AWS Single Sign-On

User Guide

AWS Single Sign-On: User Guide

Table of Contents

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 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, enumerates all your AWS accounts, and if you have organized your accounts under organizational units (OUs) you will see them displayed that way within the AWS SSO console. This enables you to 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, 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.

Use 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 on-board 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.

Getting Started

In this getting started exercise, you enable AWS Single Sign-On, connect your directory, set up SSO to your AWS accounts, and finally set up SSO to your cloud applications. Although not required, we recommend that you review Understanding Key AWS Single Sign-On Concepts (p. 4) before you begin using the console so that you are familiar with the core features and terminology.

Topics

AWS SSO Prerequisites

Before you can set up AWS SSO, you must meet the following requirements:

- You must have first set up the AWS Organizations service and have **All features** set to enabled.
- You must sign-in with the AWS Organizations master 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.
- You must have an existing Microsoft Active Directory (AD) set up in AWS Directory Service and it must reside within your organization's master account. This AWS Microsoft AD directory determines which pool of users has SSO access to the user portal. You can connect only one AWS Microsoft AD directory at a time. However, you can change it to a different AWS Microsoft AD directory at any time. For more information, see Create a Microsoft AD Directory in the *AWS Directory Service Administration Guide*.
- Your connected directory must be in the US East (N. Virginia) (us-east-1) Region where AWS SSO is also available. AWS SSO stores the assignment data in the same region as the directory. To administer AWS SSO, you must be in the us-east-1 region. Also, note that AWS SSO's user portal uses the same access URL as your connected directory.

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 SSO. 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 is granted the necessary permissions to create IAM service-linked roles in any of the AWS accounts within your AWS organization. No service-linked roles are created at this time. AWS SSO creates these roles later during the process of setting up SSO access to your AWS accounts (see Set Up SSO to Your AWS Accounts (p. 3)).

To enable AWS SSO

1. Sign in to the AWS Management Console with your AWS Organizations master account credentials.
2. Open the AWS SSO console.
3. Choose **Enable AWS SSO**.

Connect Your Directory

Connecting an AWS Microsoft AD directory to AWS SSO provides users in that directory 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 and have one-click access to all applications and AWS accounts that you have previously authorized. Optionally, you can also connect an AWS Microsoft AD directory with your on-premises Active Directory.

Depending on which directory type you are trying to set up, follow one of the procedures below:

- Connect AWS SSO to an AWS Microsoft AD Directory (p. 8)
- Connect AWS SSO to an On-Premises Active Directory (p. 8)

For more information about supported directory types, see Manage Your Connected Directory (p. 8).

Set Up SSO to Your AWS Accounts

In this step, you can grant users in your connected directory with SSO access to one or more AWS consoles for specific AWS accounts in your AWS organization. Afterward, users see only the AWS account icon (for example, Development) that they've been assigned from within their user portal. When they click 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. 10).

Set Up SSO to Your Cloud Applications

Depending on which application type you are trying to set up, follow one of the procedures below:

- Add and Configure a Cloud Application (p. 14)
- Add and Configure a Custom SAML 2.0 Application (p. 15)

For more information about supported application types, see Manage SSO to Your Applications (p. 14).

Once you complete the appropriate procedure, 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

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

User Authentications

When a user signs in to the user portal using their username, AWS SSO redirects the request to the AWS SSO authentication service based on the domain associated with their 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.

Attribute Mappings

AWS SSO retrieves user attributes from your connected 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 connected directory. For more information, see Map Attributes in Your Application to AWS SSO Attributes (p. 17).

AWS SSO also manages a set of attributes for you under the **Attribute mappings** section of your connected directory configuration page. For more information, see Map Attributes in AWS SSO to Attributes in Your Connected Directory (p. 9).

Supported Directory Attributes

The following table provides the full list of connected directory attributes that are supported and which can be mapped to user attributes in AWS SSO.

Supported attributes in your connected directory
`${dir:email}`
`${dir:displayname}`
`${dir:distinguishedName}`
`${dir:firstname}`
`${dir:guid}`
`${dir:initials}`
`${dir:lastname}`
`${dir:proxyAddresses}`
`${dir:proxyAddresses:smtp}`
`${dir:proxyAddresses:SMTP}`
`${dir:windowsUpn}`

You can specify any combination of supported connected 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 and 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 provides the full list of AWS SSO attributes that are supported and which can be mapped to user attributes in your connected directory. Later, when you set up your application attribute mappings you will be able to use these same AWS SSO attributes to map to actual attributes used by that application.

Supported attributes in AWS SSO
`${user:AD_GUID}`
`${user:email}`
`${user:familyName}`
`${user:firstName}`
`${user:middleName}`

Supported attributes in AWS SSO
`${user:name}`
`${user:preferredUsername}`
`${user:subject}`

Default Mappings

The following table shows the default mappings for user attributes in AWS SSO to the user attributes in your connected 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 connected directory
`AD_GUID`	`${dir:guid}`
`email`	`${dir:windowsUpn}`
`familyName`	`${dir:lastname}`
`givenName`	`${dir:firstname}`
`middleName`	`${dir:initials}`
`name`	`${dir:displayname}`
`preferredUsername`	`${dir:displayname}`
`subject`	`${dir:windowsUpn}`

You can change the default mappings or add more attributes to the SAML assertion based on your requirements. For example, if your cloud application requires the users email in the `User.Email` SAML attribute, and emails are stored in the `windowsUpn` attribute in your connected directory. To achieve this mapping, you would need to make changes in the following two places in the AWS SSO console:

1. On the **Connected Directory** page, under the **Attribute mappings** section, you would need to map the user attribute **`email`** to the **`${dir:windowsUpn}`** attribute (in the **Maps to this attribute in your connected directory** column)
2. On the **Applications** page, choose the application from the table, choose the **Attribute mappings** tab, and then you would need to map the `User.Email` attribute to the **`${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 ${dir:**`AttributeName`**}. For example, the `firstname` attribute in your connected directory becomes `${dir:firstname}`.

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.

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 AWS organization. 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. 2) for the first time, the AWS Organizations service grants AWS SSO the necessary permissions to create IAM roles in any of its AWS accounts. AWS SSO doesn't create roles in any of the AWS accounts at this point. It only creates a service-linked role in an AWS account after you have used the AWS SSO console to specify which account you want to assign SSO access to. For more information, see Manage SSO to Your AWS Accounts (p. 10).

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. 27).

Manage Your Connected Directory

AWS Single Sign-On enables administrators to connect their on-premises Active Directory (AD) or their AWS Microsoft AD directory using AWS Directory Service. This connected 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 makes it easy for you to set up and run a standalone Microsoft AD directory hosted in the AWS Cloud, or to connect your AWS resources with an existing on-premises Microsoft Active Directory. To configure AWS Directory Service to work with your on-premises Active Directory, you first need to set up trust relationships to extend authentication from on-premises to the cloud.

Note
AWS SSO does not support SAMBA4-based Simple AD as a connected directory.

Topics

Connect AWS SSO to an AWS Microsoft AD Directory

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

To connect AWS SSO to AWS Microsoft AD

1. Open the AWS SSO console.

   Note
   Make sure that the AWS SSO console is using one of the same regions where your AWS Microsoft AD directory is located in before you move to the next step.
2. From the **Dashboard**, choose **Connect your directory**
3. On the **Connect your directory** page, do the following:

   a. Under **Available directories**, select the AWS Microsoft AD directory you want AWS SSO to connect to.

   b. Under **User portal URL**, type the prefix to use for the user portal sign-in URL.
4. Choose **Connect directory**.

Connect AWS SSO to an On-Premises Active Directory

If you want users in your on-premises Active Directory to also have SSO access to AWS accounts and cloud applications in the AWS SSO user portal, AWS Directory Service has the following two options available:

- **Create a two-way trust relationship** – Two-way trust relationships created between AWS Microsoft AD and an on-premises Active Directory enable on-premises users to sign in with their corporate credentials to various AWS services and business applications. One-way trusts will 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 on-premises Active Directory without caching any information in the cloud. For more information, see Connect to a Directory in the *AWS Directory Service Administration Guide*.

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

Disconnect a Directory

Use this procedure when you need to remove a directory from AWS SSO.

Note
All users and groups that have been assigned any permissions to AWS accounts or cloud applications will no longer be able to sign in.

To disconnect a directory from AWS SSO

1. Open the AWS SSO console.
2. Choose **Connected directory**.
3. Under Directory details, choose **Disconnect directory**.
4. On the Disconnect a directory dialog box, type the name of the directory to remove, and then choose **Remove all assignments and disconnect directory**.

Map Attributes in AWS SSO to Attributes in Your Connected Directory

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

To map attributes in AWS SSO to attributes in your connected directory

1. Open the AWS SSO console.
2. Choose **Connected directory**.
3. 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 connected directory attribute **`${dir:windowsUpn}`**.
5. Choose **Save changes**.

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. 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. 12).

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 centrally 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 then provides users with the ability to retrieve temporary credentials for that 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. 31).

To use AWS SSO with AWS Organizations, you must first Enable AWS SSO (p. 2), which grants AWS SSO the capability to create Service-Linked Roles (p. 7) in each account in your AWS organization. These roles are not created until after you Assign User Access (p. 10) 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. 15).

Topics

Single Sign-On Access

You can set up SSO access to one or more AWS accounts in your AWS organization by granting permissions to only those users who require it. You can assign users permissions to these AWS accounts based on common job functions or 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.

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 US East (N. Virginia) (us-east-1) Region where your AWS 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 an account to which you want to assign access.

   Note
   If you see a check box in the table that is disabled, this represents the master account for your AWS organization and is by design. Please see the next step for more details.
4. On the AWS account details page, choose **Assign users**.

   Note
   If the **Assign users** button is disabled it indicates that this account is the master account for your AWS organization. You can only assign SSO access to users in member accounts. For more information about the different account types, see AWS Organizations Terminology and Concepts in the *AWS Organizations User Guide*.
5. On the **Select users or groups** page, type a user or group name and choose **Search connected directory**. Once you have selected all the accounts that you want to assign access to, choose **Next: Permission sets**. You can specify multiple users or groups by selecting the applicable accounts as they appear in search results.
6. On the **Select permission sets** page, select the permission sets that you want to apply to the user or group from the table. Then choose **Finish**. You can optionally choose to **Create a new permission set** if none of the permissions in the table meet your needs. For detailed instructions, see Create Permission Set (p. 12).
7. Choose **Finish** to begin the process of configuring your AWS account.

   Note
   If this is the first time you have assigned SSO access to this AWS account, this process creates a service-linked role in the account. For more information, see Using Service-Linked Roles for AWS SSO (p. 27).

   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.

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**.

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. 6).

Create 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. In the **Create new permission set** dialog box, choose from one of the following options, and then follow the instructions provided under that option:
    - **Use an existing job function policy**
        1. Under **Select job function policy**, select one of the default IAM job function policies in the list. For more information, see AWS Managed Policies for Job Functions.
        2. Choose **Create**.
    - **Create a custom permission set**
        1. 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.
        2. (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.
        3. 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.
        4. If you chose **Attach AWS managed policies**, under **Attach AWS Managed policies**, select up to 10 job-related or service-specific AWS managed policies from the list.
        5. 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. 24).

           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.
        6. Choose **Create**.

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**.

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**.

Manage SSO to Your Applications

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

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 using the AWS SSO console and configure it with the appropriate metadata for both the AWS SSO service and the service provider.

Once the application has been successfully added to the AWS SSO console, you can then 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:

- 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. 10).

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

Topics

Cloud Applications

AWS SSO has built-in support for commonly used cloud applications, such as Adobe Creative Cloud, AppDynamics, Box, Confluence, Domo, Dropbox, G Suite, GitHub, GoToMeeting, Jira, NewRelic, Office 365, PagerDuty, Salesforce, ServiceNow, Slack, SumoLogic, Tableau, Workplace by Facebook, ZenDesk, and Zoom.

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. Once the application has been configured, you can then assign access to the groups or users that require it.

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 nav, and then choose **Add a new application**.
2. In the **Select an application** dialog box, select the application you want to add from the list, and then choose **Add**.
3. On the **Configure <application name>** page, under **Details**, type a **Display name** for the application. For example, `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. Under **Application metadata**, next to **Application certificate**, choose **Browse** to upload the service provider's certificate. This certificate is required to establish a secure connection between AWS SSO and the service provider.
6. Choose **Save changes** to save the configuration.

Custom SAML 2.0 Applications

AWS SSO also supports applications that allow identity federation using 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 so that AWS SSO knows how to populate the SAML assertion correctly for your application. You can provide this additional SAML attribute mapping when you are setting 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, and then choose **Configure application**.
3. On the **Configure <Custom app name>** page, under **Details**, type a **Display name** for the application. For example, `MyApp`.

4. Under **AWS SSO metadata**, do the following:

   a. Next to **AWS SSO SAML metadatafile**, click **Download** to download the identity provider metadata.

   b. Next to **AWS SSO certificate**, click **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. Under **Application metadata**, next to **Application certificate**, choose **Browse** to upload the service provider's certificate. This certificate is required to establish a secure connection between AWS SSO and the service provider.
6. Choose **Save changes** to save the configuration.

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 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 US East (N. Virginia) Region where your AWS Microsoft AD directory is located before you move to 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, select the **Assigned users** tab. Then choose **Assign users**.
5. In the **Assign users** dialog box, type 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, select the **Assigned users** tab, select the user or group 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, select the **Attribute mappings** tab.
5. Choose **Add new attribute mapping**
6. In the first text box, type the application attribute.
7. In the second text box, type 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. 4).
8. In the third column of the table, select the appropriate format for the attribute from the menu.
9. Choose **Save changes**.

Authentication and Access Control 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. 10).
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. 12).

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. 18)
- Access Control (p. 19)

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. It is similar to an *IAM user*, but it is not associated with a specific person. An IAM role enables you to obtain temporary access keys that can be used to access AWS services and resources. IAM roles with temporary credentials are useful in the following situations:

  - **Federated user access** – Instead of creating an IAM user, you can use existing user 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** – You can use an IAM role in your account to grant an AWS service permissions to access your account's resources. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data from that bucket into an Amazon Redshift cluster. 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 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. 19)
- Using Identity-Based Policies (IAM Policies) for AWS SSO (p. 22)
- Using Service-Linked Roles for AWS SSO (p. 27)

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

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)

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. 22). 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. 20)), 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. 20).
- **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 IAM identities (that is, users, groups, and roles).

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. 19).

The sections in this topic cover the following:

- Permissions Required to Use the AWS SSO Console (p. 23)
- AWS Managed (Predefined) Policies for AWS SSO (p. 23)
- Customer Managed Policy Examples (p. 24)

The following shows an example of a permissions policy.

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Action" : [
        "sso:CreateApplicationInstance",
        "sso:UpdateResponseConfig",
        "sso:UpdateResponseSchemaConfig",
        "sso:UpdateSecurityConfig",
        "sso:UpdateServiceProviderConfig",
        "sso:UpdateApplicationInstanceStatus",
        "sso:UpdateApplicationInstanceDisplay",
        "sso:CreateProfile",
        "sso:SetupTrust"
      ],
      "Effect" : "Allow",
      "Resource" : "*"
    },

    {
      "Action" : [
        "organizations:xxx",
        "organizations:yyy"
       ],
      "Effect" : "Allow",
      "Resource" : "*"
    },

    {
      "Action" : [
        "ds:AuthorizeApplication"
      ],
      "Effect" : "Allow",
      "Resource" : "*"
    }
  ]
}
```

The policy includes the following:

- The first statement grants permission to manage profile associations to users and groups within your directory. It also grants permission to read all of the AWS SSO resources.
- The second statement grants permissions to search the directory for users and groups. This is required before you can create profile associations.

The policy doesn't specify the `Principal` element because in an identity-based policy you don't specify the principal who gets the permission. When you attach a policy to a user, the user is the implicit principal. When you attach a permission policy to an IAM role, the principal identified in the role's trust policy gets the permissions.

Permissions Required to Use the AWS SSO Console

For a user to work with the AWS SSO console, that user must have permissions listed in the preceding policy.

If you create an IAM policy that is more restrictive than the minimum required permissions, the console won't function as intended for users with that IAM policy.

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*.

Customer Managed Policy Examples

In this section, you can find example user policies that grant permissions for various AWS SSO actions.

Examples

Example 1: Allow a User to Set Up and Enable AWS SSO

The following permissions policy grants permissions to allow a user to open the AWS SSO console and enable the service. In order to do so, permissions such as those granted to the AWS Organizations master account are also required.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action": [
                sso:StartSSO,
                sso:GetSSOStatus
                  ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action": [
                organizations:DescribeAccount,
                organizations:EnableAWSServiceAccess
                  ],
         "Resource":"*"

      }
   ]
}
```

Example 2: Allow a User to Manage Your AWS SSO Connected Directory

The following permissions policy grants permissions to a user to manage your connected directory.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action": [
                sso:AssociateDirectory,
                sso:DisassociateDirectory,
                sso:ListDirectoryAssociations,
                sso:UpdateDirectoryAssociation
                ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action": [
                ds:DescribeDirectories
                  ],
         "Resource":"*"
      }
   ]
}
```

Example 3: Allow a User to Manage Applications in AWS SSO

The following permissions policy grants permissions to allow a user to create and manage application instances, profiles, and certificates in the AWS SSO console.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action": [
                sso:ListApplicationTemplates,
                sso:GetApplicationTemplate
                sso:ListApplicationInstances,
                sso:GetApplicationInstance,
                sso:CreateApplicationInstance,
                sso:UpdateApplicationInstanceStatus,
                sso:UpdateApplicationInstanceDisplayData,
                sso:UpdateApplicationInstanceServiceProviderConfiguration,
                sso:UpdateApplicationInstanceResponseConfiguration,
                sso:UpdateApplicationInstanceResponseSchemaConfiguration,
                sso:UpdateApplicationInstanceSecurityConfiguration,
                sso:DeleteApplicationInstance,
                sso:ImportApplicationInstanceServiceProviderMetadata,
                sso:CreateProfile,
                sso:UpdateProfile,
                sso:DeleteProfile,
                sso:GetProfile,
                sso:ListProfiles,
                sso:ListApplicationInstanceCertificates,
                sso:CreateApplicationInstanceCertificate,
                sso:UpdateApplicationInstanceActiveCertificate,
                sso:DeleteApplicationInstanceCertificate
                  ],
         "Resource":"*"
      }
   ]
}
```

Example 4: Allow a User to Manage Permissions for Your AWS Accounts in AWS SSO

The following permissions policy grants permissions to allow a user to create and manage permission sets for your AWS accounts in the AWS SSO console.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action": [
                sso:ListApplicationInstances,
                sso:GetApplicationInstance,
                sso:CreateApplicationInstance,
                sso:UpdateApplicationInstanceStatus,
                sso:UpdateApplicationInstanceDisplayData,
                sso:UpdateApplicationInstanceServiceProviderConfiguration,
                sso:UpdateApplicationInstanceResponseConfiguration,
                sso:UpdateApplicationInstanceResponseSchemaConfiguration,
                sso:UpdateApplicationInstanceSecurityConfiguration,
                sso:DeleteApplicationInstance,
                sso:ImportApplicationInstanceServiceProviderMetadata,
                sso:CreateProfile,
                sso:UpdateProfile,
                sso:DeleteProfile,
                sso:GetProfile,
                sso:ListProfiles,
                sso:ListApplicationInstanceCertificates,
                sso:CreateApplicationInstanceCertificate,
                sso:UpdateApplicationInstanceActiveCertificate,
                sso:DeleteApplicationInstanceCertificate,
                sso:CreatePermissionSet,
                sso:GetPermissionSet,
                sso:ListPermissionSets,
                sso:DeletePermissionSet,
                sso:PutPermissionsPolicy,
                sso:DeletePermissionsPolicy,
                sso:DescribePermissionsPolicies,
                sso:GetTrust,
                sso:CreateTrust,
                sso:UpdateTrust,
                sso:DeleteTrust
                  ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action": [
                organizations:DescribeOrganization
                  ],
         "Resource":"*"
      }
   ]
}
```

Example 5: Allow a User to Manage Access for Your Applications in AWS SSO

The following permissions policy grants permissions to allow a user to manage who can access your applications in the AWS SSO console.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action": [
                sso:ListApplicationInstances,
                sso:ListProfileAssociations,
                sso:AssociateProfile,
                sso:DisassociateProfile
                ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action": [
                ds:DescribeDirectories
                  ],
         "Resource":"*"
      }
   ]
}
```

Example 6: Allow a User to Find Which Cloud Applications Are Preintegrated with AWS SSO

The following permissions policy grants permissions to allow a user to locate what cloud applications are preintegrated with AWS SSO using the Add Application wizard.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action": [
                sso:ListApplicationTemplates,
                sso:GetApplicationTemplate
                  ],
         "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. 7).

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. When a user who is signed in with the AWS organization's master account assigns access to an AWS account for the first time, AWS SSO creates the service-linked role automatically in that AWS account.

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. 11) for all users and groups that have access to the AWS account.
2. Delete Permission Sets (p. 12) that you have associated with the AWS account.
3. Remove the IAM Identity Provider (p. 13) 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*.

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

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 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.

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

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. Type the name of the application. Then press **Enter**.

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. Credentials retrieved through the user portal are valid for 60 minutes. 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 Getting Temporary Credentials with AWS STS 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.

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, including how to configure and enable it, see the AWS CloudTrail User Guide.

AWS SSO Information in CloudTrail

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 actions are supported:

- `AssociateDirectory`
- `AssociateProfile`
- `CreateApplicationInstance`
- `CreateApplicationInstanceCertificate`
- `CreatePermissionSet`
- `CreateProfile`
- `DeleteApplicationInstance`
- `DeleteApplicationInstanceCertificate`
- `DeletePermissionsPolicy`
- `DeletePermissionSet`
- `DeleteProfile`
- `DescribePermissionsPolicies`
- `DisassociateDirectory`
- `DisassociateProfile`
- `GetApplicationInstance`
- `GetApplicationTemplate`
- `GetPermissionSet`
- `GetSSOStatus`
- `ImportApplicationInstanceServiceProviderMetadata`
- `ListApplicationInstances`
- `ListApplicationInstanceCertificates`
- `ListApplicationTemplates`
- `ListDirectoryAssociations`

- `ListPermissionSets`
- `ListProfileAssociations`
- `ListProfiles`
- `PutPermissionsPolicy`
- `StartSSO`
- `UpdateApplicationInstanceActiveCertificate`
- `UpdateApplicationInstanceDisplayData`
- `UpdateApplicationInstanceServiceProviderConfiguration`
- `UpdateApplicationInstanceStatus`
- `UpdateApplicationInstanceResponseConfiguration`
- `UpdateApplicationInstanceResponseSchemaConfiguration`
- `UpdateApplicationInstanceSecurityConfiguration`
- `UpdateDirectoryAssociation`
- `UpdateProfile`

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":"AKIAIIJM2K4example",
            "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"
      }
   ]
}
```

Limits in AWS SSO

The following tables describe limits within AWS SSO. For information about limits that can be changed, see AWS Service Limits.

Application Limits

Resource	Default Limit
File size of service provider SAML certificates (in PEM format)	2 kb
Number of unique Active Directory groups that can be assigned *	50
Number of connected directories that you can have at a time	1

* Users within their Active Directory can belong to many directory groups. However within AWS SSO, they can have up to 50 of their Active Directory groups assigned for using applications.

AWS Account Limits

Resource	Default Limit
Maximum number of permission sets in AWS SSO	50
Number of permission sets allowed per AWS account	20
Number of references to AWS managed policies per permission set	10
Number of inline policies per permission set	1
Maximum size of inline policy per permission set	10,000 bytes
Number of IAM roles in the AWS account that can be repaired at a time *	1

* Permission sets are provisioned in an AWS account as IAM roles. For more information, see Permission Sets (p. 6).

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.

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.

I don't know what data is in my SAML assertion that would be passed to the service provider

Use the following steps in the user portal to view what data in the SAML assertion will be sent to the application's service provider for the currently signed-in user. This procedure displays the contents in the browser window before sending it to the provider.

1. While you are signed into the portal, hold the **Shift** key and then choose the application.
2. Examine the information on the page titled **You are now in administrator mode**.
3. If the information looks good, you can choose **Send to <application>** to send the assertion to the service provider and review the outcome of the response.

Document History

The following table describes the documentation for this release of AWS Single Sign-On.

- **Latest documentation update:** December 7, 2017

Change	Description	Date
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*.