contents of SAML assertions created by AWS SSO

AWS SSO provides a web-based debug experience for the SAML assertions created and sent by AWS SSO, including attributes within these assertions, when accessing AWS accounts and SAML applications from the user portal. To see the details of a SAML assertion that AWS SSO generates, use the following steps.

1. Sign in to the AWS SSO user portal.
2. While you are signed into the portal, hold the **Shift** key down, choose the application tile, and then release the **Shift** key. If accessing an AWS account, hold the **Shift** key down while choosing the **Management console** link for the desired account and permission set.
3. Examine the information on the page titled **You are now in administrator mode**. To keep this information for future reference, choose **Copy XML**, and paste the contents elsewhere.
4. Choose **Send to <application>** to continue. This option sends the assertion to the service provider.

Note

Some browser configurations and operating systems may not support this procedure. This procedure has been tested on Windows 10 using Firefox, Chrome, and Edge browsers.

Specific users fail to synchronize into AWS SSO from an external SCIM provider

If SCIM synchronization succeeds for a subset of users configured in your IdP for provisioning into AWS SSO but fails for others, you might see an error similar to 'Request is unparsable, syntactically incorrect, or violates schema' from your identity provider. You may also see detailed provisioning failure messages in AWS CloudTrail.

This issue often indicates that the user in your IdP is configured in a way that AWS SSO does not support. Full details of the AWS SSO SCIM implementation, including the specifications of required, optional, and forbidden parameters and operations for user objects, can be found in the [AWS SSO SCIM Implementation Developer Guide](#). The *SCIM Developer Guide* should be considered authoritative for information around SCIM requirements. However, the following are a couple of common reasons for this error:

1. The user object in the IdP lacks a first (given) name, a last (family) name, and/or a display name.
 - **Solution:** Add a first (given), last (family), and display name for the user object. In addition, ensure that the SCIM provisioning mappings for user objects at your IdP are configured to send nonempty values for all of these attributes.
2. More than one value for a single attribute is being sent for the user (also known as "multi-value attributes"). For example, the user may have both a work and a home phone number specified in

the IdP, or multiple emails or physical addresses, and your IdP is configured to try to synchronize multiple or all values for that attribute.

- **Solution options:**
 - i. Update your SCIM provisioning mappings for user objects at your IdP to send only a single value for a given attribute. For example, configure a mapping that sends only the work phone number for each user.
 - ii. If the additional attributes can safely be removed from the user object at the IdP, you can remove the additional values, leaving either one or zero values set for that attribute for the user.
 - iii. If the attribute is not needed for any actions in AWS, remove the mapping for that attribute from the SCIM provisioning mappings for user objects at your IdP.
- 3. Your IdP is trying to match users in the target (AWS SSO, in this case) based on multiple attributes. Since user names are guaranteed unique within a given AWS SSO instance, you only need to specify `username` as the attribute used for matching.
 - **Solution:** Ensure that your SCIM configuration in your IdP is using only a single attribute for matching with users in AWS SSO. For example, mapping `username` or `userPrincipalName` in the IdP to the `username` attribute in SCIM for provisioning to AWS SSO will be correct and sufficient for most implementations.

Users can't sign in when their user name is in UPN format

Users might not be able to sign in to the user portal based on the format they use to enter in their user name on the sign in page. For the most part, users can sign in to the user portal using either their plain user name, their down-level logon name (DOMAIN\UserName) or their UPN logon name (UserName@Corp.Example.com). The exception to this is when AWS SSO is using a connected directory that has been enabled with MFA and the verification mode has been set to either **Context-aware** or **Always-on**. In this scenario, users must sign in with their down-level logon name (DOMAIN\UserName). For more information, see [Multi-factor authentication \(p. 83\)](#). For general information about user name formats used to sign in to Active Directory, see [User Name Formats](#) on the Microsoft documentation website.

I get a 'Cannot perform the operation on the protected role' error when modifying an IAM role

When reviewing IAM Roles in an account, you may notice role names beginning with 'AWSReservedSSO_'. These are the roles which the AWS SSO service has created in the account, and they came from assigning a permission set to the account. Attempting to modify these roles from within the IAM console will result in the following error:

```
'Cannot perform the operation on the protected role 'AWSReservedSSO_RoleName_Here' - this role is only modifiable by AWS'
```

These roles can only be modified from the AWS SSO Administrator console, which is in the management account of AWS Organizations. Once modified, you can then push the changes down to the AWS accounts that it is assigned to.

Directory users cannot reset their password

When a directory user resets their password using the **Forgot Password?** option during sign-in of the user portal, their new password must adhere to the default password policy as described in [Password requirements for the AWS SSO identity store](#) (p. 18).

If a user enters a password that adheres to the policy and then receives the error `We couldn't update your password`, check to see if AWS CloudTrail recorded the failure. This can be done by searching in the Event History console of CloudTrail using the following filter:

```
"UpdatePassword"
```

If the message states the following, then you may need to contact support:

```
"errorCode": "InternalFailure",  
  "errorMessage": "An unknown error occurred"
```

Another possible cause of this issue is in the naming convention that was applied to the user name value. Naming conventions must follow specific patterns such as 'surname.givenName'. However, some user names can be quite long, or contain special characters, and this can cause characters to be dropped in the API call, thereby resulting in an error. You may want to attempt a password reset with a test user in the same manner to verify if this is the case.

If the issue persists, contact the [AWS Support Center](#).

My user is referenced in a permission set but can't access the assigned accounts or applications

This issue can occur if you're using System for Cross-domain Identity Management (SCIM) for Automatic Provisioning with an external identity provider. Specifically, when a user, or the group the user was a member of, is deleted then re-created using the same user name (for users) or name (for groups) in the identity provider, a new unique internal identifier is created for the new user or group in AWS SSO. However, AWS SSO still has a reference to the old identifier in its permission database, such that the name of the user or group still appears in the UI, but access fails. This is because the underlying user or group ID to which the UI refers no longer exists.

To restore AWS account access in this case, you can remove access for the old user or group from the AWS account(s) where it was originally assigned, and then reassign access back to the user or group. This updates the permission set with the correct identifier for the new user or group. Similarly, to restore application access, you can remove access for the user or group from the assigned users list for that application, then add the user or group back again.

You can also check to see if AWS CloudTrail recorded the failure by searching your CloudTrail logs for SCIM synchronization events that reference the name of the user or group in question.

I cannot get my cloud application configured correctly

Each service provider of a preintegrated cloud application in AWS SSO has its own detailed instruction manual. You can access the manual from the **Configuration** tab for that application in the AWS SSO console.

If the problem is related to setting up the trust between the service provider's application and AWS SSO, make sure to check the instruction manual for troubleshooting steps.

Error 'An unexpected error has occurred' when a user tries to sign in using an external identity provider

This error may occur for multiple reasons, but one common reason is a mis-match between the user information carried in the SAML request, and the information for the user in AWS SSO.

In order for an AWS SSO user to sign in successfully when using an external IdP as the identity source, the following must be true:

- The SAML nameID format (configured at your identity provider) must be 'email'
- The nameID value must be a properly (RFC2822)-formatted string (user@domain.com)
- The nameID value must exactly match the user name of an existing user in AWS SSO (it doesn't matter if the email address in AWS SSO matches or not – the inbound match is based on username)
- The following statements apply if [Attributes for access control \(p. 59\)](#) is enabled in your AWS SSO account:
 - The number of attributes mapped in the SAML request must be 50 or less.
 - The SAML request must not contain multi-valued attributes.
 - The SAML request must not contain multiple attributes with the same name.
 - The attribute must not contain structured XML as the value.
 - The Name format must be a SAML specified format, not generic format.

Note

AWS SSO does not perform "just in time" creation of users or groups for new users or groups via SAML federation. This means that the user must be pre-created in AWS SSO, either manually or via automatic provisioning, in order to sign in to AWS SSO.

This error can also occur when the Assertion Consumer Service (ACS) endpoint configured in your identity provider does not match the ACS URL provided by your AWS SSO instance. Ensure that these two values match exactly.

Additionally, you can troubleshoot external identity provider sign-in failures further by going to AWS CloudTrail and filtering on the event name **ExternalIdPDirectoryLogin**.

Error 'Attributes for access control failed to enable'

This error may occur if the user enabling ABAC does not have the `iam:UpdateAssumeRolePolicy` permissions required to enable [Attributes for access control \(p. 59\)](#).

I get a 'Browser not supported' message when I attempt to register a device for MFA

WebAuthn is currently supported in Google Chrome, Mozilla Firefox, Microsoft Edge and Apple Safari web browsers, as well as Windows 10 and Android platforms. Some components of WebAuthn support

may be varied, such as platform authenticator support across macOS and iOS browsers. If users attempt to register WebAuthn devices on an unsupported browser or platform, they will see certain options greyed out that are not supported, or they will receive an error that all supported methods are not supported. In these cases, please refer to [FIDO2: Web Authentication \(WebAuthn\)](#) for more information about browser/platform support. For more information about WebAuthn in AWS SSO, see [Security keys and built-in authenticators](#) (p. 85).

Active Directory “Domain Users” group does not properly sync into AWS SSO

The Active Directory Domain Users group is the default “primary group” for AD user objects. Active Directory primary groups and their memberships cannot be read by AWS SSO. When assigning access to AWS SSO resources or applications, use groups other than the Domain Users group (or other groups assigned as primary groups) to have group membership properly reflected in the AWS SSO identity store.

Invalid MFA credentials error

This error can occur when a user attempts to sign in to AWS SSO using an account from an external identity provider (for example, Okta or Azure AD) before their user account has been fully provisioned to AWS SSO using the SCIM protocol. Once the user account has been provisioned to AWS SSO, this issue should be resolved. Confirm that the user account has been provisioned to AWS SSO and, if not, check the provisioning logs in the external identity provider.

I get a 'An unexpected error has occurred' message when I attempt to register or sign in using an authenticator app

Time-based one-time password (TOTP) systems, such as those used by AWS SSO in combination with code-based authenticator apps, rely on time synchronization between the client and the server. Ensure that the device where your authenticator app is installed is correctly synchronized to a reliable time source, or manually set the time on your device to match a reliable source, such as NIST (<https://www.time.gov/>) or other local/regional equivalents.

Document history

The following table describes the documentation for this release of AWS Single Sign-On.

- **Latest documentation update:** December 21, 2020

| update-history-change | update-history-description | update-history-date |
|---|---|---------------------|
| Updates for quotas | Adjustments to quota tables. | December 21, 2020 |
| New example policies | Added new customer managed policy examples and updates to the permissions required section. | December 21, 2020 |
| Support for attribute-based access control (ABAC) | Added content for ABAC feature. | November 24, 2020 |
| Support for MFA forced enrollment | Updates to require users to enroll an MFA device at sign-in. | November 23, 2020 |
| Support for WebAuthn | Added content for new WebAuthn feature. | November 20, 2020 |
| Support for ping identity | Added content to integrate with Ping Identity products as a supported external identity provider. | October 26, 2020 |
| Support for OneLogin | Added content to integrate with OneLogin as a supported external identity provider. | July 31, 2020 |
| Support for Okta | Added content to integrate with Okta as a supported external identity provider. | May 28, 2020 |
| Support for external identity providers | Changed references from directory to identity source, added content to support external identity providers. | November 26, 2019 |
| New MFA settings | Removed two-step verification topic and added new MFA topic in it's place. | October 24, 2019 |
| New setting to add two-step verification | Added content on how to enable two-step verification for users. | January 16, 2019 |
| Support for session duration on AWS accounts | Added content on how to set the session duration for an AWS account. | October 30, 2018 |
| New option to use AWS SSO directory | Added content for choosing either AWS SSO directory or connecting to an existing AD directory. | October 17, 2018 |

| | | |
|--|---|-------------------|
| Support for relay state and session duration on applications | Added content about relay state and session duration for cloud applications. | October 10, 2018 |
| Additional support for new cloud applications | Added 4me, BambooHR, Bonusly, Citrix ShareFile, ClickTime, Convo, Deputy, Deskpro, Dome9, DruvalnSync, Egnyte, Engagedly, Expensify, Freshdesk, IdeaScale, Igloo, Jitbit, Kudos, LiquidFiles, Lucidchart, PurelyHR, Samanage, ScreenSteps, Sli.do, SmartSheet, Syncplicity, TalentLMS, Trello, UserVoice, Zoho, OpsGenie, DigiCert, WeekDone, ProdPad, and UserEcho to the application catalog. | August 3, 2018 |
| Support for SSO access to management accounts | Added content about how to delegate SSO access to users in a management account. | July 9, 2018 |
| Support for new cloud applications | Added DocuSign, Keeper Security, and SugarCRM to the application catalog. | March 16, 2018 |
| Get temporary credentials for CLI access | Added information about how to get temporary credentials to run AWS CLI commands. | February 22, 2018 |
| New guide | This is the first release of the AWS SSO User Guide. | December 7, 2017 |

AWS glossary

For the latest AWS terminology, see the [AWS glossary](#) in the *AWS General Reference*.