AWS Organizations

User Guide

AWS Organizations: User Guide

Table of Contents

What is AWS Organizations?

AWS Organizations is an account (p. 6) management service that enables you to consolidate multiple AWS accounts into an *organization* that you create and centrally manage. AWS Organizations includes account management and consolidated billing capabilities that enable you to better meet the budgetary, security, and compliance needs of your business. As an administrator of an organization, you can create accounts in your organization and invite existing accounts to join the organization.

This user guide defines key concepts for AWS Organizations, provides tutorials, and explains how to create and manage an organization.

Topics

AWS Organizations features

AWS Organizations offers the following features:

Centralized management of all of your AWS accounts

You can combine your existing accounts into an organization that enables you to manage the accounts centrally. You can create accounts that automatically are a part of your organization, and you can invite other accounts to join your organization. You also can attach policies that affect some or all of your accounts.

Consolidated billing for all member accounts

Consolidated billing is a feature of AWS Organizations. You can use the management account of your organization to consolidate and pay for all member accounts. In consolidated billing, management accounts can also access the billing information, account information, and account activity of member accounts in their organization. This information may be used for services such as Cost Explorer, which can help management accounts improve their organization’s cost performance.

Hierarchical grouping of your accounts to meet your budgetary, security, or compliance needs

You can group your accounts into organizational units (OUs) and attach different access policies to each OU. For example, if you have accounts that must access only the AWS services that meet certain regulatory requirements, you can put those accounts into one OU. You then can attach a policy to that OU that blocks access to services that do not meet those regulatory requirements. You can nest OUs within other OUs to a depth of five levels, providing flexibility in how you structure your account groups.

Policies to centralize control over the AWS services and API actions that each account can access

As an administrator of the management account of an organization, you can use service control policies (SCPs) to specify the maximum permissions for member accounts in the organization. In SCPs, you can restrict which AWS services, resources, and individual API actions the users and roles in each member account can access. You can also define conditions for when to restrict access to AWS services, resources, and API actions. These restrictions even override the administrators of member accounts in the organization. When AWS Organizations blocks access to a service, resource, or API action for a member account, a user or role in that account can't access it. This block remains in effect even if an administrator of a member account explicitly grants such permissions in an IAM policy.

For more information, see Service control policies (SCPs) (p. 104).

Policies to standardize tags across the resources in your organization's accounts

You can use tag policies to maintain consistent tags, including the preferred case treatment of tag keys and tag values.

For more information, see Tag policies (p. 187)

Policies to control how AWS artificial intelligence (AI) and machine learning services can collect and store data.

You can use AI services opt-out policies to opt out of data collection and storage for any of the AWS AI services that you don't want to use.

For more information, see AI services opt-out policies (p. 139)

Policies that configure automatic backups for the resources in your organization's accounts

You can use backup policies to configure and automatically apply AWS Backup plans to resources across all your organization's accounts.

For more information, see Backup policies (p. 154)

Integration and support for AWS Identity and Access Management (IAM)

IAM provides granular control over users and roles in individual accounts. AWS Organizations expands that control to the account level by giving you control over what users and roles in an account or a group of accounts can do. The resulting permissions are the logical intersection of what is allowed by AWS Organizations at the account level and the permissions that are explicitly granted by IAM at the user or role level within that account. In other words, the user can access only what is allowed by ***both*** the AWS Organizations policies and IAM policies. If either blocks an operation, the user can't access that operation.

Integration with other AWS services

You can leverage the multi-account management services available in AWS Organizations with select AWS services to perform tasks on all accounts that are members of an organization. For a list of services and the benefits of using each service on an organization-wide level, see AWS services that you can use with AWS Organizations (p. 227).

When you enable an AWS service to perform tasks on your behalf in your organization's member accounts, AWS Organizations creates an IAM service-linked role for that service in each member account. The service-linked role has predefined IAM permissions that allow the other AWS service to perform specific tasks in your organization and its accounts. For this to work, all accounts in an organization automatically have a service-linked role. This role enables the AWS Organizations service to create the service-linked roles required by AWS services for which you enable trusted access. These additional service-linked roles are attached to IAM permission policies that enable the specified service to perform only those tasks that are required by your configuration choices. For more information, see Using AWS Organizations with other AWS services (p. 223).

Global access

AWS Organizations is a global service with a single endpoint that works from any and all AWS Regions. You don't need to explicitly select a region to operate in.

Data replication that is eventually consistent

AWS Organizations, like many other AWS services, is eventually consistent. AWS Organizations achieves high availability by replicating data across multiple servers in AWS data centers within its

Region. If a request to change some data is successful, the change is committed and safely stored. However, the change must then be replicated across the multiple servers. For more information, see Changes that I make aren't always immediately visible (p. 348).

Free to use

AWS Organizations is a feature of your AWS account offered at no additional charge. You are charged only when you access other AWS services from the accounts in your organization. For information about the pricing of other AWS products, see the Amazon Web Services pricing page.

AWS Organizations pricing

AWS Organizations is offered at no additional charge. You are charged only for AWS resources that users and roles in your member accounts use. For example, you are charged the standard fees for Amazon EC2 instances that are used by users or roles in your member accounts. For information about the pricing of other AWS services, see AWS Pricing.

Accessing AWS Organizations

You can work with AWS Organizations in any of the following ways:

AWS Management Console

The AWS Organizations console is a browser-based interface that you can use to manage your organization and your AWS resources. You can perform any task in your organization by using the console.

AWS Command Line Tools

With the AWS command line tools, you can issue commands at your system's command line to perform AWS Organizations and AWS tasks. Working with the command line can be faster and more convenient than using the console. The command line tools also are useful if you want to build scripts that perform AWS tasks.

AWS provides two sets of command line tools:

- AWS Command Line Interface (AWS CLI). For information about installing and using the AWS CLI, see the AWS Command Line Interface User Guide.
- AWS Tools for Windows PowerShell. For information about installing and using the Tools for Windows PowerShell, see the AWS Tools for Windows PowerShell User Guide.

AWS SDKs

The AWS SDKs consist of libraries and sample code for various programming languages and platforms (for example, Java, Python, Ruby, .NET, iOS, and Android). The SDKs take care of tasks such as cryptographically signing requests, managing errors, and retrying requests automatically. For more information about the AWS SDKs, including how to download and install them, see Tools for Amazon Web Services.

AWS Organizations HTTPS Query API

The AWS Organizations HTTPS Query API gives you programmatic access to AWS Organizations and AWS. The HTTPS Query API lets you issue HTTPS requests directly to the service. When you use the HTTPS API, you must include code to digitally sign requests using your credentials. For more information, see Calling the API by Making HTTP Query Requests and the AWS Organizations API Reference.

Support and feedback for AWS Organizations

We welcome your feedback. You can send your comments to feedback-awsorganizations@amazon.com. You also can post your feedback and questions in AWS Organizations support forum. For more information about the AWS Support forums, see Forums Help.

Other AWS resources

- **AWS Training and Courses** – Links to role-based and specialty courses as well as self-paced labs to help sharpen your AWS skills and gain practical experience.
- **AWS Developer Tools** – Links to developer tools and resources that provide documentation, code examples, release notes, and other information to help you build innovative applications with AWS.
- **AWS Support Center** – The hub for creating and managing your AWS Support cases. Also includes links to other helpful resources, such as forums, technical FAQs, service health status, and AWS Trusted Advisor.
- **AWS Support** – The primary webpage for information about AWS Support, a one-on-one, fast-response support channel to help you build and run applications in the cloud.
- **Contact Us** – A central contact point for inquiries concerning AWS billing, account, events, abuse, and other issues.
- **AWS Site Terms** – Detailed information about our copyright and trademark; your account, license, and site access; and other topics.

Getting started with AWS Organizations

The following topics provide information to help you start learning and using AWS Organizations.

Learn about ...

AWS Organizations terminology and concepts (p. 5)

Learn the terminology and core concepts needed to understand AWS Organizations. This section describes each of the components of an organization and the basics of how they work together to provide a new level of control over what users in those accounts can do.

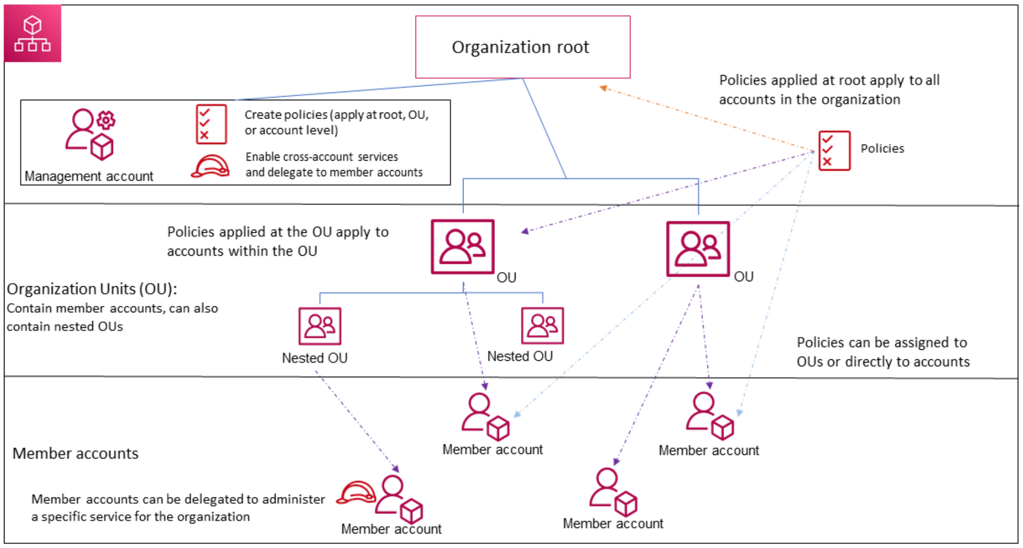
Consolidated Billing for Organizations

One of the primary features about AWS Organizations is the consolidation of the billing of all of the accounts in your organization. Learn more about how billing is handled in an organization and how various discounts work when shared across multiple accounts. This content is in the *AWS Billing User Guide*.

AWS Organizations terminology and concepts

To help you get started with AWS Organizations, this topic explains some of the key concepts.

The following diagram shows a basic organization that consists of five accounts that are organized into four organizational units (OUs) under the root. The organization also has several policies that are attached to some of the OUs or directly to accounts. For a description of each of these items, refer to the definitions in this topic.


Organization root
Policies applied at root apply to all accounts in the organization
Management account
Create policies (apply at root, OU, or account level)
Enable cross-account services and delegate to member accounts
Policies
Policies applied at the OU apply to accounts within the OU
OU
OU
Organization Units (OU):
Contain member accounts, can also contain nested OUs
Nested OU
Nested OU
Policies can be assigned to OUs or directly to accounts
Member account
Member account
Member accounts
Member accounts can be delegated to administer a specific service for the organization
Member account
Member account

Organization

An entity that you create to consolidate your AWS accounts (p. 6) so that you can administer them as a single unit. You can use the AWS Organizations console to centrally view and manage all of your accounts within your organization. An organization has one management account along with zero or more member accounts. You can organize the accounts in a hierarchical, tree-like structure with a root (p. 6) at the top and organizational units (p. 6) nested under the root. Each account can be directly in the root, or placed in one of the OUs in the hierarchy. An organization has the functionality that is determined by the feature set (p. 7) that you enable.

Root

The parent container for all the accounts for your organization. If you apply a policy to the root, it applies to all organizational units (OUs) (p. 6) and accounts (p. 6) in the organization.

> **Note**
> Currently, you can have only one root. AWS Organizations automatically creates it for you when you create an organization.

Organizational unit (OU)

A container for accounts (p. 6) within a root (p. 6). An OU also can contain other OUs, enabling you to create a hierarchy that resembles an upside-down tree, with a root at the top and branches of OUs that reach down, ending in accounts that are the leaves of the tree. When you attach a policy to one of the nodes in the hierarchy, it flows down and affects all the branches (OUs) and leaves (accounts) beneath it. An OU can have exactly one parent, and currently each account can be a member of exactly one OU.

Account

An account in Organizations is a standard AWS account that contains your AWS resources and the identities that can access those resources.

> **Tip**
> An AWS account isn't the same thing as a user account. An AWS user is an identity that you create using AWS Identity and Access Management (IAM) and takes the form of either an IAM user with long-term credentials, or an IAM role with short-term credentials. A single AWS account can, and typically does contain many users and roles.

There are two types of accounts in an organization: a single account that is designated as the management account, and one or more member accounts.

- The **management account** is the account that you use to create the organization. From the organization's management account, you can do the following:
  - Create accounts in the organization
  - Invite other existing accounts to the organization
  - Remove accounts from the organization
  - Manage invitations
  - Apply policies to entities (roots, OUs, or accounts) within the organization
  - Enable integration with supported AWS services to provide service functionality across all of the accounts in the organization.

  The management account has the responsibilities of a *payer account* and is responsible for paying all charges that are accrued by the member accounts. You can't change an organization's management account.
- **Member accounts** make up all of the rest of the accounts in an organization. An account can be a member of only one organization at a time. You can attach a policy to an account to apply controls to only that one account.

Invitation

The process of asking another account (p. 6) to join your organization (p. 6). An invitation can be issued only by the organization's management account. The invitation is extended to either the account ID or the email address that is associated with the invited account. After the invited account accepts an invitation, it becomes a member account in the organization. Invitations also can be sent to all current member accounts when the organization needs all members to approve the change from supporting only consolidated billing (p. 7) features to supporting all features (p. 7) in the organization. Invitations work by accounts exchanging handshakes (p. 7). You might not see handshakes when you work in the AWS Organizations console. But if you use the AWS CLI or AWS Organizations API, you must work directly with handshakes.

Handshake

A multi-step process of exchanging information between two parties. One of its primary uses in AWS Organizations is to serve as the underlying implementation for invitations (p. 7). Handshake messages are passed between and responded to by the handshake initiator and the recipient. The messages are passed in a way that helps ensure that both parties know what the current status is. Handshakes also are used when changing the organization from supporting only consolidated billing (p. 7) features to supporting all features (p. 7) that AWS Organizations offers. You generally need to directly interact with handshakes only if you work with the AWS Organizations API or command line tools such as the AWS CLI.

Available feature sets

- **All features** – The default feature set that is available to AWS Organizations. It includes all the functionality of consolidated billing, plus advanced features that give you more control over accounts in your organization. For example, when all features are enabled the management account of the organization has full control over what member accounts can do. The management account can apply SCPs (p. 104) to restrict the services and actions that users (including the root user) and roles in an account can access. The management account can also prevent member accounts from leaving the organization. You can also enable integration with supported AWS services to let those service provide functionality across all of the accounts in your organization.

  You can create an organization with all features already enabled, or you can enable all features in an organization that originally supported only the consolidated billing features. To enable all features, all invited member accounts must approve the change by accepting the invitation that is sent when the management account starts the process.

- **Consolidated billing** – This feature set provides shared billing functionality, but doesn't include the more advanced features of AWS Organizations. For example, you can't enable other AWS services to integrate with your organization to work across all of its accounts, or use policies to restrict what users and roles in different accounts can do. To use the advanced AWS Organizations features, you must enable all features (p. 7) in your organization.

Service control policy (SCP)

A policy that specifies the services and actions that users and roles can use in the accounts that the SCP (p. 104) affects. SCPs are similar to IAM permissions policies except that they don't grant any permissions. Instead, SCPs specify the maximum permissions for an organization, organizational unit (OU), or account. When you attach an SCP to your organization root or an OU, the SCP limits permissions for entities in member accounts.

Allow lists vs. deny lists

Allow lists and deny lists are complementary strategies that you can use to apply SCPs (p. 104) to filter the permissions that are available to accounts.

- **Allow list strategy** – You explicitly specify the access that ***is*** allowed. All other access is implicitly blocked. By default, AWS Organizations attaches an AWS managed policy called `FullAWSAccess` to all roots, OUs, and accounts. This helps ensure that, as you build your organization, nothing is blocked until you want it to be. In other words, by default all permissions are allowed. When you are ready to restrict permissions, you ***replace*** the `FullAWSAccess` policy with one that allows only the more limited, desired set of permissions. Users and roles in the affected accounts can then exercise only that level of access, even if their IAM policies allow all actions. If you replace the default policy on the root, all accounts in the organization are affected by the restrictions. You can't add permissions back at a lower level in the hierarchy because an SCP never grants permissions; it only filters them.

- **Deny list strategy** – You explicitly specify the access that isn't allowed. All other access is allowed. In this scenario, all permissions are allowed unless explicitly blocked. This is the default behavior of AWS Organizations. By default, AWS Organizations attaches an AWS managed policy called `FullAWSAccess` to all roots, OUs, and accounts. This allows any account to access any service or operation with no AWS Organizations–imposed restrictions. Unlike the allow list technique described above, when using deny lists, you leave the default `FullAWSAccess` policy in place (that allow "all"). But then you attach additional policies that explicitly ***deny*** access to the unwanted services and actions. Just as with IAM permission policies, an explicit deny of a service action overrides any allow of that action.

Artificial intelligence (AI) services opt-out policy

A type of policy that helps you standardize your opt-out settings for AWS AI services across all of the accounts in your organization. Certain AWS AI services can store and use customer content processed by those services for the development and continuous improvement of Amazon AI services and technologies. As an AWS customer, you can use AI service opt-out policies (p. 139) to choose to opt out of having your content stored or used for service improvements.

Backup policy

A type of policy that helps you standardize and implement a backup strategy for the resources across all of the accounts in your organization. In a backup policy (p. 154), you can configure and deploy backup plans for your resources.

Tag policy

A type of policy that helps you standardize tags across resources across all of the accounts in your organization. In a tag policy (p. 187), you can specify tagging rules for specific resources.

AWS Organizations tutorials

Use the tutorials in this section to learn how to perform tasks using AWS Organizations.

Tutorial: Creating and configuring an organization (p. 9)

> Get up and running with step-by-step instructions to create your organization, invite your first member accounts, create an OU hierarchy that contains your accounts, and apply some service control policies (SCPs).

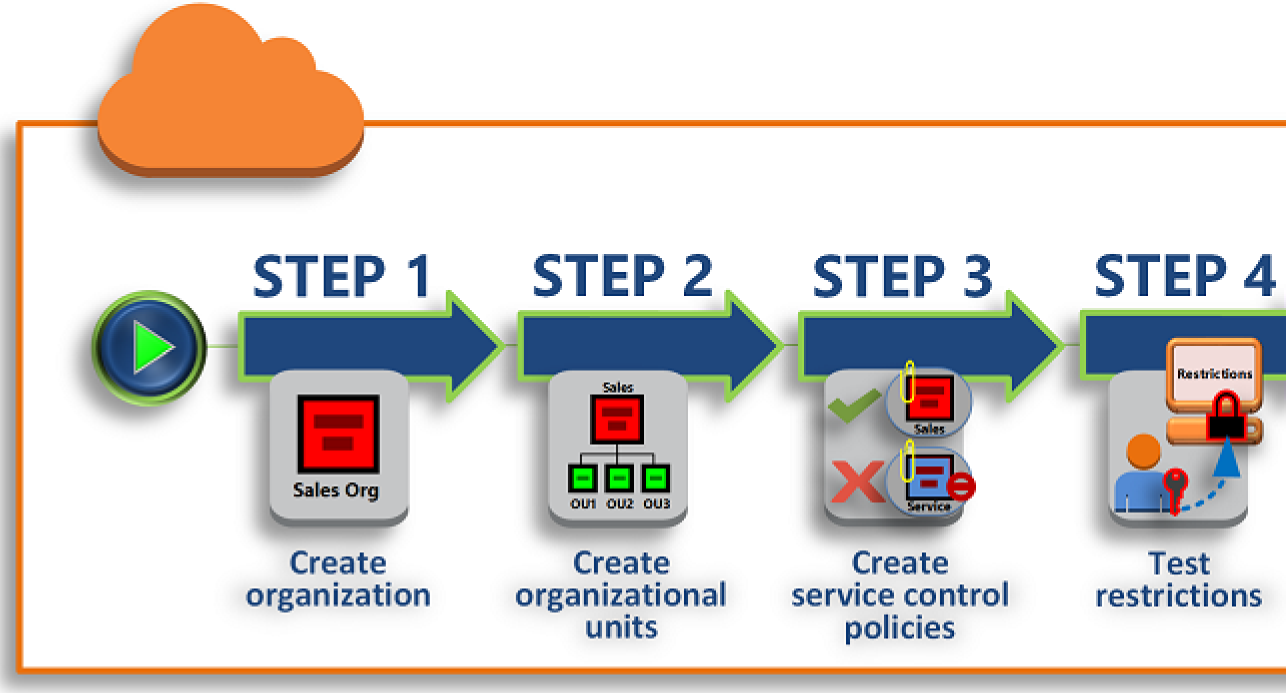
Tutorial: Monitor important changes to your organization with CloudWatch Events (p. 18)

> Monitor key changes in your organization by configuring Amazon CloudWatch Events to trigger an alarm in the form of an email, SMS text message, or log entry when actions that you designate occur in your organization. For example, many organizations want to know when a new account is created or when an account attempts to leave the organization.

Tutorial: Creating and configuring an organization

In this tutorial, you create your organization and configure it with two AWS member accounts. You create one of the member accounts in your organization, and you invite the other account to join your organization. Next, you use the allow list (p. 7) technique to specify that account administrators can delegate only explicitly listed services and actions. This allows administrators to validate any new service that AWS introduces before they permit its use by anyone else in your company. That way, if AWS introduces a new service, it remains prohibited until an administrator adds the service to the allow list in the appropriate policy. The tutorial also shows you how to use a deny list (p. 8) to ensure that no users in a member account can change the configuration for the auditing logs that AWS CloudTrail creates.

The following illustration shows the main steps of the tutorial.



Step 1: Create your organization (p. 10)

In this step, you create an organization with your current AWS account as the management account. You also invite one AWS account to join your organization, and you create a second account as a member account.

Step 2: Create the organizational units (p. 12)

Next, you create two organizational units (OUs) in your new organization and place the member accounts in those OUs.

Step 3: Create the service control policies (p. 14)

You can apply restrictions to what actions can be delegated to users and roles in the member accounts by using service control policies (SCPs) (p. 104). In this step, you create two SCPs and attach them to the OUs in your organization.

Step 4: Testing your organization's policies (p. 17)

You can sign in as users from each of the test accounts and see the effects that the SCPs have on the accounts.

None of the steps in this tutorial incurs costs to your AWS bill. AWS Organizations is a free service.

Prerequisites

This tutorial assumes that you have access to two existing AWS accounts (you create a third as part of this tutorial) and that you can sign in to each as an administrator.

The tutorial refers to the accounts as the following:

- `111111111111` – The account that you use to create the organization. This account becomes the management account. The owner of this account has an email address of `OrgAccount111@example.com`.
- `222222222222` – An account that you invite to join the organization as a member account. The owner of this account has an email address of `member222@example.com`.
- `333333333333` – An account that you create as a member of the organization. The owner of this account has an email address of `member333@example.com`.

Substitute the values above with the values that are associated with your test accounts. We recommend that you don't use production accounts for this tutorial.

Step 1: Create your organization

In this step, you sign in to account 111111111111 as an administrator, create an organization with that account as the management account, and invite an existing account, 222222222222, to join as a member account.

AWS Management Console

1. Sign in to AWS as an administrator of account 111111111111 and open the AWS Organizations console.
2. On the introduction page, choose **Create an organization**.
3. In the confirmation dialog box, choose **Create an organization**.

   Note
   By default, the organization is created with all features enabled. You can also create the organization with only consolidated billing features (p. 7) enabled.

AWS creates the organization and shows you the **AWS accounts** page. If you're on a different page then choose **AWS accounts** in the navigation pane on the left.

If the account you use has never had its email address verified by AWS, a verification email is automatically sent to the address that is associated with your management account. There might be a delay before you receive the verification email.

4. Verify your email address within 24 hours. For more information, see Email address verification (p. 34).

You now have an organization with your account as its only member. This is the management account of the organization.

Invite an existing account to join your organization

Now that you have an organization, you can begin to populate it with accounts. In the steps in this section, you invite an existing account to join as a member of your organization.

AWS Management Console

To invite an existing account to join

1. Navigate to the **AWS accounts** page, and choose **Add an AWS account**.
2. On the **Add an AWS account** page, choose **Invite an exisiting AWS account**.
3. In the box **Email address or account ID of an AWS account to invite** box, enter the email address of the owner of the account that you want to invite, similar to the following: `member222@example.com`. Alternatively, if you know the AWS account ID number, then you can enter it instead.
4. Type any text that you want into the **Message to include in the invitation email message** box. This text is included in the email that is sent to the owner of the account.
5. Choose **Send invitation**. AWS Organizations sends the invitation to the account owner.

   Important
   If you get an error that indicates that you exceeded your account limits for the organization or that you can't add an account because your organization is still initializing, wait until one hour after you created the organization and try again. If the error persists, contact AWS Support.
6. For the purposes of this tutorial, you now need to accept your own invitation. Do one of the following to get to the **Invitations** page in the console:

   - Open the email that AWS sent from the management account and choose the link to accept the invitation. When prompted to sign in, do so as an administrator in the invited member account.
   - Open the AWS Organizations console and navigate to the **Invitations** page.
7. On the **AWS accounts** page, choose **Accept** and then choose **Confirm**.
8. Sign out of your member account and sign in again as an administrator in your management account.

Create a member account

In the steps in this section, you create an AWS account that is automatically a member of the organization. We refer to this account in the tutorial as 333333333333.

AWS Management Console

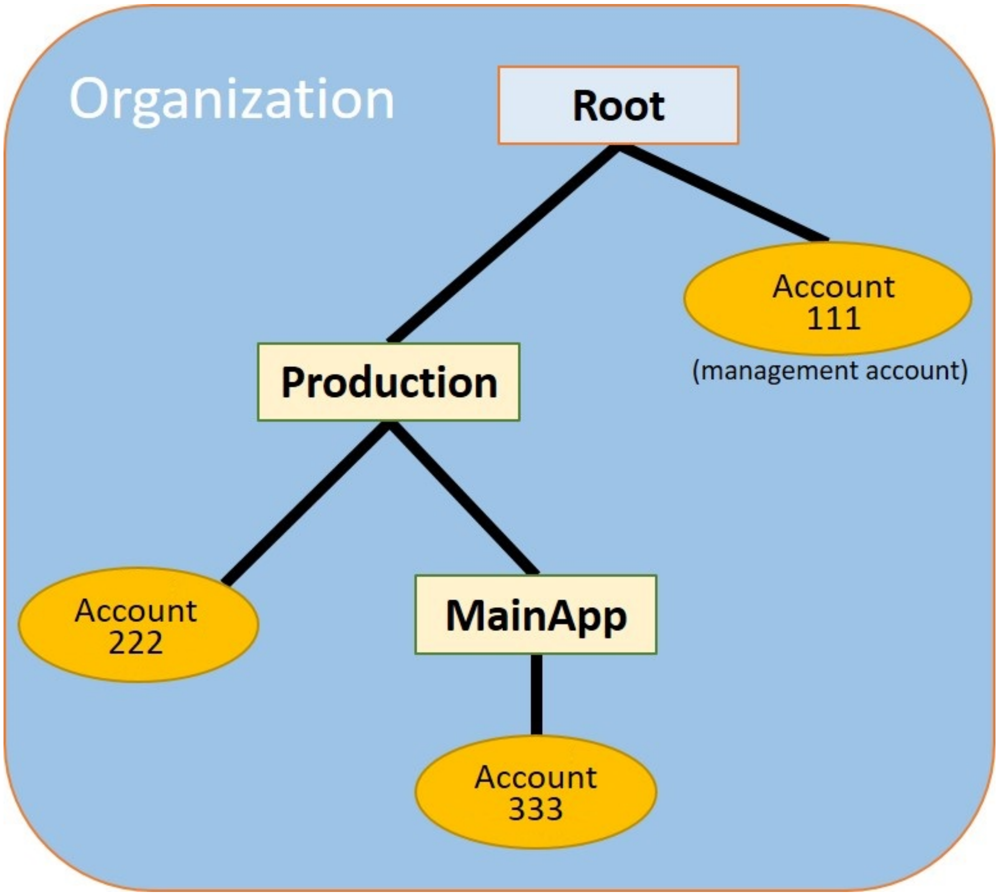
To create a member account

1. On the AWS Organizations console, on the **AWS accounts** page, choose **Add AWS account**.
2. On the **Add an AWS account** page, choose **Create an AWS account**.
3. For **AWS account name**, enter a name for the account, such as **`MainApp Account`**.
4. For **Email address of the account's root user**, enter the email address of the individual who is to receive communications on behalf of the account. This value must be globally unique. No two accounts can have the same email address. For example, you might use something like **`mainapp@example.com`**.
5. For **IAM role name**, you can leave this blank to automatically use the default role name of `OrganizationAccountAccessRole`, or you can supply your own name. This role enables you to access the new member account when signed in as an IAM user in the management account. For this tutorial, leave it blank to instruct AWS Organizations to create the role with the default name.
6. Choose **Create AWS account**. You might need to wait a short while and refresh the page to see the new account appear on the **AWS accounts** page.

   Important
   If you get an error that indicates that you exceeded your account limits for the organization or that you can't add an account because your organization is still initializing, wait until one hour after you created the organization and try again. If the error persists, contact AWS Support.

Step 2: Create the organizational units

In the steps in this section, you create organizational units (OUs) and place your member accounts in them. When you're done, your hierarchy looks like the following illustration. The management account remains in the root. One member account is moved to the Production OU, and the other member account is moved to the MainApp OU, which is a child of Production.



AWS Management Console

To create and populate the OUs

Note
In the steps that follow, you interact with objects for which you can choose either the name of the object itself, or the radio button next to the object.

- If you choose the name of the object, you open a new page that displays the objects details.
- If you choose the radio button next to the object, you are identifying that object to be acted upon by another action, such as choosing a menu option.

The steps that follow have you choose the radio button so that you can then act on the associated object by making menu choices.

1. On the AWS Organizations console navigate to the **AWS accounts** page.
2. Choose the check box ☑ next to the **Root** container.
3. On the **Children** tab, choose **Actions**, and then under **Organizational unit**, choose **Create new**.
4. On the **Create organizational unit in Root** page, for the **Organizational unit name**, enter **`Production`** and then choose **Create organizational unit**.
5. Choose the check box ☑ next to your new **Production** OU.
6. Choose **Actions**, and then under **Organizational unit**, choose **Create new**.

7. On the **Create organizational unit in Production** page, for the name of the second OU, enter `MainApp` and then choose **Create organizational unit**.

   Now you can move your member accounts into these OUs.
8. Return to the **AWS accounts** page, and then expand the tree under your **Production** OU by choosing the triangle ▶ next to it.

   This displays the **MainApp** OU as a child of **Production**.
9. Choose the check box ☑, not its name), choose **Actions**, and then under **AWS account**, choose **Move**.
10. On the **Move AWS account '*member-account-name*'** page, choose the radio button ○, not its name) and then choose **Move AWS account**.
11. Choose the check box ☑, not its name), choose **Actions**, and then under **AWS account**, choose **Move**.
12. On the **Move AWS account '*member-account-name*'** dialog box, the triangle next to **Production** to expand that branch and expose **MainApp**.
13. Choose the radio button ○, not its name) and then under **AWS account**, choose **Move AWS account**.

Step 3: Create the service control policies

In the steps in this section, you create three service control policies (SCPs) (p. 104) and attach them to the root and to the OUs to restrict what users in the organization's accounts can do. The first SCP prevents anyone in any of the member accounts from creating or modifying any AWS CloudTrail logs that you configure. The management account isn't affected by any SCP, so after you apply the CloudTrail SCP, you must create any logs from the management account.

Enable the service control policy type for the organization

Before you can attach a policy of any type to a root or to any OU within a root, you must enable the policy type for the organization. Policy types aren't enabled by default. The steps in this section show you how to enable the service control policy (SCP) type for your organization.

AWS Management Console

To enable SCPs for your organization

1. Navigate to the **Policies** page, and then choose **Service Control Policies**.
2. On the **Service control policies** page, choose **Enable service control policies**.

   A green banner appears to inform you that you can now create SCPs in your organization.

Create your SCPs

Now that service control policies are enabled in your organization, you can create the three policies that you need for this tutorial.

AWS Management Console

To create the first SCP that blocks CloudTrail configuration actions

1. Navigate to the **Policies** page, and then choose **Service Control Policies**.
2. On the **Service control policies** page, choose **Create policy**.

> **Note**
> The service control policy editor is currently available only in the original version of the AWS Organizations console. When you complete your edits, you'll automatically return to the new version of the console.

3. For **Policy name**, enter **`Block CloudTrail Configuration Actions`**.
4. In the **Policy** section, in the list of services on the left, select CloudTrail for the service. Then choose the following actions: **AddTags**, **CreateTrail**, **DeleteTrail**, **RemoveTags**, **StartLogging**, **StopLogging**, and **UpdateTrail**.
5. Still in the left pane, choose **Add resource** and specify **CloudTrail** and **All Resources**. Then choose **Add resource**.

   The policy statement on the right updates to look similar to the following.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1234567890123",
            "Effect": "Deny",
            "Action": [
                "cloudtrail:AddTags",
                "cloudtrail:CreateTrail",
                "cloudtrail:DeleteTrail",
                "cloudtrail:RemoveTags",
                "cloudtrail:StartLogging",
                "cloudtrail:StopLogging",
                "cloudtrail:UpdateTrail"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

6. Choose **Create policy**.

The second policy defines an allow list (p. 7) of all the services and actions that you want to enable for users and roles in the Production OU. When you're done, users in the Production OU can access ***only*** the listed services and actions.

AWS Management Console

To create the second policy that allows approved services for the production OU

1. From the Service control policies page, choose **Create policy**.
2. For **Policy name**, enter **`Allow List for All Approved Services`**.
3. Position your cursor in the right pane of the **Policy** section and paste in a policy like the following.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1111111111111",
            "Effect": "Allow",
            "Action": [
                "ec2:*",
                "elasticloadbalancing:*",
                "codecommit:*",
                "cloudtrail:*",
                "codedeploy:*"
              ],
            "Resource": [ "*" ]
        }
    ]
}
```

4. Choose **Create policy**.

The final policy provides a deny list (p. 8) of services that are blocked from use in the MainApp OU. For this tutorial, you block access to Amazon DynamoDB in any accounts that are in the **MainApp** OU.

AWS Management Console

To create the third policy that denies access to services that can't be used in the MainApp OU

1. From the **Service control policies** page, choose **Create policy**.
2. For **Policy name**, enter **`Deny List for MainApp Prohibited Services`**.
3. In the **Policy** section on the left, select **Amazon DynamoDB** for the service. For the action, choose **All actions**.
4. Still in the left pane, choose **Add resource** and specify **DynamoDB** and **All Resources**. Then choose **Add resource**.

   The policy statement on the right updates to look similar to the following.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [ "dynamodb:*" ],
      "Resource": [ "*" ]
    }
  ]
}
```

5. Choose **Create policy** to save the SCP.

Attach the SCPs to your OUs

Now that the SCPs exist and are enabled for your root, you can attach them to the root and OUs.

AWS Management Console

To attach the policies to the root and the OUs

1. Navigate to the **AWS accounts** page.
2. On the **AWS accounts** page, choose **Root** (its name, not the radio button) to navigate to its details page.
3. On the **Root** details page, choose the **Policies** tab, and then under **Service Control Policies**, choose **Attach**.
4. On the **Attach a service control policy** page, choose the radio button next to the SCP named `Block CloudTrail Configuration Actions`, and then choose **Attach**. In this tutorial, you

attach it to the root so that it affects all member accounts to prevent anyone from altering the way that you configured CloudTrail.

The **Root** details page, **Policies** tab now shows that two SCPs are attached to the root: the one you just attached and the default `FullAWSAccess` SCP.

5. Navigate back to the **AWS accounts** page, and choose the **Production** OU (it's name, not the radio button) to navigate to its details page.
6. On the **Production** OU's details page, choose the **Policies** tab.
7. Under **Service Control Policies**, choose **Attach**.
8. On the **Attach a service control policy** page, choose the radio button next to `Allow List for All Approved Services`, and then choose **Attach**. This enables users or roles in member accounts in the **Production** OU to access the approved services.
9. Choose the **Policies** tab again to see that two SCPs are attached to the OU: the one that you just attached and the default `FullAWSAccess` SCP. However, because the `FullAWSAccess` SCP is also an allow list that allows all services and actions, you must now detach this SCP to ensure that only your approved services are allowed.
10. To remove the default policy from the **Production** OU, choose the radio button to **FullAWSAccess**, choose **Detach**, and then on the confirmation dialog box, choose **Detach policy**.

    After you remove this default policy, all member accounts under the **Production** OU immediately lose access to all actions and services that are not on the allow list SCP that you attached in the preceding steps. Any requests to use actions that aren't included in the **Allow List for All Approved Services** SCP are denied. This is true even if an administrator in an account grants access to another service by attaching an IAM permissions policy to a user in one of the member accounts.
11. Now you can attach the SCP named `Deny List for MainApp Prohibited services` to prevent anyone in the accounts in the MainApp OU from using any of the restricted services.

    To do this, navigate to the **AWS accounts** page, choose the triangle icon to expand the **Production** OU's branch, and then choose the **MainApp** OU (it's name, not the radio button) to navigate to its contents.
12. On the **MainApp** details page, choose the **Policies** tab.
13. Under **Service Control Policies**, choose Attach, and then in the list of available policies, choose the radio button next to **Deny List for MainApp Prohibited Services**, and then choose **Attach policy**.

Step 4: Testing your organization's policies

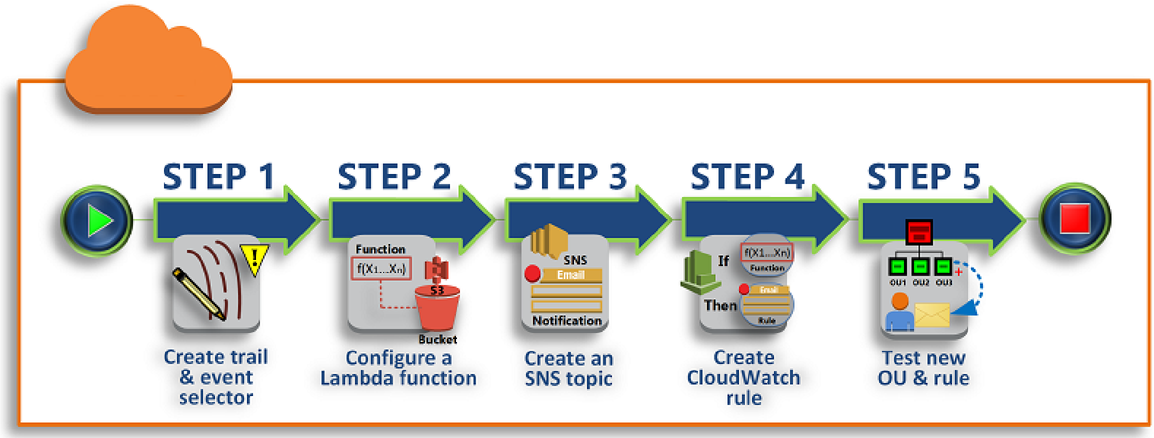
You now can sign in as a user in any of the member accounts and try to perform various AWS actions:

- If you sign in as a user in the management account, you can perform any operation that is allowed by your IAM permissions policies. The SCPs don't affect any user or role in the management account, no matter which root or OU the account is located in.
- If you sign in as the root user or an IAM user in account 222222222222, you can perform any actions that are allowed by the allow list. AWS Organizations denies any attempt to perform an action in any service that isn't in the allow list. Also, AWS Organizations denies any attempt to perform one of the CloudTrail configuration actions.
- If you sign in as a user in account 333333333333, you can perform any actions that are allowed by the allow list and not blocked by the deny list. AWS Organizations denies any attempt to perform an action that isn't in the allow list policy and any action that is in the deny list policy. Also, AWS Organizations denies any attempt to perform one of the CloudTrail configuration actions.

Tutorial: Monitor important changes to your organization with CloudWatch Events

This tutorial shows how to configure CloudWatch Events to monitor your organization for changes. You start by configuring a rule that is triggered when users invoke specific AWS Organizations operations. Next, you configure CloudWatch Events to run an AWS Lambda function when the rule is triggered, and you configure Amazon SNS to send an email with details about the event.

The following illustration shows the main steps of the tutorial.



Step 1: Configure a trail and event selector (p. 19)

Create a log, called a *trail*, in AWS CloudTrail. You configure it to capture all API calls.

Step 2: Configure a Lambda function (p. 20)

Create an AWS Lambda function that logs details about the event to an S3 bucket.

Step 3: Create an Amazon SNS topic that sends emails to subscribers (p. 20)

Create an Amazon SNS topic that sends emails to its subscribers, and then subscribe yourself to the topic.

Step 4: Create a CloudWatch Events rule (p. 21)

Create a rule that tells CloudWatch Events to pass details of specified API calls to the Lambda function and to SNS topic subscribers.

Step 5: Test your CloudWatch Events rule (p. 21)

Test your new rule by running one of the monitored operations. In this tutorial, the monitored operation is creating an organizational unit (OU). You view the log entry that the Lambda function creates, and you view the email that Amazon SNS sends to subscribers.

Tip

You can also use this tutorial as a guide in configuring similar operations, such as sending email notifications when account creation is complete. Because account creation is an asynchronous operation, you're not notified by default when it completes. For more information on using AWS CloudTrail and CloudWatch Events with AWS Organizations, see Logging and monitoring in AWS Organizations (p. 330).

Prerequisites

This tutorial assumes the following:

- You can sign in to the AWS Management Console as an IAM user from the management account in your organization. The IAM user must have permissions to create and configure a log in CloudTrail, a function in Lambda, a topic in Amazon SNS, and a rule in CloudWatch. For more information about granting permissions, see Access Management in the *IAM User Guide*, or the guide for the service for which you want to configure access.
- You have access to an existing Amazon Simple Storage Service (Amazon S3) bucket (or you have permissions to create a bucket) to receive the CloudTrail log that you configure in step 1.

> **Important**
> Currently, AWS Organizations is hosted in only the US East (N. Virginia) Region (even though it is available globally). To perform the steps in this tutorial, you must configure the AWS Management Console to use that region.

Step 1: Configure a trail and event selector

In this step, you sign in to the management account and configure a log (called a *trail*) in AWS CloudTrail. You also configure an event selector on the trail to capture all read/write API calls so that CloudWatch Events has calls to trigger on.

To create a trail

1. Sign in to AWS as an administrator of the organization's management account and then open the CloudTrail console at https://console.aws.amazon.com/cloudtrail/.
2. On the navigation bar in the upper-right corner of the console, choose the **US East (N. Virginia)** Region. If you choose a different region, AWS Organizations doesn't appear as an option in the CloudWatch Events configuration settings, and CloudTrail doesn't capture information about AWS Organizations.
3. In the navigation pane, choose **Trails**.
4. Choose **Create trail**.
5. For **Trail name**, enter **`My-Test-Trail`**.
6. Perform one of the following options to specify where CloudTrail is to deliver its logs:
   - If you already have a bucket, choose **No** next to **Create a new S3 bucket** and then choose the bucket name from the **S3 bucket** list.
   - If you need to create a bucket, choose **Yes** next to **Create a new S3 bucket** and then, for **S3 bucket**, enter a name for the new bucket.

     > **Note**
     > S3 bucket names must be ***globally*** unique.
7. Choose **Create**.
8. Choose the trail `My-Test-Trail` that you just created.
9. Choose the pencil icon next to **Management events**.
10. For **Read/Write events**, choose **All**, choose **Save**, and then choose **Configure**.

CloudWatch Events enables you to choose from several different ways to send alerts when an alarm rule matches an incoming API call. This tutorial demonstrates two methods: invoking a Lambda function that can log the API call and sending information to an Amazon SNS topic that sends an email or text message to the topic's subscribers. In the next two steps, you create the components you need: the Lambda function, and the Amazon SNS topic.

Step 2: Configure a Lambda function

In this step, you create a Lambda function that logs the API activity that is sent to it by the CloudWatch Events rule that you configure later.

To create a Lambda function that logs CloudWatch Events events

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. If you are new to Lambda, choose **Get Started Now** on the welcome page; otherwise, choose **Create a function**.
3. On the **Create function** page, choose **Blueprints**.
4. From the **Blueprints** search box, enter **`hello`** for the filter and choose the **hello-world** blueprint.
5. Choose **Configure**.
6. On the **Basic information** page, do the following:

   a. For the Lambda function name, enter **`LogOrganizationEvents`** in the **Name** text box.

   b. For **Role**, choose **Create a custom role** and then, at the bottom of the **AWS Lambda requires access to your resources** page, choose **Allow**. This role grants your Lambda function permissions to access the data it requires and to write its output log.

   c. Choose **Create function**.
7. On the next page, edit the code for the Lambda function, as shown in the following example.

```
console.log('Loading function');

exports.handler = async (event, context) => {
    console.log('LogOrganizationsEvents');
    console.log('Received event:', JSON.stringify(event, null, 2));
    return event.key1;  // Echo back the first key value
    // throw new Error('Something went wrong');
};
```

   This sample code logs the event with a **`LogOrganizationEvents`** marker string followed by the JSON string that makes up the event.
8. Choose **Save**.

Step 3: Create an Amazon SNS topic that sends emails to subscribers

In this step, you create an Amazon SNS topic that emails information to its subscribers. You make this topic a "target" of the CloudWatch Events rule that you create later.

To create an Amazon SNS topic to send an email to subscribers

1. Open the Amazon SNS console at https://console.aws.amazon.com/sns/v3/.
2. In the navigation pane, choose **Topics**.
3. Choose **Create new topic**.

   a. For **Topic name**, enter **`OrganizationsCloudWatchTopic`**.

   b. For **Display name**, enter **`OrgsCWEvnt`**.

   c. Choose **Create topic**.
4. Now you can create a subscription for the topic. Choose the ARN for the topic that you just created.
5. Choose **Create subscription**.

   a. On the **Create subscription** page, for **Protocol**, choose **Email**.
   b. For **Endpoint**, enter your email address.
   c. Choose **Create subscription**. AWS sends an email to the email address that you specified in the preceding step. Wait for that email to arrive, and then choose the **Confirm subscription** link in the email to verify that you successfully received the email.
   d. Return to the console and refresh the page. The **Pending confirmation** message disappears and is replaced by the now valid subscription ID.

Step 4: Create a CloudWatch Events rule

Now that the required Lambda function exists in your account, you create a CloudWatch Events rule that invokes it when the criteria in the rule are met.

To create a CloudWatch Events rule

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. As before, you must set the console to the **US East (N. Virginia)** Region or information about Organizations is not available. On the navigation bar in the upper-right corner of the console, choose the **US East (N. Virginia)** Region.
3. In the navigation pane, choose **Rules** and then choose **Create rule**.
4. For **Event source**, do the following:

   a. Choose **Event pattern**.
   b. Choose **Build event pattern to match events by service**.
   c. For **Service Name**, choose **Organizations**.
   d. For **Event Type**, choose **AWS API Call via CloudTrail**.
   e. Choose **Specific operation(s)** and then enter the APIs that you want monitored: **CreateAccount** and **CreateOrganizationalUnit**. You can select any others that you also want. For a complete list of available AWS Organizations APIs, see the AWS Organizations API Reference.
5. Under **Targets**, for **Function**, choose the function that you created in the previous procedure.
6. Under **Targets**, choose **Add target**.
7. In the new target row, choose the dropdown header and then choose **SNS topic**.
8. For **Topic**, choose the topic named **OrganizationCloudWatchTopic** that you created in the preceding procedure.
9. Choose **Configure details**.
10. On the **Configure rule details** page, for **Name** enter `OrgsMonitorRule`, leave **State** selected and then choose **Create rule**.

Step 5: Test your CloudWatch Events rule

In this step, you create an organizational unit (OU) and observe the CloudWatch Events rule generate a log entry and send an email to you with details about the event.

AWS Management Console

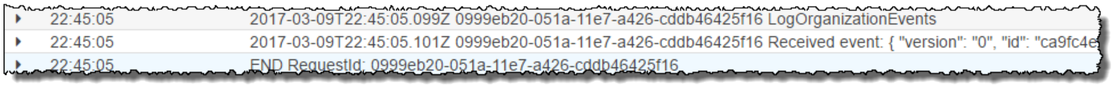
To create an OU

1. Open the AWS Organizations console to the **AWS accounts** page.
2. Choose the check box ☑ **Root** OU, choose **Actions**, and then under **Organizational unit** choose **Create new**.

3. For the name of the OU, enter **`TestCWEOU`** and then choose **Create organizational unit**.

To see the CloudWatch Events log entry

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation page, choose **Logs**.
3. Under **Log Groups**, choose the group that is associated with your Lambda function: **/aws/lambda/LogOrganizationEvents**.
4. Each group contains one or more streams, and there should be one group for today. Choose it.
5. View the log. You should see rows similar to the following.



```
▸ 22:45:05    2017-03-09T22:45:05.099Z 0999eb20-051a-11e7-a426-cddb46425f16 LogOrganizationEvents
▸ 22:45:05    2017-03-09T22:45:05.101Z 0999eb20-051a-11e7-a426-cddb46425f16 Received event: { "version": "0", "id": "ca9fc4e
▸ 22:45:05    END RequestId: 0999eb20-051a-11e7-a426-cddb46425f16
```

6. Select the middle row of the entry to see the full JSON text of the received event. You can see all the details of the API request in the `requestParameters` and `responseElements` pieces of the output.

```
2017-03-09T22:45:05.101Z 0999eb20-051a-11e7-a426-cddb46425f16 Received event:
{
    "version": "0",
    "id": "123456-EXAMPLE-GUID-123456",
    "detail-type": "AWS API Call via CloudTrail",
    "source": "aws.organizations",
    "account": "123456789012",
    "time": "2017-03-09T22:44:26Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "eventVersion": "1.04",
        "userIdentity": {
            ...
        },
        "eventTime": "2017-03-09T22:44:26Z",
        "eventSource": "organizations.amazonaws.com",
        "eventName": "CreateOrganizationalUnit",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.168.0.1",
        "userAgent": "AWS Organizations Console, aws-internal/3",
        "requestParameters": {
            "parentId": "r-exampleRootId",
            "name": "TestCWEOU"
        },
        "responseElements": {
            "organizationalUnit": {
                "name": "TestCWEOU",
                "id": "ou-exampleRootId-exampleOUId",
                "arn": "arn:aws:organizations::1234567789012:ou/o-exampleOrgId/ou-
exampleRootId-exampeOUId"
            }
        },
        "requestID": "123456-EXAMPLE-GUID-123456",
        "eventID": "123456-EXAMPLE-GUID-123456",
        "eventType": "AwsApiCall"
    }
}
```

7. Check your email account for a message from **OrgsCWEvnt** (the display name of your Amazon SNS topic). The body of the email contains the same JSON text output as the log entry that is shown in the preceding step.

Clean up: Remove the resources you no longer need

To avoid incurring charges, you should delete any AWS resources that you created as part of this tutorial that you don't want to keep.

To clean up your AWS environment

1. Use the CloudTrail console to delete the trail named **`My-Test-Trail`** that you created in step 1.
2. If you created an Amazon S3 bucket in step 1, use the Amazon S3 console to delete it.
3. Use the Lambda console to delete the function named **`LogOrganizationEvents`** that you created in step 2.
4. Use the Amazon SNS console to delete the Amazon SNS topic named **`OrganizationsCloudWatchTopic`** that you created in step 3.
5. Use the CloudWatch console to delete the CloudWatch rule named **`OrgsMonitorRule`** that you created in step 4.
6. Finally, use the Organizations console to delete the OU named **`TestCWEOU`** that you created in step 5.

That's it. In this tutorial, you configured CloudWatch Events to monitor your organization for changes. You configured a rule that is triggered when users invoke specific AWS Organizations operations. The rule ran a Lambda function that logged the event and sent an email that contains details about the event.

Best practices for AWS Organizations

We recommend that you follow these best practices when you create and operate your organization.

Topics

Best practices for the management account

Follow these recommendations to help protect the security of the management account in AWS Organizations. These recommendations assume that you also adhere to the best practice of using the root user only for those tasks that truly require it.

Topics

Use the management account only for tasks that ***require*** the management account

We recommend that you use the management account and its users and roles only for tasks that can be performed only by that account. Store all of your AWS resources in *other* AWS accounts in the organization and keep them out of the management account. The one exception is that we do recommend that you enable AWS CloudTrail and keep relevant CloudTrail trails and logs in the management account.

One important reason to keep your resources in other accounts is because Organizations service control policies (SCPs) do not work to restrict any users or roles in the management account.

Separating your resources from your management account also help you to understand the charges on your invoices.

Use a group email address for the management account's root user

- Use an email address that is managed by your business. Do not use a public email provider or one that is managed by a third party.
- Use an email address that forwards received messages directly to a list of senior business managers. In the event that AWS needs to contact the owner of the account, for example, to confirm access, the email message is distributed to multiple parties. This approach helps to reduce the risk of delays in responding, even if individuals are on vacation, out sick, or leave the business.

Use a complex password for the management account's root user

- The security of your account's root user depends on the strength of its password. We recommend that you use a password that is long, complex, and not used anywhere else. Numerous password managers and complex password generation algorithms and tools can help you achieve these goals.
- If you are using a strong password, as described in the previous point, and you rarely access the root user, we recommend that you do ***not*** periodically change the password. Changing the password more frequently than you use it increases the risk of compromise.
- Rely on your business' information security policy for managing long-term storage and access to the root user password. This approach might mean that you do any of the following:
  - Print the password, and store it in a safe.
  - Split the password into pieces, and distribute the pieces to senior business managers.
  - Store the password in a password manager system or tool under further controls and processes. If you do use a password manager, we recommend that it be offline. To avoid creating a circular dependency, do not store the root user password with tools that depend on AWS services that you sign in to with the protected account.

  Whatever method you choose, we recommend that the method is resilient and requires multiple actors to be involved to reduce collusion risks.
- Any access of the password or its storage location should be logged and monitored.

Enable MFA for your root user credentials

For instructions on how to enable multi-factor authentication (MFA), see Using multi-factor authentication (MFA) in AWS.

- Use a hardware-based device that does not rely on a battery to generate the one-time password (OTP). This approach helps ensure that the MFA is impossible to duplicate and isn't subject to battery fade risks while in long-term storage.
  - If you do decide to use a battery-based MFA, be sure to add processes to check the device periodically, and replace it when the expiry date approaches.
  - Create a plan to handle the logistics of needing to maintain 24/7 access to the token in case it is needed.
- We strongly recommend that you ***don't*** re-use that physical MFA for any other purpose than protecting this management account. If you reuse the physical MFA, it can create both operational confusion and unnecessary exposure of the MFA.
- Store the MFA device according to your information security policy but ***not*** in the same place as the associated password for the user. Make sure that the process to access the password and the process to access the MFA require different access to different resources (people, data, and tools).
- Any access of the MFA device or its storage location should be logged and monitored.

Add a phone number to the account contact information

- Although there are some credible attack vectors against landline, SIP, and mobile phone numbers, overall the risks are outweighed by the complexity of these vectors. If you use this mechanism to recover root access, other factors are available to the AWS Support representative to manage these risks. Therefore, we recommend adding a phone number as a useful additional barrier to the process.

- There are several options for provisioning a phone number, but the one we recommend is a dedicated SIM card and phone, stored long-term in a safe. It's important to ensure that the team responsible for paying the mobile bill for this phone contract understand the importance of the number even though there will be apparently no calls sent or received by it over long periods of time.
- It's important that this phone number not be well known within the business. Document it in the **AWS Contact Information** console page, and share its details with your billing team. Do not document it anywhere else. This approach helps to reduce the risk of the attack vectors associated with moving the phone number tied to the SIM to another SIM.
- Store the phone according to your existing information security policy. However, do not store the phone in the same location as the other related credential information.
- Any access of the phone or its storage location should be logged and monitored.

Review and keep track of who has access

- To ensure you maintain access to the management account, periodically review the personnel within your business who have access to the email address, password, MFA, and phone number associated with it. Align your review with existing business procedures. However, it's worth adding a monthly or quarterly review of this information to ensure that only the correct people have access.
- Ensure that the process to recover or reset access to the root user credentials is not reliant on any specific individual to complete. All processes should address the prospect of people being unavailable.

Document the processes for using the root user credentials

- It's common for important processes, such as the creation of the organization's management account, to be a planned process including multiple steps with multiple personnel. We recommend that you document and publish that plan, including the steps to be performed and their sequence of completion. This approach helps ensure that the decisions made are followed correctly.
- Document the performance of important processes as they are performed to ensure you have a record of the individuals involved in each step and the values used. It's also important to provide documentation about any exceptions and unforeseen events that occur.

  If an exception or unforeseen event does occur, document the time it occurred, who left the room, and what was taken out. You should then also document who returned to the room and what was brought back in.
- Create a suite of published processes about how to use the root user credentials in different scenarios, such as resetting the password. If you are at all unsure about the process for interacting with AWS Support in a specific scenario, create a support ticket to ask for the latest guidance about how to perform that task.

  Some scenarios you should document include the following:
  - Accessing the root user to perform one of the operations that only the root user can perform.
  - Resetting a root user password when you lose access.
  - Changing the root user password when you still have access.
  - Resetting the root user MFA when you lose access to the device.
  - Changing the root user MFA when you use a battery-based device.
  - Resetting the root user email address when you lose access to the email account.
  - Changing the root user email address when you still have access.
  - Resetting the root user phone number when you lose access to the phone number.
  - Changing the root user phone number when you still have access.

  - Deleting the organization's management account.
- Test and validate that you continue to have access to the root user and that the mobile phone number is operational on at least a quarterly basis. This schedule helps to assure the business that the process works and that you maintain access. It also demonstrates that those custodians of the access understand the steps they need to perform for the process to succeed. You never want to be in a position where the personnel involved in a process don't understand what they are supposed to do. As with fire drills, practice develops competency and reduces surprises.

  With each test, take the opportunity to reflect on the experience and propose improvements to the process. Especially examine any steps that were performed incorrectly or resulted in unexpected results. How could you change the process to improve it the next time?

  Some customers use these tests as an opportunity to rotate passwords. Our recommendation is not to rotate passwords. Instead, maintain the same complex password. You should only consider updating the password if you suspect it was compromised.

Apply controls to monitor access to the root user credentials

- Access to the root user credentials should be a rare event. Create alerts using tools like Amazon CloudWatch Events to announce the login and use of the management account root user credentials. This announcement should include, but should not be limited to, the email address used for the root user itself. This announcement should be significant and hard to miss, whether the use is valid or malicious. For an example, see Monitor and notify on AWS account root user activity.
- Ensure that personnel who receive such an announcement understand how to validate that the root user access is expected, and how to escalate if they believe that a security incident is in progress.

Best practices for member accounts

Follow these recommendations to help protect the security of the member accounts in in your organization. These recommendations assume that you also adhere to the best practice of using the root user only for those tasks that truly require it.

Topics

Use a group email address for all member account root users

- Use an email address that is managed by your business. Do not use a public email provider or one that is managed by a third party.
- Use an email address that forwards received messages directly to a list of senior business managers. In the event that AWS needs to contact the owner of the account, for example, to confirm access, the email is distributed to multiple parties. This approach helps to reduce the risk of delays in responding, even if individuals are on vacation, out sick, or leave the business.

Use a complex password for member account root user

- The security of your account's root user depends on the strength of its password. We recommend that you use a password that is long, complex, and not used anywhere else. Numerous password managers and complex password generation algorithms and tools can help you achieve these goals.
- If you are using a strong password, as described in the previous point, and you rarely access the root user, we recommend that you do ***not*** periodically change the password. Changing the password more frequently than you use it increases the risk of compromise.
- Rely on your business' information security policy for managing long-term storage and access to the passwords for your member account root users. Unlike the management account, however, it's reasonable to consider storing the password in a credible and business-approved password manager system or tool.

  Store the password in a password manager system or tool under further controls and processes. If you do use a password manager, we recommend that it be offline. To avoid creating a circular dependency, do not store the password with tools that depend on AWS services that you sign in to with the protected account.

  Whatever method you choose, we recommend that the method is resilient and requires multiple actors to be involved to reduce collusion risks.
- Alternatively, you can store the password for a member account root user in a safe, using the guidance we provide for the management account root user (p. 25).
- Consider not enabling credentials for the root user in created member accounts. By default, Organizations assigns a random, complex, and very long password that you can't retrieve. Instead, to access the root user you must perform the steps for password recovery. We recommend that you don't do this unless you need to perform a task that can only be performed by the root user in the account. For more information, see Accessing a member account as the root user (p. 64).
- However you choose to handle the root user's password, you can apply a service control policy (SCP) that prevents the member account root users from calling any AWS APIs. However, if you need to respond to a significant security event in a member account, you might need rapid access to that account's root user. Therefore, using a complex password for the member account root user and creating procedures for access and use in advance is still the recommended approach, as described in the previous points.

Enable MFA for your root user credentials

For instructions on how to enable multi-factor authentication (MFA), see Using multi-factor authentication (MFA) in AWS.

- We recommend that you use a hardware-based device that does not rely on a battery to generate the one-time password (OTP). This approach helps ensure that the MFA is impossible to duplicate and isn't subject to battery fade risks while in long-term storage
  - If you do decide to use a battery-based MFA, be sure to add processes to check the device periodically, and replace it when the expiry date approaches.
  - Create a plan to handle the logistics of needing to maintain 24/7 access to the token in case it is needed.
- If you choose to use a virtual MFA application, then unlike our recommendation for the management account root user (p. 25), for member accounts you can re-use a single MFA device for multiple member accounts. You can address geographic limitations by printing and securely storing the QR code used to configure the account in the virtual MFA application. Document the QR code's purpose, and seal and store it in accessible safes across the time zones you operate in, according to your information security policy. Then, when access is needed in a different geographic location, the local copy of the QR code can be retrieved and used to configure a virtual MFA app in the new location.
- Store the MFA device according to your information security policy but ***not*** in the same place as the associated password for the root user. Make sure that the process to access the password and the process to access the MFA device require different access procedures by different resources (people, data and tools).
- Any access of the MFA device or its storage location should be logged and monitored.
- If you lose or break your MFA device, you might need to contact Customer Support to remove the MFA from your account. Before they can do that, they must verify that the person making the request is in possession of the email address, phone number, and security questions associated with the account. So ensure you have that information and keep it up to date and securely stored.

Add the management account's phone number to the member account contact information

- You can normally rely on the phone number from the organization's management account (p. 25) for any critical account recovery. Therefore, we believe it’s unnecessary operational overhead to manage a separate telephone number to the contact information for a member account. Therefore, we recommend that you add the same phone number as the management account. Whether you use the same number as the management account or not, keep an accurate list of the phone numbers used, and any active security questions in a secure location similarly to the credentials themselves.

Review and keep track of who has access

- As we recommended for the management account (p. 26), you should periodically review the personnel within your business who have access to the email address, password, MFA, and phone number for your member account's root user. Align your review with existing business procedures. However, it’s worth adding a monthly or quarterly review of this information to ensure that only the correct people have access.
- Ensure that the process to recover or reset access to the root user credentials is not reliant on any specific individual to complete. All processes should address the possibility of people being unavailable.

Document the processes for using the root user credentials

- It’s common for important processes, such as the creation of the organization's management account, to be a planned process including multiple steps with multiple personnel. We recommend that you document and publish that plan, including the steps to be performed and their sequence of completion. This approach helps ensure that the decisions made are followed correctly.
- Document the performance of important processes as they are performed to ensure you have a record of the individuals involved in each step and the values used. It’s also important to provide documentation about any exceptions and unforeseen events that occur.

  If an exception or unforeseen event does occur, document the time it occurred, who left the room, and what was taken out. You should then also document who returned to the room and what was brought back in.
- Create and publish processes about how to use the root user credentials in different scenarios, such as resetting the password. If you are unsure about the process for interacting with AWS Support in a specific scenario, create a support ticket to ask for the latest guidance about how to perform that task.

  Some of the scenarios you should document include the following:
  - Accessing the root user to perform one of the operations that only the root user can perform.
  - Resetting a root user password when you lose access.
  - Changing the root user password when you still have access.
  - Resetting the root user MFA when you lose access to the device.
  - Changing the root user MFA when you use a battery-based device.
  - Resetting the root user email address when you lose access to the email account.
  - Changing the root user email address when you still have access.
  - Resetting the root user phone number when you lose access to the phone number.
  - Changing the root user phone number when you still have access.
  - Deleting the organization's management account.
- Test and validate that you continue to have access to the root user and that the mobile phone number for the member account (if you assigned one) is operational on at least a quarterly basis. This schedules helps assure the business that the process works and that you maintain access. It also demonstrates that those custodians of the access understand the steps they need to perform for the process to be successful. You never want to be in a position where the personnel involved in a process don’t understand what they are supposed to do. As with fire drills, practice develops competency and reduces surprises.

  With each test, take the opportunity to reflect on the experience and propose improvements to the process. Especially examine any steps that were performed incorrectly or resulted in unexpected results. How could you change the process to improve it the next time?

  Some customers use these tests as an opportunity to rotate passwords. Our recommendation is to not do this, and to maintain the same complex password. You should only look to updating the password if you suspect it was compromised.

Use an SCP to restrict what the root user in your member accounts can do

We recommend that you create a service control policy (SCP) in the organization and attach it to the organization's root so that it applies to all member accounts. The following example SCP prevents the root user in any member account from being able to make any AWS service API calls.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "*",
            "Resource": "*",
            "Condition": {
                "StringLike": { "aws:PrincipalArn": "arn:aws:iam::*:root" }
            }
        }
    ]
}
```

In the majority of circumstances, any administrative tasks can be performed by either an AWS Identity and Access Management (IAM) role in the member account that has relevant administrator permissions. Any such roles should have suitable controls applied that limit, log, and monitor activity.

Apply controls to monitor access to the root user credentials

- If you do choose to enable access to the root user credentials, such access should be a rare event. Create alerts using tools like Amazon CloudWatch Events to announce the login and use of the root user credentials. This announcement should be significant and hard to miss, whether the use is valid or malicious. For an example, see Monitor and notify on AWS account root user activity.
- Ensure that personnel who receive such an announcement understand how to validate that the root user access is expected, and how to escalate if they believe that a security incident is in progress.

Creating and managing an organization

You can perform the following tasks using the AWS Organizations console or by running an AWS Command Line Interface (AWS CLI) command or the equivalent AWS SDK API operations:

- **Create an organization (p. 32)**. Create your organization with your current account as its management account. Create member accounts within your organization, and invite other accounts to join your organization.
- **Enable all features in your organization (p. 35)**. Enabling all features is the preferred way to work with AWS Organizations. When you create an organization, you have the option to enable all features or a subset of features for consolidating billing. Enabling all features is the default, and it includes Consolidated Billing features.

  With all features enabled, you can use the advanced account management features available in AWS Organizations such as service control policies (SCPs) (p. 104). SCPs offer central control over the maximum available permissions for all accounts in your organization, helping you to keep your accounts within your organization's access control guidelines.
- **View details about your organization (p. 42)**. View details about your organization and its roots, organizational units (OUs), and accounts.
- **Delete an organization (p. 48)**. Delete an organization when you no longer need it.

Note
The procedures in this section specify the minimum permissions needed to perform the tasks. These typically apply to the API or access to the command line tool.
Performing a task in the console might require additional permissions. For example, you could grant read-only permissions to all users in your organization, and then grant other permissions that allow selected users to perform specific tasks.

Creating an organization

You can create an organization that starts with your AWS account as the management account. When you create an organization, you can choose whether the organization supports all features (recommended) or only consolidated billing features.

After creating an organization, you can add accounts to your organization in these ways from the management account:

- Create other AWS accounts (p. 60) that are automatically added to your organization as member accounts
- After verifying your email address, invite existing AWS accounts (p. 52) to join your organization as member accounts

Create an organization

You can create an organization by using either the AWS Management Console or by using a command from the AWS CLI or one of the SDK APIs.

Minimum permissions

To create an organization with your current AWS account, you must have the following permissions:

- `organizations:CreateOrganization`
- `iam:CreateServiceLinkedRole`

  You can restrict this permission to only the service principal `organizations.amazonaws.com`.

AWS Management Console

To create an organization

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. By default, the organization is created with all features enabled. However, you can choose either of the following steps:

   - To create an organization with all features enabled, on the introduction page, choose **Create an organization**.
   - To create an organization with Consolidated Billing features only, on the introduction page and under **Create an organization**, choose **consolidated billing features**, and then in the confirmation dialog box, choose **Create an organization**.

   If you accidentally choose the wrong option, you can immediately go to the **Settings** page, and then choose **Delete organization** and start over.
3. The organization is created and the **AWS accounts** page appears. The only account present is your management account, and it's currently stored in the root organizational unit (OU) (p. 6).

   If required, Organizations automatically sends a verification email to the address that is associated with your management account. There might be a delay before you receive the verification email. Verify your email address within 24 hours. For more information, see Email address verification (p. 34). You can create accounts to grow your organization without verifying your management account's email address. However, to invite existing accounts, you must first complete email verification.

   > **Note**
   > If this account previously verified its email address, then it doesn't happen again when you use the account to create an organization.

AWS CLI & AWS SDKs

To create an organization

You can use one of the following commands to create an organization:

- AWS CLI: create-organization

  The following example creates an organization and makes the currently signed-in AWS account the management account for the organization.

```
$ aws organizations create-organization
{
    "Organization": {
        "Id": "o-aa111bb222",
        "Arn": "arn:aws:organizations::123456789012:organization/o-aa111bb222",
        "FeatureSet": "ALL",
        "MasterAccountArn": "arn:aws:organizations::123456789012:account/o-
aa111bb222/123456789012",
        "MasterAccountId": "123456789012",
        "MasterAccountEmail": "admin@example.com",
        "AvailablePolicyTypes": [  ...DEPRECATED - DO NOT USE ... ]
    }
}
```

> **Important**
> The `AvailablePolicyTypes` field is deprecated and doesn't contain accurate information about the policies enabled in your organization. To see the accurate and complete list of policy types that are actually enabled for the organization, use the `ListRoots` command, as described in the AWS CLI portion of the following section.

- AWS SDKs: CreateOrganization

Now you can add additional accounts to your organization as follows:

- To create an AWS account that automatically becomes part of your AWS organization, see Creating an AWS account in your organization (p. 60).
- To invite an existing account to your organization, see Inviting an AWS account to join your organization (p. 51).

Email address verification

After you create an organization and before you can invite accounts to join, you must verify that you own the email address provided for the management account in the organization.

When you create an organization, if the management account has not been previously verified, AWS automatically sends a verification email to the specified email address. There might be a delay before you receive the verification email.

Within 24 hours, follow the instructions in the email to verify your email address.

If you don't verify your email address within 24 hours, you can resend the verification request so that you can invite other AWS accounts to your organization. If you don't receive the verification email, check that your email address is correct and, if necessary, modify it.

- To find out what email address is associated with your management account, see Viewing the details of an organization from the management account (p. 42).
- To change the email address that is associated with your management account, see Managing an AWS account in the *AWS Billing User Guide*.

AWS Management Console

To resend the verification request

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. Navigate to the **Settings** page and then choose **Send verification request**. The option is only present if the management account is not verified.
3. Verify your email address within 24 hours.

   After verifying your email address, you can invite other AWS accounts to your organization. For more information, see Inviting an AWS account to join your organization (p. 51).

If you change the email address of the management account, the account's status reverts to "email unverified," and you must complete the verification process for your new email address.

> **Note**
> If you invited accounts to join your organization before you changed the management account's email address and those invitations have not yet been accepted, they can’t be accepted until you verify the management account’s new email address. Use the previous procedure to resend the verification request. After you complete the process by responding to the email, your invited accounts can accept the invitations.

Enabling all features in your organization

AWS Organizations has two available feature sets:

- All features (p. 7) – This feature set is the preferred way to work with AWS Organizations, and it includes Consolidating Billing features. When you create an organization, enabling all features is the default. With all features enabled, you can use the advanced account management features available in AWS Organizations such as integration with supported AWS services (p. 227) and organization management policies (p. 84).
- Consolidated Billing features (p. 7) – All organizations support this subset of features, which provides basic management tools that you can use to centrally manage the accounts in your organization.

If you create an organization with consolidated billing features only, you can later enable all features. This page describes the process of enabling all features.

Before enabling all features

Before changing from an organization that supports only consolidated billing features to an organization supporting all features, note the following:

- When you start the process to enable all features, AWS Organizations sends a request to every member account that you *invited* to join your organization. Every invited account must approve enabling all features by accepting the request. Only then can you complete the process to enable all features in your organization. If an account declines the request, you must either remove the account from your organization or resend the request. The request must be accepted before you can complete the process to enable all features. Accounts that you *created* using AWS Organizations don't get a request because they don't need to approve the additional control.
- You can continue inviting accounts to your organization while enabling all features. The owner of an invited account is informed by the invitation whether they are joining an organization with consolidated billing only, or with all features enabled.
  - If you invite an account *during* the process to enable all features, the invitation states that the organization they are joining has all features enabled. If you cancel the process to enable all features before the account accepts the invitation, that invitation is canceled. You must invite the account again to be a member of an organization with consolidated billing features only.
  - If you invite an account and the invitation is not yet accepted *before* you begin the process to enable all features, that invitation is canceled because the invitation states that the organization has consolidated billing features only. You must invite the account again to be a member of an organization with all features enabled.
- You can also continue creating accounts in the organization. That process isn't affected by this change.
- AWS Organizations verifies that every member account has a service-linked role named `AWSServiceRoleForOrganizations`. This role is mandatory in all accounts to enable all features. If you deleted the role in an invited account, accepting the invitation to enable all features recreates the role. If you deleted the role in an account that was created using AWS Organizations, that account

receives an invitation specifically to recreate that role. All of these invitations must be accepted for the organization to complete the process of enabling all features.

- Because enabling all features makes it possible to use SCPs (p. 104), be sure that your account administrators understand the effects of attaching SCPs to the organization, organizational units, or accounts. SCPs can restrict what users and even administrators can do in affected accounts. For example, the management account can apply SCPs that can prevent member accounts from leaving the organization.
- The management account isn't affected by any SCP. You can't limit what users and roles in the management account can do by applying SCPs. SCPs affect only member accounts.
- The migration from consolidated billing features to all features is one-way. You can't switch an organization with all features enabled back to consolidated billing features only.
- (Not recommended) If your organization has only consolidated billing features enabled, member account administrators can choose to delete the service-linked role named `AWSServiceRoleForOrganizations`. If you later choose to enable all features in an organization, this role is required and is recreated in all accounts as part of accepting the invitation to enable all features. For more information about how AWS Organizations uses this role, see AWS Organizations and service-linked roles (p. 226).

Beginning the process to enable all features

When you sign in to your organization's management account, you can begin the process to enable all features. To do this, complete the following steps.

Minimum permissions
To enable all features in your organization, you must have the following permission:

- `organizations:EnableAllFeatures`
- `organizations:DescribeOrganization` – required only when using the Organizations console

AWS Management Console

To ask your invited member accounts to agree to enable all features in the organization

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Settings** page choose **Begin process**.
3. On the **Enable all features** page, acknowledge your understanding that you cannot return to only consolidated billing features after you switch by choosing **Begin process**.

   AWS Organizations sends a request to every invited (not created) account in the organization asking for approval to enable all features in the organization. If you have any accounts that were created using AWS Organizations and the member account administrator deleted the service-linked role named `AWSServiceRoleForOrganizations`, AWS Organizations sends that account a request to recreate the role.

   The console displays the **Request approval status** list for the invited accounts.

   Tip
   To get back to this page later, open the **Settings** page and in the **Request sent *date*** section, choose **View status**.
4. The **Enable all features** page shows the current request status for each account in the organization. Accounts that have agreed to the request show a status of **ACCEPTED**. Accounts that haven't yet agreed show a status of **OPEN**.

AWS CLI & AWS SDKs

To ask your invited member accounts to agree to enable all features in the organization

You can use one of the following commands to enable all features in an organization:

- AWS CLI: enable-all-features

  The following command begins the process to enable all features in the organization.

```
$ aws organizations enable-all-features
{
    "Handshake": {
        "Id": "h-79d8f6f114ee4304a5e55397eEXAMPLE",
        "Arn": "arn:aws:organizations::123456789012:handshake/o-aa111bb222/
enable_all_features/h-79d8f6f114ee4304a5e55397eEXAMPLE",
        "Parties": [
            {
                "Id": "a1b2c3d4e5",
                "Type": "ORGANIZATION"
            }
        ],
        "State": "REQUESTED",
        "RequestedTimestamp": "2020-11-19T16:21:46.995000-08:00",
        "ExpirationTimestamp": "2021-02-17T16:21:46.995000-08:00",
        "Action": "ENABLE_ALL_FEATURES",
        "Resources": [
            {
                "Value": "o-a1b2c3d4e5",
                "Type": "ORGANIZATION"
            }
        ]
    }
}
```

  The output shows the details of the handshake that invited member accounts must agree to.
- AWS SDKs: EnableAllFeatures

Notes

- A countdown of 90 days begins when the request is sent to the member accounts. All accounts must approve the request within that time period or the request expires. If the request expires, all requests related to this attempt are canceled, and you have to start over with step 2.
- During the time between when you make the request to enable all features and when either all accounts accept or the timeout occurs, all pending invitations for other accounts to join the organization are automatically canceled. You can't issue new invitations until the process of enabling all features is finished.
- After you complete the process of enabling all features, you once again can invite accounts to join the organization. The process doesn't change, but all invitations include information letting the recipients know that if they accept the invitation, they're subject to any applicable policies.

After all invited accounts in the organization approve their requests, you can finalize the process and enable all features. You can also immediately finalize the process if your organization doesn't have any *invited* member accounts. To finalizing the process, continue with Finalizing the process to enable all features (p. 40).

Approving the request to enable all features or to recreate the service-linked role

When you sign in to one of the organization's invited member accounts, you can approve a request from the management account. If your account was originally invited to join the organization, the invitation is to enable all features and implicitly includes approval for recreating the `AWSServiceRoleForOrganizations` role, if needed. If your account was instead created using AWS Organizations and you deleted the `AWSServiceRoleForOrganizations` service-linked role, you receive an invitation only to recreate the role. To do this, complete the following steps.

> **Important**
> If you perform the steps in the following procedure, the management account in the organization can apply policy-based controls on your member account. These controls can restrict what users and even what you as the administrator can do in your account. Such restrictions might prevent your account from leaving the organization.

> **Minimum permissions**
> To approve a request to enable all features for your member account, you must have the following permissions:
>
> - `organizations:AcceptHandshake`
> - `organizations:DescribeOrganization` – required only when using the Organizations console
> - `organizations:ListHandshakesForAccount`– required only when using the Organizations console
> - `iam:CreateServiceLinkedRole` – required only if the `AWSServiceRoleForOrganizations` role must be recreated in the member account

AWS Management Console

To agree to the request to enable all features in the organization

1. Sign in to the AWS Organizations console at AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in a member account.
2. Read what accepting the request for all features in the organization means for your account, and then choose **Accept**. The page continues to show the process as incomplete until all accounts in the organization accept the requests and the administrator of the management account finalizes the process.

AWS CLI & AWS SDKs

To agree to the request to enable all features in the organization

To agree to the request, you must accept the handshake with `"Action": "APPROVE_ALL_FEATURES"`.

- AWS CLI:
  - accept-handshake
  - list-handshakes-for-account

  The following example shows how to list the handshakes available for your account. The value of `"Id"` in the fourth line of the output is the value you need for the next command.

```
$ aws organizations list-handshakes-for-account
```

```
{
    "Handshakes": [
        {
            "Id": "h-a2d6ecb7dbdc4540bc788200aEXAMPLE",
            "Arn": "arn:aws:organizations::123456789012:handshake/o-aa111bb222/
approve_all_features/h-a2d6ecb7dbdc4540bc788200aEXAMPLE",
            "Parties": [
                {
                    "Id": "a1b2c3d4e5",
                    "Type": "ORGANIZATION"
                },
                {
                    "Id": "111122223333",
                    "Type": "ACCOUNT"
                }
            ],
            "State": "OPEN",
            "RequestedTimestamp": "2020-11-19T16:35:24.824000-08:00",
            "ExpirationTimestamp": "2021-02-17T16:35:24.035000-08:00",
            "Action": "APPROVE_ALL_FEATURES",
            "Resources": [
                {
                    "Value": "c440da758cab44068cdafc812EXAMPLE",
                    "Type": "PARENT_HANDSHAKE"
                },
                {
                    "Value": "o-aa111bb222",
                    "Type": "ORGANIZATION"
                },
                {
                    "Value": "111122223333",
                    "Type": "ACCOUNT"
                }
            ]
        }
    ]
}
```

The following example uses the Id of the handshake from the previous command to accept that handshake.

```
$ aws organizations accept-handshake --handshake-id h-
a2d6ecb7dbdc4540bc788200aEXAMPLE
{
    "Handshake": {
        "Id": "h-a2d6ecb7dbdc4540bc788200aEXAMPLE",
        "Arn": "arn:aws:organizations::123456789012:handshake/o-aa111bb222/
approve_all_features/h-a2d6ecb7dbdc4540bc788200aEXAMPLE",
        "Parties": [
            {
                "Id": "a1b2c3d4e5",
                "Type": "ORGANIZATION"
            },
            {
                "Id": "111122223333",
                "Type": "ACCOUNT"
            }
        ],
        "State": "ACCEPTED",
        "RequestedTimestamp": "2020-11-19T16:35:24.824000-08:00",
        "ExpirationTimestamp": "2021-02-17T16:35:24.035000-08:00",
        "Action": "APPROVE_ALL_FEATURES",
        "Resources": [
            {
                "Value": "c440da758cab44068cdafc812EXAMPLE",
                "Type": "PARENT_HANDSHAKE"
            },
            {
                "Value": "o-aa111bb222",
                "Type": "ORGANIZATION"
            },
            {
                "Value": "111122223333",
                "Type": "ACCOUNT"
            }
        ]
    }
}
```

- AWS SDKs:
  - list-handshakes-for-account
  - AcceptHandshake

Finalizing the process to enable all features

All invited member accounts must approve the request to enable all features. If there are no invited member accounts in the organization, the **Enable all features progress** page indicates with a green banner that you can finalize the process.

Minimum permissions

To finalize the process to enable all features for the organization, you must have the following permission:

- `organizations:AcceptHandshake`
- `organizations:ListHandshakesForOrganization`
- `organizations:DescribeOrganization` – required only when using the Organizations console

AWS Management Console

To finalize the process to enable all features

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Settings** page, if all invited accounts accept the request to enable all features, a green box appears at the top of the page to inform you. In the green box, choose **Go to finalize**.
3. On the **Enable all features** page, choose **Finalize**, and then in the confirmation dialog box, choose **Finalize** again.
4. The organization now has all features enabled.

AWS CLI & AWS SDKs

To finalize the process to enable all features

To finalize the process, you must accept the handshake with `"Action": "ENABLE_ALL_FEATURES"`.

- AWS CLI:
  - list-handshakes-for-organization

- accept-handshake

```
$ aws organizations list-handshakes-for-organization
{
    "Handshakes": [
        {
            "Id": "h-43a871103e4c4ee399868fbf2EXAMPLE",
            "Arn": "arn:aws:organizations::123456789012:handshake/o-aa111bb222/
enable_all_features/h-43a871103e4c4ee399868fbf2EXAMPLE",
            "Parties": [
                {
                    "Id": "a1b2c3d4e5",
                    "Type": "ORGANIZATION"
                }
            ],
            "State": "OPEN",
            "RequestedTimestamp": "2020-11-20T08:41:48.047000-08:00",
            "ExpirationTimestamp": "2021-02-18T08:41:48.047000-08:00",
            "Action": "ENABLE_ALL_FEATURES",
            "Resources": [
                {
                    "Value": "o-aa111bb222",
                    "Type": "ORGANIZATION"
                }
            ]
        }
    ]
}
```

The following example shows how to list the handshakes available for the organization. The value of `"Id"` in the fourth line of the output is the value you need for the next command.

```
$ aws organizations accept-handshake \
    --handshake-id h-43a871103e4c4ee399868fbf2EXAMPLE
{
    "Handshake": {
        "Id": "h-43a871103e4c4ee399868fbf2EXAMPLE",
        "Arn": "arn:aws:organizations::123456789012:handshake/o-aa111bb222/
enable_all_features/h-43a871103e4c4ee399868fbf2EXAMPLE",
        "Parties": [
            {
                "Id": "a1b2c3d4e5",
                "Type": "ORGANIZATION"
            }
        ],
        "State": "ACCEPTED",
        "RequestedTimestamp": "2020-11-20T08:41:48.047000-08:00",
        "ExpirationTimestamp": "2021-02-18T08:41:48.047000-08:00",
        "Action": "ENABLE_ALL_FEATURES",
        "Resources": [
            {
                "Value": "o-aa111bb222",
                "Type": "ORGANIZATION"
            }
        ]
    }
}
```

- AWS SDKs:
  - AcceptHandshake
  - AcceptHandshake

The next steps:

- Enable the policy types that you want to use. After that, you can attach policies to administer the accounts in your organization. For more information, see Managing AWS Organizations policies (p. 84).
- Enable integration with supported services. For more information, see Using AWS Organizations with other AWS services (p. 223).

Viewing details about your organization

You can perform the following tasks to view details about elements of your organization.

Topics

Viewing the details of an organization from the management account

When you sign in to the organization's management account in the AWS Organizations console, you can view details of the organization.

Minimum permissions
To view the details of an organization, you must have the following permission:

- `organizations:DescribeOrganization`

AWS Management Console

To view the details for your organization

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. Navigate to the **Settings** page. This page displays details about the organization, including the organization ID and the account name and email address assigned to the organization's management account.

AWS CLI & AWS SDKs

To view the details for your organization

You can use one of the following commands to view details of an organization:

- AWS CLI: describe-organization

  The following example shows the information included in the output of this command.

```
$ aws organizations describe-organization
```

```
{
    "Organization": {
        "Id": "o-aa111bb222",
        "Arn": "arn:aws:organizations::123456789012:organization/o-aa111bb222",
        "FeatureSet": "ALL",
        "MasterAccountArn": "arn:aws:organizations::128716708097:account/o-
aa111bb222/123456789012",
        "MasterAccountId": "123456789012",
        "MasterAccountEmail": "admin@example.com",
        "AvailablePolicyTypes": [ ...DEPRECATED - DO NOT USE... ]
    }
}
```

Important
The `AvailablePolicyTypes` field is deprecated and doesn't contain accurate information about the policies enabled in your organization. To see the accurate and complete list of policy types that are actually enabled for the organization, use the `ListRoots` command, as described in the AWS CLI portion of the following section.

- AWS SDKs: DescribeOrganization

Viewing the details of the root

When you sign in to the organization's management account in the AWS Organizations console, you can view details of the root.

Minimum permissions
To view the details of the root, you must have the following permissions:

- `organizations:DescribeOrganization` (console only)
- `organizations:ListRoots`

The root is the topmost container in the hierarchy of organizational units (OUs) and generally behaves as an OU. However, as the container at the very top of the hierarchy, changes to the root affect every other OU and every AWS account in the organization.

AWS Management Console

To view the details of the root

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. Navigate to the **AWS accounts** page, and choose the **Root** OU (its name, not the radio button).
3. The **Root** details page appears and displays the details of the root.

AWS CLI & AWS SDKs

To view the details of the root

You can use one of the following commands to view details of a root:

- AWS CLI: list-roots

  The following example shows how to retrieve the details of the root, including which policy types are currently enabled in the organization:

```
$ aws organizations list-roots
```

```
{
    "Roots": [
        {
            "Id": "r-a1b2",
            "Arn": "arn:aws:organizations::123456789012:root/o-aa111bb222/r-a1b2",
            "Name": "Root",
            "PolicyTypes": [
                {
                    "Type": "BACKUP_POLICY",
                    "Status": "ENABLED"
                }
            ]
        }
    ]
}
```

- AWS SDKs: ListRoots

Viewing the details of an OU

When you sign in to the organization's management account in the AWS Organizations console, you can view details of the OUs in your organization.

Minimum permissions
To view the details of an organizational unit (OU), you must have the following permissions:

- `organizations:DescribeOrganizationalUnit`
- `organizations:DescribeOrganization` – required only when using the Organizations console
- `organizations:ListOrganizationsUnitsForParent`– required only when using the Organizations console
- `organizations:ListRoots` – required only when using the Organizations console

AWS Management Console

To view details of an OU

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **AWS accounts** page, choose the name of the OU (not its radio button) that you want to examine. If the OU that you want is a child of another OU, choose the triangle icon next to its parent OU to expand it and see those in the next level of the hierarchy. Repeat until you find the OU that you want.

   The **Organizational unit details** box shows the information about the OU.

AWS CLI & AWS SDKs

To view details of an OU

You can use the following commands to view details of an OU:

- AWS CLI, AWS SDKs:
  - list-roots
  - list-children
  - describe-organizational-unit

The following example shows how to find the ID of on OU using the AWS CLI. You find the OU ID by traversing the hierarchy starting with the `list-roots` command and then performing `list-children` on the root and iteratively on each of its children until you find the one you want.

```
$ aws organizations list-roots
{
    "Roots": [
        {
            "Id": "r-a1b2",
            "Arn": "arn:aws:organizations::123456789012:root/o-aa111bb222/r-a1b2",
            "Name": "Root",
            "PolicyTypes": []
        }
    ]
}
$ aws organizations list-children --parent-id r-a1b2 --child-type ORGANIZATIONAL_UNIT
{
    "Children": [
        {
            "Id": "ou-a1b2-f6g7h111",
            "Type": "ORGANIZATIONAL_UNIT"
        }
    ]
}
```

After you have the OU's ID, the following example shows how to retrieve the details about the OU.

```
$ aws organizations describe-organizational-unit --organizational-unit-id ou-a1b2-
f6g7h111
{
    "OrganizationalUnit": {
        "Id": "ou-a1b2-f6g7h111",
        "Arn": "arn:aws:organizations::123456789012:ou/o-aa111bb222/ou-a1b2-
f6g7h111",
        "Name": "Production-Apps"
    }
}
```

- AWS SDKs:
  - ListRoots
  - ListChildren
  - DescribeOrganizationalUnit

Viewing details of an account

When you sign in to the organization's management account in the AWS Organizations console, you can view details about your accounts.

Minimum permissions
To view the details of an AWS account, you must have the following permissions:

- `organizations:DescribeAccount`
- `organizations:DescribeOrganization` – required only when using the Organizations console
- `organizations:ListAccounts` – required only when using the Organizations console

AWS Management Console

To view details of an AWS account

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. Navigate to the **AWS accounts** page and choose the name of the name of the account (not the radio button) that you want to examine. If the account that you want is a child of an OU, you might have to choose the triangle icon ▶next to an OU to expand it and see its children. Repeat until you find the account.

   The **Account details** box shows the information about the account.

AWS CLI & AWS SDKs

To view details of an AWS account

You can use the following commands to view details of an account:

- AWS CLI:
  - list-accounts – lists the details of *all* accounts in the organization
  - describe-account – lists the details of only the specified account

  Both commands return the same details for each account included in the response.

  The following example shows how to retrieve the details about a specified account.

```
$ aws organizations describe-account --account-id 123456789012

{
    "Account": {
        "Id": "123456789012",
        "Arn": "arn:aws:organizations::123456789012:account/o-
aa111bb222/123456789012",
        "Email": "admin@example.com",
        "Name": "Example.com Organization's Management Account",
        "Status": "ACTIVE",
        "JoinedMethod": "INVITED",
        "JoinedTimestamp": "2020-11-20T09:04:20.346000-08:00"
    }
}
```

- AWS SDKs:
  - ListAccounts
  - DescribeAccount

Viewing details of a policy

When you sign in to the organization's management account in the AWS Organizations console, you can view details about your policies.

Minimum permissions

To view the details of a policy, you must have the following permissions:

- `organizations:DescribePolicy`
- `organizations:ListPolicies`

AWS Management Console

To view the details of a policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. Perform one of the following:
   - Navigate to the **Policies** page, and then choose the policy type for the policy that you want to examine.
   - Navigate to the **AWS accounts** page, then navigate to an OU or account to which the policy is attached. Finally, choose the **Policies** tab to see the list of attached policies.
3. Choose the name of the policy (not the radio button).

   On the **Details** page for the policy, you can view all of the information about the policy, including the JSON policy text, and the list of OUs and accounts that the policy is attached to.

AWS CLI & AWS SDKs

To view the details of a policy

You can use one of the following commands to view details of a policy:

- AWS CLI:
  - list-policies
  - describe-policy – lists the details of only the specified policy

  The following example shows how to find the policy ID of the policy that you want to examine. You must specify a policy type, and the command returns all policies of only that type.

```
$ aws organizations list-policies --filter BACKUP_POLICY
{
    "Policies": [
        {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
backup_policy/p-i9j8k7l6m5",
            "Name": "test-backup-policy",
            "Description": "test-policy-description",
            "Type": "BACKUP_POLICY",
            "AwsManaged": false
        }
    ]
}
```

  The response includes all of the details except the JSON policy document.

  The following example shows how to retrieve the details of only the specified policy, including the JSON policy document.

```
$ aws organizations describe-policy --policy-id p-i9j8k7l6m5
{
    "Policies": [
        {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
backup_policy/p-i9j8k7l6m5",
            "Name": "test-backup-policy",
            "Description": "test-policy-description",
            "Type": "BACKUP_POLICY",
            "AwsManaged": false
        },
        "Content": "{\"plans\":{\"My-Backup-Plan\":{\"regions\":{\"@@assign\":[\"us-
west-2\"]},\"rules\":{\"My-Backup-Rule\"
                  :{\"target_backup_vault_name\":{\"@@assign\":\"My-Primary-Backup-
Vault\"}}},\"selections\":{\"tags\":{
                  \"My-Backup-Plan-Resource-Assignment\":{\"iam_role_arn\":
{\"@@assign\":\"arn:aws:iam::$account:role/
                  My-Backup-Role\"},\"tag_key\":{\"@@assign\":\"Stage\"},\"tag_value
\":{\"@@assign\":[\"Production\"]}}}}}}}}"
    ]
}
```

- AWS SDKs:
  - ListPolicies
  - DescribePolicy

Deleting the organization by removing the management account

When you no longer need your organization, you can delete it. This removes the management account from the organization and deletes the organization itself. The former management account becomes a standalone AWS account. You then have three options: You can continue to use it as a standalone account, you can use it to create a different organization, or you can accept an invitation from another organization to add the account to that organization as a member account.

Important

- If you delete an organization, you can't recover it. If you created any policies inside of the organization, they're also deleted and you can't recover them.
- You can delete an organization only after you remove all member accounts from the organization. If you created some of your member accounts using AWS Organizations, you might be blocked from removing those accounts. You can remove a member account only if it has all the information that's required to operate as a standalone AWS account. For more information about how to provide that information and then remove the account, see Leaving an organization as a member account (p. 72).
- If you closed a member account before you remove it from the organization, it enters a 'suspended' state for a period of time and you can't remove the account from the organization until it is finally closed. This can take up to 90 days and can prevent you from deleting the organization until all member accounts are completely closed.

When you remove the management account from an organization by deleting the organization, it can affect the account in the following ways:

- The account is responsible for paying only its own charges and is no longer responsible for the charges incurred by any other account.
- Integration with other services might be disabled. For example, AWS IAM Identity Center (successor to AWS Single Sign-On) requires an organization to operate, so if you remove an account from an organization that supports IAM Identity Center, the users in that account can no longer use that service.

The management account of an organization is never affected by service control policies (SCPs), so there is no change in permissions after SCPs are no longer available.

Minimum permissions

To delete an organization, you must sign in as an IAM user or role in the management account, and you must have the following permissions:

- `organizations:DeleteOrganization`
- `organizations:DescribeOrganization` – required only when using the Organizations console

AWS Management Console

To remove the management account from an organization and delete the organization

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. Before you can delete the organization, you must first remove all accounts from the organization. For more information, see Removing a member account from your organization (p. 69).
3. Navigate to the **Settings** page, and then choose **Delete organization**.
4. In the **Delete organization** confirmation dialog box, enter the organization's ID which is displayed in the line above the text box. Then, choose **Delete organization**.
5. (Optional) If you also want to close the management account, you can follow the steps at Closing an AWS account (p. 74).

AWS CLI & AWS SDKs

To remove the management account from an organization and delete the organization

You can use one of the following commands to delete an organization:

- AWS CLI: delete-organization

  The following example deletes the organization for which the AWS account whose credentials are used is the management account.

  ```
  $ aws organizations delete-organization
  ```

  This command produces no output when successful.
- AWS SDKs: DeleteOrganization

Managing the AWS accounts in your organization

An organization is a collection of AWS accounts that you manage together. You can perform the following tasks to manage the accounts that are part of your organization:

- View details of the accounts in your organization (p. 45). You can see the account's unique ID number, its Amazon Resource Name (ARN), and the policies that are attached to it.
- Export a list of all AWS accounts in your organization (p. 67). You can download a .csv file that contains account details for every account within your organization.
- Invite existing AWS accounts to join your organization (p. 51). Create invitations, manage invitations that you have created, and accept or decline invitations.
- Create an AWS account as part of your organization (p. 60). Create and access an AWS account that is automatically part of your organization.
- Update alternate contacts in your organization (p. 63). Update alternate contacts for your AWS accounts in your organization.
- Remove an AWS account from your organization (p. 69). As an administrator in the management account, remove member accounts that you no longer want to manage from your organization. As an administrator of a member account, remove your account from its organization. If the management account has attached a policy to your member account, you could be blocked from removing your account.
- Delete (or close) an AWS account (p. 74). When you no longer need an AWS account, you can close the account to prevent any usage or accrual of charges.

Impact of being in an organization

- What is the impact on an AWS account that ***joins*** an organization? (p. 50)
- What is the impact on an AWS account that you ***create*** in an organization? (p. 51)

Impact on an AWS account that joins an organization?

When you invite an AWS account to join an organization, and the owner of the account accepts the invitation, AWS Organizations automatically makes the following changes to the new member account:

- AWS Organizations creates a service-linked role called `AWSServiceRoleForOrganizations` (p. 226). The account must have this role if your organization supports all features. You can delete the role if the organization supports only the consolidated billing feature set. If you delete the role and later you enable all features in your organization, AWS Organizations recreates the role for the account.
- You might have a variety of policies attached to the organization root or the OU that contains the account. If so, those policies immediately apply to all users and roles in the invited account.

- You can enable service trust for another AWS service (p. 227) for your organization. When you do, that trusted service can create service-linked roles or perform actions in any member account in the organization, including an invited account.

> **Note**
> For invited member accounts, AWS Organizations doesn't automatically create the IAM role OrganizationAccountAccessRole (p. 66). This role grants users in the management account administrative access to the member account. If you want to enable that level of administrative control to an invited account, you can manually add the role. For more information, see Creating the OrganizationAccountAccessRole in an invited member account (p. 64).

You can invite an account to join an organization that has only the consolidated billing features enabled. If you later want to enable all features for the organization, invited accounts must approve the change.

Impact on an AWS account that you create in an organization?

When you create an AWS account in your organization, AWS Organizations automatically makes the following changes to the new member account:

- AWS Organizations creates a service-linked role called `AWSServiceRoleForOrganizations (p. 226)`. The account must have this role if your organization supports all features. You can delete the role if the organization supports only the consolidated billing feature set. If you delete the role and later you enable all features in your organization, AWS Organizations recreates the role for the account.
- AWS Organizations creates the IAM role OrganizationAccountAccessRole (p. 66). This role grants the management account access to the new member account. Although this role *can* be deleted, we recommend that you don't delete it so that it is available as a recovery option.
- If you have any policies attached to the root of the OU tree (p. 84), those policies immediately apply to all users and roles in the created account. New accounts are added to the root OU by default.
- If you have enabled service trust for another AWS service (p. 227) for your organization, that trusted service can create service-linked roles or perform actions in any member account in the organization, including your created account.

Inviting an AWS account to join your organization

After you create an organization and verify that you own the email address associated with the management account, you can invite existing AWS accounts to join your organization.

When you invite an account, AWS Organizations sends an invitation to the account owner, who decides whether to accept or decline the invitation. You can use the AWS Organizations console to initiate and manage invitations that you send to other accounts. You can send an invitation to another account only from the management account of your organization.

> **Note**
> Billing history and reports for all accounts stay with the payer account in an Organization. Before you move the account to a new Organization, download any billing and report histories for any member accounts that you want to keep. This might include Cost and Usage Reports, Detailed Billing Reports, or reports generated by Cost Explorer Service.

If you are the administrator of an AWS account, you also can accept or decline an invitation from an organization. If you accept, your account becomes a member of that organization. Your account can join only one organization, so if you receive multiple invitations to join, you can accept only one.

At the moment an account accepts the invitation to join an organization, the management account of the organization becomes liable for all charges accrued by the new member account. The payment method attached to the member account is no longer used. Instead, the payment method attached to the management account of the organization pays for all charges accrued by the member account.

When an invited account joins your organization, you *do not* automatically have full administrator control over the account, unlike created accounts. If you want the management account to have full administrative control over an invited member account, you must create the `OrganizationAccountAccessRole` IAM role in the member account and grant permission to the management account to assume the role. To configure this, after the invited account becomes a member, follow the steps in Creating the OrganizationAccountAccessRole in an invited member account (p. 64).

> **Note**
> When you create an account in your organization instead of inviting an existing account to join, AWS Organizations automatically creates an IAM role (named `OrganizationAccountAccessRole` by default) that you can use to grant users in the management account administrator access to the created account.

AWS Organizations *does* automatically create a service-linked role in invited member accounts to support integration between AWS Organizations and other AWS services. For more information, see AWS Organizations and service-linked roles (p. 226).

For the number of invitations you can send per day, see Maximum and minimum values (p. 340). Accepted invitations don't count against this quota. As soon as one invitation is accepted, you can send another invitation that same day. Each invitation must be responded to within 15 days, or it expires.

An invitation that is sent to an account counts against the quota of accounts in your organization. The count is restored if the invited account declines, the management account cancels the invitation, or the invitation expires.

To create an account that automatically is part of your organization, see Creating an AWS account in your organization (p. 60).

> **Important**
> Because of legal and billing constraints, you can invite AWS accounts only from the same AWS seller and AWS partition as the management account. For example, in an AWS EMEA organization, you can have both an AWS Inc. and an AWS Canada account.
>
> - All accounts in an organization must come from the same seller of record as the management account if your organization's management account was created by Amazon Internet Services Pvt. Ltd (AISPL). For example, as an AWS seller in India, you can invite only other AISPL accounts to your organization. You can't combine accounts from AISPL and AWS or from any other AWS seller.
> - All accounts in an organization must come from the same AWS partition as the management account. Accounts in the commercial AWS Regions partition can't be in an organization with accounts from the China Regions partition or accounts in the AWS GovCloud (US) Regions partition.

Sending invitations to AWS accounts

To invite accounts to your organization, you must first verify that you own the email address associated with the management account. For more information, see Email address verification (p. 34). After you verify your email address, complete the following steps to invite accounts to your organization.

> **Minimum permissions**
> To invite an AWS account to join your organization, you must have the following permissions:
>
> - `organizations:DescribeOrganization` (console only)

- `organizations:InviteAccountToOrganization`

AWS Management Console

To invite another account to join your organization

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. If you already verified your email address with AWS, skip this step.

   If you haven't yet verified your email address, follow the instructions in the verification email (p. 34) within 24 hours after you create the organization. There might be a delay before you receive the verification email message. You can't invite an account to join your organization until you verify your email address.
3. Navigate to the **AWS accounts** page, and choose **Add an AWS account**.
4. On the **Add an AWS account** page, choose **Invite an existing AWS account**.
5. On the **Invite an existing AWS** page, for **Email address or account ID of the AWS account to invite** enter either the email address associated with the account to be invited, or its account ID number.
6. (Optional) For **Message to include in the invitation email message**, enter any text that you want to include in the email invitation to the invited account owner.
7. (Optional) In the **Add tags** section, specify one or more tags that are automatically applied to the account after its administrator accepts the invitation. To do this, choose **Add tag** and then enter a key and an optional value. Leaving the value blank sets it to an empty string; it isn't `null`. You can attach up to 50 tags to an AWS account.
8. Choose **Send invitation**.

   > **Important**
   > If you get a message that you exceeded your account quotas for the organization or that you can't add an account because your organization is still initializing, contact AWS Support.
9. The console redirects you to the **Invitations** page page where you can view all open and accepted invitations here. The invitation that you just created appears at the top of the list with its status set to **OPEN**.

   AWS Organizations sends an invitation to the email address of the owner of the account that you invited to the organization. This email message includes a link to the AWS Organizations console, where the account owner can view the details and choose to accept or decline the invitation. Alternatively, the owner of the invited account can bypass the email message, go directly to the AWS Organizations console, view the invitation, and accept or decline it.

   The invitation to this account immediately counts against the maximum number of accounts that you can have in your organization. AWS Organizations doesn't wait until the account accepts the invitation. If the invited account declines, the management account cancels the invitation. If the invited account doesn't respond within the specified time period, the invitation expires. In either case, the invitation no longer counts against your quota.

AWS CLI & AWS SDKs

To invite another account to join your organization

You can use one of the following commands to invite another account to join your organization:

- AWS CLI: invite-account-to-organization

```
$ aws organizations invite-account-to-organization \
    --target '{"Type": "EMAIL", "Id": "juan@example.com"}' \
    --notes "This is a request for Juan's account to join Bill's organization."
{
    "Handshake": {
        "Action": "INVITE",
        "Arn": "arn:aws:organizations::111111111111:handshake/o-exampleorgid/invite/
h-examplehandshakeid111",
        "ExpirationTimestamp": 1482952459.257,
        "Id": "h-examplehandshakeid111",
        "Parties": [
            {
                "Id": "o-exampleorgid",
                 "Type": "ORGANIZATION"
            },
            {
                 "Id": "juan@example.com",
                 "Type": "EMAIL"
            }
        ],
        "RequestedTimestamp": 1481656459.257,
        "Resources": [
            {
                "Resources": [
                    {
                        "Type": "MASTER_EMAIL",
                        "Value": "bill@amazon.com"
                    },
                    {
                         "Type": "MASTER_NAME",
                         "Value": "Management Account"
                    },
                    {
                         "Type": "ORGANIZATION_FEATURE_SET",
                         "Value": "FULL"
                    }
                ],
                "Type": "ORGANIZATION",
                "Value": "o-exampleorgid"
            },
            {
                "Type": "EMAIL",
                "Value": "juan@example.com"
            }
        ],
        "State": "OPEN"
    }
}
```

- AWS SDKs: InviteAccountToOrganization

Managing pending invitations for your organization

When you sign in to your management account, you can view all the linked AWS accounts in your organization and cancel any pending (open) invitations. To do this, complete the following steps.

Minimum permissions

To manage pending invitations for your organization, you must have the following permissions:

- `organizations:DescribeOrganization` – required only when using the Organizations console
- `organizations:ListHandshakesForOrganization`
- `organizations:CancelHandshake`

AWS Management Console

To view or cancel invitations that are sent from your organization to other accounts

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. Navigate to the **Invitations** page.

   This page displays all invitations that are sent from your organization and their current status.

   > **Note**
   > Accepted, canceled, and declined invitations continue to appear in the list for 30 days. After that, they're deleted and no longer appear in the list.

3. Choose the radio button ○next to the invitation that you want to cancel, and then choose **Cancel invitation**. If the radio button is grayed out, then that invitation can't be canceled.

   The status of the invitation changes from **OPEN** to **CANCELED**.

   AWS sends an email message to the account owner stating that you canceled the invitation. The account can no longer join the organization unless you send a new invitation.

AWS CLI & AWS SDKs

To view or cancel invitations that are sent from your organization to other accounts

You can use the following commands to view or cancel invitations:

- AWS CLI: list-handshakes-for-organization, cancel-handshake
- The following example shows the invitations sent by this organization to other accounts.

```
$ aws organizations list-handshakes-for-organization
{
    "Handshakes": [
        {
            "Action": "INVITE",
            "Arn": "arn:aws:organizations::111111111111:handshake/o-exampleorgid/
invite/h-examplehandshakeid111",
            "ExpirationTimestamp": 1482952459.257,
            "Id": "h-examplehandshakeid111",
            "Parties": [
                {
                    "Id": "o-exampleorgid",
                    "Type": "ORGANIZATION"
                },
                {
                    "Id": "juan@example.com",
                    "Type": "EMAIL"
                }
            ],
            "RequestedTimestamp": 1481656459.257,
            "Resources": [
                {
                    "Resources": [
                        {
                            "Type": "MASTER_EMAIL",
                            "Value": "bill@amazon.com"
                        },
                        {
                            "Type": "MASTER_NAME",
                            "Value": "Management Account"
                        },
                        {
                            "Type": "ORGANIZATION_FEATURE_SET",
                            "Value": "FULL"
                        }
                    ],
                    "Type": "ORGANIZATION",
                    "Value": "o-exampleorgid"
                },
                {
                    "Type": "EMAIL",
                    "Value": "juan@example.com"
                },
                {
                    "Type":"NOTES",
                    "Value":"This is an invitation to Juan's account to join Bill's
 organization."
                }
            ],
            "State": "OPEN"
        },
        {
            "Action": "INVITE",
            "State":"ACCEPTED",
            "Arn": "arn:aws:organizations::111111111111:handshake/o-exampleorgid/
invite/h-examplehandshakeid111",
            "ExpirationTimestamp": 1.471797437427E9,
            "Id": "h-examplehandshakeid222",
            "Parties": [
                {
                    "Id": "o-exampleorgid",
                    "Type": "ORGANIZATION"
                },
                {
                    "Id": "anika@example.com",
                    "Type": "EMAIL"
                }
            ],
            "RequestedTimestamp": 1.469205437427E9,
            "Resources": [
                {
                    "Resources": [
                        {
                            "Type":"MASTER_EMAIL",
                             "Value":"bill@example.com"
                        },
                        {
                            "Type":"MASTER_NAME",
                            "Value":"Management Account"
                        }
                    ],
                    "Type":"ORGANIZATION",
                    "Value":"o-exampleorgid"
                },
                {
                    "Type":"EMAIL",
                    "Value":"anika@example.com"
                },
                {
                    "Type":"NOTES",
                    "Value":"This is an invitation to Anika's account to join Bill's
 organization."
                }
            ]
        }
    ]
```

```
}
```

The following example shows how to cancel an invitation to an account.

```
$ aws organizations cancel-handshake --handshake-id h-examplehandshakeid111
{
    "Handshake": {
        "Id": "h-examplehandshakeid111",
        "State":"CANCELED",
        "Action": "INVITE",
        "Arn": "arn:aws:organizations::111111111111:handshake/o-exampleorgid/invite/
h-examplehandshakeid111",
        "Parties": [
            {
                "Id": "o-exampleorgid",
                "Type": "ORGANIZATION"
            },
            {
                "Id": "susan@example.com",
                "Type": "EMAIL"
            }
        ],
        "Resources": [
            {
                "Type": "ORGANIZATION",
                "Value": "o-exampleorgid",
                "Resources": [
                    {
                        "Type": "MASTER_EMAIL",
                        "Value": "bill@example.com"
                    },
                    {
                        "Type": "MASTER_NAME",
                        "Value": "Management Account"
                    },
                    {
                        "Type": "ORGANIZATION_FEATURE_SET",
                        "Value": "CONSOLIDATED_BILLING"
                    }
                ]
            },
            {
                "Type": "EMAIL",
                "Value": "anika@example.com"
            },
            {
                "Type": "NOTES",
                "Value": "This is a request for Susan's account to join Bob's
 organization."
            }
        ],
        "RequestedTimestamp": 1.47008383521E9,
        "ExpirationTimestamp": 1.47137983521E9
    }
}
```

- AWS SDKs: ListHandshakesForOrganization, CancelHandshake

Accepting or declining an invitation from an organization

Your AWS account might receive an invitation to join an organization. You can accept or decline the invitation. To do this, complete the following steps.

> **Note**
>
> An account’s status with an organization affects what cost and usage data is visible:
>
> - If a member account leaves an organization and becomes a standalone account, the account no longer has access to cost and usage data from the time range when the account was a member of the organization. The account has access only to the data that is generated as a standalone account.
> - If a member account leaves organization A to join organization B, the account no longer has access to cost and usage data from the time range when the account was a member of organization A. The account has access only to the data that is generated as a member of organization B.
> - If an account rejoins an organization that it previously belonged to, the account regains access to its historical cost and usage data.

Minimum permissions

To accept or decline an invitation to join an AWS organization, you must have the following permissions:

- `organizations:ListHandshakesForAccount` – Required to see the list of invitations in the AWS Organizations console.
- `organizations:AcceptHandshake`.
- `organizations:DeclineHandshake`.
- `iam:CreateServiceLinkedRole` – Required only when accepting the invitation requires the creation of a service-linked role in the member account to support integration with other AWS services. For more information, see AWS Organizations and service-linked roles (p. 226).

AWS Management Console

To accept or decline an invitation

1. An invitation to join an organization is sent to the email address of the account owner. If you are an account owner and you receive an invitation email message, follow the instructions in the email invitation or go to AWS Organizations console in your browser, and then choose **Invitations**, or go straight to the **member account's Invitation** page.
2. If prompted, sign in to the invited account as an IAM user, assume an IAM role, or sign in as the account's root user (not recommended).
3. The **member account's Invitation** page displays your account's open invitations to join organizations.

   Choose **Accept invitation** or **Decline invitation** as appropriate.

   - If you choose **Accept invitation** in the preceding step, the console redirects you to the Organization overview page with details about the organization that your account is now a member of. You can view the organization's ID and the owner's email address.

Note

Accepted invitations continue to appear in the list for 30 days. After that, they are deleted and no longer appear in the list.

AWS Organizations automatically creates a service-linked role in the new member account to support integration between AWS Organizations and other AWS services. For more information, see AWS Organizations and service-linked roles (p. 226).

AWS sends an email message to the owner of the organization's management account stating that you accepted the invitation. It also sends an email message to the member account owner stating that the account is now a member of the organization.

- If you choose **Decline** in the preceding step, your account remains on the **member account's Invitation** page that lists any other pending invitations.

  AWS sends an email message to the organization's management account owner stating that you declined the invitation.

  Note

  Declined invitations continue to appear in the list for 30 days. After that, they are deleted and no longer appear in the list.

AWS CLI & AWS SDKs

To accept or decline an invitation

You can use the following commands to accept or decline an invitation:

- AWS CLI: accept-handshake, decline-handshake

  The following example shows how to accept an invitation to join an organization.

```
$ aws organizations accept-handshake --handshake-id h-examplehandshakeid111
{
    "Handshake": {
        "Action": "INVITE",
        "Arn": "arn:aws:organizations::111111111111:handshake/o-exampleorgid/invite/
h-examplehandshakeid111",
        "RequestedTimestamp": 1481656459.257,
        "ExpirationTimestamp": 1482952459.257,
        "Id": "h-examplehandshakeid111",
        "Parties": [
            {
                "Id": "o-exampleorgid",
                "Type": "ORGANIZATION"
            },
            {
                "Id": "juan@example.com",
                "Type": "EMAIL"
            }
        ],
        "Resources": [
            {
                "Resources": [
                    {
                        "Type": "MASTER_EMAIL",
                        "Value": "bill@amazon.com"
                    },
                    {
                        "Type": "MASTER_NAME",
                        "Value": "Management Account"
                    },
                    {
                        "Type": "ORGANIZATION_FEATURE_SET",
                         "Value": "ALL"
                    }
                ],
                "Type": "ORGANIZATION",
                "Value": "o-exampleorgid"
            },
            {
                "Type": "EMAIL",
                "Value": "juan@example.com"
            }
        ],
        "State": "ACCEPTED"
    }
}
```

The following example shows how to decline an invitation to join an organization.

- AWS SDKs: AcceptHandshake, DeclineHandshake

Creating an AWS account in your organization

This page describes how to create accounts within your organization in AWS Organizations. To learn about getting started with AWS and creating a single AWS account, see the Getting Started Resource Center.

An organization is a collection of AWS accounts that you centrally manage. You can perform the following procedures to manage the accounts that are part of your organization:

- Creating an AWS account that is part of your organization (p. 61)
- Accessing a member account that has a management account access role (p. 66)

Important

- When you create a member account in your organization, AWS Organizations automatically creates an AWS Identity and Access Management (IAM) role `OrganizationAccountAccessRole` in the member account that enables IAM users in the management account to exercise full administrative control over the member account. This role is subject to any service control policies (SCPs) (p. 104) that apply to the member account.

  AWS Organizations also automatically adds a managed policy with the `OrganizationAccountAccessRole` role to the member account. This allows centralized control, so that any additional accounts attached to the same managed policy will be updated automatically whenever the policy gets updated. Previously, new accounts created within an organization got an inline policy added that only applied to that single account. To learn more about inline and managed policies, see Managed policies and inline policies in the *IAM User Guide*.

  AWS Organizations also automatically creates a service-linked role named `AWSServiceRoleForOrganizations` that enables integration with select AWS services. You must configure the other services to allow the integration. For more information, see AWS Organizations and service-linked roles (p. 226).
- If this organization is managed with AWS Control Tower, then create your accounts by using the AWS Control Tower account factory in the AWS Control Tower console or APIs. If you create an account in Organizations, then that account isn't enrolled with AWS Control Tower.

For more information, see Referring to Resources Outside of AWS Control Tower in the *AWS Control Tower User Guide*.

Note
AWS accounts that you create as part of an organization are not automatically subscribed to AWS marketing emails. To opt-in your accounts to receive marketing emails, see https://pages.awscloud.com/communication-preferences.

Creating an AWS account that is part of your organization

When you sign in to the organization's management account, you can create member accounts that are automatically part of your organization. To do this, complete the following steps.

When you create an account using the following procedure, Organizations automatically copies the following information from the management account to the new member account:

- Account name
- Phone number
- Company name
- Customer URL
- Company contact email
- Communication language
- Marketplace (vendor of the account in some AWS Regions)

AWS does ***not*** automatically collect all the information required for an account to operate as a standalone account. If you ever need to remove the account from the organization and make it a standalone account, you must provide that information for the account before you can remove it. For more information, see Leaving an organization as a member account (p. 72).

Minimum permissions
To create a member account in your organization, you must have the following permissions:

- `organizations:CreateAccount`
- `organizations:DescribeOrganization` – required only when using the Organizations console
- `iam:CreateServiceLinkedRole` (granted to principal `organizations.amazonaws.com` to enable creating the required service-linked role in the member accounts).

AWS Management Console

To create an AWS account that is automatically part of your organization

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page, choose **Add an AWS account**.
3. On the **Add an AWS account** page, choose **Create an AWS account** (it is chosen by default).
4. On the **Create an AWS account** page, for **AWS account name** enter the name that you want to assign to the account. This name helps you distinguish the account from all other accounts in the organization and is separate from the IAM alias or the email name of the owner.

5. For **Email address of the account's owner**, enter the email address of the account's owner. This email address cannot already be associated with another AWS account because it becomes the user name credential for the root user of the account.
6. (Optional) Specify the name to assign to the IAM role that is automatically created in the new account. This role grants the organization's management account permission to access the newly created member account. If you don't specify a name, AWS Organizations gives the role a default name of `OrganizationAccountAccessRole`. We recommend that you use the default name across all of your accounts for consistency.

   Important
   Remember this role name. You need it later to grant access to the new account for IAM users in the management account.
7. (Optional) In the **Add tags** section, add one or more tags to the new account by choosing **Add tag** and then entering a key and an optional value. Leaving the value blank sets it to an empty string; it isn't `null`. You can attach up to 50 tags to an account.
8. Choose **Create AWS account**.

   - If you get an error that indicates that you exceeded your account quota for the organization, see I get a "quota exceeded" message when I try to add an account to my organization (p. 347).
   - If you get an error that indicates that you can't add an account because your organization is still initializing, wait one hour and try again.
   - You can also check the AWS CloudTrail log for information on whether the account creation was successful. For more information, see Logging and monitoring in AWS Organizations (p. 330).
   - If the error persists, contact AWS Support.

   The **AWS accounts** page appears, with your new account added to the list.
9. Now that the account exists and has an IAM role that grants administrator access to users in the management account, you can access the account by following the steps in Accessing and administering the member accounts in your organization (p. 63).

   Note
   When you create an account, AWS Organizations initially assigns a long (64 characters), complex, randomly generated password to the root user. You can't retrieve this initial password. To access the account as the root user for the first time, you must go through the process for password recovery. For more information, see Accessing a member account as the root user (p. 64).

AWS CLI & AWS SDKs

To create an AWS account that automatically is part of your organization

You can use one of the following commands to create an account:

- AWS CLI: create-account

```
$ aws organizations create-account \
    --email susan@example.com \
    --account-name "Production Account"
{
        "CreateAccountStatus": {
                "State": "IN_PROGRESS",
                "Id": "car-examplecreateaccountrequestid111"
        }
}
```

You can then check the status of the account creation with the following command.

```
$ aws organizations describe-create-account-status \
    --create-account-request-id car-examplecreateaccountrequestid111
{
  "CreateAccountStatus": {
    "State": "SUCCEEDED",
    "AccountId": "555555555555",
    "AccountName": "Production account",
    "RequestedTimestamp": 1470684478.687,
    "CompletedTimestamp": 1470684532.472,
    "Id": "car-examplecreateaccountrequestid111"
  }
}
```

- AWS SDKs: CreateAccount

Updating alternate contacts in your organization

You can update alternate contacts for accounts within your organization using the AWS Organizations console, or programmatically using the AWS CLI or AWS SDKs. To learn how to update alternate contacts, see Accessing or updating the alternate contacts in the *AWS Account Management Reference*.

Accessing and administering the member accounts in your organization

When you create an account in your organization, in addition to the root user, AWS Organizations automatically creates an IAM role that is by default named `OrganizationAccountAccessRole`. You can specify a different name when you create it, however we recommend that you name it consistently across all of your accounts. We refer to the role in this guide by the default name. AWS Organizations doesn't create any other IAM users, groups, or other roles. To access the accounts in your organization, you must use one of the following methods:

- The account has a root user that you can use to sign in. We recommend that you use the root user only to create IAM users, groups, and roles and then always sign in with one of those. See Accessing a member account as the root user (p. 64).
- If you create an account by using the tools provided as part of AWS Organizations, you can access the account by using the preconfigured role named `OrganizationAccountAccessRole` that exists in all new accounts that you create this way. See Accessing a member account that has a management account access role (p. 66).
- If you invite an existing account to join your organization and the account accepts the invitation, you can then choose to create an IAM role that allows the management account to access the invited member account. This role is intended to be identical to the role automatically added to an account that is created with AWS Organizations. To create this role, see Creating the OrganizationAccountAccessRole in an invited member account (p. 64). After you create the role, you can access it using the steps in Accessing a member account that has a management account access role (p. 66).
- Use AWS IAM Identity Center (successor to AWS Single Sign-On) and enable trusted access for IAM Identity Center with AWS Organizations. This allows users to sign in to the AWS access portal with their corporate credentials and access resources in their assigned management account or member accounts.

For more information, see Multi-account permissions in the *AWS IAM Identity Center (successor to AWS Single Sign-On) User Guide.* For information about setting up trusted access for IAM Identity Center, see AWS IAM Identity Center (successor to AWS Single Sign-On) and AWS Organizations (p. 302).

Minimum permissions
To access an AWS account from any other account in your organization, you must have the following permission:

- `sts:AssumeRole` – The `Resource` element must be set to either an asterisk (*) or the account ID number of the account with the user who needs to access the new member account

Accessing a member account as the root user

When you create a new account, AWS Organizations initially assigns a password to the root user that is a minimum of 64 characters long. All characters are randomly generated with no guarantees on the appearance of certain character sets. You can't retrieve this initial password. To access the account as the root user for the first time, you must go through the process for password recovery.

Notes

- As a best practice, we recommend that you don't use the root user to access your account except to create other users and roles with more limited permissions. Then sign in as one of those users or roles.
- We also recommend that you set multi-factor authentication (MFA) on the root user. Reset the password, and assign an MFA device to the root user.
- If you created a member account in an organization with an incorrect email address, you can’t sign in to the account as the root user. Contact AWS Billing and Support for assistance.

AWS Management Console

To request a new password for the root user of the member account

1. Go to the **Sign in** page of the AWS console at https://console.aws.amazon.com/. If you are already signed in to AWS, you have to sign out to see the sign-in page.
2. If the **Sign in** page shows three text boxes for **Account ID or alias**, **IAM user name**, and **Password**, choose **Sign in using root account credentials**.
3. Enter the email address that is associated with your AWS account and then choose **Next**.
4. Choose **Forgot your password?** and then enter the information that is required to reset the password to a new one that you provide. To do this, you must be able to access incoming mail sent to the email address that is associated with the account.

Creating the OrganizationAccountAccessRole in an invited member account

By default, if you create a member account as part of your organization, AWS automatically creates a role in the account that grants administrator permissions to IAM users in the management account who can assume the role. By default, that role is named `OrganizationAccountAccessRole`. For more information, see Accessing a member account that has a management account access role (p. 66).

However, member accounts that you *invite* to join your organization ***do not*** automatically get an administrator role created. You have to do this manually, as shown in the following procedure. This essentially duplicates the role automatically set up for created accounts. We recommend that you use the same name, `OrganizationAccountAccessRole`, for your manually created roles for consistency and ease of remembering.

AWS Management Console

To create an AWS Organizations administrator role in a member account

1. Sign in to the IAM console at https://console.aws.amazon.com/iam/. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the member account. The user or role must have permission to create IAM roles and policies.
2. In the IAM console, navigate to **Roles** and then choose **Create Role**.
3. Choose **Another AWS account**.
4. Enter the 12-digit account ID number of the management account that you want to grant administrator access to and choose **Next: Permissions**.

   For this role, because the accounts are internal to your company, you should **not** choose **Require external ID**. For more information about the external ID option, see When Should I Use the External ID? in the *IAM User Guide*.
5. If you have MFA enabled and configured, you can optionally choose to require authentication using an MFA device. For more information about MFA, see Using Multi-Factor Authentication (MFA) in AWS in the *IAM User Guide*.
6. On the **Attach permissions policies** page, choose the AWS managed policy named `AdministratorAccess` and then choose **Next: Tags**.
7. On the **Add tags (optional)** page, choose **Next: Review**.
8. On the **Review** page, specify a role name and an optional description. We recommend that you use `OrganizationAccountAccessRole`, for consistency with the default name assigned to the role in new accounts. To commit your changes, choose **Create role**.
9. Your new role appears on the list of available roles. Choose the new role's name to view its details, paying special note to the link URL that is provided. Give this URL to users in the member account who need to access the role. Also, note the **Role ARN** because you need it in step 15.
10. Sign in to the IAM console at https://console.aws.amazon.com/iam/. This time, sign in as a user in the management account who has permissions to create policies and assign the policies to users or groups.
11. Navigate to **Policies** and then choose **Create Policy**.
12. For **Service**, choose **STS**.
13. For **Actions**, start typing **`AssumeRole`** in the **Filter** box and then select the check box next to it when it appears.
14. Choose **Resources**, ensure that **Specific** is selected and then choose **Add ARN**.
15. Enter the AWS member account ID number and then enter the name of the role that you previously created in steps 1–8. Choose **Add**.
16. If you're granting permission to assume the role in multiple member accounts, repeats steps 14 and 15 for each account.
17. Choose **Review policy**.
18. Enter a name for the new policy and then choose **Create policy** to save your changes.
19. Choose **Groups** in the navigation pane and then choose the name of the group (not the check box) that you want to use to delegate administration of the member account.
20. Choose the **Permissions** tab.
21. Choose **Attach Policy**, select the policy that you created in steps 11–18, and then choose **Attach Policy**.

The users who are members of the selected group now can use the URLs that you captured in step 9 to access each member account's role. They can access these member accounts the same way as they would if accessing an account that you create in the organization. For more information about using the role to administer a member account, see Accessing a member account that has a management account access role (p. 66).

Accessing a member account that has a management account access role

When you create a member account using the AWS Organizations console, AWS Organizations *automatically* creates an IAM role named `OrganizationAccountAccessRole` in the account. This role has full administrative permissions in the member account. The scope of access for this role includes all principals in the management account, such that the role is configured to grant that access to the organization's management account. You can create an identical role for an invited member account by following the steps in Creating the OrganizationAccountAccessRole in an invited member account (p. 64). To use this role to access the member account, you must sign in as a user from the management account that has permissions to assume the role. To configure these permissions, perform the following procedure. We recommend that you grant permissions to groups instead of users for ease of maintenance.

AWS Management Console

To grant permissions to members of an IAM group in the management account to access the role

1. Sign in to the IAM console at https://console.aws.amazon.com/iam/ as a user with administrator permissions in the management account. This is required to delegate permissions to the IAM group whose users will access the role in the member account.
2. Start by creating the managed policy that you need later in Step 11 (p. 66).

   In the navigation pane, choose **Policies** and then choose **Create policy**.
3. On the Visual editor tab, choose **Choose a service**, type **`STS`** in the search box to filter the list, and then choose the **STS** option.
4. In the **Actions** section, type **`assume`** in the search box to filter the list, and then choose the **AssumeRole** option.
5. In the Resources section, choose **Specific**, choose **Add ARN to restrict access**, and then type the member account number and the name of the role that you created in the previous section (we recommended naming it `OrganizationAccountAccessRole`).
6. Choose Add when the dialog box displays the correct ARN.
7. (Optional) If you want to require multi-factor authentication (MFA), or restrict access to the role from a specified IP address range, then expand the Request conditions section, and select the options you want to enforce.
8. Choose **Review policy**.
9. In the **Name** field, enter a name for your policy. For example : **`GrantAccessToOrganizationAccountAccessRole`**. You can also add an optional description.
10. Choose **Create policy** to save your new managed policy.
11. Now that you have the policy available, you can attach it to a group.

    In the navigation pane, choose **Groups** and then choose the name of the group (not the check box) whose members you want to be able to assume the role in the member account. If necessary, you can create a new group.
12. Choose the **Permissions** tab and then under **Managed Policies,** choose **Attach policy**.

13. (Optional) In the **Search** box, you can start typing the name of your policy to filter the list until you can see the name of the policy you just created in Step 2 (p. 66) through Step 10 (p. 66). You can also filter out all of the AWS managed policies by choosing **Policy Type** and then choosing **Customer Managed**.
14. Check the box next to your policy, and then choose **Attach Policy**.

IAM users that are members of the group now have permissions to switch to the new role in the AWS Organizations console by using the following procedure.

AWS Management Console

To switch to the role for the member account

When using the role, the user has administrator permissions in the new member account. Instruct your IAM users who are members of the group to do the following to switch to the new role.

1. From the upper-right corner of the AWS Organizations console, choose the link that contains your current sign-in name and then choose **Switch Role**.
2. Enter the administrator-provided account ID number and role name.
3. For **Display Name**, enter the text that you want to show on the navigation bar in the upper-right corner in place of your user name while you are using the role. You can optionally choose a color.
4. Choose **Switch Role**. Now all actions that you perform are done with the permissions granted to the role that you switched to. You no longer have the permissions associated with your original IAM user until you switch back.
5. When you finish performing actions that require the permissions of the role, you can switch back to your normal IAM user. Choose the role name in the upper-right corner (whatever you specified as the **Display Name**) and then choose **Back to *`UserName`***.

Additional resources

- For more information about granting permissions to switch roles, see Granting a User Permissions to Switch Roles in the *IAM User Guide*.
- For more information about using a role that you have been granted permissions to assume, see Switching to a Role (AWS Management Console) in the *IAM User Guide*.
- For a tutorial about using roles for cross-account access, see Tutorial: Delegate Access Across AWS accounts Using IAM Roles in the *IAM User Guide*.
- For information about closing AWS accounts, see Closing an AWS account (p. 74).

Exporting all AWS account details for your organization

With AWS Organizations, management account users and delegated administrators for an organization can export a .csv file with all account details within an organization. As a result, organization administrators can easily view accounts and filter by status: `ACTIVE`, `SUSPENDED`, or `PENDING`. If your organization has many accounts, the .csv file download option provides an easy way to view and sort account details in a spreadsheet.

Previously, the only way to view accounts was to look at the account hierarchy or list display in the AWS Organizations console.

Note
Only principals in the management account can download the account list.

Exporting a list of all AWS accounts in your organization

When you sign in to the organization's management account, you can get a list of all accounts that are part of your organization as a .csv file. The list contains individual account details; however, it doesn't specify to which organizational unit (OU) the account belongs.

The .csv file contains the following information for each account:

- **Account ID** - Numeric account identifier. For example: 123456789012
- **ARN** - Amazon Resource Name for the account. For example: `arn:aws:organizations::123456789012account/o-o1gb0d1234/123456789012`
- **Email** - Email address associated with the account. For example: marymajor@example.com
- **Name** - Account name provided by account creator. For example: stage testing account
- **Status** - Account status within the organization. Value can be `PENDING`, `ACTIVE` or `SUSPENDED`.
- **Joined method** - Specifies how the account was created. Value can be `INVITED`, `CREATED` or `ADDED`.
- **Joined timestamp** - Date and time the account joined the organization.

Minimum permissions
To export a .csv file with all member accounts in your organization, you must have the following permissions:

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

AWS Management Console

To export a .csv file for all AWS accounts in your organization

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. Choose **Actions**, then for **AWS account** choose **Export account list**. The blue banner at the top of the page indicates "Export is in progress!"
3. When the file is ready, the banner turns green and indicates: "Download is ready!" Choose **Download CSV**. The file `Organization_accounts_information.csv` downloads to your device.

AWS CLI & AWS SDKs

The only way to export the .csv file with account details is by using the AWS Management Console. You can't export the account list .csv file using the AWS CLI.

Removing a member account from your organization

Part of managing accounts in an organization is removing *member* accounts that you no longer need. This page describes what you need to know before removing an account and provides procedures for removing accounts.

For information on removing the *management account*, see Deleting the organization by removing the management account (p. 48).

Topics

Before removing an account from an organization

Before you remove an account, it's important to know the following:

- You can remove an account from your organization only if the account has the information that is required for it to operate as a standalone account. When you create an account in an organization using the AWS Organizations console, API, or AWS CLI commands, all the information that is required of standalone accounts is not automatically collected. For each account that you want to make standalone, you must choose a support plan, provide and verify the required contact information, and provide a current payment method. AWS uses the payment method to charge for any billable (not AWS Free Tier) AWS activity that occurs while the account isn't attached to an organization.
- To remove an account that you created in the organization, you must wait until at least seven days after the account was created. Invited accounts aren't subject to this waiting period.
- At the moment the account successfully leaves the organization, the owner of the AWS account becomes responsible for all new AWS costs accrued, and the account's payment method is used. The management account of the organization is no longer responsible.
- The account that you want to remove must not be a delegated administrator account for any AWS service enabled for your organization. If the account is a delegated administrator, you must first change the delegated administrator account to another account that is remaining in the organization. For more information about how to disable or change the delegated administrator account for an AWS service, see the documentation for that service
- Even after the removal of created accounts (accounts created using the AWS Organizations console or the `CreateAccount` API) from within an organization, (i) created accounts are governed by the terms of the creating management account's agreement with us, and (ii) the creating management account remains jointly and severally liable for any actions taken by its created accounts. Customers' agreements with us, and the rights and obligations under those agreements, cannot be assigned or transferred without our prior consent. To obtain our consent, contact us at https://aws.amazon.com/contact-us/.
- When a member account leaves an organization, that account no longer has access to cost and usage data from the time range when the account was a member of the organization. However, the management account of the organization can still access the data. If the account rejoins the organization, the account can access that data again.
- When a member account leaves an organization, all tags attached to the account are deleted.

Effects of removing an account from an organization

When you remove an account from an organization, no direct changes are made to the account. However, the following indirect effects occur:

- The account is now responsible for paying its own charges and must have a valid payment method attached to the account.
- The principals in the account are no longer affected by any policies (p. 84) that applied in the organization. This means that restrictions imposed by SCPs are gone, and the users and roles in the account might have more permissions than they had before. Other organization policy types can no longer be enforced or processed.
- If you use the `aws:PrincipalOrgID` condition key in any policies to restrict access to only users and roles from AWS accounts in your organization, then you should review, and possibly update these policies before removing the member account. If you don't update the policies, then users and roles in the account could lose access to the resources when the account leaves the organization.
- Integration with other services might be disabled. If you remove an account from an organization that has integration with an AWS service enabled, the users in that account can no longer use that service.

Removing a member account from your organization

When you sign in to the organization's management account, you can remove member accounts from the organization that you no longer need. To do this, complete the following procedure. These procedures apply only to member accounts. To remove the management account, you must delete the organization (p. 48).

> **Note**
> If a member account is removed from an organization, that member account will no longer be covered by organization agreements. Management account administrators should communicate this to member accounts before removing member accounts from the organization, so that member accounts can put new agreements in place if necessary. A list of active organization agreements can be viewed in the AWS Artifact console on the AWS Artifact Organization Agreements page.
>
> **Minimum permissions**
> To remove one or more member accounts from your organization, you must sign in as an IAM user or role in the management account with the following permissions:
>
> - `organizations:DescribeOrganization` – required only when using the Organizations console
> - `organizations:RemoveAccountFromOrganization`
>
> If you choose to sign in as an IAM user or role in a member account in step 6, then that user or role must have the following permissions:
>
> - `organizations:DescribeOrganization` – required only when using the Organizations console.
> - `organizations:LeaveOrganization` – Note that the organization administrator can apply a policy to your account that removes this permission, preventing you from removing your account from the organization.
> - If you sign in as an IAM user and the account is missing payment information, the IAM user must have the permissions `aws-portal:ModifyBilling` and `aws-portal:ModifyPaymentMethods`. Also, the member account must have IAM user access to billing enabled. If this isn't already enabled, see Activating Access to the Billing and Cost Management Console in the *AWS Billing User Guide*.

AWS Management Console

To remove a member account from your organization

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page, find and choose the check box☑ next to each member account that you want to remove from your organization. You can navigate the OU hierarchy or enable **View AWS accounts only** to see a flat list of accounts without the OU structure. If you have a lot of accounts, you might have to choose **Load more accounts in '*ou-name*'** at the bottom of the list to find all of those you want to move.

   On the **AWS accounts** page, find and choose the name of the member account that you want to remove from your organization. You might have to expand OUs (choose the ▶) to find the account that you want.
3. Choose **Actions**, then under **AWS account**, choose **Remove from organization**.
4. In the **Remove account '*account-name*' (#*account-id-num*) from organization?** dialog box, choose **Remove account**.
5. If AWS Organizations fails to remove one or more of the accounts, it's typically because you have not provided all the required information for the account to operate as a standalone account. Perform the following steps:

   a. Sign in to the failed accounts. We recommend that you sign in to the member account by choosing **Copy link**, and then pasting it into the address bar of a new incognito browser window. If you don't use an incognito window, you're signed out of the management account and won't be able to navigate back to this dialog box.

   b. The browser takes you directly to the sign-up process to complete any steps that are missing for this account. Complete all the steps presented. They might include the following:

      - Provide contact information
      - Provide a valid payment method
      - Verify the phone number
      - Select a support plan option

   c. After you complete the last sign-up step, AWS automatically redirects your browser to the AWS Organizations console for the member account. Choose **Leave organization**, and then confirm your choice in the confirmation dialog box. You are redirected to the **Getting Started** page of the AWS Organizations console, where you can view any pending invitations for your account to join other organizations.

AWS CLI & AWS SDKs

To remove a member account from your organization

You can use one of the following commands to remove a member account:

- AWS CLI: remove-account-from-organization

  ```
  $ aws organizations remove-account-from-organization \
      --account-id 123456789012
  ```

  This command produces no output when successful.
- AWS SDKs: RemoveAccountFromOrganization

Leaving an organization as a member account

When you sign in to a member account, you can remove that one account from its organization. To do this, complete the following procedure. The management account can't leave the organization using this technique. To remove the management account, you must delete the organization.

The account that you want to remove must not be a delegated administrator account for any AWS service enabled for your organization. If the account is a delegated administrator, you must first change the delegated administrator account to another account that is remaining in the organization. For more information about how to disable or change the delegated administrator account for an AWS service, see the documentation for that service

Important
If you leave an organization, you are no longer covered by organization agreements that were accepted on your behalf by the management account of the organization. You can view a list of these organization agreements in the AWS Artifact console on the AWS Artifact Organization Agreements page. Before leaving the organization, you should determine (with the assistance of your legal, privacy, or compliance teams where appropriate) whether it is necessary for you to have new agreement(s) in place.

Note
An account’s status with an organization affects what cost and usage data is visible:

- If a member account leaves an organization and becomes a standalone account, the account no longer has access to cost and usage data from the time range when the account was a member of the organization. The account has access only to the data that is generated as a standalone account.
- If a member account leaves organization A to join organization B, the account no longer has access to cost and usage data from the time range when the account was a member of organization A. The account has access only to the data that is generated as a member of organization B.
- If an account rejoins an organization that it previously belonged to, the account regains access to its historical cost and usage data.

Minimum permissions
To leave an AWS organization, you must have the following permissions:

- `organizations:DescribeOrganization` – required only when using the Organizations console.
- `organizations:LeaveOrganization` – Note that the organization administrator can apply a policy to your account that removes this permission, preventing you from removing your account from the organization.
- If you sign in as an IAM user and the account is missing payment information, the IAM user must have the permissions `aws-portal:ModifyBilling` and `aws-portal:ModifyPaymentMethods`. Also, the member account must have IAM user access to billing enabled. If this isn't already enabled, see Activating Access to the Billing and Cost Management Console in the *AWS Billing User Guide*.

AWS Management Console

To leave an organization as a member account

1. Sign in to the AWS Organizations console at AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in a member account.

By default, you don't have access to the root user password in a member account that was created using AWS Organizations. If required, recover the root user password by following the steps at Accessing a member account as the root user (p. 64).

2. On the **Organizations Dashboard** page, choose **Leave organization**.
3. Perform one of the following steps:

   - If your account has all the required information to operate as a standalone account, a confirmation dialog box appears. Confirm your choice to remove the account. You are redirected to the **Getting Started** page of the AWS Organizations console, where you can view any pending invitations for your account to join other organizations.
   - If your account doesn't have all the required information, perform the following steps:

     a. A dialog box appears to explain that you must complete some additional steps. Click the link.

     b. Complete all the sign-up steps that are presented. They might include the following:

        - Provide contact information
        - Provide a valid payment method
        - Verify the phone number
        - Select a support plan option

     c. When you see the dialog box stating that the sign-up process is complete, choose **Leave organization**.

     d. A confirmation dialog box appears. Confirm your choice to remove the account. You are redirected to the **Getting Started** page of the AWS Organizations console, where you can view any pending invitations for your account to join other organizations.
4. Remove the IAM roles that grant access to your account from the organization.

   > **Important**
   > If your account was created in the organization, then Organizations automatically created an IAM role in the account that enabled access by the organization's management account. If the account was invited to join, then Organizations did not automatically create such a role, but you or another administrator might have created one to get the same benefits. In either case, when you remove the account from the organization, any such role isn't automatically deleted. If you want to terminate this access from the former organization's management account, then you must manually delete this IAM role. For information about how to delete a role, see Deleting roles or instance profiles in the *IAM User Guide*.

AWS CLI & AWS SDKs

To leave an organization as a member account

You can use one of the following commands to leave an organization:

- AWS CLI: leave-organization

  The following example causes the account whose credentials are used to run the command to leave the organization.

  ```
  $ aws organizations leave-organization
  ```

  This command produces no output when successful.
- AWS SDKs: LeaveOrganization

Closing an AWS account

This topic applies *only* to AWS accounts
To close an **Amazon.com shopping account**, see http://www.amazon.com/gp/help/customer/display.html?nodeId=GDK92DNLSGWTV6MP.

If you no longer need a member account in your organization, and want to ensure that no one can accrue charges for it, you can close the account from the AWS Organizations console following the instructions in this section. You can also close an AWS account from the AWS Billing Console. To learn more, see Closing an account in the *AWS Billing User Guide*.

Before closing your account, back up any applications and data that you want to retain.

Immediately, the account can no longer be used for any AWS activity other than signing in as the root user to view past bills or to contact AWS Support. For more information, see Contacting Customer Support About Your Bill.

Impacts of closing an account

When you close an AWS account, there are some impacts that you should consider prior to account closure.

- The root user's email address can't be reused if you close an account.
- To close the management account for the organization, you must first either remove (p. 70) or close all member accounts in the organization. By closing the management account, it automatically deletes the organization, as long as there are no member accounts in the organization.
- If you use AWS Control Tower, you need to unmanage the accounts before you attempt to close an account. See Unmanage a member account in the *AWS Control Tower User Guide*.
- If you have an AWS account that is linked to a AWS GovCloud (US) account, you need to close the standard account before you close the AWS GovCloud (US) account. To learn important pre-closure details, see Closing an AWS GovCloud (US) account in the *AWS GovCloud (US) User Guide*.

Best practice we recommend but is not required for security:

As a best practice, we recommend that you remove references from any IAM permissions or policies to closed accounts, in accordance with the security best practice of granting the least privilege needed to get the job done. This is not a security issue because AWS never reuses an ID number after the account is closed. IAM Access Analyzer will notify you if you have an ID for a closed account in an IAM policy.

From the time you close the account until 90 days expire:

- Closed accounts are visible in your organization with the SUSPENDED state.
- Some active resources that are not terminated prior to account closure can continue to incur fees if you decide to reopen the account within 90 days. For more information, see How do I terminate active resources I no longer need on my AWS account? in the Knowledge Center.
- You will be able to log in to view past bills and access AWS Support.

After the 90 day grace period expires:

- A closed AWS account is no longer visible in your organization.
- AWS accounts are no longer eligible for reinstatement. At this point, any AWS resources that were in the account can’t be recovered.

Closing an AWS account

When you sign in to the organization's management account, you can close member accounts that are part of your organization. To do this, complete the following steps.

AWS Management Console

To close an AWS account

> **Note**
> Optionally, you can close an AWS member account from the Billing and Cost Management console. To learn more, see Closing an account in the *AWS Billing User Guide*.

1. Before closing your account, back up any applications and data that you want to retain.
2. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
3. On the **AWS accounts** page, find and choose the name of the member account you want to close. You can navigate the OU hierarchy, or look at a flat list of accounts without the OU structure.
4. Choose **Close** next to the account name at the top of the page.
5. Select each check box to acknowledge all required account closure statements.
6. Enter the member account ID and then choose **Close Account**.

After you close an AWS account, you can no longer use it to access AWS services or resources. You can contact AWS Support within the Post-Closure Period to reopen the account. For more information, see How do I reopen my closed AWS account? in the Knowledge Center.

AWS CLI & AWS SDKs

To close an AWS account

You can use one of the following commands to close an AWS account:

- AWS CLI: close-account

  ```
  $ aws organizations close-account \
      --account-id 123456789012
  ```

  This command produces no output when successful.
- AWS SDKs: CloseAccount

Protecting accounts from closure

If you want to protect an AWS account from accidental closure, you can create an IAM policy to specify which accounts are exempt from closure. Any member account protected with these policies can't be closed. This can't be accomplished with an SCP, because they don't affect principals in the management account.

You can create an IAM policy that denies closing accounts in either of two ways:

- Explicitly list each account that you want to protect in the policy by including the `arn` in the `Resource` element. To see an example, see Prevent accounts listed in this policy from getting closed (p. 76).
- Tag individual accounts to prevent them from getting closed. Use the `aws:ResourceTag` tag global condition key in your policy to prevent any account with the tag from being closed. To learn how to tag

an account, see Tagging Organizations resources (p. 220). To see an example, see Prevent accounts with tag from getting closed (p. 76).

Example IAM policies that prevent AWS account closures

Examples in this category

- Prevent accounts with tag from getting closed (p. 76)
- Prevent accounts listed in this policy from getting closed (p. 76)

Prevent accounts with tag from getting closed

You can attach the following policy to an identity in your management account. This policy prevents principals in the management account from closing any member account that is tagged with the `aws:ResourceTag` tag global condition key, the `AccountType` key and the `Critical` tag value.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PreventCloseAccountForTaggedAccts",
            "Effect": "Deny",
            "Action": "organizations:CloseAccount",
            "Resource": "*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/AccountType": "Critical"}
            }
        }
    ]
}
```

Prevent accounts listed in this policy from getting closed

You can attach the following policy to an identity in your management account. This policy prevents principals in the management account from closing accounts explicitly specified in the `Resource` element.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PreventCloseAccount",
            "Effect": "Deny",
            "Action": "organizations:CloseAccount",
            "Resource": [
                "arn:aws:organizations::555555555555:account/o-12345abcdef/123456789012",
                "arn:aws:organizations::555555555555:account/o-12345abcdef/123456789014"
            ]
        }
    ]
}
```

Managing organizational units (OUs)

You can use organizational units (OUs) to group accounts together to administer as a single unit. This greatly simplifies the management of your accounts. For example, you can attach a policy-based control to an OU, and all accounts within the OU automatically inherit the policy. You can create multiple OUs within a single organization, and you can create OUs within other OUs. Each OU can contain multiple accounts, and you can move accounts from one OU to another. However, OU names must be unique within a parent OU or root.

> **Note**
> There is one root in the organization, which AWS Organizations creates for you when you first set up your organization.

To structure the accounts in your organization, you can perform the following tasks:

- Viewing details of an OU (p. 44)
- Creating an OU (p. 78)
- Renaming an OU (p. 79)
- Edit tags attached to an OU (p. 80)
- Moving an account to an OU or between the root and OUs (p. 81)

Navigating the root and OU hierarchy

To navigate to different OUs or to the root when moving accounts or attaching policies, you can use the default "tree" view.

AWS Management Console

To navigate the organization as a 'tree'

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page, at the top of the **Organization** section, select the **Hierarchy** toggle (instead of **List**).
3. The tree initially appears showing the root, displaying only the first level of child OUs and accounts. To expand the tree to show deeper levels, choose the expand icon (▶)next to any parent entity. To reduce clutter and collapse a branch of the tree, choose the collapse icon (▼) next to an expanded parent entity.
4. Choose the name of an OU or root to view its details and perform certain operations. Alternatively, you can choose the radio button next to the name, and perform certain operations on that entity in the **Actions** menu.

You can also view the list of only the accounts in your organization in tabular form, without having to first navigate to an OU to find them. In this view you can't see any of the OUs or manipulate the policies attached to them.

AWS Management Console

To view the organization as a flat list of accounts with no hierarchy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.

2. On the **AWS accounts** page, at the top of the Organization section, choose the **View AWS accounts only** switch icon to turn it on. ●.
3. The list of accounts is displayed without any hierarchy.

Creating an OU

When you sign in to your organization's management account, you can create an OU in your organization's root. OUs can be nested up to five levels deep. To create an OU, complete the following steps.

Important

If this organization is managed with AWS Control Tower, then create your OUs with the AWS Control Tower console or APIs. If you create the OU in Organizations, then that OU isn't registered with AWS Control Tower. For more information, see Referring to Resources Outside of AWS Control Tower in the *AWS Control Tower User Guide*.

Minimum permissions

To create an OU within a root in your organization, you must have the following permissions:

- `organizations:DescribeOrganization` – required only when using the Organizations console
- `organizations:CreateOrganizationalUnit`

AWS Management Console

To create an OU

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. Navigate to the **AWS accounts** page.

   The console displays the Root OU and its contents. The first time you visit the Root, the console displays all of your AWS accounts in that top-level view. If you previously created OUs and moved accounts into them, the console shows only the top-level OUs and any accounts that you have not yet moved into an OU.
3. (Optional) If you want to create an OU inside an existing OU, navigate to the child OU (p. 77) by choosing the name (not the check box) of the child OU, or by choosing the ▶ next to OUs in the tree view until you see the one you want, and then choosing its name.
4. When you've selected the correct parent OU in the hierarchy, on the **Actions** menu, under **Organizational Unit**, choose **Create new**
5. In the **Create organizational unit** dialog box, enter the name of the OU that you want to create.
6. (Optional) Add one or more tags by choosing **Add tag** and then entering a key and an optional value. Leaving the value blank sets it to an empty string; it isn't `null`. You can attach up to 50 tags to an OU.
7. Finally, choose **Create organizational unit**.

Your new OU appears inside the parent. You now can move accounts to this OU (p. 81) or attach policies to it.

AWS CLI & AWS SDKs

To create an OU

You can use one of the following commands to create an OU:

- AWS CLI: create-organizational-unit

  To create an OU, you must first find the identity of the root or OU that you want to be the parent of the new OU.

  To find the identity of the root, use the list-roots command. To find the identity of an OU, use the list-children to navigate to the OU you want.

  The following example shows how to find the identity of the root, and then find the identity of an OU under the root. The last command shows how to create a new OU in that found OU.

```
$ aws organizations list-roots
{
    "Roots": [
        {
            "Id": "r-a1b2",
            "Arn": "arn:aws:organizations::123456789012:root/o-aa111bb222/r-a1b2",
            "Name": "Root",
            "PolicyTypes": []
        }
    ]
}
$ aws organizations list-children \
    --parent-id r-a1b2 \
    --child-type ORGANIZATIONAL_UNIT
{
    "Children": [
        {
            "Id": "ou-a1b2-f6g7h111",
            "Type": "ORGANIZATIONAL_UNIT"
        }
    ]
}
$ aws organizations create-organizational-unit \
    --parent-id ou-a1b2-f6g7h111 \
    --name New-Child-OU
{
    "OrganizationalUnit": {
        "Id": "ou-a1b2-f6g7h222",
        "Arn": "arn:aws:organizations::123456789012:ou/o-aa111bb222/ou-a1b2-
f6g7h222",
        "Name": "New-Child-OU"
    }
}
```

- AWS SDKs: CreateOrganizationalUnit

Renaming an OU

When you sign in to your organization's management account, you can rename an OU. To do this, complete the following steps.

Minimum permissions
To rename an OU within a root in your AWS organization, you must have the following permissions:

- `organizations:DescribeOrganization` – required only when using the Organizations console
- `organizations:UpdateOrganizationalUnit`

AWS Management Console

To rename an OU

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **AWS accounts** page, navigate to the OU (p. 77) that you want to rename, and then do one of the following steps:
   - Choose the radio button ◉ next to the OU that you want to rename. Then, on the **Actions** menu, under **Organizational unit**, choose **Rename**.
   - Choose the OU's name, to access the OU's detail page. Then, at the top of the page choose **Rename**.
3. In the **Rename organizational unit** dialog box, enter a new name, and then choose **Save changes**.

AWS CLI & AWS SDKs

To rename an OU

You can use one of the following commands to rename an OU:

- AWS CLI: update-organizational-unit

  The following example shows how to rename an OU.

```
$ aws organizations update-organizational-unit \
    --organizational-unit-id ou-a1b2-f6g7h222 \
    --name "Renamed-OU"
{
    "OrganizationalUnit": {
        "Id": "ou-a1b2-f6g7h222",
        "Arn": "arn:aws:organizations::123456789012:ou/o-aa111bb222/ou-a1b2-
f6g7h222",
        "Name": "Renamed-OU"
    }
}
```

- AWS SDKs: UpdateOrganizationalUnit

Editing tags attached to an OU

When you sign in to your organization's management account, you can add or remove the tags attached to an OU. To do this, complete the following steps.

Minimum permissions
To edit the tags attached to an OU within a root in your AWS organization, you must have the following permissions:

- `organizations:DescribeOrganization` – required only when using the Organizations console
- `organizations:DescribeOrganizationalUnit`– required only when using the Organizations console
- `organizations:TagResource`
- `organizations:UntagResource`

AWS Management Console

To edit the tags attached to an OU

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **AWS accounts** page, navigate to and choose the name of the OU (p. 77) whose tags you want to edit.
3. On the OU's details page, choose the **Tags** tab, and then choose **Manage tags**.
4. You can perform any of these actions on this tab:
   - Edit the value for any tag by entering a new value over the old one. You can't modify the tag key. To change a key, you must delete the tag with the old key and add a tag with the new key.
   - Remove an existing tag by choosing **Remove** next to the tag you want to remive.
   - Add a new tag key and value pair. Choose **Add tag**, then enter the new key name and optional value in the provided boxes. If you leave the **Value** box empty, the value is an empty string; it isn't `null`.
5. Choose **Save changes** after you've made all the additions, removals, and edits you want to make.

AWS CLI & AWS SDKs

To edit the tags attached to an OU

You can use one of the following commands to change the tags attached to an OU:

- AWS CLI: tag-resource and untag-resource

  The following example attaches the tag `"Department"="12345"` to an OU. Note that `Key` and `Value` are case sensitive.

  ```
  $ aws organizations tag-resource \
      --resource-id ou-a1b2-f6g7h222 \
      --tags Key=Department,Value=12345
  ```

  This command produces no output when successful.

  The following example removes the `Department` tag from an OU.

  ```
  $ aws organizations untag-resource \
      --resource-id ou-a1b2-f6g7h222 \
      --tag-keys Department
  ```

  This command produces no output when successful.
- AWS SDKs: TagResource and UntagResource

Moving accounts to an OU or between the root and OUs

When you sign in to your organization's management account, you can move accounts in your organization from the root to an OU, from one OU to another, or back to the root from an OU. Placing an account inside an OU makes it subject to any policies that are attached to the parent OU and any OUs in the parent chain up to the root. If an account isn't in an OU, it's subject to only the policies that are attached directly to the root and any policies that are attached directly to the account. To move accounts, complete the following steps.

Minimum permissions

To move accounts to a new location in the OU hierarchy, you must have the following permissions:

- `organizations:DescribeOrganization` – required only when using the Organizations console
- `organizations:MoveAccount`

AWS Management Console

To move accounts to an OU

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **AWS accounts** page, find the account or accounts that you want to move. You can navigate the OU hierarchy or enable **View AWS accounts only** to see a flat list of accounts without the OU structure. If you have a lot of accounts, you might have to choose **Load more accounts in '*ou-name*'** at the bottom of the list to find all of those you want to move.
3. Choose the check box ☑ next to the name of each account that you want to move.
4. On the **Actions** menu, under **AWS account**, choose **Move** .
5. In the **Move AWS account** dialog box, navigate to and then choose the OU or root that you want to move the account to, and then choose **Move AWS account**.

AWS CLI & AWS SDKs

To move an account to an OU

You can use one of the following commands to move an account:

- AWS CLI: move-account

  The following example moves an AWS account from the root to an OU. Note that you must specify the IDs of both the source and destination containers.

  ```
  $ aws organizations move-account \
      --account-id 111122223333 \
      --source-parent-id r-a1b2 \
      --destination-parent-id ou-a1b2-f6g7h111
  ```

  This command produces no output when successful.
- AWS SDKs: MoveAccount

Deleting OUs

When you sign in to your organization's management account, you can delete any OUs that you no longer need.

You must first move all accounts out of the OU and any child OUs, and then you can delete the child OUs.

Minimum permissions

To delete an OU, you must have the following permissions:

- `organizations:DescribeOrganization` – required only when using the Organizations console
- `organizations:DeleteOrganizationalUnit`

AWS Management Console

To delete an OU

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **AWS accounts** page, find the OUs that you want to delete and choose the check box ☑ next to each OU's name.
3. Choose **Actions**, and then under **Organizational unit**, choose **Delete**.
4. To confirm that you want to delete the OUs, enter the OU's name (if you chose to delete only one) or the word 'delete' (if you chose more than one), and then choose **Delete**.

   AWS Organizations deletes the OUs and removes them from the list.

AWS CLI & AWS SDKs

To delete an OU

You can use one of the following commands to delete an OU:

- AWS CLI: delete-organizational-unit

  The following example shows how to delete an OU.

  ```
  $ aws organizations delete-organizational-unit \
      --organizational-unit-id ou-a1b2-f6g7h222
  ```

  This command produces no output when successful.
- AWS SDKs: DeleteOrganizationalUnit

Managing AWS Organizations policies

Policies in AWS Organizations enable you to apply additional types of management to the AWS accounts in your organization. You can use policies when all features are enabled (p. 35) in your organization.

The AWS Organizations console displays the enabled or disabled status for each policy type. On the **Organize accounts** tab, choose the `Root` in the left navigation pane. The details pane on the right side of the screen shows all of the available policy types. The list indicates which are enabled and which are disabled in that organization root. If the option to **Enable** a type is present, that type is currently disabled. If the option to **Disable** a type is present, that type is currently enabled.

Policy types

Organizations offers policy types in the following two broad categories:

Authorization policies

Authorization policies help you to centrally manage the security of the AWS accounts in your organization.

- **Service control policies (SCPs) (p. 104)** offer central control over the maximum available permissions for all of the accounts in your organization.

Management policies

Management policies enable you to centrally configure and manage AWS services and their features.

- **Artificial Intelligence (AI) services opt-out policies (p. 139)** enable you to control data collection for AWS AI services for all of your organization's accounts.
- **Backup policies (p. 154)** help you centrally manage and apply backup plans to the AWS resources across your organization's accounts.
- **Tag policies (p. 187)** help you standardize the tags attached to the AWS resources in your organization's accounts.

The following table summarizes some of the characteristics of each policy type:

Policy type	Affects management account	Maximum number you can attach to a root, OU, or account	Maximum size	Supports viewing effective policy for OU or account
SCP	⊗ No	5	5120 bytes	⊗ No
AI services opt-out policy	✓ Yes	5	2500 characters	✓ Yes
Backup policy	✓ Yes	10	10,000 characters	✓ Yes

Policy type	Affects management account	Maximum number you can attach to a root, OU, or account	Maximum size	Supports viewing effective policy for OU or account
Tag policy	✓ Yes	5	2500 characters	✓ Yes

Using policies in your organization

Enabling and disabling policy types

Enabling a policy type

Before you can create and attach a policy to your organization, you must enable that policy type for use. Enabling a policy type is a one-time task on the organization root. You can enable a policy type from only the organization's management account.

Minimum permissions

To enable a policy type, you need permission to run the following actions:

- `organizations:EnablePolicyType`
- `organizations:DescribeOrganization` – required only when using the Organizations console
- `organizations:ListRoots` – required only when using the Organizations console

AWS Management Console

To enable a policy type

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Policies** page, choose the name of the policy type that you want to enable.
3. On the policy type page, choose **Enable** ***policy type***.

   The page is replaced by a list of the available policies of the specified type.

AWS CLI & AWS SDKs

To enable a policy type

You can use one of the following commands to enable a policy type:

- AWS CLI: enable-policy-type

The following example shows how to enable backup policies for your organization. Note that you must specify the ID of your organization's root.

```
$ aws organizations enable-policy-type \
    --root-id r-a1b2 \
    --policy-type BACKUP_POLICY
{
    "Root": {
        "Id": "r-a1b2",
        "Arn": "arn:aws:organizations::123456789012:root/o-aa111bb222/r-a1b2",
        "Name": "Root",
        "PolicyTypes": [
            {
                "Type": "BACKUP_POLICY",
                "Status": "ENABLED"
            }
        ]
    }
}
```

The list of `PolicyTypes` in the output now includes the specified policy type with the `Status` of `ENABLED`.

- AWS SDKs: EnablePolicyType

Disabling a policy type

If you no longer want to use a certain policy type in your organization, you can disable that type to prevent its accidental use. You can disable a policy type from only the organization's management account.

Important

- When you disable a policy type, all policies of the specified type are automatically detached from all entities in the organization root. The policies are ***not*** deleted.
- (Service control policy type only) If you re-enable the SCP policy type later, all entities in the organization root are initially attached to only the default `FullAWSAccess` SCP. Attachments of SCPs to entities are lost when the SCPs are disabled in the organization. If you later want to re-enable SCPs, you must reattach them to the organization's root, OUs, and accounts, as appropriate.

Minimum permissions

To disable SCPs, you need permission to run the following actions:

- `organizations:DisablePolicyType`
- `organizations:DescribeOrganization` – required only when using the Organizations console
- `organizations:ListRoots` – required only when using the Organizations console

AWS Management Console

To disable a policy type

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **Policies** page, choose the name of the policy type that you want to disable.

3. On the policy type page, choose **Disable *policy type***.
4. On the confirmation dialog box, enter the word **disable**, and then choose **Disable**.

   The list of available policies of the specified type disappears.

AWS CLI & AWS SDKs

To disable a policy type

You can use one of the following commands to disable a policy type:

- AWS CLI: disable-policy-type

  The following example shows how to disable backup policies for your organization. Note that you must specify the ID of your organization's root.

```
$ aws organizations disable-policy-type \
    --root-id r-a1b2 \
    --policy-type BACKUP_POLICY
{
    "Root": {
        "Id": "r-a1b2",
        "Arn": "arn:aws:organizations::123456789012:root/o-aa111bb222/r-a1b2",
        "Name": "Root",
        "PolicyTypes": []
    }
}
```

  The list of `PolicyTypes` in the output no longer includes the specified policy type.
- AWS SDKs: DisablePolicyType

Getting information about your organization's policies

This section describes various ways to get details about the policies in your organization. These procedures apply to *all* policy types. You must enable a policy type on the organization root before you can attach policies of that type to any entities in that organization root.

Listing all policies

Minimum permissions
To list the policies within your organization, you must have the following permission:

- `organizations:ListPolicies`

You can view the policies in your organization in the AWS Management Console or by using an AWS Command Line Interface (AWS CLI) command or an AWS SDK operation.

AWS Management Console

To list all of the policies in your organization

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.

2. On the **Policies** page, choose the policy type that you want to list.

   If the specified policy type is enabled, the console displays a list of all of the policies of that type that are currently available in the organization.
3. Return to the **Policies** page and repeat for each policy type.

AWS CLI & AWS SDKs

To list all of the policies in your organization

You can use one of the following commands to list policies in an organization:

- AWS CLI: list-policies

  The following example shows how to get a list of all of the service control policies in your organization. You must specify the type of policy you want see. Repeat the command for each policy type you want to include.

```
$ aws organizations list-policies \
    --filter SERVICE_CONTROL_POLICY
{
    "Policies": [
        {
            "Id": "p-FullAWSAccess",
            "Arn": "arn:aws:organizations::aws:policy/service_control_policy/p-
FullAWSAccess",
            "Name": "FullAWSAccess",
            "Description": "Allows access to every operation",
            "Type": "SERVICE_CONTROL_POLICY",
            "AwsManaged": true
        }
    ]
}
```

- AWS SDKs: ListPolicies

Listing the policies attached to a root, OU, or account

Minimum permissions

To list the policies that are attached to a root, organizational unit (OU), or account within your organization, you must have the following permission:

- `organizations:ListPoliciesForTarget` with a `Resource` element in the same policy statement that includes the Amazon Resource Name (ARN) of the specified target (or "*")

AWS Management Console

To list all policies that are attached directly to a specified root, OU, or account

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page, choose the name of the root, OU, or account whose policies you want to view. You might have to expand OUs (choose the ▶) to find the OU that you want.
3. On the Root, OU, or account page, choose the **Policies** tab.

   The **Policies** tab displays all of the policies attached to that root, OU, or account, grouped by policy type.

AWS CLI & AWS SDKs

To list all policies that are attached directly to a specified root, OU, or account

You can use one of the following commands to list policies that are attached to an entity:

- AWS CLI: list-policies-for-target

  The following example lists all of the service control policies attached to the specified OU. You must specify both the ID of the root, OU, or account, and the type of policy that you want to list.

```
$ aws organizations list-policies-for-target \
    --target-id ou-a1b2-f6g7h222 \
    --filter SERVICE_CONTROL_POLICY
{
    "Policies": [
        {
            "Id": "p-FullAWSAccess",
            "Arn": "arn:aws:organizations::aws:policy/service_control_policy/p-
FullAWSAccess",
            "Name": "FullAWSAccess",
            "Description": "Allows access to every operation",
            "Type": "SERVICE_CONTROL_POLICY",
            "AwsManaged": true
        }
    ]
}
```

- AWS SDKs: ListPoliciesForTarget

Listing all roots, OUs, and accounts that a policy is attached to

Minimum permissions
To list the entities that a policy is attached to, you must have the following permission:

- `organizations:ListTargetsForPolicy` with a `Resource` element in the same policy statement that includes the ARN of the specified policy (or "*")

AWS Management Console

To list all roots, OUs, and accounts that have a specified policy attached

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Policies** page, choose the policy type, and then choose the name of the policy whose attachments you want to examine.
3. Choose the **Targets** tab, to display a table of every root, OU, and account that the chosen policy is attached to.

AWS CLI & AWS SDKs

To list all roots, OUs, and accounts that have a specified policy attached

You can use one of the following commands to list entities that have a policy:

- AWS CLI: list-targets-for-policy

The following example shows all of the attachments to root, OUs, and accounts for the specified policy.

```
$ aws organizations list-targets-for-policy \
    --policy-id p-FullAWSAccess
{
    "Targets": [
        {
            "TargetId": "ou-a1b2-f6g7h111",
            "Arn": "arn:aws:organizations::123456789012:ou/o-aa111bb222/ou-a1b2-
f6g7h111",
            "Name": "testou2",
            "Type": "ORGANIZATIONAL_UNIT"
        },
        {
            "TargetId": "ou-a1b2-f6g7h222",
            "Arn": "arn:aws:organizations::123456789012:ou/o-aa111bb222/ou-a1b2-
f6g7h222",
            "Name": "testou1",
            "Type": "ORGANIZATIONAL_UNIT"
        },
        {
            "TargetId": "123456789012",
            "Arn": "arn:aws:organizations::123456789012:account/o-
aa111bb222/123456789012",
            "Name": "My Management Account (bisdavid)",
            "Type": "ACCOUNT"
        },
        {
            "TargetId": "r-a1b2",
            "Arn": "arn:aws:organizations::123456789012:root/o-aa111bb222/r-a1b2",
            "Name": "Root",
            "Type": "ROOT"
        }
    ]
}
```

- AWS SDKs: ListTargetsForPolicy

Getting details about a policy

Minimum permissions

To display the details of a policy, you must have the following permission:

- `organizations:DescribePolicy` with a `Resource` element in the same policy statement that includes the ARN of the specified policy (or "*")

AWS Management Console

To get details about a policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Policies** page, choose the policy type of the policy that you want to examine, and then choose the name of the policy.

   The policy page displays the available information about the policy, including its ARN, description, and attached targets.

- The **Content** tab shows the current contents of the policy in JSON format.
- The **Targets** tab shows a list of the roots, OUs, and accounts to which the policy is attached.
- The **Tags** tab shows the tags attached to the policy. Note: the Tags tab is not available for AWS managed policies.

To edit the policy, choose **Edit policy**. Because each policy type has different editing requirements, see the instructions for creating and updating policies of your specified policy type.

AWS CLI & AWS SDKs

To get details about a policy

You can use one of the following commands to get details about a policy:

- AWS CLI: describe-policy

  The following example displays the details for the specified policy.

```
$ aws organizations describe-policy \
    --policy-id p-FullAWSAccess
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-FullAWSAccess",
            "Arn": "arn:aws:organizations::aws:policy/service_control_policy/p-
FullAWSAccess",
            "Name": "FullAWSAccess",
            "Description": "Allows access to every operation",
            "Type": "SERVICE_CONTROL_POLICY",
            "AwsManaged": true
        },
        "Content": "{\n  \"Version\": \"2012-10-17\",\n  \"Statement\": [\n    {\n
    \"Effect\": \"Allow\",\n      \"Action\": \"*\",\n      \"Resource\": \"*\"\n
  }\n  ]\n}"
    }
}
```

- AWS SDKs: DescribePolicy

Understanding policy inheritance

You can attach policies to organization entities (organization root, organizational unit (OU), or account) in your organization:

- When you attach a policy to the organization root, all OUs and accounts in the organization inherit that policy.
- When you attach a policy to a specific OU, accounts that are directly under that OU or any child OU inherit the policy.
- When you attach a policy to a specific account, it affects only that account.

Because you can attach policies to multiple levels in the organization, accounts can inherit multiple policies.

Exactly how policies affect the OUs and accounts that inherit them depends on the type of policy:

- Service control policies (SCPs) (p. 92)
- Management policy types (p. 94)
    - AI services opt-out policies
    - Backup policies
    - Tag policies

Inheritance for service control policies

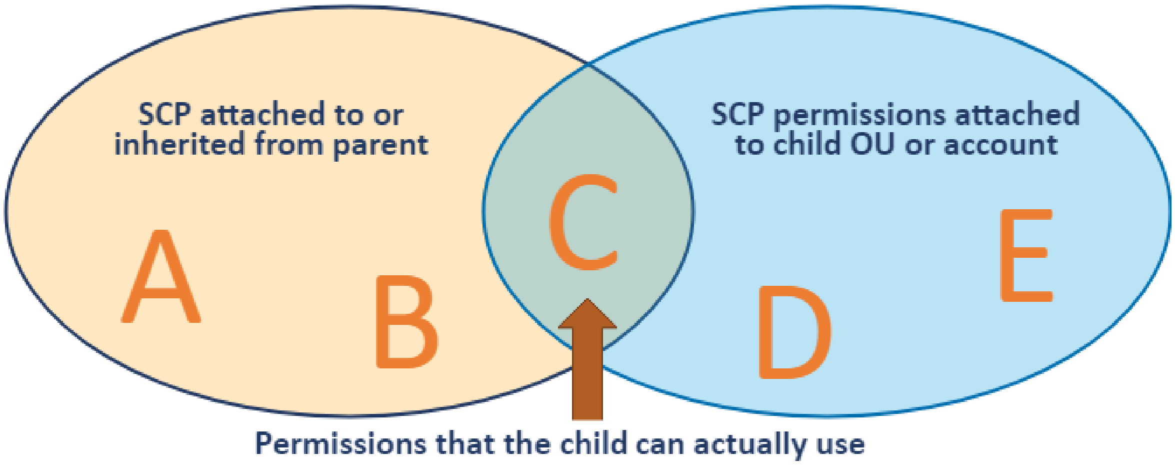
Important
The information in this section does ***not*** apply to management policy types, including AI services opt-out policies, backup policies, or tag policies. See the next section Policy syntax and inheritance for management policy types (p. 94).

Service control policies (SCPs)

Inheritance for service control policies behaves like a filter through which permissions flow to all parts of the tree below. Imagine that the inverted tree structure of the organization is made of branches that connect from the root to all of the OUs and end at the accounts. All AWS permissions flow into the root of the tree. Those permissions must then flow past the SCPs attached to the root, OUs, and the account to get to the principal (an IAM role or user) making a request. Each SCP can filter the permissions passing through to the levels below it. If an action is blocked by a `Deny` statement, then all OUs and accounts affected by that SCP are denied access to that action. An SCP at a lower level can't add a permission after it is blocked by an SCP at a higher level. SCPs can ***only*** filter; they never add permissions.

SCPs do not support inheritance operators that alter how elements of the policy are inherited by child OUs and accounts.

The following illustration shows how SCPs (p. 104) work.


SCP attached to or inherited from parent
SCP permissions attached to child OU or account
A
B
C
D
E
Permissions that the child can actually use

In this illustration, assume that the oval on the left represents an SCP that is attached to the organization's root. It allows permissions A, B, and C. The root contains an organizational unit (OU) to which is attached a second SCP represented by the oval on the right. That second SCP allows permissions C, D, and E. Because the SCP attached to the root doesn't allow D or E, no OUs or accounts in the organization can use them. Even though the SCP attached to the OU explicitly allows D and E, they are blocked because they're blocked by the SCP attached to the root. Because the OU's SCP doesn't allow A or B, those permissions are blocked for the OU and any of its child OUs or accounts. However, other OUs that might exist under the root can still allow A and B.

As you traverse down the hierarchy of OUs to the account, the process in the previous paragraph is repeated at each OU and finally with the account. At each level, the result of the evaluation at the parent becomes the policy on the left of the diagram and is compared to the SCPs attached to the child OU.

For example, if you were to move down the tree to some child OU called X, imagine that the oval on the left represents the inherited, effective permissions permitted by all of the SCPs above OU X in the hierarchy. The oval on the right represents the SCP attached to an OU or an AWS account contained in OU X. Again, the intersection of those permissions is what is available to be used the entity on the right. If that entity is an AWS account, then the intersection is the set of permissions that can be granted to users and roles in that account. If the entity is an OU, then the intersection is the set of permissions that can be inherited by that OU's children. Repeat the process for each level down the OU hierarchy until you reach the account itself. That final effective policy is the list of permissions that were allowed by every SCP above that account and attached to it.

This means that to allow an AWS service API at the member account level, you must allow that API at ***every*** level between the member account and the root of your organization. You must attach SCPs to every level from your organization’s root to the member account that allows the given AWS service API (such as ec2:RunInstances). You can use either of the following strategies for this:

- A deny list strategy (p. 117) makes use of the `FullAWSAccess` SCP that is attached by default to every OU and account. This SCP overrides the default implicit deny, and explicitly allows all permissions to flow down from the root to every account, unless you explicitly deny a permission with an additional SCP that you create and attach to the appropriate OU or account. This strategy works because an explicit `deny` in a policy always overrides any kind of `allow`. No account below the level of the OU with the deny policy can use the denied API, and there is no way to add the permission back lower in the hierarchy. For more information, see Using SCPs as a deny list (p. 117).
- An allow list strategy (p. 118) has you remove the `FullAWSAccess` SCP that is attached by default to every OU and account. This means that no APIs are permitted anywhere unless you explicitly allow them. To allow a service API to operate in an AWS account, you must create your own SCPs and attach them to the account and every OU above it, up to and including the root. Every SCP in the hierarchy, starting at the root, must explicitly allow the APIs that you want to be usable in the OUs and accounts below it. This strategy works because an explicit `allow` in an SCP overrides an implicit `deny`. For more information, see Using SCPs as an allow list (p. 118).

To review how policies are evaluated in terms of implicit vs. explicit allows and denies, and what overrides what, see Determining whether a request is allowed or denied within an account.

Users and roles in accounts must still be granted permissions using AWS Identity and Access Management (IAM) permission policies attached to them or to groups. The SCPs only determine what permissions are ***available*** to be granted by such policies. The user can't perform any actions that the applicable SCPs don't allow. Actions allowed by the SCPs can be used if they are granted to the user or role by one or more IAM permission policies.

When you attach SCPs to the organization root, OUs, or directly to accounts, all policies that affect a given account are evaluated together using the same rules that govern IAM permission policies:

- Users and roles in affected accounts can't perform any actions that are listed in the SCP's `Deny` statement. An explicit `Deny` statement overrides any `Allow` that other SCPs might grant.
- Any action that has an explicit `Allow` in an SCP (such as the default "*" SCP or by any other SCP that calls out a specific service or action) can be delegated to users and roles in the affected accounts.
- Any action that isn't explicitly allowed by an SCP is implicitly denied and can't be delegated to users or roles in the affected accounts.

By default, an SCP named `FullAWSAccess` is attached to every organization root, OU, and account. This default SCP allows all actions and all services. So in a new organization, until you start creating or manipulating the SCPs, all of your existing IAM permissions continue to operate as they did. As soon as you apply a new or modified SCP to the organization root or an OU that contains an account, the permissions that your users have in that account become filtered by the SCP. Permissions that used to work might now be denied if they're not allowed by the SCP at every level of the hierarchy down to the specified account.

If you disable the SCP policy type on the organization root, all SCPs are automatically detached from all entities in the organization root. If you reenable SCPs on the organization root, all the original attachments are lost, and all entities are reset to being attached to only the default `FullAWSAccess` SCP.

For details about the syntax of SCPs, see SCP syntax (p. 119).

You can see a list of all policies applied to an account and where that policy comes from. To do this, choose an account in the AWS Organizations console. On the account details page, choose **Policies** and then choose **Service Control Policies** in the right-hand details pane. The same policy might apply to the account multiple times because the policy can be attached to any or all of the parent containers of the account. The effective policy that applies to the account is the intersection of allowed permissions of all applicable policies.

For more information about how to use SCPs, see Service control policies (SCPs) (p. 104).

Policy syntax and inheritance for management policy types

> **Important**
> The information in this section does ***not*** apply to SCPs. See the previous section Inheritance for service control policies (p. 92).

Management policy types include:

- Artificial Intelligence (AI) services opt-out policies (p. 139)
- Backup policies (p. 154)
- Tag policies (p. 187)

Inheritance behaves differently for management policy types than it does for service control policies. The syntax for management policy types includes *inheritance operators*, which enable you to specify with fine granularity what elements from the parent policies are applied and what elements can be overridden or modified when inherited by child OUs and accounts.

The *effective policy* is the set of rules that are inherited from the organization root and OUs along with those directly attached to the account. The effective policy specifies the final set of rules that apply to the account. You can view the effective policy for an account that includes the effect of all of the inheritance operators in the policies applied. For more information, see Viewing effective tag policies (p. 202).

This section explains how parent policies and child policies are processed into the effective policy for an account.

Terminology

This topic uses the following terms when discussing policy inheritance.

Policy inheritance

> The interaction of policies at differing levels of an organization, moving from the top-level root of the organization, down through the organizational unit (OU) hierarchy to individual accounts.

You can attach policies to the organization root, OUs, individual accounts, and to any combination of these organization entities. Policy inheritance refers to policies that are attached to the organization root or to an OU. All accounts that are members of the organization root or OU where a policy is attached *inherit*that policy.

For example, when policies are attached to the organization root, all accounts in the organization inherit that policy. That's because all accounts in an organization are always under the organization root. When you attach a policy to a specific OU, accounts that are directly under that OU or any child OU inherit that policy. Because you can attach policies to multiple levels in the organization, accounts might inherit multiple policy documents for a single policy type.

Parent policies

Policies that are attached higher in the organizational tree than policies that are attached to entities lower in the tree.

For example, if you attach policy A to the organization root, it's just a policy. If you also attach policy B to an OU under that root, policy A is the parent policy of Policy B. Policy B is the child policy of Policy A. Policy A and policy B merge to create the effective tag policy for accounts in the OU.

Child policies

Policies that are attached at a lower level in the organization tree than the parent policy.

Effective policies

The final, single policy document that specifies the rules that apply to an account. The effective policy is the aggregation of any policies the account inherits, plus any policy that is directly attached to the account. For example, tag policies enable you to view the effective tag policy that applies to any of your accounts. For more information, see Viewing effective tag policies (p. 202).

Inheritance operators (p. 95)

Operators that control how inherited policies merge into a single effective policy. These operators are considered an advanced feature. Experienced policy authors can use them to limit what changes a child policy can make and how settings in policies merge.

Inheritance operators

Inheritance operators control how inherited policies and account policies merge into the account's effective policy. These operators include value-setting operators and child control operators.

When you use the visual editor in the AWS Organizations console, you can use only the `@@assign` operator. Other operators are considered an advanced feature. To use the other operators, you must manually author the JSON policy. Experienced policy authors can use inheritance operators to control what values are applied to the effective policy and limit what changes child policies can make.

Value-setting operators

You can use the following value-setting operators to control how your policy interacts with its parent policies:

- `@@assign` – **Overwrites** any inherited policy settings with the specified settings. If the specified setting isn't inherited, this operator adds it to the effective policy. This operator can apply to any policy setting of any type.
  - For single-valued settings, this operator replaces the inherited value with the specified value.
  - For multi-valued settings (JSON arrays), this operator removes any inherited values and replaces them with the values specified by this policy.

- `@@append` – **Adds** the specified settings (without removing any) to the inherited ones. If the specified setting isn't inherited, this operator adds it to the effective policy. You can use this operator with only multi-valued settings.
  - This operator adds the specified values to any values in the inherited array.
- `@@remove` – **Removes** the specified inherited settings from the effective policy, if they exist. You can use this operator with only multi-valued settings.
  - This operator removes only the specified values from the array of values inherited from the parent policies. Other values can continue to exist in the array and can be inherited by child policies.

Child control operators

Using child control operators is optional. You can use the `@@operators_allowed_for_child_policies` operator to control which value-setting operators child policies can use. You can allow all operators, some specific operators, or no operators. By default, all operators (`@@all`) are allowed.

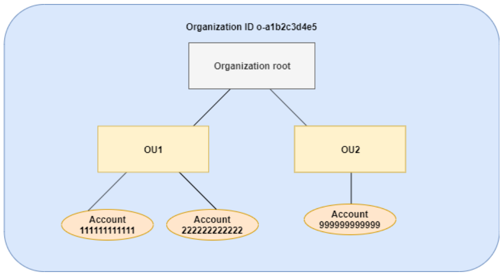
- `"@@operators_allowed_for_child_policies":["@@all"]` – Child OUs and accounts can use any operator in policies. By default, all operators are allowed in child policies.
- `"@@operators_allowed_for_child_policies":["@@assign", "@@append", "@@remove"]` – Child OUs and accounts can use only the specified operators in child policies. You can specify one or more value-setting operators in this child control operator.
- `"@@operators_allowed_for_child_policies":["@@none"]` – Child OUs and accounts can't use operators in policies. You can use this operator to effectively lock in the values that are defined in a parent policy so that child policies can't add, append, or remove those values.

> **Note**
> If an inherited child control operator limits the use of an operator, you can't reverse that rule in a child policy. If you include child control operators in a parent policy, they limit the value-setting operators in all child policies.

Policy inheritance examples

These examples show how policy inheritance works by showing how parent and child tag policies are merged into an effective tag policy for an account.

The examples assume that you have the organization structure shown in the following diagram.



Examples

- Example 1: Allow child policies to overwrite only tag values (p. 97)
- Example 2: Append new values to inherited tags (p. 98)
- Example 3: Remove values from inherited tags (p. 99)
- Example 4: Restrict changes to child policies (p. 101)
- Example 5: Conflicts with child control operators (p. 102)
- Example 6: Conflicts with appending values at same hierarchy level (p. 103)

Example 1: Allow child policies to overwrite *only* tag values

The following tag policy defines the `CostCenter` tag key and two acceptable values, `Development` and `Support`. If you attach it to the organization root, the tag policy is in effect for all accounts in the organization.

Policy A – Organization root tag policy

```
{
    "tags": {
        "costcenter": {
            "tag_key": {
                "@@assign": "CostCenter"
            },
            "tag_value": {
                "@@assign": [
                    "Development",
                    "Support"
                ]
            }
        }
    }
}
```

Assume that you want users in OU1 to use a different tag value for a key, and you want to enforce the tag policy for specific resource types. Because policy A doesn't specify which child control operators are allowed, all operators are allowed. You can use the `@@assign` operator and create a tag policy like the following to attach to OU1.

Policy B – OU1 tag policy

```
{
    "tags": {
        "costcenter": {
            "tag_key": {
                "@@assign": "CostCenter"
            },
            "tag_value": {
                "@@assign": [
                    "Sandbox"
                ]
            },
            "enforced_for": {
                "@@assign": [
                    "redshift:*",
                    "dynamodb:table"
                ]
            }
        }
    }
}
```

Specifying the `@@assign` operator for the tag does the following when policy A and policy B merge to form the *effective tag policy* for an account:

- Policy B overwrites the two tag values that were specified in the parent policy, policy A. The result is that `Sandbox` is the only compliant value for the `CostCenter` tag key.
- The addition of `enforced_for` specifies that the `CostCenter` tag *must* be the specified tag value on all Amazon Redshift resources and Amazon DynamoDB tables.

As shown in the diagram, OU1 includes two accounts: 111111111111 and 222222222222.

Resultant effective tag policy for accounts 111111111111 and 222222222222

> **Note**
> You can't directly use the contents of a displayed effective policy as the contents of a new policy. The syntax doesn't include the operators needed to control merging with other child and parent policies. The display of an effective policy is intended only for understanding the results of the merger.

```
{
    "tags": {
        "costcenter": {
            "tag_key": "CostCenter",
            "tag_value": [
                "Sandbox"
            ],
            "enforced_for": [
                "redshift:*",
                "dynamodb:table"
            ]
        }
    }
}
```

Example 2: Append new values to inherited tags

There may be cases where you want all accounts in your organization to specify a tag key with a short list of acceptable values. For accounts in one OU, you may want to allow an additional value that only those accounts can specify when creating resources. This example specifies how to do that by using the `@@append` operator. The `@@append` operator is an advanced feature.

Like example 1, this example starts with policy A for the organization root tag policy.

Policy A – Organization root tag policy

```
{
    "tags": {
        "costcenter": {
            "tag_key": {
                "@@assign": "CostCenter"
            },
            "tag_value": {
                "@@assign": [
                    "Development",
                    "Support"
                ]
            }
        }
    }
}
```

For this example, attach policy C to OU2. The difference in this example is that using the `@@append` operator in policy C *adds to*, rather than overwrites, the list of acceptable values and the `enforced_for` rule.

Policy C – OU2 tag policy for appending values

```
{
    "tags": {
        "costcenter": {
            "tag_key": {
                "@@assign": "CostCenter"
            },
            "tag_value": {
                "@@append": [
                    "Marketing"
                ]
            },
            "enforced_for": {
                "@@append": [
                    "redshift:*",
                    "dynamodb:table"
                ]
            }
        }
    }
}
```

Attaching policy C to OU2 has the following effects when policy A and policy C merge to form the effective tag policy for an account:

- Because policy C includes the `@@append` operator, it allows for *adding to*, not overwriting, the list of acceptable tag values that are specified in Policy A.
- As in policy B, the addition of `enforced_for` specifies that the `CostCenter` tag must be used as the specified tag value on all Amazon Redshift resources and Amazon DynamoDB tables. Overwriting (`@@assign`) and adding (`@@append`) have the same effect if the parent policy doesn't include a child control operator that restricts what a child policy can specify.

As shown in the diagram, OU2 includes one account: 999999999999. Policy A and policy C merge to create the effective tag policy for account 999999999999.

Effective tag policy for account 999999999999

> **Note**
> You can't directly use the contents of a displayed effective policy as the contents of a new policy. The syntax doesn't include the operators needed to control merging with other child and parent policies. The display of an effective policy is intended only for understanding the results of the merger.

```
{
    "tags": {
        "costcenter": {
            "tag_key": "CostCenter",
            "tag_value": [
                "Development",
                "Support",
                "Marketing"
            ],
            "enforced_for": [
                "redshift:*",
                "dynamodb:table"
            ]
        }
    }
}
```

Example 3: Remove values from inherited tags

There may be cases where the tag policy that is attached to the organization defines more tag values than you want an account to use. This example explains how to revise a tag policy using the `@@remove` operator. The `@@remove` is an advanced feature.

Like the other examples, this example starts with policy A for the organization root tag policy.

Policy A – Organization root tag policy

```
{
    "tags": {
        "costcenter": {
            "tag_key": {
                "@@assign": "CostCenter"
            },
            "tag_value": {
                "@@assign": [
                    "Development",
                    "Support"
                ]
            }
        }
    }
}
```

For this example, attach policy D to account 999999999999.

Policy D – Account 999999999999 tag policy for removing values

```
{
    "tags": {
        "costcenter": {
            "tag_key": {
                "@@assign": "CostCenter"
            },
            "tag_value": {
                "@@remove": [
                    "Development",
                    "Marketing"
                ],
                "enforced_for": {
                    "@@remove": [
                        "redshift:*",
                        "dynamodb:table"
                    ]
                }
            }
        }
    }
}
```

Attaching policy D to account 999999999999 has the following effects when policy A, policy C, and policy D merge to form the effective tag policy:

- Assuming you performed all of the previous examples, policies B, C, and C are child policies of A. Policy B is only attached to OU1, so it has no effect on account 9999999999999.
- For account 9999999999999, the only acceptable value for the `CostCenter` tag key is `Support`.
- Compliance is not enforced for the `CostCenter` tag key.

New effective tag policy for account 999999999999

> **Note**
> You can't directly use the contents of a displayed effective policy as the contents of a new policy. The syntax doesn't include the operators needed to control merging with other child and parent

policies. The display of an effective policy is intended only for understanding the results of the merger.

```
{
    "tags": {
        "costcenter": {
            "tag_key": "CostCenter",
            "tag_value": [
                "Support"
            ]
        }
    }
}
```

If you later add more accounts to OU2, their effective tag policies would be different than for account 999999999999. That's because the more restrictive policy D is only attached at the account level, and not to the OU.

Example 4: Restrict changes to child policies

There may be cases where you want to restrict changes in child policies. This example explains how to do that using child control operators.

This example starts with a new organization root tag policy and assumes that tag policies aren't yet attached to organization entities.

Policy E – Organization root tag policy for restricting changes in child policies

```
{
    "tags": {
        "project": {
            "tag_key": {
                "@@operators_allowed_for_child_policies": ["@@none"],
                "@@assign": "Project"
            },
            "tag_value": {
                "@@operators_allowed_for_child_policies": ["@@append"],
                "@@assign": [
                    "Maintenance",
                    "Escalations"
                ]
            }
        }
    }
}
```

When you attach policy E to the organization root, the policy prevents child policies from changing the `Project` tag key. However, child policies can overwrite or append tag values.

Assume you then attach the following policy F to an OU.

Policy F – OU tag policy

```
{
    "tags": {
        "project": {
            "tag_key": {
                "@@assign": "PROJECT"
            },
            "tag_value": {
                "@@append": [
                    "Escalations - research"
                ]
            }
        }
    }
}
```

Merging policy E and policy F have the following effects on the OU's accounts:

- Policy F is a child policy to Policy E.
- Policy F attempts to change the case treatment, but it can't. That's because policy E includes the `"@@operators_allowed_for_child_policies": ["@@none"]` operator for the tag key.
- However, policy F can append tag values for the key. That's because policy E includes `"@@operators_allowed_for_child_policies": ["@@append"]` for the tag value.

Effective policy for accounts in the OU

> **Note**
> You can't directly use the contents of a displayed effective policy as the contents of a new policy. The syntax doesn't include the operators needed to control merging with other child and parent policies. The display of an effective policy is intended only for understanding the results of the merger.

```
{
    "tags": {
        "project": {
            "tag_key": "Project",
            "tag_value": [
                "Maintenance",
                "Escalations",
                "Escalations - research"
            ]
        }
    }
}
```

Example 5: Conflicts with child control operators

Child control operators can exist in tag policies that are attached at the same level in the organization hierarchy. When that happens, the intersection of the allowed operators is used when the policies merge to form the effective policy for accounts.

Assume policy G and policy H are attached to the organization root.

Policy G – Organization root tag policy 1

```
{
    "tags": {
        "project": {
            "tag_value": {
                "@@operators_allowed_for_child_policies": ["@@append"],
                "@@assign": [
                    "Maintenance"
                ]
            }
        }
    }
}
```

Policy H – Organization root tag policy 2

```
{
    "tags": {
        "project": {
            "tag_value": {
                "@@operators_allowed_for_child_policies": ["@@append", "@@remove"]
            }
        }
    }
}
```

In this example, one policy at the organization root defines that the values for the tag key can only be appended to. The other policy attached to the organization root allows child policies to both append and remove values. The intersection of these two permissions is used for child policies. The result is that child policies can append values, but not remove values. Therefore, the child policy can append a value to the list of tag values but can't remove the `Maintenance` value.

Example 6: Conflicts with appending values at same hierarchy level

You can attach multiple tag policies to each organization entity. When you do this, the tag policies that are attached to the same organization entity might include conflicting information. Policies are evaluated based on the order in which they were attached to the organization entity. To change which policy is evaluated first, you can detach a policy and then reattach it.

Assume policy J is attached to the organization root first, and then policy K is attached to the organization root.

Policy J – First tag policy attached to the organization root

```
{
    "tags": {
        "project": {
            "tag_key": {
                "@@assign": "PROJECT"
            },
            "tag_value": {
                "@@append": ["Maintenance"]
            }
        }
    }
}
```

Policy K – Second tag policy attached to the organization root

```
{
    "tags": {
        "project": {
            "tag_key": {
                "@@assign": "project"
            }
        }
    }
}
```

In this example, the tag key `PROJECT` is used in the effective tag policy because the policy that defined it was attached to the organization root first.

Policy JK – Effective tag policy for account

The effective policy for the account is as follows.

> **Note**
> You can't directly use the contents of a displayed effective policy as the contents of a new policy. The syntax doesn't include the operators needed to control merging with other child and parent policies. The display of an effective policy is intended only for understanding the results of the merger.

```
{
    "tags": {
        "project": {
            "tag_key": "PROJECT",
            "tag_value": [
                "Maintenance"
            ]
        }
    }
}
```

Service control policies (SCPs)

For information and procedures common to all policy types, see the following topics:

- Enable and disable policy types (p. 85)
- Get details about your policies (p. 87)
- Policy syntax and inheritance (p. 92)

Service control policies (SCPs) are a type of organization policy that you can use to manage permissions in your organization. SCPs offer central control over the maximum available permissions for all accounts in your organization. SCPs help you to ensure your accounts stay within your organization’s access control guidelines. SCPs are available only in an organization that has all features enabled (p. 35). SCPs aren't available if your organization has enabled only the consolidated billing features. For instructions on enabling SCPs, see Enabling and disabling policy types (p. 85).

SCPs alone are not sufficient to granting permissions to the accounts in your organization. No permissions are granted by an SCP. An SCP defines a guardrail, or sets limits, on the actions that the account's administrator can delegate to the IAM users and roles in the affected accounts. The administrator must still attach identity-based or resource-based policies to IAM users or roles, or to the resources in your accounts to actually grant permissions. The effective permissions (p. 105) are the logical intersection between what is allowed by the SCP and what is allowed by the IAM and resource-based policies.

> **Important**
> SCPs don't affect users or roles in the management account. They affect only the member accounts in your organization.

Topics on this page

Testing effects of SCPs

AWS strongly recommends that you don't attach SCPs to the root of your organization without thoroughly testing the impact that the policy has on accounts. Instead, create an OU that you can move your accounts into one at a time, or at least in small numbers, to ensure that you don't inadvertently lock users out of key services. One way to determine whether a service is used by an account is to examine the service last accessed data in IAM. Another way is to use AWS CloudTrail to log service usage at the API level.

Maximum size of SCPs

All characters in your SCP count against its maximum size (p. 340). The examples in this guide show the SCPs formatted with extra white space to improve their readability. However, to save space if your policy size approaches the maximum size, you can delete any white space, such as space characters and line breaks that are outside quotation marks.

> **Tip**
> Use the visual editor to build your SCP. It automatically removes extra white space.

Inheritance of SCPs in the OU hierarchy

For a detailed explanation of how SCP inheritance works, see Inheritance for service control policies (p. 92)

SCP effects on permissions

SCPs are similar to AWS Identity and Access Management (IAM) permission policies and use almost the same syntax. However, an SCP never grants permissions. Instead, SCPs are JSON policies that specify the maximum permissions for the affected accounts. For more information, see Policy Evaluation Logic in the *IAM User Guide*.

- SCPs ***affect only IAM users and roles*** that are managed by accounts that are part of the organization. SCPs don't affect resource-based policies directly. They also don't affect users or roles from accounts outside the organization. For example, consider an Amazon S3 bucket that's owned by account A in an organization. The bucket policy (a resource-based policy) grants access to users from account B outside the organization. Account A has an SCP attached. That SCP doesn't apply to those outside users in account B. The SCP applies only to users that are managed by account A in the organization.
- An SCP restricts permissions for IAM users and roles in member accounts, including the member account's root user. Any account has only those permissions permitted by ***every*** parent above it. If a permission is blocked at any level above the account, either implicitly (by not being included in an `Allow` policy statement) or explicitly (by being included in a `Deny` policy statement), a user or role in the affected account can't use that permission, even if the account administrator attaches the `AdministratorAccess` IAM policy with */* permissions to the user.
- SCPs affect only ***member*** accounts in the organization. They have no effect on users or roles in the management account.

- Users and roles must still be granted permissions with appropriate IAM permission policies. A user without any IAM permission policies has no access, even if the applicable SCPs allow all services and all actions.
- If a user or role has an IAM permission policy that grants access to an action that is also allowed by the applicable SCPs, the user or role can perform that action.
- If a user or role has an IAM permission policy that grants access to an action that is either not allowed or explicitly denied by the applicable SCPs, the user or role can't perform that action.
- SCPs affect all users and roles in attached accounts, ***including the root user***. The only exceptions are those described in Tasks and entities not restricted by SCPs (p. 106).
- SCPs ***do not*** affect any service-linked role. Service-linked roles enable other AWS services to integrate with AWS Organizations and can't be restricted by SCPs.
- When you disable the SCP policy type in a root, all SCPs are automatically detached from all AWS Organizations entities in that root. AWS Organizations entities include organizational units, organizations, and accounts. If you reenable SCPs in a root, that root reverts to only the default `FullAWSAccess` policy automatically attached to all entities in the root. Any attachments of SCPs to AWS Organizations entities from before SCPs were disabled are lost and aren't automatically recoverable, although you can manually reattach them.
- If both a permissions boundary (an advanced IAM feature) and an SCP are present, then the boundary, the SCP, and the identity-based policy must all allow the action.

Using access data to improve SCPs

When signed in with management account credentials, you can view service last accessed data for an AWS Organizations entity or policy in the **AWS Organizations** section of the IAM console. You can also use the AWS Command Line Interface (AWS CLI) or AWS API in IAM to retrieve service last accessed data. This data includes information about which allowed services that the IAM users and roles in an AWS Organizations account last attempted to access and when. You can use this information to identify unused permissions so that you can refine your SCPs to better adhere to the principle of least privilege.

For example, you might have a deny list SCP (p. 117) that prohibits access to three AWS services. All services that aren't listed in the SCP's `Deny` statement are allowed. The service last accessed data in IAM tells you which AWS services are allowed by the SCP but are never used. With that information, you can update the SCP to deny access to services that you don't need.

For more information, see the following topics in the *IAM User Guide*:

- Viewing Organizations Service Last Accessed Data for Organizations
- Using Data to Refine Permissions for an Organizational Unit

Tasks and entities not restricted by SCPs

You ***can't*** use SCPs to restrict the following tasks:

- Any action performed by the management account
- Any action performed using permissions that are attached to a service-linked role
- Register for the Enterprise support plan as the root user
- Change the AWS support level as the root user
- Provide trusted signer functionality for CloudFront private content
- Configure reverse DNS for an Amazon Lightsail email server and Amazon EC2 instance as the root user
- Tasks on some AWS-related services:

- Alexa Top Sites
- Alexa Web Information Service
- Amazon Mechanical Turk
- Amazon Product Marketing API

Exceptions for only member accounts created before September 15, 2017

For **some** accounts created *before* September 15, 2017, you ***can't*** use SCPs to prevent the root user in those member accounts from performing the following tasks:

> **Important**
> For **all** accounts created *after* September 15, 2017, the following exceptions don't apply and you ***can*** use SCPs to prevent the root user in those member accounts from performing the following tasks. However, unless you are certain that **all** of the accounts in your organization were created after September 15, 2017, we recommend that you don't rely on SCPs to try to restrict these operations:

- Enable or disable multi-factor authentication on the root user
- Create, update, or delete x.509 keys for the root user
- Change the root user's password
- Create, update, or delete root access keys

Creating, updating, and deleting service control policies

When you sign in to your organization's management account, you can create and update service control policies (SCPs) (p. 104). You create SCPs by building statements that deny or allow access to services and actions that you specify.

The default configuration for working with SCPs is to use a "block list" strategy where all actions are implicitly allowed except for those actions you want to block by creating statements that deny access. With deny statements, you can specify resources and conditions for the statement and use the NotAction element. For allow statements, you can specify services and actions only. For more information about statements that deny access and allow access, see Strategies for using SCPs (p. 117).

> **Tip**
> You can use service last accessed data in IAM as a data point for updating your SCPs to restrict access to only the AWS services that you need. For more information, see Viewing Organizations Service Last Accessed Data for Organizations in the *IAM User Guide.*

In this topic:

- After you enable service control policies (p. 85) for your organization, you can create a policy (p. 107).
- When your SCP requirements change, you can update an existing policy (p. 111).
- When you no longer need a policy and after you detach it from all organizational units (OUs) and accounts, you can delete it (p. 113).

Creating an SCP

> **Minimum permissions**
> To create SCPs, you need permission to run the following action:

- `organizations:CreatePolicy`

AWS Management Console

To create a service control policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Service control policies** page, choose **Create policy**.
3. On the **Create new service control policy** page, enter a **Policy name** and an optional **Policy description**.
4. (Optional) Add one or more tags by choosing **Add tag** and then entering a key and an optional value. Leaving the value blank sets it to an empty string; it isn't `null`. You can attach up to 50 tags to a policy. For more information, see Tagging AWS Organizations resources (p. 220).

   Note
   In most of the steps that follow, we discuss using the controls on the right side of the JSON editor to construct the policy, element by element. Alternatively, you can, at any time, simply enter text in the JSON editor on the left side of the window. You can directly type, or use copy and paste.
5. To build the policy, your next steps vary depending on whether you want to add a statement that denies (p. 117) or allows (p. 118) access. For more information, see Strategies for using SCPs (p. 117). If you use `Deny` statements, you have additional control because you can restrict access to specific resources, define conditions for when SCPs are in effect, and use the NotAction element. For details about syntax, see SCP syntax (p. 119).

   To add a statement that *denies* access:

   a. In the right **Edit statement** pane of the editor, under **1. Add actions**, choose an AWS service.

      As you choose options on the right, the JSON editor updates to show the corresponding JSON policy on left.

   b. After you select a service, a list opens that contains the available actions for that service. You can choose **All actions**, or choose one or more individual actions that you want to deny.

      The JSON on the left updates to include the actions you selected.

      Note
      If you select an individual action and then also go back and also select **All actions**, the expected entry for *`servicename`*`/*` is added to the JSON, but the individual actions that you previously selected are left in the JSON and not removed.

   c. If you want to add actions from additional services, you can choose **All services** at the top of the **Statement** box, and then repeat the previous two steps as needed.

   d. Specify resources to include in the statement.

      - Next to **2. Add a resource**, choose **Add**.
      - In the **Add resource** dialog, choose the service whose resources you want to control from the list. You can select from among only those services you selected in the previous step.
      - Under **Resource type**, choose the type of resource you want to control.
      - Finally, complete the Amazon Resource Name (ARN) in **Resource ARN** to identify the specific resource to which you want to control access. You must replace all placeholders that are surrounded by curly braces `{}`. You can specify wild cards (`*`) where that resource type's ARN syntax permits. See the documentation for a specific resource type for information about where you can use wild cards.

- Save your addition to the policy by choosing **Add resource**. The `Resource` element in the JSON reflects your additions or changes. The **Resource** element is required.

  Tip
  If you want to specify all resources for the selected service, either choose the **All resources** option in the list, or edit the `Resource` statement directly in the JSON to read `"Resource":"*"`.

e. (Optional) To specify conditions that limit when a policy statement is in effect, next to **3. Add condition**, choose **Add**.

- **Condition key** – From the list you can choose any condition key that is available for all AWS services (for example, `aws:SourceIp`) or a service-specific key for only one of the services that you selected for this statement.
- **Qualifier** – (Optional) If you provide multiple values for the condition (dependent on the specified condition key), you can specify a qualifier for testing requests against the values.
  - **Default** – Tests a single value in the request against the condition key value in the policy. The condition returns true if the value in the request matches the value in the policy. If the policy specifies more than one value then they are treated as an "or" test, and the condition returns true if the request values matches any of the policy values.
  - **For any value in a request** – When the request can have multiple values, this option tests whether *at least one* of the request values matches at least one of the condition key values in the policy. The condition returns true if any one of the key values in the request matches any one of the condition values in the policy. For no matching key or a null dataset, the condition returns false.
  - **For all values in a request** – When the request can have multiple values, this option tests whether *every* request value matches a condition key value in the policy. The condition returns true if every key value in the request matches at least one value in the policy. It also returns true if there are no keys in the request, or if the key values resolve to a null data set, such as an empty string.
- **Operator** – The operator specifies the type of comparison to make. The options that are presented depend on the data type of the condition key. For example, the `aws:CurrentTime` global condition key lets you pick from any of the date comparison operators, or `Null`, which you can use to test whether the value is present in the request.

  For any condition operator except the `Null` test, you can choose the IfExists option.
- **Value** – (Optional) Specify one or more values for which you want to test the request.

Choose **Add condition**.

For more information about condition keys, see IAM JSON Policy Elements: Condition in the *IAM User Guide*.

f. (Optional) To use the `NotAction` element to deny access to all actions ***except*** those specified, replace `Action` in the left pane with `NotAction`, just after the `"Effect": "Deny",` element. For more information, see IAM JSON Policy Elements: NotAction in the *IAM User Guide*.

6. To add a statement that *allows* access:

a. In the JSON editor on the left, change the line `"Effect": "Deny"` to `"Effect": "Allow"`.

As you choose options on the right, the JSON editor updates to show the corresponding JSON policy on the left.

b. After you select a service, a list opens that contains the available actions for that service. You can choose **All actions**, or choose one or more individual actions that you want to allow.

   The JSON on the left updates to include the actions you selected.

   > **Note**
   > If you select an individual action and then also go back and also select **All actions**, the expected entry for *`servicename`*`/*` is added to the JSON, but the individual actions that you previously selected are left in the JSON and not removed.

c. If you want to add actions from additional services, you can choose **All services** at the top of the **Statement** box, and then repeat the previous two steps as needed.

7. (Optional) To add another statement to the policy, choose **Add statement** and use the visual editor to build the next statement.
8. When you're finished adding statements, choose **Create policy** to save the completed SCP.

Your new SCP appears in the list of the organization's policies. You can now attach your SCP to the root, OUs, or accounts (p. 114).

AWS CLI & AWS SDKs

To create a service control policy

You can use one of the following commands to create an SCP:

- AWS CLI: create-policy

  The following example assumes that you have a file named `Deny-IAM.json` with the JSON policy text in it. It uses that file to create a new service control policy.

```
$ aws organizations create-policy \
    --content file://Deny-IAM.json \
    --description "Deny all IAM actions" \
    --name DenyIAMSCP \
    --type SERVICE_CONTROL_POLICY
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
service_control_policy/p-i9j8k7l6m5",
            "Name": "DenyIAMSCP",
            "Description": "Deny all IAM actions",
            "Type": "SERVICE_CONTROL_POLICY",
            "AwsManaged": false
        },
         "Content": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":
\"Statement1\",\"Effect\":\"Deny\",\"Action\":[\"iam:*\"],\"Resource\":[\"*\"]}]}"
    }
}
```

- AWS SDKs: CreatePolicy

Note
SCPs don't take effect on the management account and in a few other situations. For more information, see Tasks and entities not restricted by SCPs (p. 106).

Updating an SCP

When you sign in to your organization's management account, you can rename or change the contents of a policy. Changing the contents of an SCP immediately affects any users, groups, and roles in all attached accounts.

Minimum permissions
To update an SCP, you need permission to run the following actions:

- `organizations:UpdatePolicy` with a `Resource` element in the same policy statement that includes the ARN of the specified policy (or "*")
- `organizations:DescribePolicy` with a `Resource` element in the same policy statement that includes the ARN of the specified policy (or "*")

AWS Management Console

To update a policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **Service control policies** pagechoose the name of the policy that you want to update.
3. On the policy's detail page, choose **Edit policy**.
4. Make any or all of the following changes:
    - You can rename the policy by entering a new name in **Policy name**.
    - You can change the description by entering new text in **Policy description**.
    - You can edit the policy text by editing the policy in JSON format in the left pane. Alternatively, you can choose a statement in the editor on the left, and then alter its elements by using the controls on the left. For more details about each control, see the Creating an SCP procedure (p. 107) earlier in this topic.
5. When you're finished, choose **Save changes**.

AWS CLI & AWS SDKs

To update a policy

You can use one of the following commands to update a policy:

- AWS CLI: update-policy

  The following example renames a policy.

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --name "MyRenamedPolicy"
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
service_control_policy/p-i9j8k7l6m5",
            "Name": "MyRenamedPolicy",
            "Description": "Blocks all IAM actions",
            "Type": "SERVICE_CONTROL_POLICY",
            "AwsManaged": false
        },
        "Content": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":
\"Statement1\",\"Effect\":\"Deny\",\"Action\":[\"iam:*\"],\"Resource\":[\"*\"]}]}"
    }
}
```

The following example adds or changes the description for a service control policy.

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --description "My new policy description"
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
service_control_policy/p-i9j8k7l6m5",
            "Name": "MyRenamedPolicy",
            "Description": "My new policy description",
            "Type": "SERVICE_CONTROL_POLICY",
            "AwsManaged": false
        },
        "Content": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":
\"Statement1\",\"Effect\":\"Deny\",\"Action\":[\"iam:*\"],\"Resource\":[\"*\"]}]}"
    }
}
```

The following example changes the policy document of the SCP by specifying a file that contains the new JSON policy text.

```
$ aws organizations update-policy \
    --policy-id p-zlfw1r64
    --content file://MyNewPolicyText.json
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
service_control_policy/p-i9j8k7l6m5",
            "Name": "MyRenamedPolicy",
            "Description": "My new policy description",
            "Type": "SERVICE_CONTROL_POLICY",
            "AwsManaged": false
        },
        "Content": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":
\"AModifiedPolicy\",\"Effect\":\"Deny\",\"Action\":[\"iam:*\"],\"Resource\":[\"*
\"]}]}"
    }
}
```

- AWS SDKs: UpdatePolicy

For more information

For more information about creating SCPs, see the following topics:

- Example service control policies (p. 126)
- SCP syntax (p. 119)

Editing tags attached to an SCP

When you sign in to your organization's management account, you can add or remove the tags attached to an SCP. For more information about tagging, see Tagging AWS Organizations resources (p. 220).

Minimum permissions
To edit the tags attached to an SCP in your AWS organization, you must have the following permissions:

- `organizations:DescribeOrganization` – required only when using the Organizations console
- `organizations:DescribePolicy` – required only when using the Organizations console
- `organizations:TagResource`
- `organizations:UntagResource`

AWS Management Console

To edit the tags attached to an SCP

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Service control policies** page choose the name of the policy with the tags that you want to edit.
3. On the policy details page, choose the **Tags** tab, and then choose**Manage tags**.
4. Make any or all of the following changes:
   - Change the value of a tag by entering a new value over the old one. You can't directly modify the tag key. To change a key, you must delete the tag with the old key and then add a tag with the new key.
   - Remove an existing tag by choosing **Remove**.
   - Add a new tag key and value pair. Choose **Add tag**, then enter the new key name and optional value in the provided boxes. If you leave the **Value** box empty, the value is an empty string; it isn't `null`.
5. When you're finished, choose **Save changes**.

AWS CLI & AWS SDKs

To edit the tags attached to an SCP

You can use one of the following commands to edit the tags attached to an SCP:

- AWS CLI: tag-resource and untag-resource
- AWS SDKs: TagResource and UntagResource

Deleting an SCP

When you sign in to your organization's management account, you can delete a policy that you no longer need in your organization.

Notes

- Before you can delete a policy, you must first detach it from all attached entities.

- You can't delete any AWS managed SCP such as the SCP named `FullAWSAccess`.

Minimum permissions
To delete an SCP, you need permission to run the following action:

- `organizations:DeletePolicy`

AWS Management Console

To delete an SCP

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **Service control policies** page, choose the name of the SCP that you want to delete.
3. You must first detach the policy that you want to delete from all roots, OUs, and accounts. Choose the **Targets** tab, choose the radio button next to each root, OU, or account that is shown in the **Targets** list, and then choose **Detach**. In the confirmation dialog box, choose **Detach**. Repeat until you remove all targets.
4. Choose **Delete** at the top of the page.
5. On the confirmation dialog box, enter the name of the policy, and then choose **Delete**.

AWS CLI & AWS SDKs

To delete an SCP

You can use one of the following commands to delete a policy:

- AWS CLI: delete-policy

  The following example deletes the specified SCP.

  ```
  $ aws organizations delete-policy \
      --policy-id p-i9j8k7l6m5
  ```

  This command produces no output when successful.
- AWS SDKs: DeletePolicy

Attaching and detaching service control policies

When you sign in to your organization's management account, you can attach a service control policy (SCP) that you previously created. You can attach an SCP to the organization root, to an organizational unit (OU), or directly to an account. To attach an SCP, complete the following steps.

Minimum permissions
To attach an SCP to a root, OU, or account, you need permission to run the following action:

- `organizations:AttachPolicy` with a `Resource` element in the same policy statement that includes "*" or the Amazon Resource Name (ARN) of the specified policy and the ARN of the root, OU, or account that you want to attach the policy to

AWS Management Console

You can attach an SCP by either navigating to the policy or to the root, OU, or account that you want to attach the policy to.

To attach an SCP by navigating to the root, OU, or account

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page, navigate to and then choose the check box next to the root, OU, or account that you want to attach an SCP to. You might have to expand OUs (choose the ▶) to find the OU or account that you want.
3. In the **Policies** tab, in the entry for **Service control policies**, choose **Attach**.
4. Find the policy that you want and choose **Attach policy**.

   The list of attached SCPs on the **Policies** tab is updated to include the new addition. The policy change takes effect immediately, affecting the permissions of IAM users and roles in the attached account or all accounts under the attached root or OU.

To attach an SCP by navigating to the policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Service control policies** page, choose the name of the policy that you want to attach.
3. On the **Targets** tab, choose **Attach**.
4. Choose the radio button next to the root, OU, or account that you want to attach the policy to. You might have to expand OUs (choose the ▶) to find the OU or account that you want.
5. Choose **Attach policy**.

   The list of attached SCPs on the **Targets** tab is updated to include the new addition. The policy change takes effect immediately, affecting the permissions of IAM users and roles in the attached account or all accounts under the attached root or OU.

AWS CLI & AWS SDKs

To attach an SCP by navigating to the root, OU, or account

You can use one of the following commands to attach an SCP:

- AWS CLI: attach-policy

  The following example attaches an SCP to an OU.

  ```
  $ aws organizations attach-policy \
      --policy-id p-i9j8k7l6m5 \
      --target-id ou-a1b2-f6g7h222
  ```

  This command produces no output when successful.
- AWS SDKs: AttachPolicy

The policy change takes effect immediately, affecting the permissions of IAM users and roles in the attached account or all accounts under the attached root or OU.

Detaching an SCP from the organization root, OUs, or accounts

When you sign in to your organization's management account, you can detach an SCP from the organization root, OU, or account that it is attached to. After you detach an SCP from an entity, that

SCP no longer applies to any account that was affected by the now detached entity. To detach an SCP, complete the following steps.

> **Note**
> You can't detach the last SCP from a root, an OU, or an account. There must be at least one SCP attached to every root, OU, and account at all times.
>
> **Minimum permissions**
> To detach an SCP from the root, OU, or account, you need permission to run the following action:
>
> - `organizations:DetachPolicy`

AWS Management Console

You can detach an SCP by either navigating to the policy or to the root, OU, or account that you want to detach the policy from.

To detach an SCP by navigating to the root, OU, or account it's attached to

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page navigate to the Root, OU, or account that you want to detach a policy from. You might have to expand OUs (choose the ▶) to find the OU or account that you want. Choose the name of the Root, OU, or account.
3. On the **Policies** tab, choose the radio button next to the SCP that you want to detach, and then choose **Detach**.
4. In the confirmation dialog box, choose **Detach policy**.

   The list of attached SCPs is updated. The policy change caused by detaching the SCP takes effect immediately. For example, detaching an SCP immediately affects the permissions of IAM users and roles in the formerly attached account or accounts under the formerly attached organization root or OU.

To detach an SCP by navigating to the policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Service control policies** page, choose the name of the policy that you want to detach from a root, OU, or account.
3. On the **Targets** tab, choose the radio button next to the root, OU, or account that you want to detach the policy from. You might have to expand OUs (choose the ▶) to find the OU or account that you want.
4. Choose **Detach**.
5. In the confirmation dialog box, choose **Detach**.

   The list of attached SCPs is updated. The policy change caused by detaching the SCP takes effect immediately. For example, detaching an SCP immediately affects the permissions of IAM users and roles in the formerly attached account or accounts under the formerly attached organization root or OU.

AWS CLI & AWS SDKs

To detach an SCP from a root, OU, or account

You can use one of the following commands to detach an SCP:

- AWS CLI: detach-policy

  The following example detaches the specified SCP from the specified OU.

  ```
  $ aws organizations detach-policy \
      --policy-id p-i9j8k7l6m5 \
      --target-id ou-a1b2-f6g7h222
  ```
- AWS SDKs: DetachPolicy

The policy change takes effect immediately, affecting the permissions of IAM users and roles in the attached account or all accounts under the attached root or OU

Strategies for using SCPs

You can configure the service control policies (SCPs) in your organization to work as either of the following:

- A deny list (p. 117) – actions are allowed by default, and you specify what services and actions are prohibited
- An allow list (p. 118) – actions are prohibited by default, and you specify what services and actions are allowed

Tip
You can use service last accessed data in IAM to update your SCPs to restrict access to only the AWS services that you need. For more information, see Viewing Organizations Service Last Accessed Data for Organizations in the *IAM User Guide.*

Using SCPs as a deny list

The default configuration of AWS Organizations supports using SCPs as deny lists. Using a deny list strategy, account administrators can delegate all services and actions until you create and attach an SCP that denies a specific service or set of actions. Deny statements require less maintenance, because you don't need to update them when AWS adds new services. Deny statements usually use less space, thus making it easier to stay within the maximum size for SCPs (p. 340). In a statement where the `Effect` element has a value of `Deny`, you can also restrict access to specific resources, or define conditions for when SCPs are in effect.

To support this, AWS Organizations attaches an AWS managed SCP named FullAWSAccess to every root and OU when it's created. This policy allows all services and actions. It's always available for you to attach or detach from the entities in your organization as needed. Because the policy is an AWS managed SCP, you can't modify or delete it. The policy looks like the following.

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

This policy enables account administrators to delegate permissions for any service or operation until you create and attach an SCP that denies some access. You can attach an SCP that explicitly prohibits actions that you don't want users and roles in certain accounts to perform.

Such a policy might look like the following example, which prevents users in the affected accounts from performing any actions for the Amazon DynamoDB service. The organization administrator can detach the `FullAWSAccess` policy and attach this one instead. This SCP still allows all other services and their actions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowsAllActions",
            "Effect": "Allow",
            "Action": "*",
            "Resource": "*"
        },
        {
            "Sid": "DenyDynamoDB",
            "Effect": "Deny",
            "Action": "dynamodb:*",
            "Resource": "*"
        }
    ]
}
```

The users in the affected accounts can't perform DynamoDB actions because the explicit `Deny` element in the second statement overrides the explicit `Allow` in the first. You could also configure this by leaving the `FullAWSAccess` policy in place and then attaching a second policy that has only the `Deny` statement in it, as shown here.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "dynamodb:*",
            "Resource": "*"
        }
    ]
}
```

The combination of the `FullAWSAccess` policy and the `Deny` statement in the preceding DynamoDB policy that is applied to a root or OU has the same effect as the single policy that contains both statements. All policies that apply at a specified level are combined. Each statement, no matter which policy originated it, is evaluated according to the rules discussed earlier (that is, an ***explicit*** `Deny` overrides an ***explicit*** `Allow`, which overrides the default ***implicit*** `Deny`).

Using SCPs as an allow list

To use SCPs as an allow list, you must replace the AWS managed `FullAWSAccess` SCP with an SCP that explicitly permits only those services and actions that you want to allow. By removing the default `FullAWSAccess` SCP, all actions for all services are now implicitly denied. Your custom SCP then overrides the implicit `Deny` with an explicit `Allow` for only those actions that you want to permit. For a permission to be enabled for a specified account, every SCP from the root through each OU in the direct path to the account, and even attached to the account itself, must allow that permission.

Notes

- An `Allow` statement in an SCP permits the `Resource` element to only have a `"*"` entry.

- An Allow statement in an SCP can't have a Condition element at all.

An allow list policy might look like the following example, which enables account users to perform operations for Amazon Elastic Compute Cloud (Amazon EC2) and Amazon CloudWatch, but no other service. All SCPs in parent OUs and the root also must explicitly allow these permissions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:*",
                "cloudwatch:*"
            ],
            "Resource": "*"
        }
    ]
}
```

SCP syntax

Service control policies (SCPs) use a similar syntax to that used by AWS Identity and Access Management (IAM) permission policies and resource-based policies (like Amazon S3 bucket policies). For more information about IAM policies and their syntax, see Overview of IAM Policies in the *IAM User Guide*.

An SCP is a plaintext file that is structured according to the rules of JSON. It uses the elements that are described in this topic.

> **Note**
> All characters in your SCP count against its maximum size (p. 340). The examples in this guide show the SCPs formatted with extra white space to improve their readability. However, to save space if your policy size approaches the maximum size, you can delete any white space, such as space characters and line breaks that are outside quotation marks.

For general information about SCPs, see Service control policies (SCPs) (p. 104).

Elements summary

The following table summarizes the policy elements that you can use in SCPs. Some policy elements are available only in SCPs that deny actions. The **Supported effects** column lists the effect type that you can use with each policy element in SCPs.

Element	Purpose	Supported effects
Version (p. 121)	Specifies the language syntax rules to use for processing the policy.	Allow, Deny
Statement (p. 121)	Serves as the container for policy elements.	Allow, Deny

Element	Purpose	Supported effects
	You can have multiple statements in SCPs.	
Statement ID (Sid) (p. 122)	(Optional) Provides a friendly name for the statement.	`Allow`, `Deny`
Effect (p. 122)	Defines whether the SCP statement allows (p. 7) or denies (p. 8) access to the IAM users and roles in an account.	`Allow`, `Deny`
Action (p. 123)	Specifies AWS service and actions that the SCP allows or denies.	`Allow`, `Deny`
NotAction (p. 123)	Specifies AWS service and actions that are exempt from the SCP. Used instead of the `Action` element.	`Deny`
Resource (p. 124)	Specifies the AWS resources that the SCP applies to.	`Deny`

Element	Purpose	Supported effects
Condition (p. 125)	Specifies conditions for when the statement is in effect.	`Deny`

The following sections provide more information and examples of how policy elements are used in SCPs.

`Version` element

Every SCP must include a `Version` element with the value `"2012-10-17"`. This is the same version value as the most recent version of IAM permission policies.

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

For more information, see IAM JSON Policy Elements: Version in the *IAM User Guide*.

`Statement` element

An SCP consists of one or more `Statement` elements. You can have only one `Statement` keyword in a policy, but the value can be a JSON array of statements (surrounded by [ ] characters).

The following example shows a single statement that consists of single `Effect`, `Action`, and `Resource` elements.

```
    "Statement": {
        "Effect": "Allow",
        "Action": "*",
        "Resource": "*"
    }
```

The following example includes two statements as an array list inside one `Statement` element. The first statement allows all actions, while the second denies any EC2 actions. The result is that an administrator in the account can delegate any permission *except* those from Amazon Elastic Compute Cloud (Amazon EC2).

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "*",
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": "ec2:*",
            "Resource": "*"
        }
    ]
```

For more information, see IAM JSON Policy Elements: Statement in the *IAM User Guide*.

Statement ID (`Sid`) element

The `Sid` is an optional identifier that you provide for the policy statement. You can assign a `Sid` value to each statement in a statement array. The following example SCP shows a sample `Sid` statement.

```
{
    "Statement": {
        "Sid": "AllowsAllActions",
        "Effect": "Allow",
        "Action": "*",
        "Resource": "*"
    }
}
```

For more information, see IAM JSON Policy Elements: Id in the *IAM User Guide*.

`Effect` element

Each statement must contain one `Effect` element. The value can be either `Allow` or `Deny`. It affects any actions listed in the same statement.

For more information, see IAM JSON Policy Elements: Effect in the *IAM User Guide*.

`"Effect": "Allow"`

The following example shows an SCP with a statement that contains an `Effect` element with a value of `Allow` that permits account users to perform actions for the Amazon S3 service. This example is useful in an organization that uses the allow list strategy (p. 7) (where the default `FullAWSAccess` policies are all detached so that permissions are implicitly denied by default). The result is that the statement allows (p. 7) the Amazon S3 permissions for any attached accounts:

```
{
    "Statement": {
        "Effect": "Allow",
        "Action": "s3:*",
        "Resource": "*"
    }
}
```

Even though this statement uses the same `Allow` value keyword as an IAM permission policy, in an SCP it doesn't actually grant a user permission to do anything. Instead, SCPs act as *filters* that specify the maximum permissions for the accounts in an organization, organizational unit (OU), or account. In the preceding example, even if a user in the account had the `AdministratorAccess` managed policy attached, this SCP limits ***all*** users in affected accounts to only Amazon S3 actions.

`"Effect": "Deny"`

In a statement where the `Effect` element has a value of `Deny`, you can also restrict access to specific resources or define conditions for when SCPs are in effect.

The following shows an example of how to use a condition key in a deny statement.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Deny",
        "Action": "ec2:RunInstances",
        "Resource": "arn:aws:ec2:*:*:instance/*",
        "Condition": {
            "StringNotEquals": {
                "ec2:InstanceType": "t2.micro"
            }
        }
    }
}
```

This statement in an SCP sets a guardrail to prevent affected accounts (where the SCP is attached to the account itself or to the organization root or OU that contains the account), from launching Amazon EC2 instances if the Amazon EC2 instance isn't set to `t2.micro`. Even if an IAM policy that allows this action is attached to the account, the guardrail created by the SCP prevents it.

`Action` and `NotAction` elements

Each statement must contain one of the following:

- In allow and deny statements, an `Action` element.
- In deny statements only (where the value of the `Effect` element is `Deny`), an `Action` *or* `NotAction` element.

The value for the `Action` or `NotAction` element is a list (a JSON array) of strings that identify AWS services and actions that are allowed or denied by the statement.

Each string consists of the abbreviation for the service (such as "s3", "ec2", "iam", or "organizations"), in all lowercase, followed by a colon and then an action from that service. The actions and notactions are case sensitive and must be typed as shown in each service's documentation. Generally, they are all typed with each word starting with an uppercase letter and the rest lowercase. For example: `"s3:ListAllMyBuckets"`.

You also can use wildcard characters such as asterisk (*) or question mark (?) in an SCP:

- Use an asterisk (*) as a wildcard to match multiple actions that share part of a name. The value `"s3:*"` means all actions in the Amazon S3 service. The value `"ec2:Describe*"` matches only the EC2 actions that begin with "Describe".
- Use the question mark (?) wildcard to match a single character.

> **Note**
> In an SCP, the wildcard characters (*) and (?) in an `Action` or `NotAction` element can be used only by itself or at the end of the string. It can't appear at the beginning or middle of the string. Therefore, `"servicename:action*"` is valid, but `"servicename:*action"` and `"servicename:some*action"` are both invalid in SCPs.

For a list of all the services and the actions that they support in both AWS Organizations SCPs and IAM permission policies, see Actions, Resources, and Condition Keys for AWS Services in the *IAM User Guide*.

For more information, see IAM JSON Policy Elements: Action and IAM JSON Policy Elements: NotAction in the *IAM User Guide*.

Example of `Action` element

The following example shows an SCP with a statement that permits account administrators to delegate describe, start, stop, and terminate permissions for EC2 instances in the account. This is an example of an allow list (p. 7), and is useful when the default `Allow *` policies are ***not*** attached so that, by default, permissions are implicitly denied. If the default `Allow *` policy is still attached to the root, OU, or account to which the following policy is attached, the policy has no effect.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
          "ec2:DescribeInstances", "ec2:DescribeImages", "ec2:DescribeKeyPairs",
          "ec2:DescribeSecurityGroups", "ec2:DescribeAvailabilityZones",
 "ec2:RunInstances",
          "ec2:TerminateInstances", "ec2:StopInstances", "ec2:StartInstances"
        ],
        "Resource": "*"
    }
}
```

The following example shows how you can deny access (p. 8) to services that you don't want used in attached accounts. It assumes that the default `"Allow *"` SCPs are still attached to all OUs and the root. This example policy prevents the account administrators in attached accounts from delegating any permissions for the IAM, Amazon EC2, and Amazon RDS services. Any action from other services can be delegated as long as there isn't another attached policy that denies them.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Deny",
        "Action": [ "iam:*", "ec2:*", "rds:*" ],
        "Resource": "*"
    }
}
```

Example of `NotAction` element

The following example shows how you can use a `NotAction` element to exclude AWS services from the effect of the policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "LimitActionsInRegion",
      "Effect": "Deny",
      "NotAction": "iam:*",
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:RequestedRegion": "us-west-1"
         }
       }
     }
   ]
}
```

With this statement, affected accounts are limited to taking actions in the specified AWS Region, except when using IAM actions.

`Resource` element

In statements where the `Effect` element has a value of `Allow`, you can specify only "*" in the `Resource` element of an SCP. You can't specify individual resource Amazon Resource Names (ARNs).

You also can use wildcard characters such as asterisk (*) or question mark (?) in the resource element:

- Use an asterisk (*) as a wildcard to match multiple actions that share part of a name.
- Use the question mark (?) wildcard to match a single character.

In statements where the `Effect` element has a value of `Deny`, you *can* specify individual ARNs, as shown in the following example.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAccessToAdminRole",
      "Effect": "Deny",
      "Action": [
        "iam:AttachRolePolicy",
        "iam:DeleteRole",
        "iam:DeleteRolePermissionsBoundary",
        "iam:DeleteRolePolicy",
        "iam:DetachRolePolicy",
        "iam:PutRolePermissionsBoundary",
        "iam:PutRolePolicy",
        "iam:UpdateAssumeRolePolicy",
        "iam:UpdateRole",
        "iam:UpdateRoleDescription"
      ],
      "Resource": [
        "arn:aws:iam::*:role/role-to-deny"
      ]
    }
  ]
}
```

This SCP restricts IAM users and roles in affected accounts from making changes to a common administrative IAM role created in all accounts in your organization.

For more information, see IAM JSON Policy Elements: Resource in the *IAM User Guide*.

`Condition` element

You can specify a `Condition` element in deny statements in an SCP.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyAllOutsideEU",
            "Effect": "Deny",
            "NotAction": [
                "cloudfront:*",
                "iam:*",
                "route53:*",
                "support:*"
            ],
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestedRegion": [
                        "eu-central-1",
                        "eu-west-1"
                    ]
                }
            }
        }
    ]
}
```

This SCP denies access to any operations outside the `eu-central-1` and `eu-west-1` Regions, except for actions in the listed services.

For more information, see IAM JSON Policy Elements: Condition in the *IAM User Guide*.

Unsupported elements

The following elements aren't supported in SCPs:

- `Principal`
- `NotPrincipal`
- `NotResource`

Example service control policies

The example service control policies (SCPs) (p. 104) displayed in this topic are for information purposes only.

Before using these examples
Before you use these example SCPs in your organization, do the following:

- Carefully review and customize the SCPs for your unique requirements.
- Thoroughly test the SCPs in your environment with the AWS services that you use.

  The example policies in this section demonstrate the implementation and use of SCPs. They're ***not*** intended to be interpreted as official AWS recommendations or best practices to be implemented exactly as shown. It is your responsibility to carefully test any deny-based policies for its suitability to solve the business requirements of your environment. Deny-based service control policies can unintentionally limit or block your use of AWS services unless you add the necessary exceptions to the policy. For an example of such an exception, see the first example that exempts global services from the rules that block access to unwanted AWS Regions.
- Remember that an SCP affects every user and role, including the root user, in every account that it's attached to.

Tip
You can use service last accessed data in IAM to update your SCPs to restrict access to only the AWS services that you need. For more information, see Viewing Organizations Service Last Accessed Data for Organizations in the *IAM User Guide.*

Each of the following policies is an example of a deny list policy (p. 117) strategy. Deny list policies must be attached along with other policies that allow the approved actions in the affected accounts. For example, the default `FullAWSAccess` policy permits the use of all services in an account. This policy is attached by default to the root, all organizational units (OUs), and all accounts. It doesn't actually grant the permissions; no SCP does. Instead, it enables administrators in that account to delegate access to those actions by attaching standard AWS Identity and Access Management (IAM) permissions policies to users, roles, or groups in the account. Each of these deny list policies then overrides any policy by blocking access to the specified services or actions.

Examples

- General examples (p. 127)

General examples

Deny access to AWS based on the requested AWS Region

This SCP denies access to any operations outside of the specified Regions. Replace `eu-central-1` and `eu-west-1` with the AWS Regions you want to use. It provides exemptions for operations in approved global services. This example also shows how to exempt requests made by either of two specified administrator roles.

> **Note**
> To use the Region deny SCP with AWS Control Tower, see Deny access to AWS based on the requested AWS Region (p. 127).

This policy uses the `Deny` effect to deny access to all requests for operations that don't target one of the two approved regions (`eu-central-1` and `eu-west-1`). The NotAction element enables you to list services whose operations (or individual operations) are exempted from this restriction. Because global services have endpoints that are physically hosted by the `us-east-1` Region , they must be exempted in this way. With an SCP structured this way, requests made to global services in the `us-east-1` Region are allowed if the requested service is included in the `NotAction` element. Any other requests to services in the `us-east-1` Region are denied by this example policy.

> **Note**
> ***This example might not include all of the latest global AWS services or operations.*** Replace the list of services and operations with the global services used by accounts in your organization.

Tip

You can view the service last accessed data in the IAM console to determine what global services your organization uses. The **Access Advisor** tab on the details page for an IAM user, group, or role displays the AWS services that have been used by that entity, sorted by most recent access.

Considerations

- AWS KMS and AWS Certificate Manager support Regional endpoints. However, if you want to use them with a global service such as Amazon CloudFront you must include them in the global service exclusion list in the following example SCP. A global service like Amazon CloudFront typically requires access to AWS KMS and ACM in the same region, which for a global service is the US East (N. Virginia) Region (`us-east-1`).
- By default, AWS STS is a global service and must be included in the global service exclusion list. However, you can enable AWS STS to use Region endpoints instead of a single global endpoint. If you do this, you can remove STS from the global service exemption list in the following example SCP. For more information see Managing AWS STS in an AWS Region.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyAllOutsideEU",
            "Effect": "Deny",
            "NotAction": [
                "a4b:*",
                "acm:*",
                "aws-marketplace-management:*",
                "aws-marketplace:*",
                "aws-portal:*",
                "budgets:*",
                "ce:*",
                "chime:*",
                "cloudfront:*",
                "config:*",
                "cur:*",
                "directconnect:*",
                "ec2:DescribeRegions",
                "ec2:DescribeTransitGateways",
                "ec2:DescribeVpnGateways",
                "fms:*",
                "globalaccelerator:*",
                "health:*",
                "iam:*",
                "importexport:*",
                "kms:*",
                "mobileanalytics:*",
                "networkmanager:*",
                "organizations:*",
                "pricing:*",
                "route53:*",
                "route53domains:*",
                "s3:GetAccountPublic*",
                "s3:ListAllMyBuckets",
                "s3:PutAccountPublic*",
                "shield:*",
                "sts:*",
                "support:*",
                "trustedadvisor:*",
                "waf-regional:*",
                "waf:*",
                "wafv2:*",
                "wellarchitected:*"
            ],
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestedRegion": [
                        "eu-central-1",
                        "eu-west-1"
                    ]
                },
                "ArnNotLike": {
                    "aws:PrincipalARN": [
                        "arn:aws:iam::*:role/Role1AllowedToBypassThisSCP",
                        "arn:aws:iam::*:role/Role2AllowedToBypassThisSCP"
                    ]
                }
            }
        }
    ]
}
```

Prevent IAM users and roles from making certain changes

This SCP restricts IAM users and roles from making changes to the specified IAM role that you created in all accounts in your organization.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAccessToASpecificRole",
      "Effect": "Deny",
      "Action": [
        "iam:AttachRolePolicy",
        "iam:DeleteRole",
        "iam:DeleteRolePermissionsBoundary",
        "iam:DeleteRolePolicy",
        "iam:DetachRolePolicy",
        "iam:PutRolePermissionsBoundary",
        "iam:PutRolePolicy",
        "iam:UpdateAssumeRolePolicy",
        "iam:UpdateRole",
        "iam:UpdateRoleDescription"
      ],
      "Resource": [
        "arn:aws:iam::*:role/name-of-role-to-deny"
      ]
    }
  ]
}
```

Prevent IAM users and roles from making specified changes, with an exception for a specified admin role

This SCP builds on the previous example to make an exception for administrators. It prevents IAM users and roles in affected accounts from making changes to a common administrative IAM role created in all accounts in your organization *except* for administrators using a specified role.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAccessWithException",
      "Effect": "Deny",
      "Action": [
        "iam:AttachRolePolicy",
        "iam:DeleteRole",
        "iam:DeleteRolePermissionsBoundary",
        "iam:DeleteRolePolicy",
        "iam:DetachRolePolicy",
        "iam:PutRolePermissionsBoundary",
        "iam:PutRolePolicy",
        "iam:UpdateAssumeRolePolicy",
        "iam:UpdateRole",
        "iam:UpdateRoleDescription"
      ],
      "Resource": [
        "arn:aws:iam::*:role/name-of-role-to-deny"
      ],
      "Condition": {
        "StringNotLike": {
          "aws:PrincipalARN":"arn:aws:iam::*:role/name-of-admin-role-to-allow"
        }
      }
    }
  ]
}
```

Require MFA to perform an API action

Use an SCP like the following to require that multi-factor authentication (MFA) is enabled before an IAM user or role can perform an action. In this example, the action is to stop an Amazon EC2 instance.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyStopAndTerminateWhenMFAIsNotPresent",
      "Effect": "Deny",
      "Action": [
        "ec2:StopInstances",
        "ec2:TerminateInstances"
      ],
      "Resource": "*",
      "Condition": {"BoolIfExists": {"aws:MultiFactorAuthPresent": false}}
    }
  ]
}
```

Block service access for the root user

The following policy restricts all access to the specified actions for the root user in a member account. If you want to prevent your accounts from using root credentials in specific ways, add your own actions to this policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RestrictEC2ForRoot",
      "Effect": "Deny",
      "Action": [
        "ec2:*"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringLike": {
          "aws:PrincipalArn": [
            "arn:aws:iam::*:root"
          ]
        }
      }
    }
  ]
}
```

Prevent member accounts from leaving the organization

The following policy blocks use of the `LeaveOrganization` API operation so that administrators of member accounts can't remove their accounts from the organization.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "organizations:LeaveOrganization"
            ],
            "Resource": "*"
        }
    ]
}
```

Example SCPs for Amazon CloudWatch

Examples in this category

- Prevent users from disabling CloudWatch or altering its configuration (p. 131)

Prevent users from disabling CloudWatch or altering its configuration

A lower-level CloudWatch operator needs to monitor dashboards and alarms. However, the operator must not be able to delete or change any dashboard or alarm that senior people might put into place. This SCP prevents users or roles in any affected account from running any of the CloudWatch commands that could delete or change your dashboards or alarms.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "cloudwatch:DeleteAlarms",
        "cloudwatch:DeleteDashboards",
        "cloudwatch:DisableAlarmActions",
        "cloudwatch:PutDashboard",
        "cloudwatch:PutMetricAlarm",
        "cloudwatch:SetAlarmState"
      ],
      "Resource": "*"
    }
  ]
}
```

Example SCPs for AWS Config

Examples in this category

- Prevent users from disabling AWS Config or changing its rules (p. 132)

Prevent users from disabling AWS Config or changing its rules

This SCP prevents users or roles in any affected account from running AWS Config operations that could disable AWS Config or alter its rules or triggers.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "config:DeleteConfigRule",
        "config:DeleteConfigurationRecorder",
        "config:DeleteDeliveryChannel",
        "config:StopConfigurationRecorder"
      ],
      "Resource": "*"
    }
  ]
}
```

Example SCPs for Amazon Elastic Compute Cloud (Amazon EC2)

Examples in this category

- Require Amazon EC2 instances to use a specific type (p. 132)

Require Amazon EC2 instances to use a specific type

With this SCP, any instance launches not using the `t2.micro` instance type are denied.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RequireMicroInstanceType",
      "Effect": "Deny",
      "Action": "ec2:RunInstances",
      "Resource": [
        "arn:aws:ec2:*:*:instance/*"
      ],
      "Condition": {
        "StringNotEquals": {
          "ec2:InstanceType": "t2.micro"
        }
      }
    }
  ]
}
```

Example SCPs for Amazon GuardDuty

Examples in this category

- Prevent users from disabling GuardDuty or modifying its configuration (p. 133)

Prevent users from disabling GuardDuty or modifying its configuration

This SCP prevents users or roles in any affected account from disabling GuardDuty or altering its configuration, either directly as a command or through the console. It effectively enables read-only access to the GuardDuty information and resources.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "guardduty:AcceptInvitation",
                "guardduty:ArchiveFindings",
                "guardduty:CreateDetector",
                "guardduty:CreateFilter",
                "guardduty:CreateIPSet",
                "guardduty:CreateMembers",
                "guardduty:CreatePublishingDestination",
                "guardduty:CreateSampleFindings",
                "guardduty:CreateThreatIntelSet",
                "guardduty:DeclineInvitations",
                "guardduty:DeleteDetector",
                "guardduty:DeleteFilter",
                "guardduty:DeleteInvitations",
                "guardduty:DeleteIPSet",
                "guardduty:DeleteMembers",
                "guardduty:DeletePublishingDestination",
                "guardduty:DeleteThreatIntelSet",
                "guardduty:DisassociateFromMasterAccount",
                "guardduty:DisassociateMembers",
                "guardduty:InviteMembers",
                "guardduty:StartMonitoringMembers",
                "guardduty:StopMonitoringMembers",
                "guardduty:TagResource",
                "guardduty:UnarchiveFindings",
                "guardduty:UntagResource",
                "guardduty:UpdateDetector",
                "guardduty:UpdateFilter",
                "guardduty:UpdateFindingsFeedback",
                "guardduty:UpdateIPSet",
                "guardduty:UpdatePublishingDestination",
                "guardduty:UpdateThreatIntelSet"
            ],
            "Resource": "*"
        }
    ]
}
```

Example SCPs for AWS Resource Access Manager

Examples in this category

- Preventing external sharing (p. 134)
- Allowing specific accounts to share only specified resource types (p. 134)

- Prevent sharing with organizations or organizational units (OUs) (p. 135)
- Allow sharing with only specified IAM users and roles (p. 135)

Preventing external sharing

The following example SCP prevents users from creating resource shares that allow sharing with IAM users and roles that aren't part of the organization.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "ram:CreateResourceShare",
                "ram:UpdateResourceShare"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "ram:RequestedAllowsExternalPrincipals": "true"
                }
            }
        }
    ]
}
```

Allowing specific accounts to share only specified resource types

The following SCP allows accounts `111111111111` and `222222222222` to create resource shares that share prefix lists, and to associate prefix lists with existing resource shares.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "OnlyNamedAccountsCanSharePrefixLists",
            "Effect": "Deny",
            "Action": [
                "ram:AssociateResourceShare",
                "ram:CreateResourceShare"
            ],
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:PrincipalAccount": [
                        "111111111111",
                        "222222222222"
                    ]
                },
                "StringEquals": {
                    "ram:RequestedResourceType": "ec2:PrefixList"
                }
            }
        }
    ]
}
```

Prevent sharing with organizations or organizational units (OUs)

The following SCP prevents users from creating resource shares that share resources with an AWS Organization or OUs.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "ram:CreateResourceShare",
                "ram:AssociateResourceShare"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringLike": {
                    "ram:Principal": [
                        "arn:aws:organizations::*:organization/*",
                        "arn:aws:organizations::*:ou/*"
                    ]
                }
            }
        }
    ]
}
```

Allow sharing with only specified IAM users and roles

The following example SCP allows users to share resources with *only* organization `o-12345abcdef`, organizational unit `ou-98765fedcba`, and account `111111111111`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "ram:AssociateResourceShare",
                "ram:CreateResourceShare"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringNotEquals": {
                    "ram:Principal": [
                        "arn:aws:organizations::123456789012:organization/o-12345abcdef",
                        "arn:aws:organizations::123456789012:ou/o-12345abcdef/
ou-98765fedcba",
                        "111111111111"
                    ]
                }
            }
        }
    ]
}
```

Example SCPs for tagging resources

Examples in this category

Require a tag on specified created resources

The following SCP prevents IAM users and roles in the affected accounts from creating certain resource types if the request doesn't include the specified tags.

Important

Remember to test Deny-based policies with the services you use in your environment. The following example is a simple block of creating untagged secrets or running untagged Amazon EC2 instances, and doesn't include any exceptions.

The following example policy is not compatible with AWS CloudFormation as written, because that service creates a secret and then tags it as two separate steps. This example policy effectively blocks AWS CloudFormation from creating a secret as part of a stack, because such an action would result, however briefly, in a secret that is not tagged as required.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyCreateSecretWithNoProjectTag",
      "Effect": "Deny",
      "Action": "secretsmanager:CreateSecret",
      "Resource": "*",
      "Condition": {
        "Null": {
          "aws:RequestTag/Project": "true"
        }
      }
    },
    {
      "Sid": "DenyRunInstanceWithNoProjectTag",
      "Effect": "Deny",
      "Action": "ec2:RunInstances",
      "Resource": [
        "arn:aws:ec2:*:*:instance/*",
        "arn:aws:ec2:*:*:volume/*"
      ],
      "Condition": {
        "Null": {
          "aws:RequestTag/Project": "true"
        }
      }
    },
    {
      "Sid": "DenyCreateSecretWithNoCostCenterTag",
      "Effect": "Deny",
      "Action": "secretsmanager:CreateSecret",
      "Resource": "*",
      "Condition": {
        "Null": {
          "aws:RequestTag/CostCenter": "true"
        }
      }
    },
    {
      "Sid": "DenyRunInstanceWithNoCostCenterTag",
      "Effect": "Deny",
      "Action": "ec2:RunInstances",
      "Resource": [
        "arn:aws:ec2:*:*:instance/*",
        "arn:aws:ec2:*:*:volume/*"
      ],
      "Condition": {
        "Null": {
          "aws:RequestTag/CostCenter": "true"
```

```
                }
            }
        }
    ]
}
```

For a list of all the services and the actions that they support in both AWS Organizations SCPs and IAM permission policies, see Actions, Resources, and Condition Keys for AWS Services in the *IAM User Guide*.

Prevent tags from being modified except by authorized principals

The following SCP shows how a policy can allow only authorized principals to modify the tags attached to your resources. This is an important part of using attribute-based access control (ABAC) as part of your AWS cloud security strategy. The policy allows a caller to modify the tags on only those resources where the authorization tag (in this example, `access-project`) exactly matches the same authorization tag attached to the user or role making the request. The policy also prevents the authorized user from changing the *value* of the tag that is used for authorization. The calling principal must have the authorization tag to make any changes at all.

This policy only blocks unauthorized users from changing tags. An authorized user who isn't blocked by this policy must still have a separate IAM policy that explicitly grants the `Allow` permission on the relevant tagging APIs. As an example, if your user has an administrator policy with `Allow */*` (allow all services and all operations), then the combination results in the administrator user being allowed to change *only* those tags that have an authorization tag value that matches the authorization tag value attached to the user's principal. This is because the explicit `Deny` in the this policy overrides the explicit `Allow` in the administrator policy.

Important

This is not a complete policy solution and should not be used as shown here. This example is intended only to illustrate part of an ABAC strategy and needs to be customized and tested for production environments.

For the complete policy with a detailed analysis of how it works, see Securing resource tags used for authorization using a service control policy in AWS Organizations

Remember to test Deny-based policies with the services you use in your environment.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyModifyTagsIfResAuthzTagAndPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ec2:CreateTags",
                "ec2:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "ec2:ResourceTag/access-project": "${aws:PrincipalTag/access-project}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-admins/iam-
admin"
                },
                "Null": {
                    "ec2:ResourceTag/access-project": false
                }
            }
        },
        {
            "Sid": "DenyModifyResAuthzTagIfPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ec2:CreateTags",
                "ec2:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestTag/access-project": "${aws:PrincipalTag/access-project}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-admins/iam-
admin"
                },
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": [
                        "access-project"
                    ]
                }
            }
        },
        {
            "Sid": "DenyModifyTagsIfPrinTagNotExists",
            "Effect": "Deny",
            "Action": [
                "ec2:CreateTags",
                "ec2:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-admins/iam-
admin"
                },
                "Null": {
                    "aws:PrincipalTag/access-project": true
                }
            }
        }
    ]
}
```

Example SCPs for Amazon Virtual Private Cloud (Amazon VPC)

Examples in this category

- Prevent users from deleting Amazon VPC flow logs (p. 138)
- Prevent any VPC that doesn't already have internet access from getting it (p. 139)

Prevent users from deleting Amazon VPC flow logs

This SCP prevents users or roles in any affected account from deleting Amazon Elastic Compute Cloud (Amazon EC2) flow logs or CloudWatch log groups or log streams.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "ec2:DeleteFlowLogs",
        "logs:DeleteLogGroup",
        "logs:DeleteLogStream"
      ],
      "Resource": "*"
    }
  ]
 }
```

Prevent any VPC that doesn't already have internet access from getting it

This SCP prevents users or roles in any affected account from changing the configuration of your Amazon EC2 virtual private clouds (VPCs) to grant them direct access to the internet. It doesn't block existing direct access or any access that routes through your on-premises network environment.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "ec2:AttachInternetGateway",
        "ec2:CreateInternetGateway",
        "ec2:CreateEgressOnlyInternetGateway",
        "ec2:CreateVpcPeeringConnection",
        "ec2:AcceptVpcPeeringConnection",
        "globalaccelerator:Create*",
        "globalaccelerator:Update*"
      ],
      "Resource": "*"
    }
  ]
}
```

AI services opt-out policies

For information and procedures common to all policy types, see the following topics:

- Enable and disable policy types (p. 85)
- Get details about your policies (p. 87)
- Policy syntax and inheritance (p. 92)

Certain AWS artificial intelligence (AI) services, including Amazon CodeGuru Profiler, Amazon Comprehend, Amazon Fraud Detector, Amazon Lex, Amazon Polly, Amazon Rekognition, Amazon Textract, Amazon Transcribe, Amazon Translate, Amazon GuardDuty, and Contact Lens for Amazon Connect, may store and use customer content processed by those services for the development and continuous improvement of other AWS services. As an AWS customer, you can opt out of having your content stored or used for service improvements.

> **Note**
> The AWS artificial intelligence (AI) services need to store your data even if you opt-out from AWS using your data for service improvements.

Instead of configuring this setting individually for each AWS account that your organization uses, you can configure an organization policy that enforces your setting choice on all accounts that are members of the organization. You can choose to opt out of content storage and use for an individual AI service, or for all of the covered services at once. You can query the effective policy applicable to each account to see the effects of your setting choices.

Important

- When you specify an opt in or opt out preference for a service, that setting is global and applied to all AWS Regions. Setting the value from within one AWS Region replicates to all other Regions.
- When you opt out of content use by an AWS AI service, that service deletes all of the associated historical content that was shared with AWS before you set the option. This deletion should be limited to data stored that is not required to provide service functions.

Getting started with AI services opt-out policies

Follow these steps to get started using Artificial Intelligence (AI) services opt-out policies.

1. Enable AI services opt-out policies for your organization (p. 85).
2. Create an AI services opt-out policy (p. 140).
3. Attach the AI services opt-out policy to your organization's root, OU, or account (p. 145).
4. View the combined effective AI services opt-out policy that applies to an account (p. 148).

For all of these steps, you sign in as an AWS Identity and Access Management (IAM) user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.

Other information

- Learn policy syntax for AI services opt-out policies and see policy examples (p. 150)

Creating, updating, and deleting AI services opt-out policies

In this topic:

- After you enable AI service opt-out policies (p. 85) for your organization, you can create a policy (p. 140).
- When your opt-out requirements change, you can update an existing policy (p. 142).
- When you no longer need a policy and after you detach it from all organizational units (OUs) and accounts, you can delete it (p. 145).

Creating an AI services opt-out policy

Minimum permissions

To create an AI services opt-out policy, you need permission to run the following action:

- `organizations:CreatePolicy`

AWS Management Console

To create an AI services opt-out policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **AI services opt-out policies** page, choose **Create policy**.

3. On the **Create new AI services opt-out policy** page, enter a **Policy name** and an optional **Policy description**.
4. (Optional) You can add one or more tags to the policy by choosing **Add tag** and then entering a key and an optional value. Leaving the value blank sets it to an empty string; it isn't `null`. You can attach up to 50 tags to a policy. For more information, see Tagging AWS Organizations resources (p. 220).
5. Enter or paste the policy text in the **JSON** tab. For information about AI services opt-out policy syntax, see AI services opt-out policy syntax and examples (p. 150). For example policies that you can use as a starting point, see AI services opt-out policy examples (p. 152).
6. When you're finished editing your policy, choose **Create policy** at the lower-right corner of the page.

AWS CLI & AWS SDKs

To create an AI services opt-out policy

You can use one of the following to create a tag policy:

- AWS CLI: create-policy

1. Create an AI services opt-out policy like the following, and store it in a text file. Note that "`optOut`" and "`optIn`" are case-sensitive.

```
{
    "services": {
        "default": {
            "opt_out_policy": {
                "@@assign": "optOut"
            }
        },
        "rekognition": {
            "opt_out_policy": {
                "@@assign": "optIn"
            }
        }
    }
}
```

This AI services opt-out policy specifies that all accounts affected by the policy are opted out of all AI services except for Amazon Rekognition.

2. Import the JSON policy file to create a new policy in the organization. In this example, the previous JSON file was named `policy.json`.

```
$ aws organizations create-policy \
    --type AISERVICES_OPT_OUT_POLICY \
    --name "MyTestPolicy" \
    --description "My test policy" \
    --content file://policy.json
{
    "Policy": {
        "Content": "{\"services\":{\"default\":{\"opt_out_policy\":{\"@@assign\":
\"optOut\"}},\"rekognition\":{\"opt_out_policy\":{\"@@assign\":\"optIn\"}}}}",
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5"
            "Arn": "arn:aws:organizations::o-aa111bb222:policy/
aiservices_opt_out_policy/p-i9j8k7l6m5",
            "Description": "My test policy",
            "Name": "MyTestPolicy",
            "Type": "AISERVICES_OPT_OUT_POLICY"
            }
        }
}
```

- AWS SDKs: CreatePolicy

What to do next

After you create an AI services opt-out policy, you can put your opt-out choices into effect. To do that, you can attach the policy (p. 199) to the organization root, organizational units (OUs), AWS accounts within your organization, or a combination of all of those.

Updating an AI services opt-out policy

Minimum permissions

To update an AI services opt-out policy, you must have permission to run the following actions:

- `organizations:UpdatePolicy` with a `Resource` element in the same policy statement that includes the ARN of the specified policy (or "*")
- `organizations:DescribePolicy` with a `Resource` element in the same policy statement that includes the Amazon Resource Name (ARN) of the specified policy (or "*")

AWS Management Console

To update an AI services opt-out policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AI services opt-out policies** page, choose the name of the policy that you want to update.
3. On the policy's detail page, choose **Edit policy**.
4. You can enter a new **Policy name**, **Policy description**, or edit the **JSON** policy text. For information about AI services opt-out policy syntax, see AI services opt-out policy syntax and examples (p. 150). For example policies that you can use as a starting point, see AI services opt-out policy examples (p. 152).
5. When you're finished updating the policy, choose **Save changes**.

AWS CLI & AWS SDKs

To update a policy

You can use one of the following to update a policy:

- AWS CLI: update-policy

  The following example renames an AI services opt-out policy.

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --name "Renamed policy"
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
aiservices_opt_out_policy/p-i9j8k7l6m5",
            "Name": "Renamed policy",
            "Type": "AISERVICES_OPT_OUT_POLICY",
            "AwsManaged": false
        },
        "Content": "{\"services\":{\"default\":{\"opt_out_policy\":   ....TRUNCATED
 FOR BREVITY...   :{\"@@assign\":\"optIn\"}}}}"
    }
}
```

The following example adds or changes the description for an AI services opt-out policy.

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --description "My new description"
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
aiservices_opt_out_policy/p-i9j8k7l6m5",
            "Name": "Renamed policy",
            "Description": "My new description",
            "Type": "AISERVICES_OPT_OUT_POLICY",
            "AwsManaged": false
        },
        "Content": "{\"services\":{\"default\":{\"opt_out_policy\":   ....TRUNCATED
 FOR BREVITY...   :{\"@@assign\":\"optIn\"}}}}"
    }
}
```

The following example changes the JSON policy document attached to an AI services opt-out policy. In this example, the content is taken from a file called `policy.json` with the following text:

```
{
    "services": {
        "default": {
            "opt_out_policy": {
                "@@assign": "optOut"
            }
        },
        "comprehend": {
            "opt_out_policy": {
                "@@operators_allowed_for_child_policies": ["@@none"],
                "@@assign": "optOut"
            }
        },
        "rekognition": {
            "opt_out_policy": {
                "@@assign": "optIn"
            }
        }
    }
}
```

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --content file://policy.json
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
aiservices_opt_out_policy/p-i9j8k7l6m5",
            "Name": "Renamed policy",
            "Description": "My new description",
            "Type": "AISERVICES_OPT_OUT_POLICY",
            "AwsManaged": false
        },
         "Content": "{\n\"services\": {\n\"default\": {\n\"   ....TRUNCATED FOR
 BREVITY....    ": \"optIn\"\n}\n}\n}\n}\n"}
}
```

- AWS SDKs: UpdatePolicy

Editing tags attached to an AI services opt-out policy

When you sign in to your organization's management account, you can add or remove the tags attached to an AI services opt-out policy. For more information about tagging, see Tagging AWS Organizations resources (p. 220).

Minimum permissions
To edit the tags attached to an AI services opt-out policy in your AWS organization, you must have the following permissions:

- `organizations:DescribeOrganization`– required only when using the Organizations console
- `organizations:DescribePolicy`– required only when using the Organizations console
- `organizations:TagResource`
- `organizations:UntagResource`

AWS Management Console

To edit the tags attached to an AI services opt-out policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **AI services opt-out policies** page, choose the name of the policy with the tags that you want to edit.
3. On the chosen policy's detail page, choose the **Tags** tab, and then choose **Manage tags**.
4. You can perform any of these actions on this page:
   - Edit the value for any tag by entering a new value over the old one. You can't modify the key. To change a key, you must delete the tag with the old key and add a tag with the new key.
   - Remove an existing tag by choosing **Remove**.
   - Add a new tag key and value pair. Choose **Add tag**, then enter the new key name and optional value in the provided boxes. If you leave the **Value** box empty, the value is an empty string; it isn't `null`.
5. Choose **Save changes** after you've made all the additions, removals, and edits you want to make.

AWS CLI & AWS SDKs

To edit the tags attached to a AI services opt-out policy

You can use one of the following commands to edit the tags attached to a AI services opt-out policy:

- AWS CLI: tag-resource and untag-resource

- AWS SDKs: TagResource and UntagResource

Deleting an AI services opt-out policy

When you sign in to your organization's management account, you can delete a policy that you no longer need in your organization.

Before you can delete a policy, you must first detach it from all attached entities.

Minimum permissions
To delete a policy, you must have permission to run the following action:

- `organizations:DescribePolicy` (console only – to navigate to the policy)
- `organizations:DeletePolicy`

AWS Management Console

To delete an AI services opt-out policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **AI services opt-out policies** page, choose the name of the policy that you want to delete.
3. You must first detach the policy that you want to delete from all roots, OUs, and accounts. Choose the **Targets** tab, choose the radio button next to each root, OU, or account that is shown in the **Targets** list, and then choose **Detach**. In the confirmation dialog box, choose **Detach**. Repeat until you remove all targets.
4. Choose **Delete** at the top of the page.
5. On the confirmation dialog box, enter the name of the policy, and then choose **Delete**.

AWS CLI & AWS SDKs

To delete an AI services opt-out policy

You can use one of the following to delete a policy:

- AWS CLI: delete-policy

  The following example deletes the specified policy. It works only if the policy is not attached to any root, OU, or account.

```
$ aws organizations delete-policy \
    --policy-id p-i9j8k7l6m5
```

  This command produces no output when successful.
- AWS SDKs: DeletePolicy

Attaching and detaching AI services opt-out policies

You can use Artificial Intelligence (AI) services opt-out policies on an entire organization as well as on organizational units (OUs) and individual accounts. What the AI services opt-out policy applies to depends on what organization element you attach it to:

- When you attach an AI services opt-out policy to your *organization root*, the policy applies to all of that root's member OUs and accounts.
- When you attach an AI services opt-out policy to an *OU*, that policy applies to the accounts that belong to the OU or any of its child OUs. Those accounts are also subject to any policy attached to the organization root.
- When you attach an AI services opt-out policy to an *account*, that policy applies to only that account. The account is also subject to any policy attached to the organization root and any OUs that the account belongs to.

The aggregation of any AI services opt-out policies the account inherits from the root and parent OUs, as well as any policies directly attached to the account, is the *effective policy* (p. 148). For information about how policies are merged to the effective policy, see Understanding policy inheritance (p. 91).

Minimum permissions

To attach AI services opt-out policies, you must have permission to run the following action:

- `organizations:AttachPolicy`

AWS Management Console

You can attach an AI services opt-out policy by either navigating to the policy or to the root, OU, or account that you want to attach the policy to.

To attach an AI services opt-out policy by navigating to the root, OU, or account

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **AWS accounts** page, navigate to and then choose the name of the root, OU, or account that you want to attach a policy to. You might have to expand OUs (choose the ▶) to find the OU or account that you want.
3. In the **Policies** tab, in the entry for **AI service opt-out policies**, choose **Attach**.
4. Find the policy that you want and choose **Attach policy**.

   The list of attached AI services opt-out policies on the **Policies** tab is updated to include the new addition. The policy change takes effect immediately.

To attach an AI services opt-out policy by navigating to the policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **AI services opt-out policies** page, choose the name of the policy that you want to attach.
3. On the **Targets** tab, choose **Attach**.
4. Choose the radio button next to the root, OU, or account that you want to attach the policy to. You might have to expand OUs (choose the ▶) to find the OU or account that you want.
5. Choose **Attach policy**.

   The list of attached AI services opt-out policies on the **Targets** tab is updated to include the new addition. The policy change takes effect immediately.

AWS CLI & AWS SDKs

To attach an AI services opt-out policy to the organization root, OU, or account

You can use one of the following to attach an AI services opt-out policy:

- AWS CLI: attach-policy

  The following example attaches a policy to an OU.

  ```
  $ aws organizations attach-policy \
      --target-id ou-a1b2-f6g7h222 \
      --policy-id p-i9j8k7l6m5
  ```

  This command produces no output when successful.
- AWS SDKs: AttachPolicy

The policy change takes effect immediately.

Detaching an AI services opt-out policy

When you sign in to your organization's management account, you can detach an AI services opt-out policy from the organization root, OU, or account that it is attached to. After you detach an AI services opt-out policy from an entity, that policy no longer applies to any account that was previously affected by the now detached entity. To detach a policy, complete the following steps.

Minimum permissions

To detach an AI services opt-out policy from the organization root, OU, or account, you must have permission to run the following action:

- `organizations:DetachPolicy`

AWS Management Console

You can detach an AI services opt-out policy by either navigating to the policy or to the root, OU, or account that you want to detach the policy from.

To detach an AI services opt-out policy by navigating to the root, OU, or account it's attached to

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page navigate to the Root, OU, or account that you want to detach a policy from. You might have to expand OUs (choose the ▶) to find the OU or account that you want. Choose the name of the Root, OU, or account.
3. On the **Policies** tab, choose the radio button next to the AI services opt-out policy that you want to detach, and then choose **Detach**.
4. In the confirmation dialog box, choose **Detach policy**.

   The list of attached AI services opt-out policies is updated. The policy change takes effect immediately.

To detach an AI services opt-out policy by navigating to the policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AI services opt-out policies** page, choose the name of the policy that you want to detach from a root, OU, or account.

3. On the **Targets** tab, choose the radio button next to the root, OU, or account that you want to detach the policy from. You might have to expand OUs (choose the ▶) to find the OU or account that you want.
4. Choose **Detach**.
5. In the confirmation dialog box, choose **Detach**.

   The list of attached AI services opt-out policies is updated. The policy change takes effect immediately.

AWS CLI & AWS SDKs

To detach an AI services opt-out policy from the organization root, OU, or account

You can use one of the following to detach an AI services opt-out policy:

- AWS CLI: detach-policy

  The following example detaches a policy from an OU.

```
$ aws organizations detach-policy \
    --target-id ou-a1b2-f6g7h222 \
    --policy-id p-i9j8k7l6m5
```

  This command produces no output when successful.
- AWS SDKs: DetachPolicy

The policy change takes effect immediately.

Viewing effective AI services opt-out policies

Determine the effective Artificial Intelligence (AI) services opt-out policy for an account in your organization.

What is the effective AI services opt-out policy?

The *effective AI services opt-out policy* specifies the final rules that apply to an AWS account. It is the aggregation of any AI services opt-out policies that the account inherits, plus any AI services opt-out policies that are directly attached to the account. When you attach an AI services opt-out policy to the organization's root, it applies to all accounts in your organization. When you attach an AI services opt-out policy to an OU, it applies to all accounts and OUs that belong to the OU. When you attach a policy directly to an account, it applies only to that one AWS account.

For example, the AI services opt-out policy attached to the organization root might specify that all accounts in the organization opt out of content use by all AWS machine learning services. A separate AI services opt-out policy attached directly to one member account specifies that it opts in to content use for only Amazon Rekognition. The combination of these AI services opt-out policies comprises the effective AI services opt-out policy. The result is that all accounts in the organization are opted out of all AWS services, with the exception of one account that opts in to Amazon Rekognition.

For information about how policies are combined into the final effective policy, see Understanding policy inheritance (p. 91).

How to view the effective AI services opt-out policy

You can view the effective AI services opt-out policy for an account from the AWS Management Console, AWS API, or AWS Command Line Interface.

Minimum permissions
To view the effective AI services opt-out policy for an account, you must have permission to run the following actions:

- `organizations:DescribeEffectivePolicy`
- `organizations:DescribeOrganization` – required only when using the Organizations console

AWS Management Console

To view the effective AI services opt-out policy for an account

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page, choose the name of the account for which you want to view the effective AI services opt-out policy. You might have to expand OUs (choose the ▶) to find the account that you want.
3. On the **Policies** tab, in the **AI services opt-out policies** section, choose **View the effective AI policy for this AWS account**.

   The console displays the effective policy applied to the specified account.

   Note
   You can't copy and paste an effective policy and use it as the JSON for another AI services opt-out policy without significant changes. AI services opt-out policy documents must include the inheritance operators (p. 95) that specify how each setting is merged into the final effective policy.

AWS CLI & AWS SDKs

To view the effective AI services opt-out policy for an account

You can use one of the following to view the effective AI services opt-out policy:

- AWS CLI: describe-effective-policy

  The following example shows the effective AI services opt-out policy for an account.

```
$ aws organizations describe-effective-policy \
    --policy-type AISERVICES_OPT_OUT_POLICY \
    --target-id 123456789012
{
    "EffectivePolicy": {
        "PolicyContent": "{\"services\":{\"comprehend\":{\"opt_out_policy\":\"optOut
\"},   ....TRUNCATED FOR BREVITY....   "opt_out_policy\":\"optIn\"}}}",
        "LastUpdatedTimestamp": "2020-12-09T12:58:53.548000-08:00",
        "TargetId": "123456789012",
        "PolicyType": "AISERVICES_OPT_OUT_POLICY"
    }
}
```

- AWS SDKs: DescribeEffectivePolicy

AI services opt-out policy syntax and examples

This topic describes Artificial Intelligence (AI) services opt-out policy syntax and provides examples.

Syntax for AI services opt-out policies

An AI services opt-out policy is a plaintext file that is structured according to the rules of JSON. The syntax for AI services opt-out policies follows the syntax for management policy types. For a complete discussion of that syntax, see Policy syntax and inheritance for management policy types (p. 94). This topic focuses on applying that general syntax to the specific requirements of the AI services opt-out policy type.

> **Important**
> The capitalization of the values discussed in this section are important. Enter the values with upper and lower case letters as shown in this topic. The policies do not work if you use unexpected capitalization.

The following policy shows the basic AI services opt-out policy syntax. If this example was attached directly to an account, that account would be explicitly opted out of one service and opted in to another. Other services could be opted in or opted out by policies inherited from higher levels (OU or root policies).

```
{
    "services": {
        "rekognition": {
            "opt_out_policy": {
                "@@assign": "optOut"
            }
        },
        "lex": {
            "opt_out_policy": {
                "@@assign": "optIn"
            }
        }
    }
}
```

Imagine the following example policy attached to the organization's root. It sets the default for the organization to opt out of all AI services. This automatically includes any AI services not otherwise explicitly exempted, including any AI services that AWS might deploy in the future. You can attach child policies to OUs or directly to accounts to override this setting for any AI service except Amazon Comprehend. The second entry in the following example uses `@@operators_allowed_for_child_policies` set to `none` to prevent it from being overridden. The third entry in the example makes an organization-wide exemption for Amazon Rekognition. It opts in the entire organization for that service, but the policy does allow child policies to override where appropriate.

```
{
    "services": {
        "default": {
            "opt_out_policy": {
                "@@assign": "optOut"
            }
        },
        "comprehend": {
            "opt_out_policy": {
                "@@operators_allowed_for_child_policies": ["@@none"],
                "@@assign": "optOut"
            }
        },
        "rekognition": {
            "opt_out_policy": {
                "@@assign": "optIn"
            }
        }
    }
}
```

AI services opt-out policy syntax includes the following elements:

- The `services` element. An AI services opt-out policy is identified by this fixed name as the outermost JSON containing element.

  An AI services opt-out policy can have one or more statements under the `services` element. Each statement contains the following elements:

  - A *service name key* that identifies an AWS AI service. The following key names are valid values for this field:

    - **`default`** – represents **all** currently available AI services and implicitly and automatically includes any AI services that might be added in the future.
    - `codeguruprofiler`
    - `comprehend`
    - `connectamd`
    - `connectoptimization`
    - `contactlens`
    - `frauddetector`
    - `guardduty`
    - `lex`
    - `polly`
    - `rekognition`
    - `textract`
    - `transcribe`
    - `translate`

    Each policy statement identified by a service name key can contain the following elements:

    - The `opt_out_policy` key. This key must be present. This is the only key you can place under a service name key.

      The `opt_out_policy` key can contain ***only*** the `@@assign` operator with one of the following values:

      - `optOut` – you choose to opt out of content use for the specified AI service.
      - `optIn` – you choose to opt in to content use for the specified AI service.

        Notes

        - You can't use the `@@append` and `@@remove` inheritance operators in AI services opt-out policies.
        - You can't use the `@@enforced_for` operator in AI services opt-out policies.

  - At any level, you can specify the `@@operators_allowed_for_child_policies` operator to control what child policies can do to override settings imposed by parent policies. You can specify one of the following values:

    - `@@assign` – child policies of this policy can use the `@@assign` operator to override the inherited value with a different value.
    - `@@none` – child policies of this policy can't change the value.

The behavior of the `@@operators_allowed_for_child_policies` depends on where you place it. You can use the following locations:

- Under the `services` key – controls whether a child policy can add to or change the list of services in the effective policy.
- Under the key for a specific AI service or the `default` key - controls whether a child policy can add to or change the list of keys under this specific entry.
- Under the `opt_out_policies` key for a specific service – controls whether a child policy can change only the setting for this specific service.

AI services opt-out policy examples

The example policies that follow are for information purposes only.

Example 1: Opt out of all AI services for all accounts in the organization

The following example shows a policy that you could attach to your organization's root to opt out of AI services for accounts in your organization.

Tip
If you copy the following example using the copy button in the example's upper-right corner, the copy doesn't include the line numbers. It's ready to paste.

```
    | {
    |     "services": {
[1] |         "@@operators_allowed_for_child_policies": ["@@none"],
    |         "default": {
[2] |             "@@operators_allowed_for_child_policies": ["@@none"],
    |             "opt_out_policy": {
[3] |                 "@@operators_allowed_for_child_policies": ["@@none"],
    |                 "@@assign": "optOut"
    |             }
    |         }
    |     }
    | }
```

- [1] – The `"@@operators_allowed_for_child_policies": ["@@none"]` that is under `services` prevents any child policy from adding any new sections for individual services other than the `default` section that is already there. `Default` is the placeholder that represents "all AI services".
- [2] – The `"@@operators_allowed_for_child_policies": ["@@none"]` that is under `default` prevents any child policy from adding any new sections other than the `opt_out_policy` section that is already there.
- [3] – The `"@@operators_allowed_for_child_policies": ["@@none"]` that is under `opt_out_policy` prevents child policies from changing the value of the `optOut` setting or adding any additional settings.

Example 2: Set an organization default setting for all services, but allow child policies to override the setting for individual services

The following example policy sets an organization-wide default for all AI services. The value for `default` prevents a child policy from change the `optOut` value for service `default`, the placeholder for all AI services. If this policy is applied as a parent policy by attaching it to the root or to an OU, child policies can still change the opt-out setting for individual services, as shown in the second policy.

- Because there is no `"@@operators_allowed_for_child_policies": ["@@none"]` under the `services` key, child policies can add new sections for individual services.

- The `"@@operators_allowed_for_child_policies": ["@@none"]` that is under `default` prevents any child policy from adding any new sections other than the `opt_out_policy` section that is already there.
- The `"@@operators_allowed_for_child_policies": ["@@none"]` that is under `opt_out_policy` prevents child policies from changing the value of the `optOut` setting or adding any additional settings.

Organization root AI services opt-out parent policy

```
{
    "services": {
        "default": {
            "@@operators_allowed_for_child_policies": ["@@none"],
            "opt_out_policy": {
                "@@operators_allowed_for_child_policies": ["@@none"],
                "@@assign": "optOut"
            }
        }
    }
}
```

The following example policy assumes that the previous example policy is attached to either the organization root or to a parent OU, and that you attach this example to an account affected by the parent policy. It overrides the default opt-out setting and explicitly opts in to only the Amazon Lex service.

AI services opt-out child policy

```
{
    "services": {
        "lex": {
            "opt_out_policy": {
                "@@assign": "optIn"
            }
        }
    }
}
```

The resulting effective policy for the AWS account is that the account opts in to only Amazon Lex, and opts out of all other AWS AI services because of the inherited `default` opt-out setting from the parent policy.

Example 3: Define an organization-wide AI services opt-out policy for a single service

The following example shows an AI services opt-out policy that defines an `optOut` setting for a single AI service. If this policy is attached to the organization's root, it prevents any child policy from overriding the `optOut` setting for this one service. Other services are not addressed by this policy, but could be affected by child policies in other OUs or accounts.

```
{
    "services": {
        "rekognition": {
            "opt_out_policy": {
                "@@assign": "optOut",
                "@@operators_allowed_for_child_policies": ["@@none"]
            }
        }
    }
}
```

Backup policies

For information and procedures common to all policy types, see the following topics:

- Enable and disable policy types (p. 85)
- Get details about your policies (p. 87)
- Policy syntax and inheritance (p. 92)

AWS Backup enables you to create backup plans that define how to back up your AWS resources. The rules in the plan include a variety of settings, such as the backup frequency, the time window during which the backup occurs, the AWS Region containing the resources to back up and the vault in which to store the backup. You can then apply a backup plan to groups of AWS resources identified by using tags. You must also identify an AWS Identity and Access Management (IAM) role that grants AWS Backup permission to perform the backup operation on your behalf.

Backup policies in AWS Organizations combine all of those pieces into JSON text documents. You can attach a backup policy to any of the elements in your organization's structure, such as the root, organizational units (OUs), and individual accounts. Organizations applies inheritance rules to combine the policies in the organization's root, any parent OUs, or attached to the account. This results in an effective backup policy (p. 167) for each account. This effective policy instructs AWS Backup how to automatically back up your AWS resources.

Backup policies give you granular control over backing up your resources at whatever level your organization requires. For example, you can specify in a policy attached to the organization's root that all Amazon DynamoDB tables must be backed up. That policy can include a default backup frequency. You can then attach a backup policy to OUs that override the backup frequency according to the requirements of each OU. For example, the `Developers` OU might specify a backup frequency of once per week, while the `Production` OU specifies once per day.

You can create partial backup policies that individually include only part of the required information to successfully back up your resources. You can attach these policies to different parts of the organization tree, such as the root or a parent OU, with the intention of those partial policies being inherited by lower-level OUs and accounts. When Organizations combines all of the policies for an account by using inheritance rules, the resulting effective policy must have all the required elements. Otherwise, AWS Backup considers the policy not valid and does not back up the affected resources.

> **Important**
> AWS Backup can only perform a successful backup when it is invoked by a *complete* effective policy that has all of the required elements.
> Although a partial policy strategy as described earlier can work, if an effective policy for an account is incomplete, it results in errors or resources that are not successfully backed up.
> As an alternate strategy, consider requiring that all backup policies be complete and valid by themselves. Use *default* values supplied by policies attached higher in the hierarchy, and override them where needed in child policies by including inheritance child control operators (p. 95).

The effective backup plan for each AWS account in the organization appears in the AWS Backup console as an immutable plan for that account. You can view it, but not change it.

When AWS Backup begins a backup based on a policy-created backup plan, you can see the status of the backup job in the AWS Backup console. A user in a member account can see the status and any errors for the backup jobs in that member account. If you also enable trusted service access with AWS Backup, a user in the organization's management account can see the status and errors for all backup jobs in the organization. For more information, see Enabling cross-account management in the *AWS Backup Developer Guide*.

Getting started with backup policies

Follow these steps to get started using backup policies.

1. Learn about the permissions you must have to perform backup policy tasks (p. 155).
2. Learn about some best practices we recommend when using backup policies (p. 156).
3. Enable backup policies for your organization (p. 85).
4. Create a backup policy (p. 158).
5. Attach the backup policy to your organization's root, OU, or account (p. 165).
6. View the combined effective backup policy that applies to an account (p. 167).

For all of these steps, you sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.

Other information

- Learn backup policy syntax and see example policies (p. 169)

Prerequisites and permissions for managing backup policies

This page describes the prerequisites and required permissions to manage backup policies in AWS Organizations.

Topics

Prerequisites for managing backup policies

To manage backup policies in an organization requires the following:

- Your organization must have all features enabled (p. 35).
- You must be signed in to your organization's management account.
- Your AWS Identity and Access Management (IAM) user or role must have the permissions that are listed in the following section.

Permissions for managing backup policies

The following example IAM policy provides permissions to manage all aspects of backup policies in an organization.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ManageBackupPolicies",
            "Effect": "Allow",
            "Action": [
                "organizations:AttachPolicy",
                "organizations:CreatePolicy",
                "organizations:DeletePolicy",
                "organizations:DescribeAccount",
                "organizations:DescribeCreateAccountStatus",
                "organizations:DescribeEffectivePolicy",
                "organizations:DescribeOrganization",
                "organizations:DescribeOrganizationalUnit",
                "organizations:DescribePolicy",
                "organizations:DetachPolicy",
                "organizations:DisableAWSServiceAccess",
                "organizations:DisablePolicyType",
                "organizations:EnableAWSServiceAccess",
                "organizations:EnablePolicyType",
                "organizations:ListAccounts",
                "organizations:ListAccountsForParent",
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:ListCreateAccountStatus",
                "organizations:ListOrganizationalUnitsForParent",
                "organizations:ListParents",
                "organizations:ListPolicies",
                "organizations:ListPoliciesForTarget",
                "organizations:ListRoots",
                "organizations:ListTargetsForPolicy",
                "organizations:UpdatePolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

For more information about IAM policies and permissions, see the IAM User Guide.

Best practices for using backup policies

AWS recommends the following best practices for using backup policies.

Decide on a backup policy strategy

You can create backup policies in incomplete pieces that are inherited and merged to make a complete policy for each member account. If you do this, you risk ending up with an effective policy that is not complete if you make a change at one level without carefully considering the change's impact on all accounts below that level. To prevent this, we recommend that you instead ensure that the backup policies you implement at all levels are complete by themselves. Treat the parent policies as default policies that can be overridden by settings specified in child policies. That way, even if a child policy doesn't exist, the inherited policy is complete and uses the default values. You can control which settings can be added to, changed, or removed by child policies by using the child control inheritance operators (p. 96).

Validate changes to your backup policies checking using `GetEffectivePolicy`

After you make a change to a backup policy, check the effective policies for representative accounts below the level where you made the change. You can view the effective policy by using the AWS Management Console (p. 167), or by using the GetEffectivePolicy API operation or one of its AWS CLI or AWS SDK variants. Ensure that the change you made had the intended impact on the effective policy.

Start simply and make small changes

To simplify debugging, start with simple policies and make changes one item at a time. Validate the behavior and impact of each change before making the next change. This approach reduces the number of variables you have to account for when an error or unexpected result does happen.

Store copies of your backups in other AWS Regions and accounts in your organization

To improve your disaster recovery position, you can store copies of your backups.

- **A different region** – If you store copies of the backup in additional AWS Regions, you help protect the backup against accidental corruption or deletion in the original Region. Use the `copy_actions` section of the policy to specify a vault in one or more Regions of the same account in which the backup plan runs. To do this, identify the account by using the `$account` variable when you specify the ARN of the backup vault in which to store the copy of the backup. The `$account` variable is automatically replaced at run time with the account ID in which the backup policy is running.
- **A different account** – If you store copies of the backup in additional AWS accounts, you add a security barrier that helps protect against a malicious actor who compromises one of your accounts. Use the `copy_actions` section of the policy to specify a vault in one or more accounts in your organization, separate from the account in which the backup plan runs . To do this, identify the account by using its actual account ID number when you specify the ARN of the backup vault in which to store the copy of the backup.

Limit the number of plans per policy

Policies that contain multiple plans are more complicated to troubleshoot because of the larger number of outputs that must all be validated. Instead, have each policy contain one and only one backup plan to simplify debugging and troubleshooting. You can then add additional policies with other plans to meet other requirements. This approach helps keep any issues with a plan isolated to one policy, and it prevents those issues from complicating the troubleshooting of issues with other policies and their plans.

Use stack sets to create the required backup vaults and IAM roles

Use AWS CloudFormation stack sets integration with Organizations to automatically create the required backup vaults and AWS Identity and Access Management (IAM) roles in each of the member accounts in your organization. You can create a stack set that includes the resources you want automatically available in every AWS account in your organization. This approach enables you to run your backup plans with assurance that the dependencies are already met. For more information, see Create a Stack Set with Self-Managed Permissions in the *AWS CloudFormation User Guide*.

Check your results by reviewing the first backup created in each account

When you make a change to a policy, check the next backup created after that change to ensure the change had the desired impact. This step goes beyond looking at the effective policy and ensures that AWS Backup interprets your policies and implements the backup plans the way you intended.

Creating, updating, and deleting backup policies

In this topic:

- After you enable backup policies (p. 85) for your organization, you can create a policy (p. 158).
- When your backup requirements change, you can update an existing policy (p. 161).
- When you no longer need a policy and after you detach it from all organizational units (OUs) and accounts, you can delete it (p. 164).

Creating a backup policy

Minimum permissions
To create a backup policy, you need permission to run the following action:

- `organizations:CreatePolicy`

AWS Management Console

You can create a backup policy in the AWS Management Console in one of two ways:

- A visual editor that lets you choose options and generates the JSON policy text for you.
- A text editor that lets you directly create the JSON policy text yourself.

The visual editor makes the process easy, but it limits your flexibility. It's a great way to create your first policies and get comfortable with using them. After you understand how they work and have started to be limited by what the visual editor provides, you can add advanced features to your policies by editing the JSON policy text yourself. The visual editor uses only the @@assign value-setting operator (p. 95), and it doesn't provide any access to the child control operators (p. 96). You can add the child control operators only if you manually edit the JSON policy text.

To create a backup policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **Backup policies** page, choose **Create policy**.
3. On the **Create policy** page, enter a **Policy name** and an optional **Policy description**.
4. (Optional) You can add one or more tags to the policy by choosing **Add tag** and then entering a key and an optional value. Leaving the value blank sets it to an empty string; it isn't `null`. You can attach up to 50 tags to a policy. For more information about tagging, see Tagging AWS Organizations resources (p. 220).
5. You can build the policy using the **Visual editor** as described in this procedure. You can also enter or paste policy text in the **JSON** tab. For information about backup policy syntax, see Backup policy syntax and examples (p. 169).

   If you choose to use the **Visual editor**, select the backup options appropriate for your scenario. A backup plan consists of three parts. For more information about these backup plan elements, see Creating a backup plan and Assigning resources in the *AWS Backup Developer Guide*.

   a. Backup plan general details
      - The **Backup plan name** can consist of only alphanumeric, hyphen, and underline characters.
      - You must select at least one **Backup plan region** from the list. The plan can back up resources in only the selected AWS Regions.

b. One or more backup rules that specify how and when AWS Backup is to operate. Each backup rule defines the following items:

   - A schedule that includes the frequency of the backup and the time window in which the backup can occur.
   - The name of the backup vault to use. The **Backup vault name** can consist of only alphanumeric, hyphen, and underline characters. The backup vault must exist before the plan can successfully run. Create the vault using the AWS Backup console or AWS CLI commands.
   - (Optional) One or more **Copy to region** rules to also copy the backup to vaults in other AWS Regions.
   - One or more tag key and value pairs to attach to the backup recovery points created each time this backup plan runs.
   - Lifecycle options that specify when the backup transitions to cold storage, and when the backup expires.

   Choose **Add rule** to add each rule you need to the plan.

   For more information about backup rules, see Backup Rules in the *AWS Backup Developer Guide*.

c. A resource assignment that specifies which resources that AWS Backup should backup with this plan. The assignment is made by specifying tag pairs that AWS Backup uses to find and match resources

   - The **Resource assignment name** can consist of only alphanumeric, hyphen, and underline characters.
   - Specify the **IAM role** for AWS Backup to use to perform the backup by its name.

     In the console, you don't specify the entire Amazon Resource Name (ARN). You must include both the role name and its prefix that specifies the type of role. The prefixes are typically `role` or `service-role` , and they are separated from the role name by a forward slash ('/'). For example, you might enter `role/MyRoleName` or `service-role/MyManagedRoleName`. This is converted to a full ARN for you when stored in the underlying JSON.

     > **Important**
     > The specified IAM role must already exist in the account the policy is applied to. If it does not, the backup plan might successfully start backup jobs, but those backup jobs will fail.

   - Specify one or more **Resource tag key** and **Tag values** pairs to identify resources that you want backed up. If there is more than one tag value, separate the values with commas.

   Choose **Add assignment** to add each configured resource assignment to the backup plan.

   For more information, see Assign Resources to a Backup Plan in the *AWS Backup Developer Guide*.

6. When you're finished creating your policy, choose **Create policy**. The policy appears in your list of available backup policies.

AWS CLI & AWS SDKs

To create a backup policy

You can use one of the following to create a backup policy:

- AWS CLI: create-policy

Create a backup plan as JSON text similar to the following, and store it in a text file. For complete rules for the syntax, see Backup policy syntax and examples (p. 169).

```
{
    "plans": {
        "PII_Backup_Plan": {
            "regions": { "@@assign": [ "ap-northeast-2", "us-east-1", "eu-
north-1" ] },
            "rules": {
                "Hourly": {
                    "schedule_expression": { "@@assign": "cron(0 5/1 ? * * *)" },
                    "start_backup_window_minutes": { "@@assign": "480" },
                    "complete_backup_window_minutes": { "@@assign": "10080" },
                    "lifecycle": {
                        "move_to_cold_storage_after_days": { "@@assign": "180" },
                        "delete_after_days": { "@@assign": "270" }
                    },
                    "target_backup_vault_name": { "@@assign": "FortKnox" },
                    "copy_actions": {
                        "arn:aws:backup:us-east-1:$account:backup-vault:secondary-
vault": {
                            "lifecycle": {
                                "move_to_cold_storage_after_days": { "@@assign":
 "10" },
                                "delete_after_days": { "@@assign": "100" }
                            }
                        }
                    }
                }
            },
            "selections": {
                "tags": {
                    "datatype": {
                        "iam_role_arn": { "@@assign": "arn:aws:iam::$account:role/
MyIamRole" },
                        "tag_key": { "@@assign": "dataType" },
                        "tag_value": { "@@assign": [ "PII" ] }
                    }
                }
            }
        }
    }
}
```

This backup plan specifies that AWS Backup should back up all resources in the affected AWS accounts that are in the specified AWS Regions and that have the tag `dataType` with a value of `PII`.

Next, import the JSON policy file backup plan to create a new backup policy in the organization. Note the policy ID at the end of the policy ARN in the output.

```
$ aws organizations create-policy \
    --name "MyBackupPolicy" \
    --type BACKUP_POLICY \
    --description "My backup policy" \
    --content file://policy.json{
    "Policy": {
        "PolicySummary": {
            "Arn": "arn:aws:organizations::o-aa111bb222:policy/backup_policy/p-
i9j8k7l6m5",
            "Description": "My backup policy",
            "Name": "MyBackupPolicy",
            "Type": "BACKUP_POLICY"
        }
        "Content": "...a condensed version of the JSON policy document you provided
 in the file...",
    }
}
```

- AWS SDKs: CreatePolicy

What to do next

After you create a backup policy, you can put your policy into effect. To do that, you can attach the policy (p. 199) to the organization root, organizational units (OUs), AWS accounts within your organization, or a combination of all of those.

Updating a backup policy

When you sign in to your organization's management account, you can edit a policy that requires changes in your organization.

Minimum permissions
To update a backup policy, you must have permission to run the following actions:

- `organizations:UpdatePolicy` with a `Resource` element in the same policy statement that includes the ARN of the policy to update (or "*")
- `organizations:DescribePolicy` with a `Resource` element in the same policy statement that includes the ARN of the policy to update (or "*")

AWS Management Console

To update a backup policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **Backup policies** page, choose the name of the policy that you want to update.
3. Choose **Edit policy**.
4. You can enter a new **Policy name**, **Policy description**. You can change the policy content by using either the **Visual editor** or by directly editing the **JSON**.
5. When you're finished updating the policy, choose **Save changes**.

AWS CLI & AWS SDKs

To update a backup policy

You can use one of the following to update a backup policy:

- AWS CLI: update-policy

  The following example renames a backup policy.

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --name "Renamed policy"
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
backup_policy/p-i9j8k7l6m5",
            "Name": "Renamed policy",
            "Type": "BACKUP_POLICY",
            "AwsManaged": false
        },
         "Content": "{\"plans\":{\"TestBackupPlan\":{\"regions\":{\"@@assign\":
  ....TRUNCATED FOR BREVITY....   "@@assign\":[\"Yes\"]}}}}}}}"
    }
}
```

The following example adds or changes the description for a backup policy.

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --description "My new description"
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
backup_policy/p-i9j8k7l6m5",
            "Name": "Renamed policy",
            "Description": "My new description",
            "Type": "BACKUP_POLICY",
            "AwsManaged": false
        },
        "Content": "{\"plans\":{\"TestBackupPlan\":{\"regions\":{\"@@assign\":
  ....TRUNCATED FOR BREVITY....   "@@assign\":[\"Yes\"]}}}}}}}"
    }
}
```

The following example changes the JSON policy document attached to a backup policy. In this example, the content is taken from a file called `policy.json` with the following text:

```
{
    "plans": {
        "PII_Backup_Plan": {
            "regions": { "@@assign": [ "ap-northeast-2", "us-east-1", "eu-
north-1" ] },
            "rules": {
                "Hourly": {
                    "schedule_expression": { "@@assign": "cron(0 5/1 ? * * *)" },
                    "start_backup_window_minutes": { "@@assign": "480" },
                    "complete_backup_window_minutes": { "@@assign": "10080" },
                    "lifecycle": {
                        "move_to_cold_storage_after_days": { "@@assign": "180" },
                        "delete_after_days": { "@@assign": "270" }
                    },
                    "target_backup_vault_name": { "@@assign": "FortKnox" },
                    "copy_actions": {
                        "arn:aws:backup:us-east-1:$account:backup-vault:secondary-
vault": {
                            "lifecycle": {
                                "move_to_cold_storage_after_days": { "@@assign":
 "10" },
                                "delete_after_days": { "@@assign": "100" }
                            }
                        }
                    }
                }
            },
            "selections": {
                "tags": {
                    "datatype": {
                        "iam_role_arn": { "@@assign": "arn:aws:iam::$account:role/
MyIamRole" },
                        "tag_key": { "@@assign": "dataType" },
                        "tag_value": { "@@assign": [ "PII" ] }
                    }
                }
            }
        }
    }
}
```

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --content file://policy.json
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
backup_policy/p-i9j8k7l6m5",
            "Name": "Renamed policy",
            "Description": "My new description",
            "Type": "BACKUP_POLICY",
            "AwsManaged": false
        },
         "Content": "{\"plans\":{\"TestBackupPlan\":{\"regions\":{\"@@assign\":
  ....TRUNCATED FOR BREVITY....   "@@assign\":[\"Yes\"]}}}}}}}"
}
```

- AWS SDKs: UpdatePolicy

Editing tags attached to a backup policy

When you sign in to your organization's management account, you can add or remove the tags attached to a backup policy. For more information about tagging, see Tagging AWS Organizations resources (p. 220).

Minimum permissions
To edit the tags attached to a backup policy in your AWS organization, you must have the following permissions:

- `organizations:DescribeOrganization` (console only – to navigate to the policy)
- `organizations:DescribePolicy` (console only – to navigate to the policy)
- `organizations:TagResource`
- `organizations:UntagResource`

AWS Management Console

To edit the tags attached to an backup policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. **Backup policies** page
3. Choose the name of the policy with the tags that you want to edit.

   The policy detail page appears.

4. On the **Tags** tab, choose **Manage tags**.
5. You can perform any of these actions on this page:
    - Edit the value for any tag by entering a new value over the old one. You can't modify the key. To change a key, you must delete the tag with the old key and add a tag with the new key.
    - Remove an existing tag by choosing **Remove**.
    - Add a new tag key and value pair. Choose **Add tag**, then enter the new key name and optional value in the provided boxes. If you leave the **Value** box empty, the value is an empty string; it isn't `null`.
6. Choose **Save changes** after you've made all the additions, removals, and edits you want to make.

AWS CLI & AWS SDKs

To edit the tags attached to a backup policy

You can use one of the following commands to edit the tags attached to a backup policy:

- AWS CLI: tag-resource and untag-resource
- AWS SDKs: TagResource and UntagResource

Deleting a backup policy

When you sign in to your organization's management account, you can delete a policy that you no longer need in your organization.

Before you can delete a policy, you must first detach it from all attached entities.

Minimum permissions

To delete a policy, you must have permission to run the following action:

- `organizations:DeletePolicy` with a `Resource` element in the same policy statement that includes the ARN of the policy to delete (or "*")

AWS Management Console

To delete a backup policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **Backup policies** page, choose the name of the backup policy that you want to delete.
3. You must first detach the backup policy that you want to delete from all roots, OUs, and accounts. Choose the **Targets** tab, choose the radio button next to each root, OU, or account that is shown in the **Targets** list, and then choose **Detach**. In the confirmation dialog box, choose **Detach**. Repeat until you remove all targets.
4. Choose **Delete** at the top of the page.
5. On the confirmation dialog box, enter the name of the policy, and then choose **Delete**.

AWS CLI & AWS SDKs

To delete a backup policy

You can use one of the following to delete a policy:

- AWS CLI: delete-policy

The following example deletes the specified policy. It works only if the policy is not attached to any root, OU, or account.

```
$ aws organizations delete-policy \
    --policy-id p-i9j8k7l6m5
```

This command produces no output when successful.

- AWS SDKs: DeletePolicy

Attaching and detaching backup policies

You can use backup policies on an entire organization as well as on organizational units (OUs) and individual accounts. Keep the following points in mind:

- When you attach a backup policy to your *organization root*, the policy applies to all of that root's member OUs and accounts.
- When you attach a backup policy to an *OU*, that policy applies to the accounts that belong to the OU or any of its child OUs. Those accounts are also subject to any policy attached to the organization root.
- When you attach a backup policy to an *account*, that policy applies to only that account. The account is also subject to any policy attached to the organization root and any OUs that the account belongs to.

The aggregation of any backup policies the account inherits from the root and parent OUs, as well as any policies directly attached to the account, is the *effective policy* (p. 167). For information about how policies are merged to the effective policy, see Policy syntax and inheritance for management policy types (p. 94).

Attaching a backup policy

When you sign in to your organization's management account, you can attach a backup policy to the organization's root, OU, or directly to an account.

Minimum permissions
To attach backup policies, you must have permission to run the following action:

- `organizations:AttachPolicy`

AWS Management Console

You can attach a backup policy by either navigating to the policy or to the root, OU, or account that you want to attach the policy to.

To attach a backup policy by navigating to the root, OU, or account

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page, navigate to and then choose the name of the root, OU, or account that you want to attach a policy to. You might have to expand OUs (choose the ▶) to find the OU or account that you want.
3. In the **Policies** tab, in the entry for **Backup policies**, choose **Attach**.
4. Find the policy that you want and choose **Attach policy**.

   The list of attached backup policies on the **Policies** tab is updated to include the new addition. The policy change takes effect immediately.

To attach a backup policy by navigating to the policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **Backup policies** page, choose the name of the policy that you want to attach.
3. On the **Targets** tab, choose **Attach**.
4. Choose the radio button next to the root, OU, or account that you want to attach the policy to. You might have to expand OUs (choose the ▶) to find the OU or account that you want.
5. Choose **Attach policy**.

   The list of attached backup policies on the **Targets** tab is updated to include the new addition. The policy change takes effect immediately.

AWS CLI & AWS SDKs

To attach a backup policy to the organization root, OU, or account

You can use one of the following commands to attach a backup policy:

- AWS CLI: attach-policy

```
$ aws organizations attach-policy \
    --target-id 123456789012 \
    --policy-id p-i9j8k7l6m5
```

- AWS SDKs: AttachPolicy

The policy change takes effect immediately.

Detaching a backup policy

When you sign in to your organization's management account, you can detach a backup policy from the organization's root, OU, or account that it is attached to. After you detach a backup policy from an entity, that policy no longer applies to any account that was previously affected by the now detached entity. To detach a policy, complete the following steps.

Minimum permissions

To detach a backup policy from the organization root, OU, or account, you must have permission to run the following action:

- `organizations:DetachPolicy`

AWS Management Console

You can detach a backup policy by either navigating to the policy or to the root, OU, or account that you want to detach the policy from.

To detach a backup policy by navigating to the root, OU, or account it's attached to

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **AWS accounts** page navigate to the Root, OU, or account that you want to detach a policy from. You might have to expand OUs (choose the ▶) to find the OU or account that you want. Choose the name of the Root, OU, or account.

3. On the **Policies** tab, choose the radio button next to the backup policy that you want to detach, and then choose **Detach**.
4. In the confirmation dialog box, choose **Detach policy**.

   The list of attached backup policies is updated. The policy change takes effect immediately.

To detach a backup policy by navigating to the policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **Backup policies** page, choose the name of the policy that you want to detach from a root, OU, or account.
3. On the **Targets** tab, choose the radio button next to the root, OU, or account that you want to detach the policy from. You might have to expand OUs (choose the ▶) to find the OU or account that you want.
4. Choose **Detach**.
5. In the confirmation dialog box, choose **Detach**.

   The list of attached backup policies is updated. The policy change takes effect immediately.

AWS CLI & AWS SDKs

To detach a backup policy from the organization root, OU, or account

You can use one of the following commands to detach a backup policy:

- AWS CLI: detach-policy

  The following example detaches a policy from an OU.

```
$ aws organizations detach-policy \
    --target-id ou-a1b2-f6g7h222 \
    --policy-id p-i9j8k7l6m5
```

  This command produces no output when successful.
- AWS SDKs: DetachPolicy

The policy change takes effect immediately.

Viewing effective backup policies

You can view the effective backup policy for an account from the AWS Management Console, AWS API, or AWS Command Line Interface. The following section provides a brief overview of the effective backup policy, including an example.

What is the effective backup policy?

The *effective backup policy* specifies the final backup plan settings that apply to an AWS account. It is the aggregation of any backup policies that the account inherits, plus any backup policy that is directly attached to the account. When you attach a backup policy to the organization's root, it applies to all accounts in your organization. When you attach an backup policy to an organizational unit (OU), it applies to all accounts and OUs that belong to the OU. When you attach a policy directly to an account, it applies only to that one AWS account.

For example, the backup policy attached to the organization root might specify that all accounts in the organization back up all Amazon DynamoDB tables with a default backup frequency of once per week. A separate backup policy attached directly to one member account with critical information in a table can override the frequency with a value of once per day. The combination of these backup policies comprises the effective backup policy. This effective backup policy is determined for each account in the organization individually. In this example, the result is that all accounts in the organization back up their DynamoDB tables once per week, with the exception of one account that backs up its tables daily.

For information about how backup policies are combined into the final effective backup policy, see Policy syntax and inheritance for management policy types (p. 94).

Viewing the effective backup policy

You can view the effective backup policy for an account by using the AWS Management Console, AWS API, or AWS Command Line Interface.

Minimum permissions
To view the effective backup policy for an account, you must have permission to run the following actions:

- `organizations:DescribeEffectivePolicy`
- `organizations:DescribeOrganization` – required only when using the Organizations console

AWS Management Console

To view the effective backup policy for an account

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page, choose the name of the account for which you want to view the effective backup policy. You might have to expand OUs (choose the ▶) to find the account that you want.
3. On the **Policies** tab, in the **Backup policies** section, choose **View the effective backup policy for this AWS account**.

   The console displays the effective policy applied to the specified account.

   Note
   You can't copy and paste an effective policy and use it as the JSON for another backup policy without significant changes. backup policy documents must include the inheritance operators (p. 95) that specify how each setting is merged into the final effective policy.

AWS CLI & AWS SDKs

To view the effective backup policy for an account

You can use one of the following commands to view the effective backup policy:

- AWS CLI: describe-effective-policy

  The following example displays the details of a backup policy.

```
$ aws organizations describe-effective-policy \
--policy-type BACKUP_POLICY \
--target-id 123456789012{
    "EffectivePolicy": {
        "LastUpdatedTimestamp": "2020-06-22T14:31:50.748000-07:00",
        "TargetId": "123456789012",
        "PolicyType": "BACKUP_POLICY",
        "PolicyContent": "{\"plans\":{\"pii_backup_plan\":{\"regions\":[\"ap-
northeast-2\",\"us-east-1\",\"eu-north-1\"],\
"selections\":{\"tags\":{\"datatype\":{\"iam_role_arn\":\"arn:aws:iam::$account:role/
MyIamRole\",\"tag_value\":[\"PII\"],\
"tag_key\":\"dataType\"}}},\"rules\":{\"hourly\":{\"complete_backup_window_minutes\":
\"10080\",\"target_backup_vault_name\
":\"FortKnox\",\"start_backup_window_minutes\":\"480\",\"schedule_expression\":
\"cron(0 5/1 ? * * *)\",\"lifecycle\":{\"mo
ve_to_cold_storage_after_days\":\"180\",\"delete_after_days\":\"270\"},\"copy_actions
\":{\"arn:aws:backup:us-east-1:$accou
nt:backup-vault:secondary-vault\":{\"lifecycle\":{\"move_to_cold_storage_after_days
\":\"10\",\"delete_after_days\":\"100\"
}}}}}}}}"
    }
}
```

- AWS SDKs: DescribeEffectivePolicy

Backup policy syntax and examples

This page describes backup policy syntax and provides examples.

Syntax for backup policies

A backup policy is a plaintext file that is structured according to the rules of JSON. The syntax for backup policies follows the syntax for all management policy types. For a complete discussion of that syntax, see Policy syntax and inheritance for management policy types. This topic focuses on applying that general syntax to the specific requirements of the backup policy type.

The bulk of a backup policy is the backup plan and its rules. The syntax for the backup plan within an AWS Organizations backup policy is structurally identical to the syntax used by AWS Backup, but the key names are different. In the descriptions of the policy key names below, each includes the equivalent AWS Backup plan key name. For more information about AWS Backup plans, see CreateBackupPlan in the *AWS Backup Developer Guide*.

To be complete and functional, an effective backup policy (p. 167) must include more than just a backup plan with its schedule and rules. The policy must also identify the AWS Regions and the resources to be backed up, and the AWS Identity and Access Management (IAM) role that AWS Backup can use to perform the backup.

The following functionally complete policy shows the basic backup policy syntax. If this example was attached directly to an account, AWS Backup would back up all resources for that account in the `us-east-1` and `eu-north-1` Regions that have the tag `dataType` with a value of either `PII` or `RED` . It backs up those resources daily at 5:00 AM to `My_Backup_Vault` and also stores a copy in `My_Secondary_Vault`. Both of those vaults are in the same account as the resource. It also stores a copy of the backup in the `My_Tertiary_Vault` in a different, explicitly specified account. The vaults must already exist in each of the specified AWS Regions for each AWS account that receives the effective policy. If any of the backed up resources are EC2 instances, then support for Microsoft Volume Shadow Copy Service (VSS) is enabled for the backups on those instances. The backup applies the tag `Owner:Backup` to each recovery point.

```
{
    "plans": {
        "PII_Backup_Plan": {
            "rules": {
                "My_Hourly_Rule": {
                    "schedule_expression": {"@@assign": "cron(0 5 ? * * *)"},
                    "start_backup_window_minutes": {"@@assign": "60"},
                    "complete_backup_window_minutes": {"@@assign": "604800"},
                    "enable_continuous_backup": {"@@assign": false},
                    "target_backup_vault_name": {"@@assign": "My_Backup_Vault"},
                    "recovery_point_tags": {
                        "Owner": {
                            "tag_key": {"@@assign": "Owner"},
                            "tag_value": {"@@assign": "Backup"}
                        }
                    },
                    "lifecycle": {
                        "move_to_cold_storage_after_days": {"@@assign": "180"},
                        "delete_after_days": {"@@assign": "270"}
                    },
                    "copy_actions": {
                        "arn:aws:backup:us-west-2:$account:backup-
vault:My_Secondary_Vault": {
                            "target_backup_vault_arn": {
                                "@@assign": "arn:aws:backup:us-west-2:$account:backup-
vault:My_Secondary_Vault"
                            },
                            "lifecycle": {
                                "move_to_cold_storage_after_days": {"@@assign": "180"},
                                "delete_after_days": {"@@assign": "270"}
                            }
                        },
                        "arn:aws:backup:us-east-1:$account:backup-vault:My_Tertiary_Vault":
 {
                            "target_backup_vault_arn": {
                                "@@assign": "arn:aws:backup:us-east-1:111111111111:backup-
vault:My_Tertiary_Vault"
                            },
                            "lifecycle": {
                                "move_to_cold_storage_after_days": {"@@assign": "180"},
                                "delete_after_days": {"@@assign": "270"}
                            }
                        }
                    }
                }
            },
            "regions": {
                "@@append": [
                    "us-east-1",
                    "eu-north-1"
                ]
            },
            "selections": {
                "tags": {
                    "My_Backup_Assignment": {
                        "iam_role_arn": {"@@assign": "arn:aws:iam::$account:role/
MyIamRole"},
                        "tag_key": {"@@assign": "dataType"},
                        "tag_value": {
                            "@@assign": [
                                "PII",
                                "RED"
                            ]
                        }
                    }
                }
            },
            "advanced_backup_settings": {
                "ec2": {
                    "windows_vss": {"@@assign": "enabled"}
                }
            },
            "backup_plan_tags": {
                "stage": {
                    "tag_key": {"@@assign": "Stage"},
                    "tag_value": {"@@assign": "Beta"}
                }
            }
        }
    }
}
```

Backup policy syntax includes the following components:

- `$account` variables – In certain text strings in the policies, you can use the `$account` variable to represent the current AWS account. When AWS Backup runs a plan in the effective policy, it automatically replaces this variable with the current AWS account in which the effective policy and its plans are running.

  > **Important**
  > You can use the `$account` variable only in policy elements that can include an Amazon Resource Name (ARN), such as those that specify the backup vault to store the backup in, or the IAM role with permissions to perform the backup.

  For example, the following requires that a vault named `My_Vault` exist in each AWS account that the policy applies to.

  ```
  arn:aws:backup:us-west-2:$account:vault:My_Vault"
  ```

  We recommend that you use AWS CloudFormation stack sets and its integration with Organizations to automatically create and configure backup vaults and IAM roles for each member account in the organization. For more information, see Create a stack set with self-managed permissions in the *AWS CloudFormation User Guide*.
- Inheritance operators – Backup policies can use both the inheritance value-setting operators (p. 95) and the child control operators (p. 96).
- `plans`

  At the top level key of the policy is the `plans` key. A backup policy must always start with this fixed key name at the top of the policy file. You can have one or more backup plans under this key.
- Each plan under the `plans` top level key has a key name that consists of the backup plan name assigned by the user. In the preceding example, the backup plan name is `PII_Backup_Plan`. You can have multiple plans in a policy, each with its own `rules`, `regions`, `selections`, and `tags`.

  This backup plan key name in a backup policy maps to the value of the `BackupPlanName` key in an AWS Backup plan.

  Each plan can contain the following elements:
  - `rules (p. 171)` – This key contains a collection of rules. Each rule translates to a scheduled task, with a start time and window in which to back up the resources identified by the `selections` and `regions` elements in the effective backup policy.
  - `regions (p. 174)` – This key contains an array list of AWS Regions whose resources can be backed up by this policy.
  - `selections (p. 174)` – This key contains one or more collections of resources (within the specified `regions`) that are backed up by the specified `rules`.
  - `advanced_backup_settings (p. 175)` – This key contains settings specific to backups running on certain resources.
  - `backup_plan_tags (p. 175)` – This specifies tags that are attached to the backup plan itself.
- `rules`

The `rules` policy key maps to the `Rules` key in an AWS Backup plan. You can have one or more rules under the `rules` key. Each rule becomes a scheduled task to perform a backup of the selected resources.

Each rule contains a key whose name is the name of the rule. In the previous example, the rule name is "My_Hourly_Rule". The value of the rule key is the following collection of rule elements:

- `schedule_expression` – This policy key maps to the `ScheduleExpression` key in an AWS Backup plan.

  Specifies the start time of the backup. This key contains the `@@assign` inheritance value operator (p. 95) and a string value with a CRON expression that specifies when AWS Backup is to initiate a backup job. The general format of the CRON string is: "cron( )". Each is a number or wildcard. For example, `cron(0 5 ? * 1,3,5 *)` starts the backup at 5 AM every Monday, Wednesday, and Friday. `cron(0 0/1 ? * * *)` starts the backup every hour on the hour, every day of the week.
- `target_backup_vault_name` – This policy key maps to the `TargetBackupVaultName` key in an AWS Backup plan.

  Specifies the name of the backup vault in which to store the backup. You create the value by using AWS Backup. This key contains the `@@assign` inheritance value operator (p. 95) and a string value with a vault name.

  > **Important**
  > The vault must already exist when the backup plan is launched the first time. We recommend that you use AWS CloudFormation stack sets and its integration with Organizations to automatically create and configure backup vaults and IAM roles for each member account in the organization. For more information, see Create a stack set with self-managed permissions in the *AWS CloudFormation User Guide*.
- `start_backup_window_minutes` – This policy key maps to the `StartWindowMinutes` key in an AWS Backup plan.

  (Optional) Specifies the number of minutes to wait before canceling a job that does not start successfully. This key contains the `@@assign` inheritance value operator (p. 95) and a value with an integer number of minutes.
- `complete_backup_window_minutes` – This policy key maps to the `CompletionWindowMinutes` key in an AWS Backup plan.

  (Optional) Specifies the number of minutes after a backup job successfully starts before it must complete or it is canceled by AWS Backup. This key contains the `@@assign` inheritance value operator (p. 95) and a value with an integer number of minutes.
- `enable_continuous_backup` – This policy key maps to the `EnableContinuousBackup` key in an AWS Backup plan.

  (Optional) Specifies whether AWS Backup creates continuous backups. `True` causes AWS Backup to create continuous backups capable of point-in-time restore (PITR). `False` (or not specified) causes AWS Backup to create snapshot backups.

  > **Note**
  > Because PITR-enabled backups can be retained for a maximum of 35 days, you must either choose `False` or don't specify a value ***if*** you set either of the following options:
  >
  > - Set `delete_after_days` to greater than 35.
  > - Set `move_to_code_storage_after_days` to any value.

  For more information about continuous backups, see Point-in-time recovery in the *AWS Backup Developer Guide*.
- `lifecycle` – This policy key maps to the `Lifecycle` key in an AWS Backup plan.

(Optional) Specifies when AWS Backup transitions this backup to cold storage and when it expires.

- `move_to_cold_storage_after_days` – This policy key maps to the `MoveToColdStorageAfterDays` key in an AWS Backup plan.

  Specifies the number of days after the backup occurs before AWS Backup moves the recovery point to cold storage. This key contains the `@@assign` inheritance value operator (p. 95) and a value with an integer number of days.
- `delete_after_days` – This policy key maps to the `DeleteAfterDays` key in an AWS Backup plan.

  Specifies the number of days after the backup occurs before AWS Backup deletes the recovery point. This key contains the `@@assign` inheritance value operator (p. 95) and a value with an integer number of days. If you transition a backup to cold storage, it must stay there a minimum of 90 days, so this value must be a minimum of 90 days greater than the `move_to_cold_storage_after_days` value.

- `copy_actions` – This policy key maps to the `CopyActions` key in an AWS Backup plan.

  (Optional) Specifies that AWS Backup should copy the backup to one or more additional locations. Each backup copy location is described as follows:

  - A key whose name uniquely identifies this copy action. At this time, the key name must be the Amazon Resource Name (ARN) of the backup vault. This key contains two entries.
    - `target_backup_vault_arn` – This policy key maps to the `DestinationBackupVaultArn` key in an AWS Backup plan.

      (Optional) Specifies the vault in which AWS Backup stores an additional copy of the backup. The value of this key contains the `@@assign` inheritance value operator (p. 95) and the ARN of the vault.
      - To reference a vault in the AWS account that the backup policy is running in, use the `$account` variable in the ARN in place of the account ID number. When AWS Backup runs the backup plan, it automatically replaces the variable with the account ID number of the AWS account in which the policy is running. This enables the backup to run correctly when the backup policy applies to more than one account in an organization.
      - To reference a vault in a different AWS account in the same organization, use the actual account ID number in the ARN.

        Important

        - If this key is missing, then an all lower-case version of the ARN in the parent key name is used. Because ARNs are case sensitive, this string might not match the actual ARN of the fault and the plan fails. For this reason, we recommend you always supply this key and value.
        - The backup vault that you want to copy the backup to must already exist the first time you launch the backup plan. We recommend that you use AWS CloudFormation stack sets and its integration with Organizations to automatically create and configure backup vaults and IAM roles for each member account in the organization. For more information, see Create a stack set with self-managed permissions in the *AWS CloudFormation User Guide*.
    - `lifecycle` – This policy key maps to the `Lifecycle` key under the `CopyAction` key in an AWS Backup plan.

      (Optional) Specifies when AWS Backup transitions this copy of a backup to cold storage and when it expires.
      - `move_to_cold_storage_after_days` – This policy key maps to the `MoveToColdStorageAfterDays` key in an AWS Backup plan.

        Specifies the number of days after the backup occurs before AWS Backup moves the recovery point to cold storage. This key contains the `@@assign` inheritance value operator (p. 95) and a value with an integer number of days.

    - `delete_after_days` – This policy key maps to the `DeleteAfterDays` key in an AWS Backup plan.

        Specifies the number of days after the backup occurs before AWS Backup deletes the recovery point. This key contains the `@@assign` inheritance value operator (p. 95) and a value with an integer number of days. If you transition a backup to cold storage, it must stay there a minimum of 90 days, so this value must be a minimum of 90 days greater than the `move_to_cold_storage_after_days` value.

  - `recovery_point_tags` – This policy key maps to the `RecoveryPointTags` key in an AWS Backup plan.

    (Optional) Specifies tags that AWS Backup attaches to each backup that it creates from this plan. This key's value contains one or more of the following elements:

    - An identifier for this key name and value pair. This name for each element under `recovery_point_tags` is the tag key name in all lower case, even if the `tag_key` has a different case treatment. This identifier is ***not*** case sensitive. In the previous example, this key pair was identified by the name `Owner`. Each key pair contains the following elements:
      - `tag_key` – Specifies the tag key name to attach to the backup plan. This key contains the `@@assign` inheritance value operator (p. 95) and a string value. The value is case sensitive.
      - `tag_value` – Specifies the value that is attached to the backup plan and associated with the `tag_key`. This key contains any of the inheritance value operators (p. 95), and one or more values to replace, append, or remove from the effective policy. The values are case sensitive.

- `regions`

  The `regions` policy key specifies which AWS Regions that AWS Backup looks in to find the resources that match the conditions in the `selections` key. This key contains any of the inheritance value operators (p. 95) and one or more string values for AWS Region codes, for example: `["us-east-1", "eu-north-1"]`.

- `selections`

  The `selections` policy key specifies the resources that are backed up by the plan rules in this policy. This key roughly corresponds to the BackupSelection object in AWS Backup. The resources are specified by a query for matching tag key names and values. The `selections` key contains one key under it – `tags`.

  - `tags` – Specifies the tags that identify the resources, and the IAM role that has permission to both query the resources and back them up. This key's value contains one or more of the following elements:
    - An identifier for this tag element. This identifier under `tags` is the tag key name in all lower case, even if the tag to query has a different case treatment. This identifier is ***not*** case sensitive. In the previous example, one element was identified by the name `My_Backup_Assignment`. Each identifier under `tags` contains the following elements:
      - `iam_role_arn` – Specifies the IAM role that has permission to access the resources identified by the tag query in the AWS Regions specified by the `regions` key. This value contains the `@@assign` inheritance value operator (p. 95) and a string value that contains the ARN of the role. AWS Backup uses this role to query for and discover the resources and to perform the backup.

        You can use the `$account` variable in the ARN in place of the account ID number. When the backup plan is run by AWS Backup, it automatically replaces the variable with the actual account ID number of the AWS account in which the policy is running.

> **Important**
>
> The role must already exist when you launch the backup plan the first time. We recommend that you use AWS CloudFormation stack sets and its integration with Organizations to automatically create and configure backup vaults and IAM roles for each member account in the organization. For more information, see Create a stack set with self-managed permissions in the *AWS CloudFormation User Guide*.

    - `tag_key` – Specifies the tag key name to search for. This key contains the `@@assign` inheritance value operator (p. 95) and a string value. The value is case sensitive.
    - `tag_value` – Specifies the value that must be associated with a key name that matches `tag_key`. AWS Backup includes the resource in the backup only if both the `tag_key` and `tag_value` match. This key contains any of the inheritance value operators (p. 95) and one or more values to replace, append, or remove from the effective policy. The values are case sensitive.

- `advanced_backup_settings` – Specifies settings for specific backup scenarios. This key contains one or more settings. Each setting is a JSON object string with the following elements:
  - Object key name – A string that specifies the type of resource to which the following advanced settings apply.
  - Object value – A JSON object string that contains one or more backup settings specific to the associated resource type.

  At this time, the only advanced backup setting that is supported enables Microsoft Volume Shadow Copy Service (VSS) backups for Windows or SQL Server running on an Amazon EC2 instance. The key name must be the `"ec2"` resource type, and the value specifies that `"windows_vss"` support is either `enabled` or `disabled` for backups performed on those Amazon EC2 instances. For more information about this feature, see Creating a VSS-Enabled Windows Backup in the *AWS Backup Developer Guide*.

```
"advanced_backup_settings": {
    "ec2": {
        "windows_vss": {
            "@@assign": "enabled"
        }
    }
}
```

- `backup_plan_tags` – Specifies tags that are attached to the backup plan itself. This does not impact the tags specified in any rules or selections.

  (Optional) You can attach tags to your backup plans. This key's value is a collection of elements.

  The key name for each element under `backup_plan_tags` is the tag key name in all lower case, even if the tag to query has a different case treatment. This identifier is ***not*** case sensitive. The value for each of these entries consists of the following keys:
  - `tag_key` – Specifies the tag key name to attach to the backup plan. This key contains the `@@assign` inheritance value operator (p. 95) and a string value. This value is case sensitive.
  - `tag_value` – Specifies the value that is attached to the backup plan and associated with the `tag_key`. This key contains the `@@assign` inheritance value operator (p. 95) and a string value. This value is case sensitive.

Backup policy examples

The example backup policies that follow are for information purposes only. In some of the following examples, the JSON whitespace formatting might be compressed to save space.

Example 1: Policy assigned to a parent node

The following example shows a backup policy that is assigned to one of the parent nodes of an account.

Parent policy – This policy can be attached to the organization's root, or to any OU that is a parent of all of the intended accounts.

```
{
    "plans": {
        "PII_Backup_Plan": {
            "regions": {
                "@@assign": [
                    "ap-northeast-2",
                    "us-east-1",
                    "eu-north-1"
                ]
            },
            "rules": {
                "Hourly": {
                    "schedule_expression": {
                        "@@assign": "cron(0 5/1 ? * * *)"
                    },
                    "start_backup_window_minutes": {
                        "@@assign": "480"
                    },
                    "complete_backup_window_minutes": {
                        "@@assign": "10080"
                    },
                    "lifecycle": {
                        "move_to_cold_storage_after_days": {
                            "@@assign": "180"
                        },
                        "delete_after_days": {
                            "@@assign": "270"
                        }
                    },
                    "target_backup_vault_name": {
                        "@@assign": "FortKnox"
                    },
                    "copy_actions": {
                        "arn:aws:backup:us-east-1:$account:backup-vault:secondary_vault": {
                            "target_backup_vault_arn": {
                                "@@assign": "arn:aws:backup:us-east-1:$account:backup-
vault:secondary_vault"
                            },
                            "lifecycle": {
                                "move_to_cold_storage_after_days": {
                                    "@@assign": "30"
                                },
                                "delete_after_days": {
                                    "@@assign": "120"
                                }
                            }
                        },
                        "arn:aws:backup:us-west-1:111111111111:backup-
vault:tertiary_vault": {
                            "target_backup_vault_arn": {
                                "@@assign": "arn:aws:backup:us-west-1:111111111111:backup-
vault:tertiary_vault"
                            },
                            "lifecycle": {
                                "move_to_cold_storage_after_days": {
                                    "@@assign": "30"
                                },
                                "delete_after_days": {
                                    "@@assign": "120"
                                }
                            }
                        }                    }
                }
            },
            "selections": {
                "tags": {
                    "datatype": {
                        "iam_role_arn": {
                            "@@assign": "arn:aws:iam::$account:role/MyIamRole"
                        },
                        "tag_key": {
                            "@@assign": "dataType"
                        },
                        "tag_value": {
                            "@@assign": [
                                "PII",
                                "RED"
                            ]
                        }
                    }
                }
            },
            "advanced_backup_settings": {
                "ec2": {
                    "windows_vss": {
                        "@@assign": "enabled"
                    }
                }
            }
        }
    }
}
```

If no other policies are inherited or attached to the accounts, the effective policy rendered in each applicable AWS account looks like the following example. The CRON expression causes the backup to run once an hour on the hour. The account ID 123456789012 will be the actual account ID for each account.

```
{
    "plans": {
        "PII_Backup_Plan": {
            "regions": [
                "us-east-1",
                "ap-northeast-3",
                "eu-north-1"
            ],
            "rules": {
                "hourly": {
                    "schedule_expression": "cron(0 0/1 ? * * *)",
                    "start_backup_window_minutes": "60",
                    "target_backup_vault_name": "FortKnox",
                    "lifecycle": {
                        "to_delete_after_days": "2",
                        "move_to_cold_storage_after_days": "180"
                    },
                    "copy_actions": {
                        "arn:aws:backup:us-east-1:$account:vault:secondary_vault": {
                            "target_backup_vault_arn": {
                                "@@assign": "arn:aws:backup:us-east-1:
$account:vault:secondary_vault"
                            },
                            "lifecycle": {
                                "to_delete_after_days": "28",
                                "move_to_cold_storage_after_days": "180"
                            }
                        },
                        "arn:aws:backup:us-west-1:111111111111:vault:tertiary_vault": {
                            "target_backup_vault_arn": {
                                "@@assign": "arn:aws:backup:us-
west-1:111111111111:vault:tertiary_vault"
                            },
                            "lifecycle": {
                                "to_delete_after_days": "28",
                                "move_to_cold_storage_after_days": "180"
                            }
                        }
                    }
                }
            },
            "selections": {
                "tags": {
                    "datatype": {
                        "iam_role_arn": "arn:aws:iam::123456789012:role/MyIamRole",
                        "tag_key": "dataType",
                        "tag_value": [
                            "PII",
                            "RED"
                        ]
                    }
                }
            },
            "advanced_backup_settings": {
                "ec2": {
                    "windows_vss": "enabled"
                }
            }
        }
    }
}
```

Example 2: A parent policy is merged with a child policy

In the following example, an inherited parent policy and a child policy either inherited or directly attached to an AWS account merge to form the effective policy.

Parent policy – This policy can be attached to the organization's root or to any parent OU.

```
{
    "plans": {
       "PII_Backup_Plan": {
            "regions": { "@@append":[ "us-east-1", "ap-northeast-3", "eu-north-1" ] },
            "rules": {
                "Hourly": {
                    "schedule_expression": { "@@assign": "cron(0 0/1 ? * * *)" },
                    "start_backup_window_minutes": { "@@assign": "60" },
                    "target_backup_vault_name": { "@@assign": "FortKnox" },
                    "lifecycle": {
                        "move_to_cold_storage_after_days": { "@@assign": "28" },
                        "to_delete_after_days": { "@@assign": "180" }
                    },
                    "copy_actions": {
                        "arn:aws:backup:us-east-1:$account:vault:secondary_vault" : {
                            "target_backup_vault_arn" : {
                                "@@assign" : "arn:aws:backup:us-east-1:
$account:vault:secondary_vault"
                            },
                            "lifecycle": {
                                "move_to_cold_storage_after_days": { "@@assign": "28" },
                                "to_delete_after_days": { "@@assign": "180" }
                            }
                        }
                    }
                }
            },
            "selections": {
                "tags": {
                    "datatype": {
                        "iam_role_arn": { "@@assign": "arn:aws:iam::$account:role/
MyIamRole" },
                        "tag_key": { "@@assign": "dataType" },
                        "tag_value": { "@@assign": [ "PII", "RED" ] }
                    }
                }
            }
        }
    }
}
```

Child policy – This policy can be attached directly to the account or to an OU any level below the one the parent policy is attached to.

```
{
    "plans": {
       "Monthly_Backup_Plan": {
            "regions": {
                "@@append":[ "us-east-1", "eu-central-1" ] },
            "rules": {
                "Monthly": {
                    "schedule_expression": { "@@assign": "cron(0 5 1 * ? *)" },
                    "start_backup_window_minutes": { "@@assign": "480" },
                    "target_backup_vault_name": { "@@assign": "Default" },
                    "lifecycle": {
                        "move_to_cold_storage_after_days": { "@@assign": "30" },
                        "to_delete_after_days": { "@@assign": "365" }
                    },
                    "copy_actions": {
                        "arn:aws:backup:us-east-1:$account:vault:Default" : {
                            "target_backup_vault_arn" : {
                                "@@assign" : "arn:aws:backup:us-east-1:
$account:vault:Default"
                            },
                            "lifecycle": {
                                "move_to_cold_storage_after_days": { "@@assign": "30" },
                                "to_delete_after_days": { "@@assign": "365" }
                            }
                        }
                    }
                }
            },
            "selections": {
                "tags": {
                    "MonthlyDatatype": {
                        "iam_role_arn": { "@@assign": "arn:aws:iam::$account:role/
MyMonthlyBackupIamRole" },
                        "tag_key": { "@@assign": "BackupType" },
                        "tag_value": { "@@assign": [ "MONTHLY", "RED" ] }
                    }
                }
            }
        }
    }
}
```

Resulting effective policy – The effective policy applied to the accounts contains two plans, each with its own set of rules and set of resources to apply the rules to.

```
{
    "plans": {
       "PII_Backup_Plan": {
            "regions": [ "us-east-1", "ap-northeast-3", "eu-north-1" ],
            "rules": {
                "hourly": {
                    "schedule_expression": "cron(0 0/1 ? * * *)",
                    "start_backup_window_minutes": "60",
                    "target_backup_vault_name": "FortKnox",
                    "lifecycle": {
                        "to_delete_after_days": "2",
                        "move_to_cold_storage_after_days": "180"
                    },
                    "copy_actions": {
                        "arn:aws:backup:us-east-1:$account:vault:secondary_vault" : {
                            "target_backup_vault_arn" : {
                                "@@assign" : "arn:aws:backup:us-east-1:
$account:vault:secondary_vault"
                            },
                            "lifecycle": {
                                "move_to_cold_storage_after_days": "28",
                                "to_delete_after_days": "180"
                            }
                        }
                    }
                }
            },
            "selections": {
                "tags": {
                    "datatype": {
                        "iam_role_arn": "arn:aws:iam::$account:role/MyIamRole",
                        "tag_key": "dataType",
                        "tag_value": [ "PII", "RED" ]
                    }
                }
            }
        },
        "Monthly_Backup_Plan": {
            "regions": [ "us-east-1", "eu-central-1" ],
            "rules": {
                "monthly": {
                    "schedule_expression": "cron(0 5 1 * ? *)",
                    "start_backup_window_minutes": "480",
                    "target_backup_vault_name": "Default",
                    "lifecycle": {
                        "to_delete_after_days": "365",
                        "move_to_cold_storage_after_days": "30"
                    },
                    "copy_actions": {
                        "arn:aws:backup:us-east-1:$account:vault:Default" : {
                            "target_backup_vault_arn": {
                                "@@assign" : "arn:aws:backup:us-east-1:
$account:vault:Default"
                            },
                            "lifecycle": {
                                "move_to_cold_storage_after_days": "30",
                                "to_delete_after_days": "365"
                            }
                        }
                    }
                }
            },
            "selections": {
                "tags": {
                    "monthlydatatype": {
                        "iam_role_arn": "arn:aws:iam::&ExampleAWSAccountNo3;:role/
MyMonthlyBackupIamRole",
                        "tag_key": "BackupType",
                        "tag_value": [ "MONTHLY", "RED" ]
                    }
                }
            }
        }
    }
}
```

Example 3: A parent policy prevents any changes by a child policy

In the following example, an inherited parent policy uses the child control operators (p. 96) to enforce all settings and prevents them from being changed or overridden by a child policy.

Parent policy – This policy can be attached to the organization's root or to any parent OU. The presence of `"@@operators_allowed_for_child_policies": ["@@none"]` at every node of the policy means that a child policy can't make changes of any kind to the plan. Nor can a child policy add additional plans to the effective policy. This policy becomes the effective policy for every OU and account under the OU to which it is attached.

```
{
    "plans": {
        "@@operators_allowed_for_child_policies": ["@@none"],
        "PII_Backup_Plan": {
            "@@operators_allowed_for_child_policies": ["@@none"],
            "regions": {
                "@@operators_allowed_for_child_policies": ["@@none"],
                "@@append": [
                    "us-east-1",
                    "ap-northeast-3",
                    "eu-north-1"
                ]
            },
            "rules": {
                "@@operators_allowed_for_child_policies": ["@@none"],
                "Hourly": {
                    "@@operators_allowed_for_child_policies": ["@@none"],
                    "schedule_expression": {
                        "@@operators_allowed_for_child_policies": ["@@none"],
                        "@@assign": "cron(0 0/1 ? * * *)"
                    },
                    "start_backup_window_minutes": {
                        "@@operators_allowed_for_child_policies": ["@@none"],
                        "@@assign": "60"
                    },
                    "target_backup_vault_name": {
                        "@@operators_allowed_for_child_policies": ["@@none"],
                        "@@assign": "FortKnox"
                    },
                    "lifecycle": {
                        "@@operators_allowed_for_child_policies": ["@@none"],
                        "move_to_cold_storage_after_days": {
                            "@@operators_allowed_for_child_policies": ["@@none"],
                            "@@assign": "28"
                        },
                        "to_delete_after_days": {
                            "@@operators_allowed_for_child_policies": ["@@none"],
                            "@@assign": "180"
                        }
                    },
                    "copy_actions": {
                        "@@operators_allowed_for_child_policies": ["@@none"],
                        "arn:aws:backup:us-east-1:$account:vault:secondary_vault": {
                            "@@operators_allowed_for_child_policies": ["@@none"],
                            "target_backup_vault_arn": {
                                "@@assign": "arn:aws:backup:us-east-1:
$account:vault:secondary_vault",
                                "@@operators_allowed_for_child_policies": ["@@none"]
                            },
                            "lifecycle": {
                                "@@operators_allowed_for_child_policies": ["@@none"],
                                "to_delete_after_days": {
                                    "@@operators_allowed_for_child_policies": ["@@none"],
                                    "@@assign": "28"
                                },
                                "move_to_cold_storage_after_days": {
                                    "@@operators_allowed_for_child_policies": ["@@none"],
                                    "@@assign": "180"
                                }
                            }
                        }
                    }
                }
            },
            "selections": {
                "@@operators_allowed_for_child_policies": ["@@none"],
                "tags": {
                    "@@operators_allowed_for_child_policies": ["@@none"],
                    "datatype": {
                        "@@operators_allowed_for_child_policies": ["@@none"],
                        "iam_role_arn": {
                            "@@operators_allowed_for_child_policies": ["@@none"],
                            "@@assign": "arn:aws:iam::$account:role/MyIamRole"
                        },
                        "tag_key": {
                            "@@operators_allowed_for_child_policies": ["@@none"],
                            "@@assign": "dataType"
                        },
                        "tag_value": {
                            "@@operators_allowed_for_child_policies": ["@@none"],
                            "@@assign": [
                                "PII",
                                "RED"
                            ]
                        }
                    }
                }
            },
            "advanced_backup_settings": {
                "@@operators_allowed_for_child_policies": ["@@none"],
                "ec2": {
                    "@@operators_allowed_for_child_policies": ["@@none"],
                    "windows_vss": {
                        "@@assign": "enabled",
                        "@@operators_allowed_for_child_policies": ["@@none"]
                    }
                }
            }
        }
    }
}
```

Resulting effective policy – If any child backup policies exist, they are ignored and the parent policy becomes the effective policy.

```
{
    "plans": {
        "PII_Backup_Plan": {
            "regions": [
                "us-east-1",
                "ap-northeast-3",
                "eu-north-1"
            ],
            "rules": {
                "hourly": {
                    "schedule_expression": "cron(0 0/1 ? * * *)",
                    "start_backup_window_minutes": "60",
                    "target_backup_vault_name": "FortKnox",
                    "lifecycle": {
                        "to_delete_after_days": "2",
                        "move_to_cold_storage_after_days": "180"
                    },
                    "copy_actions": {
                        "target_backup_vault_arn": "arn:aws:backup:us-
east-1:123456789012:vault:secondary_vault",
                        "lifecycle": {
                            "move_to_cold_storage_after_days": "28",
                            "to_delete_after_days": "180"
                        }
                    }
                }
            },
            "selections": {
                "tags": {
                    "datatype": {
                        "iam_role_arn": "arn:aws:iam::123456789012:role/MyIamRole",
                        "tag_key": "dataType",
                        "tag_value": [
                            "PII",
                            "RED"
                        ]
                    }
                }
            },
            "advanced_backup_settings": {
                "ec2": {"windows_vss": "enabled"}
            }
        }
    }
}
```

Example 4: A parent policy prevents changes to one backup plan by a child policy

In the following example, an inherited parent policy uses the child control operators (p. 96) to enforce the settings for a single plan and prevents them from being changed or overridden by a child policy. The child policy can still add additional plans.

Parent policy – This policy can be attached to the organization's root or to any parent OU. This example is similar to the previous example with all child inheritance operators blocked, except at the `plans` top level. The `@@append` setting at that level enables child policies to add other plans to the collection in the effective policy. Any changes to the inherited plan are still blocked.

The sections in the plan are truncated for clarity.

```
{
    "plans": {
        "@@operators_allowed_for_child_policies": ["@@append"],
        "PII_Backup_Plan": {
            "@@operators_allowed_for_child_policies": ["@@none"],
            "regions": { ... },
            "rules": { ... },
            "selections": { ... }
        }
    }
}
```

Child policy – This policy can be attached directly to the account or to an OU any level below the one the parent policy is attached to. This child policy defines a new plan.

The sections in the plan are truncated for clarity.

```
{
    "plans": {
        "MonthlyBackupPlan": {
            "regions": { ... },
            "rules": { ... },
            "selections": { … }
        }
    }
}
```

Resulting effective policy – The effective policy includes both plans.

```
{
    "plans": {
        "PII_Backup_Plan": {
            "regions": { ... },
            "rules": { ... },
            "selections": { ... }
        },
        "MonthlyBackupPlan": {
            "regions": { ... },
            "rules": { ... },
            "selections": { … }
        }
    }
}
```

Example 5: A child policy overrides settings in a parent policy

In the following example, a child policy uses value-setting operators (p. 95) to override some of the settings inherited from a parent policy.

Parent policy – This policy can be attached to the organization's root or to any parent OU. Any of the settings can be overridden by a child policy because the default behavior, in the absence of a child-control operator (p. 96) that prevents it, is to allow the child policy to `@@assign`, `@@append`, or `@@remove`. The parent policy contains all of the required elements for a valid backup plan, so it backs up your resources successfully if it is inherited as is.

```
{
    "plans": {
        "PII_Backup_Plan": {
            "regions": {
                "@@append": [
                    "us-east-1",
                    "ap-northeast-3",
                    "eu-north-1"
                ]
            },
            "rules": {
                "Hourly": {
                    "schedule_expression": {"@@assign": "cron(0 0/1 ? * * *)"},
                    "start_backup_window_minutes": {"@@assign": "60"},
                    "target_backup_vault_name": {"@@assign": "FortKnox"},
                    "lifecycle": {
                        "to_delete_after_days": {"@@assign": "2"},
                        "move_to_cold_storage_after_days": {"@@assign": "180"}
                    },
                    "copy_actions": {
                        "arn:aws:backup:us-east-1:$account:vault:t2": {
                            "target_backup_vault_arn": {"@@assign": "arn:aws:backup:us-
east-1:$account:vault:t2"},
                            "lifecycle": {
                                "move_to_cold_storage_after_days": {"@@assign": "28"},
                                "to_delete_after_days": {"@@assign": "180"}
                            }
                        }
                    }
                }
            },
            "selections": {
                "tags": {
                    "datatype": {
                        "iam_role_arn": {"@@assign": "arn:aws:iam::$account:role/
MyIamRole"},
                        "tag_key": {"@@assign": "dataType"},
                        "tag_value": {
                            "@@assign": [
                                "PII",
                                "RED"
                            ]
                        }
                    }
                }
            }
        }
    }
}
```

Child policy – The child policy includes only the settings that need to be different from the inherited parent policy. There must be an inherited parent policy that provides the other required settings when merged into an effective policy. Otherwise, the effective backup policy contains a backup plan that is not valid and doesn't back up your resources as expected.

```
{
    "plans": {
        "PII_Backup_Plan": {
            "regions": {
                "@@assign": [
                    "us-west-2",
                    "eu-central-1"
                ]
            },
            "rules": {
                "Hourly": {
                    "schedule_expression": {"@@assign": "cron(0 0/2 ? * * *)"},
                    "start_backup_window_minutes": {"@@assign": "80"},
                    "target_backup_vault_name": {"@@assign": "Default"},
                    "lifecycle": {
                        "move_to_cold_storage_after_days": {"@@assign": "30"},
                        "to_delete_after_days": {"@@assign": "365"}
                    }
                }
            }
        }
    }
}
```

Resulting effective policy – The effective policy includes settings from both policies, with the settings provided by the child policy overriding the settings inherited from the parent. In this example, the following changes occur:

- The list of Regions is replaced with a completely different list. If you wanted to add a Region to the inherited list, use `@@append` instead of `@@assign` in the child policy.
- AWS Backup performs every other hour instead of hourly.
- AWS Backup allows 80 minutes for the backup to start instead of 60 minutes.
- AWS Backup uses the `Default` vault instead of `FortKnox`.
- The lifecycle is extended for both the transfer to cold storage and the eventual deletion of the backup.

```
{
    "plans": {
        "PII_Backup_Plan": {
            "regions": [
                "us-west-2",
                "eu-central-1"
            ],
            "rules": {
                "hourly": {
                    "schedule_expression": "cron(0 0/2 ? * * *)",
                    "start_backup_window_minutes": "80",
                    "target_backup_vault_name": "Default",
                    "lifecycle": {
                        "to_delete_after_days": "365",
                        "move_to_cold_storage_after_days": "30"
                    },
                    "copy_actions": {
                        "arn:aws:backup:us-east-1:$account:vault:secondary_vault": {
                            "target_backup_vault_arn": {"@@assign": "arn:aws:backup:us-
east-1:$account:vault:secondary_vault"},
                            "lifecycle": {
                                "move_to_cold_storage_after_days": "28",
                                "to_delete_after_days": "180"
                            }
                        }
                    }
                }
            },
            "selections": {
                "tags": {
                    "datatype": {
                        "iam_role_arn": "arn:aws:iam::$account:role/MyIamRole",
                        "tag_key": "dataType",
                        "tag_value": [
                            "PII",
                            "RED"
                        ]
                    }
                }
            }
        }
    }
}
```

Tag policies

For information and procedures common to all policy types, see the following topics:

- Enable and disable policy types (p. 85)
- Get details about your policies (p. 87)
- Policy syntax and inheritance (p. 92)

You can use tag policies to maintain consistent tags, including the preferred case treatment of tag keys and tag values.

What are tags?

Tags are custom attribute labels that you assign or that AWS assigns to AWS resources. Each tag has two parts:

- A *tag key* (for example, `CostCenter`, `Environment`, or `Project`). Tag keys are case sensitive.
- An optional field known as a *tag value* (for example, `111122223333` or `Production`). Omitting the tag value is the same as using an empty string. Like tag keys, tag values are case sensitive.

The rest of this page describes tag policies. For more information about tags, see the following sources:

- For more general information on tagging, including naming and usage conventions, see Tagging AWS Resources in the *AWS General Reference*.
- For a list of services that support using tags, see the *Resource Groups Tagging API Reference*.
- For information on tagging Organizations resources, see Tagging AWS Organizations resources (p. 220).
- For information on tagging resources in other AWS services, see the documentation for each.
- For information about using tags to categorize resources, see AWS Tagging Strategies.

What are tag policies?

Tag policies are a type of policy that can help you standardize tags across resources in your organization's accounts. In a tag policy, you specify tagging rules applicable to resources when they are tagged.

For example, a tag policy can specify that when the `CostCenter` tag is attached to a resource, it must use the case treatment and tag values that the tag policy defines. A tag policy can also specify that noncompliant tagging operations on specified resource types are *enforced*. In other words, noncompliant tagging requests on specified resource types are prevented from completing. Untagged resources or tags that aren't defined in the tag policy aren't evaluated for compliance with the tag policy.

Using tag policies involves working with multiple AWS services:

- Use **AWS Organizations** to manage *tag policies*. When you sign in to the organization's management account, you use Organizations to enable the tag policies feature. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account. Then you can create tag policies and attach them to the organization entities to put those tagging rules in effect.

- Use **AWS Resource Groups** to manage *compliance* with tag policies. When you sign in to an account in your organization, you use Resource Groups to find noncompliant tags on resources in the account. You can correct noncompliant tags in the AWS service where you created the resource.

  If you sign in to the management account in your organization, you can view compliance information for all your organization's accounts.

Tag policies are available only in an organization that has all features enabled (p. 35). For more information on what's required to use tag policies, see Prerequisites and permissions for managing tag policies (p. 188).

> **Important**
> To get started with tag policies, AWS strongly recommends that you follow the example workflow described in Getting started with tag policies (p. 190) before moving on to more advanced tag policies. It's best to understand the effects of attaching a simple tag policy to a single account before expanding tag policies to an entire OU or organization. It's especially important to understand a tag policy's effects before you *enforce* compliance with any tag policy. The tables on the Getting started with tag policies (p. 190) page also provide links to instructions for more advanced policy-related tasks.

Prerequisites and permissions for managing tag policies

This page describes the prerequisites and required permissions for managing tag policies in AWS Organizations.

Topics

Prerequisites for managing tag policies

Using tag policies requires the following:

- Your organization must have all features enabled (p. 35).
- You must be signed in to your organization's management account.
- You need the permissions that are listed in Permissions for managing tag policies (p. 188).

To evaluate compliance with tag policies, you use AWS Resource Groups. For information on requirements for evaluating compliance, see Prerequisites and Permissions in the *AWS Resource Groups User Guide*.

Permissions for managing tag policies

The following example IAM policy provides permissions for managing tag policies.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ManageTagPolicies",
            "Effect": "Allow",
            "Action": [
                "organizations:ListPoliciesForTarget",
                "organizations:ListTargetsForPolicy",
                "organizations:DescribeEffectivePolicy",
                "organizations:DescribePolicy",
                "organizations:ListRoots",
                "organizations:DisableAWSServiceAccess",
                "organizations:DetachPolicy",
                "organizations:DeletePolicy",
                "organizations:DescribeAccount",
                "organizations:DisablePolicyType",
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:ListPolicies",
                "organizations:ListAccountsForParent",
                "organizations:ListAccounts",
                "organizations:EnableAWSServiceAccess",
                "organizations:ListCreateAccountStatus",
                "organizations:DescribeOrganization",
                "organizations:UpdatePolicy",
                "organizations:EnablePolicyType",
                "organizations:DescribeOrganizationalUnit",
                "organizations:AttachPolicy",
                "organizations:ListParents",
                "organizations:ListOrganizationalUnitsForParent",
                "organizations:CreatePolicy",
                "organizations:DescribeCreateAccountStatus"
            ],
            "Resource": "*"
        }
    ]
}
```

For more information on IAM policies and permissions, see the IAM User Guide.

Best practices for using tag policies

AWS recommends the following best practices for using tag policies.

Decide on a tag capitalization strategy

Determine how you want to capitalize tags and consistently implement that strategy across all resource types. For example, decide whether to use `Costcenter`, `costcenter`, or `CostCenter`, and use the same convention for all tags. For consistent results in compliance reports, avoid using similar tags with inconsistent case treatment. This strategy will help you define tag policies for your organization.

Use the recommended workflow

Start small by creating a simple tag policy. Then attach it to a member account that you can use for testing purposes. Use the workflows described in Getting started with tag policies (p. 190).

Determine tagging rules

This will depend on your organization's needs. For example, you may want to specify that when a `CostCenter` tag is attached to AWS Secrets Manager secrets, it must use the specified case treatment. Create tag policies that define compliant tags and attach them to the organization entities where you want those tagging rules to be in effect.

Educate account administrators

When you're ready to expand your use of tag policies, educate account administrators as follows:

- Communicate your tagging strategy.
- Emphasize that administrators need to use tags on specific resource types.

  This is important, as untagged resources don't show as noncompliant in compliance results.
- Provide guidance on checking compliance with tag policies. Instruct administrators to find and correct noncompliant tags on resources in their account using the procedure described in Evaluating Compliance for an Account in the *AWS Resource Groups User Guide.* Let them know how often you want them to check for compliance.

Use caution in enforcing compliance

Enforcing compliance could prevent users in your organization's accounts from tagging the resources they need. Review the information in Understanding enforcement (p. 204). Also see the workflows described in Getting started with tag policies (p. 190).

Consider creating an SCP to set guardrails around resource creation requests

Resources that have never had tags attached to them don't show as noncompliant in reports. Account administrators can still create untagged resources. In some cases, you can use a service control policy (SCP) to set guardrails around resource creation requests. For an example SCP, see Require a tag on specified created resources (p. 136). To learn whether an AWS service supports controlling access using tags, see AWS Services That Work with IAM in the *IAM User Guide*. Look for the services that have **Yes** in the **Authorization based on tags** column. Choose the name of the service to view the authorization and access control documentation for that service.

Getting started with tag policies

Using tag policies involves working with multiple AWS services. To get started, review the following pages. Then follow the workflows on this page to get familiar with tag policies and their effects.

- Prerequisites and permissions for managing tag policies (p. 188)
- Best practices for using tag policies (p. 189)

Using tag policies for the first time

Follow these steps to get started using tag policies for the first time.

Task	Account to sign in to	AWS service console to use
Step 1: Enable tag policies for your organization. (p. 85)	The organization's management account.¹	AWS Organizations
Step 2: Create a tag policy (p. 193). Keep your first tag policy simple. Enter one tag key in the case treatment you want to use and leave all other options at their defaults.	The organization's management account.¹	AWS Organizations

Task	Account to sign in to	AWS service console to use
Step 3: Attach a tag policy to a single member account that you can use for testing. (p. 199) You'll need to sign in to this account in the next step.	The organization's management account.[1]	AWS Organizations
Step 4: Create some resources with compliant tags and some with noncompliant tags.	The member account that you're using for testing purposes.	Any AWS service that you are comfortable with. For example, you can use AWS Secrets Manager and follow the procedure in Creating a Basic Secret to create secrets with compliant and non-compliant secrets.
Step 5: View the effective tag policy and evaluate the compliance status of the account.	The member account that you're using for testing purposes.	Resource Groups and the AWS service where the resource was created. If you created resources with compliant and non-compliant tags, you should see the non-compliant tags in the results.
Step 6: Repeat the process of finding and correcting compliance issues until the resources in the test account are compliant with your tag policy.	The member account that you're using for testing purposes.	Resource Groups and the AWS service where the resource was created.
At any time, you can evaluate organization-wide compliance.	The organization's management account.[1]	Resource Groups

[1] You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.

Expanding use of tag policies

You can perform the following tasks in any order to expand your use of tag policies.

Advanced task	Account to sign in to	AWS service console to use
Create more advanced tag policies (p. 193). Follow the same process as for first-time users, but try other tasks. For example, define additional keys or values or specify different case treatment for a tag key. You can use the information in Understanding policy	The organization's management account.[1]	AWS Organizations

Advanced task	Account to sign in to	AWS service console to use
inheritance (p. 91) and Tag policy syntax (p. 214) to create more detailed tag policies.		
Attach tag policies to additional accounts or OUs. (p. 199) Check the effective tag policy for an account (p. 202) after you attach more policies to it or to any OU in which the account is a member.	The organization's management account.[1]	AWS Organizations
Create an SCP to require tags when anyone creates new resources. For an example, see Require a tag on specified created resources (p. 136).	The organization's management account.[1]	AWS Organizations
Continue to evaluate the compliance status of the account against the effective tag policy as it changes. Correct noncompliant tags.	A member account with an effective tag policy.	Resource Groups and the AWS service where the resource was created.
Evaluate organization-wide compliance.	The organization's management account.[1]	Resource Groups

[1] You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.

Enforcing tag policies for the first time

To enforce tag policies for the first time, follow a workflow similar to using tag policies for the first time and use a test account.

> **Warning**
> Use caution in enforcing compliance. Make sure that you understand the effects of using tag policies and follow the recommended workflow. Test how enforcement works on a test account before expanding it to more accounts. Otherwise, you could prevent users in your organization's accounts from tagging the resources they need. For more information, see Understanding enforcement (p. 204).

Enforcement tasks	Account to sign in to	AWS service console to use
Step 1: Create a tag policy (p. 193). Keep your first enforced tag policy simple. Enter one tag key in the case treatment you want to use, and choose the **Prevent noncompliant operations for this tag** option. Then specify one resource type to enforce it	The organization's management account.[1]	AWS Organizations

Enforcement tasks	Account to sign in to	AWS service console to use
on. Continuing with our earlier example, you can choose to enforce it on Secrets Manager secrets.		
Step 2: Attach a tag policy to a single, test account. (p. 199)	The organization's management account.[1]	AWS Organizations
Step 3: Try creating some resources with compliant tags, and some with noncompliant tags. You shouldn't be allowed to create a tag a resource of the type specified in the tag policy with a noncompliant tag.	The member account that you're using for testing purposes.	Any AWS service that you are comfortable with. For example, you can use AWS Secrets Manager and follow the procedure in Creating a Basic Secret to create secrets with compliant and non-compliant secrets.
Step 4: Evaluate the compliance status of the account against the effective tag policy and correct noncompliant tags.	The member account that you're using for testing purposes.	Resource Groups and the AWS service where the resource was created.
Step 5: Repeat the process of finding and correcting compliance issues until the resources in the test account are compliant with your tag policy.	The member account that you're using for testing purposes.	Resource Groups and the AWS service where the resource was created.
At any time, you can evaluate organization-wide compliance.	The organization's management account.[1]	Resource Groups

[1] You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.

Creating, updating, and deleting tag policies

In this topic:

- After you enable tag policies (p. 85) for your organization, you can create a policy (p. 193).
- When your tagging requirements change, you can update an existing policy (p. 196).
- When you no longer need a policy and after you detach it from all organizational units (OUs) and accounts, you can delete it (p. 199).

> **Important**
> Untagged resources don’t appear as noncompliant in results.

Creating a tag policy

> **Minimum permissions**
> To create tag policies, you need permission to run the following action:
>
> - `organizations:CreatePolicy`

You can create a tag policy in the AWS Management Console in one of two ways:

- A visual editor that lets you choose options and generates the JSON policy text for you.
- A text editor that lets you directly create the JSON policy text yourself.

The visual editor makes the process easy, but it limits your flexibility. It's a great way to create your first policies and get comfortable with using them. After you understand how they work and have started to be limited by what the visual editor provides, you can add advanced features to your policies by editing the JSON policy text yourself. The visual editor uses only the @@assign value-setting operator (p. 95), and it doesn't provide any access to the child control operators (p. 96). You can add the child control operators only if you manually edit the JSON policy text.

AWS Management Console

To create a tag policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **Tag policies** page, choose **Create policy**.
3. On the **Create policy** page, enter a **Policy name** and an optional **Policy description**.
4. (Optional) You can add one or more tags to the policy object itself. These tags are not part of the policy. To do this, choose **Add tag** and then enter a key and an optional value. Leaving the value blank sets it to an empty string; it isn't `null`. You can attach up to 50 tags to a policy. For more information, see Tagging AWS Organizations resources (p. 220).
5. You can build the tag policy using the **Visual editor** as described in this procedure. You can also type or paste a tag policy in the **JSON** tab. For information about tag policy syntax, see Tag policy syntax (p. 214).

   For **New tag key 1**, specify the name of a tag key to add.
6. For **Tag key capitalization compliance**, leave this option cleared (the default) to specify that the inherited parent tag policy, if any exists, should define the case treatment for the tag key.

   Enable this option if you want to mandate a specific capitalization for the tag key using this policy. If you select this option, the capitalization you specified for **Tag Key** overrides the case treatment specified in an inherited parent policy.

   If a parent policy doesn't exist and you don't enable this option, tag keys in all lowercase characters are considered compliant. For more information about inheritance from parent policies, see Understanding policy inheritance (p. 91).

   > **Tip**
   > Consider using the example tag policy shown in Example 1: Define organization-wide tag key case (p. 216) as a guide in creating a tag policy that define tag keys and their case treatment. Attach it to the organization root. Later, you can create and attach additional tag policies to OUs or accounts to create additional tagging rules.
7. For **Tag value compliance**, enable this option if you want to add allowed values for this tag key to any values inherited from a parent policy.

   By default, this option is cleared, which means that only those values defined in and inherited from a parent policy are considered compliant. If a parent policy doesn't exist and you don't specify tag values then any value (including no value at all) is considered compliant.

   To update the list of acceptable tag values, select **Specify allowed values for this tag key** and then choose **Specify values**. When prompted, enter the new values (one value per box), and then choose **Save changes**.
8. For **Prevent noncompliant operations for this tag**, we recommend that you leave this option cleared (the default) unless you are experienced with using tag policies. Make sure that you have reviewed the recommendations in Understanding enforcement (p. 204), and test thoroughly.

Otherwise, you could prevent users in your organization's accounts from tagging the resources they need.

If you do want to enforce compliance with this tag key, select the check box and then **Specify resource types**. When prompted, select the resource types to include in the policy. Then choose **Save changes**.

> **Important**
> When you select this option, any operations that manipulate tags for resources of the specified types succeed only if the operation results in tags that are compliant with the policy.

9. (Optional) To add another tag key to this tag policy, choose **Add tag key**. Then perform steps 6–9 to define the tag key.
10. When you're finished building your tag policy, choose **Save changes**.

AWS CLI & AWS SDKs

To create a tag policy

You can use one of the following to create a tag policy:

- AWS CLI: create-policy

  You can use any text editor to create a tag policy. Use JSON syntax and save the tag policy as a file with any name and extension in a location of your choosing. Tag policies can have a maximum of 2,500 characters, including spaces. For information about tag policy syntax, see Tag policy syntax (p. 214).

To create a tag policy

1. Create a tag policy in a text file that looks similar to the following:

   Contents of `testpolicy.json`:

```
{
    "tags": {
        "CostCenter": {
            "tag_key": {
                "@@assign": "CostCenter"
            }
        }
    }
}
```

   This tag policy defines the `CostCenter` tag key. The tag can accept any value or no value. A policy like this means that a resource that has the CostCenter tag attached with or without a value is compliant.

2. Create a policy that contains the policy content from the file. Extra white space in the output has been truncated for readability.

```
$ aws organizations create-policy \
    --name "MyTestTagPolicy" \
    --description "My Test policy" \
    --content file://testpolicy.json \
    --type TAG_POLICY
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-a1b2c3d4e5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
tag_policy/p-a1b2c3d4e5",
            "Name": "MyTestTagPolicy",
            "Description": "My Test policy",
            "Type": "TAG_POLICY",
            "AwsManaged": false
        },
        "Content": "{\n\"tags\":{\n\"CostCenter\":{\n\"tag_key\":{\n\"@@assign\":
\"CostCenter\"\n}\n}\n}\n}\n\n"
    }
}
```

- AWS SDKs: CreatePolicy

What to Do Next

After you create a tag policy, you can put your tagging rules into effect. To do that, attach the policy (p. 199) to the organization root, organizational units (OUs), AWS accounts within your organization, or a combination of organization entities.

Updating a tag policy

Minimum permissions

To update a tag policy, you must have permission to run the following actions:

- `organizations:UpdatePolicy` with a `Resource` element in the same policy statement that includes the ARN of the specified policy (or "*")
- `organizations:DescribePolicy` with a `Resource` element in the same policy statement that includes the ARN of the specified policy (or "*")

AWS Management Console

To update a tag policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **Tag policies page** page, choose the tag policy that you want to update.
3. Choose **Edit policy**.
4. You can enter a new **Policy name**, **Policy description**. You can change the policy content by using either the **Visual editor** or by editing the **JSON**.
5. When you're finished updating the tag policy, choose **Save changes**.

AWS CLI & AWS SDKs

To update a policy

You can use one of the following to update a policy:

- AWS CLI: update-policy

  The following example renames a tag policy.

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --name "Renamed tag policy"
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
tag_policy/p-i9j8k7l6m5",
            "Name": "Renamed tag policy",
            "Type": "TAG_POLICY",
            "AwsManaged": false
        },
        "Content": "{\n\"tags\":{\n\"CostCenter\":{\n\"tag_key\":{\n\"@@assign\":
\"CostCenter\"\n}\n}\n}\n}\n\n"
    }
}
```

The following example adds or changes the description for a tag policy.

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --description "My new tag policy description"
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
tag_policy/p-i9j8k7l6m5",
            "Name": "Renamed tag policy",
            "Description": "My new tag policy description",
            "Type": "TAG_POLICY",
            "AwsManaged": false
        },
       "Content": "{\n\"tags\":{\n\"CostCenter\":{\n\"tag_key\":{\n\"@@assign\":
\"CostCenter\"\n}\n}\n}\n}\n\n"
    }
}
```

The following example changes the JSON policy document attached to an AI services opt-out policy. In this example, the content is taken from a file called `policy.json` with the following text:

```
{
  "tags": {
    "Stage": {
      "tag_key": {
        "@@assign": "Stage"
      },
      "tag_value": {
        "@@assign": [
          "Production",
          "Test"
        ]
      }
    }
  }
}
```

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --content file://policy.json
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
tag_policy/p-i9j8k7l6m5",
            "Name": "Renamed tag policy",
            "Description": "My new tag policy description",
            "Type": "TAG_POLICY",
            "AwsManaged": false
        },
         "Content": "{\"tags\":{\"Stage\":{\"tag_key\":{\"@@assign\":\"Stage\"},
\"tag_value\":{\"@@assign\":[\"Production\",\"Test\"]},\"enforced_for\":{\"@@assign
\":[\"ec2:instance\"]}}}}"
}
```

- AWS SDKs: UpdatePolicy

Editing tags attached to a tag policy

When you sign in to your organization's management account, you can add or remove the tags attached to a tag policy. To do this, complete the following steps.

Minimum permissions

To edit the tags attached to a tag policy in your AWS organization, you must have the following permissions:

- `organizations:DescribeOrganization` (console only – to navigate to the policy)
- `organizations:DescribePolicy` (console only – to navigate to the policy)
- `organizations:TagResource`
- `organizations:UntagResource`

AWS Management Console

To edit the tags attached to an AI services opt-out policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **Tag policies page** page, choose the name of the policy with the tags that you want to edit.
3. On the chosen policy's detail page, choose the **Tags** tab, and then choose **Manage tags**.
4. You can perform any of these actions on this page:
   - Edit the value for any tag by entering a new value over the old one. You can't modify the key. To change a key, you must delete the tag with the old key and add a tag with the new key.
   - Remove an existing tag by choosing **Remove**.
   - Add a new tag key and value pair. Choose **Add tag**, then enter the new key name and optional value in the provided boxes. If you leave the **Value** box empty, the value is an empty string; it isn't `null`.
5. Choose **Save changes** after you've made all the additions, removals, and edits you want to make.

AWS CLI & AWS SDKs

To edit the tags attached to a tag policy

You can use one of the following commands to edit the tags attached to a tag policy:

- AWS CLI: tag-resource and untag-resource
- AWS SDKs: TagResource and UntagResource

Deleting a tag policy

When you sign in to your organization's management account, you can delete a policy that you no longer need in your organization.

Before you can delete a policy, you must first detach it from all attached entities.

Minimum permissions
To delete a tag policy, you must have permission to run the following action:

- `organizations:DeletePolicy`

AWS Management Console

To delete a tag policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2.
3. On the **Tag policies** **page** page, choose the policy that you want to delete.
4. You must first detach the policy that you want to delete from all roots, OUs, and accounts. Choose the **Targets** tab, choose the radio button next to each root, OU, or account that's shown in the **Targets** list, and then choose **Detach**. In the confirmation dialog box, choose **Detach**.
5. Choose **Delete** at the top of the page.
6. On the confirmation dialog box, enter the name of the policy, and then choose **Delete**.

AWS CLI & AWS SDKs

To delete a tag policy

You can use one of the following to delete a policy:

- AWS CLI: delete-policy

  The following example deletes the specified policy. It works only if the policy isn't attached to any root, OU, or account.

  ```
  $ aws organizations delete-policy \
      --policy-id p-i9j8k7l6m5
  ```

  This command produces no output when successful.
- AWS SDKs: DeletePolicy

Attaching and detaching tag policies

You can use tag policies on an entire organization as well as on organizational units (OUs) and individual accounts.

- When you attach a tag policy to your *organization root*, the tag policy applies to all of that root's member OUs and accounts.
- When you attach a tag policy to an *OU*, that tag policy applies to the accounts that belong to the OU. Those accounts are also subject to any tag policy attached to the organization root.
- When you attach a tag policy to an *account*, that tag policy, applies to the account. In addition, that account is subject to any tag policy attached to the organization root, *plus* any tag policy attached to an OU that the account belongs to.

The aggregation of any tag policies the account inherits, plus any tag policy directly attached to the account is the *effective tag policy* (p. 202). For more information, see Understanding policy inheritance (p. 91).

Important
Untagged resources don't appear as noncompliant in results.

Minimum permissions
To attach tag policies, you must have permission to run the following action:

- `organizations:AttachPolicy`

AWS Management Console

You can attach a tag policy by either navigating to the policy or to the root, OU, or account that you want to attach the policy to.

To attach a tag policy by navigating to the root, OU, or account

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page, navigate to and then choose the name of the root, OU, or account that you want to attach a policy to. You might have to expand OUs (choose the ▶) to find the OU or account that you want.
3. In the **Policies** tab, in the entry for **Tag policies**, choose **Attach**.
4. Find the policy that you want and choose **Attach policy**.

   The list of attached tag policies on the **Policies** tab is updated to include the new addition. The policy change takes effect immediately.

To attach a tag policy by navigating to the policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Tag policies** page, choose the name of the policy that you want to attach.
3. On the **Targets** tab, choose **Attach**.
4. Choose the radio button next to the root, OU, or account that you want to attach the policy to. You might have to expand OUs (choose the ▶) to find the OU or account that you want.
5. Choose **Attach policy**.

   The list of attached tag policies on the **Targets** tab is updated to include the new addition. The policy change takes effect immediately.

AWS CLI & AWS SDKs

To attach a tag policy to the organization root, OU, or account

You can use one of the following to attach a tag policy:

- AWS CLI: attach-policy

  The following procedure shows how to attach the tag policy you just created to a single test account.

  - Attach the tag policy to your test account by running a command like the following:

    ```
    $ aws organizations attach-policy \
    ```

```
    --target-id <account-id> \
    --policy-id p-a1b2c3d4e5
```

This command has no output if it is successful.

- AWS SDKs: AttachPolicy

The policy change takes effect immediately.

What to Do Next

After you attach a tag policy, you can find out how compliant your resources are with that tag policy. To do this, use the Resource Groups console. For information, see Evaluating an Account's Compliance in the *AWS Resource Groups User Guide*.

Detaching a tag policy

When you sign in to your organization's management account, you can detach a tag policy from the organization root, OU, or account that it is attached to. After you detach a tag policy from an entity, that policy no longer applies to any account that was affected by the now detached entity. To detach a policy, complete the following steps.

Minimum permissions
To detach a tag policy from the organization root, OU, or account, you must have permission to run the following action:

- `organizations:DetachPolicy`

AWS Management Console

You can detach a tag policy by either navigating to the policy or to the root, OU, or account that you want to detach the policy from.

To detach a tag policy by navigating to the root, OU, or account it's attached to

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **AWS accounts** page navigate to the Root, OU, or account that you want to detach a policy from. You might have to expand OUs (choose the ▶) to find the OU or account that you want. Choose the name of the Root, OU, or account.
3. On the **Policies** tab, choose the radio button next to the tag policy that you want to detach, and then choose **Detach**.
4. In the confirmation dialog box, choose **Detach policy**.

   The list of attached tag policies is updated. The policy change takes effect immediately.

To detach a tag policy by navigating to the policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **Tag policies** page, choose the name of the policy that you want to detach from a root, OU, or account.
3. On the **Targets** tab, choose the radio button next to the root, OU, or account that you want to detach the policy from. You might have to expand OUs (choose the ▶) to find the OU or account that you want.

4. Choose **Detach**.
5. In the confirmation dialog box, choose **Detach**.

   The list of attached tag policies is updated. The policy change takes effect immediately.

AWS CLI & AWS SDKs

To detach a tag policy from the organization root, OU, or account

You can use one of the following to detach a tag policy:

- AWS CLI: detach-policy
- AWS SDKs: DetachPolicy

The policy change takes effect immediately.

Viewing effective tag policies

Before you start checking compliance status for tagged resources in an account, it's helpful to first determine the effective tag policy for an account.

What is the effective tag policy?

The *effective tag policy* specifies the tagging rules that apply to an account. It is the aggregation of any tag policies the account inherits, plus any tag policy directly attached to the account. When you attach a tag policy to the organization root, it applies to all accounts in your organization. When you attach a tag policy to an OU, it applies to all accounts and OUs that belong to the OU.

For example, the tag policy attached to the organization root may define a `CostCenter` tag with four compliant values. A separate tag policy attached to the account may restrict the `CostCenter` key to only two of the four compliant values. The combination of these tag policies comprises the effective tag policy. The result is that only two of the four compliant tag values defined in the organization root tag policy are compliant for the account.

For more information and more advanced examples of how effective tag policies are generated, see Understanding policy inheritance (p. 91).

How to view the effective tag policy

You can view the effective tag policy for an account from the AWS Management Console, AWS API, or AWS Command Line Interface.

Minimum permissions
To view the effective tag policy for an account, you must have permission to run the following actions:

- `organizations:DescribeEffectivePolicy`
- `organizations:DescribeOrganization`

AWS Management Console

To view the effective tag policy for an account

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.

2. On the **AWS accounts** page, choose the name of the account for which you want to view the effective tag policy. You might have to expand OUs (choose the ▶) to find the account that you want.
3. On the **Policies** tab, in the **Tag policies** section, choose **View the effective tag policy for this AWS account**.

   The console displays the effective policy applied to the specified account.

   > **Note**
   > You can't copy and paste an effective policy and use it as the JSON for another tag policy without significant changes. Tag policy documents must include the inheritance operators (p. 95) that specify how each setting is merged into the final effective policy.

AWS CLI & AWS SDKs

To view the effective tag policy for an account

You can use one of the following to view the effective tag policy:

- AWS CLI: describe-effective-policy

  To determine what tagging rules are inherited by or attached to an account, run the following from the account and save the results to a file:

```
$ aws organizations describe-effective-policy \
    --policy-type TAG_POLICY
{
    "EffectivePolicy": {
        "PolicyContent": "{\"tags\":{\"costcenter\":{\"tag_value\":[\"*\"],\"tag_key
\":\"CostCenter\"}}}",
        "LastUpdatedTimestamp": "2020-06-09T08:34:25.103000-07:00",
        "TargetId": "123456789012",
        "PolicyType": "TAG_POLICY"
    }
}
```

  If a tag policy is attached to the account as well as to the root or any OUs, the combination of all of the inherited policies defines the account's effective tag policy. In these cases, running `describe-effective-policy` from the account returns the merged content of all tag policies in the account's hierarchy.
- AWS SDKs: DescribeEffectivePolicy

Using CloudWatch Events to monitor noncompliant tags

You can use CloudWatch Events to monitor when noncompliant tags are introduced. In the following example event, the `"false"` value for `tag-policy-compliant` indicates that a new tag is noncompliant with the effective tag policy.

```
{
  "detail-type": "Tag Change on Resource",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ec2:us-east-1:123456789012:instance/i-0000000aaaaaaaaaa"
  ],
  "detail": {
    "changed-tag-keys": [
      "a-new-key"
    ],
    "service": "ec2",
    "resource-type": "instance",
    "version": 3,
    "tag-policy-compliant": "false",
    "tags": {
      "a-new-key": "tag-value-on-new-key-just-added"
    }
  }
}
```

You can subscribe to events and specify strings or patterns to monitor. For more information on CloudWatch Events, see the *Amazon CloudWatch Events User Guide*.

Understanding enforcement

A tag policy can specify that noncompliant tagging operations on specified resource types are *enforced*. In other words, noncompliant tagging requests on specified resource types are prevented from completing.

> **Important**
> Enforcement has no effect on resources that are created without tags.

To enforce compliance with tag policies, do one of the following when you create a tag policy (p. 193):

- From the **Visual editor** tab, select **Prevent noncompliant operations for this tag** (p. 194).
- From the **JSON** tab, use the `enforced_for` field. For information on tag policy syntax, see Tag policy syntax and examples (p. 214).

Follow these best practices for enforcing compliance with tag policies:

- **Use caution in enforcing compliance** – Make sure you understand the effects of using tag policies, and follow the recommended workflows described in Getting started with tag policies (p. 190). Test how enforcement works on a test account before expanding it to more accounts. Otherwise, you could prevent users in your organization's accounts from tagging the resources they need.
- **Be aware of what resource types you can enforce on** – You can only enforce compliance with tag policies on supported resource types (p. 206). Resource types that support enforcing compliance are listed when you use the visual editor to build a tag policy.
- **Understand interactions with some services** – Some AWS services have container-like groupings of resources that automatically create resources for you, and tags can propagate from a resource in one service to another. For example, tags on Amazon EC2 Auto Scaling groups and Amazon EMR clusters can automatically propagate to the contained Amazon EC2 instances. You may have tag policies for Amazon EC2 that are more strict than for Auto Scaling groups or EMR clusters. If you enable enforcement, the tag policy prevents resources from being tagged and may block dynamic scaling and provisioning.

The following sections show how you can find non-compliant resources, and correct them to be compliant.

Finding non-compliant resources for an account

For each account, you can get information about non-compliant resources. You should run this command from every Region in which the account has resources.

To find non-compliant resources for an account that with a tag policy, you can run the following command When you sign in to the account and save the results to a file:

```
$ aws resourcegroupstaggingapi get-resources  --region us-east-1 \
    --include-compliance-details \
    --exclude-compliant-resources > outputfile.txt
```

Correcting non-compliant tags in resources

After finding non-compliant tags, make corrections using any of the following methods. You must be signed in to the account that has the resource with non-compliant tags:

- Use the console or tagging API operations of the AWS service that created the non-compliant resources.
- Use the AWS Resource Groups TagResources and UntagResources operations to add tags that are compliant with the effective policy or to remove non-compliant tags.

Finding and correcting additional non-compliance issues

Finding and correcting compliance issues is an iterative process. Repeat the steps in the two previous sections until the resources you care about are compliant with your tag policy.

Generating an organization-wide compliance report

At any time, you can generate a report that lists all tagged resources in the AWS accounts across your organization. The report shows whether each resource is compliant with the effective tag policy. Note that it can take up to 48 hours for changes you make to a tag policy or resources to be reflected in the organization-wide compliance report. For example, assume that you have a tag policy that defines a new standardized tag for a resource type. Resources of that type that don't have this tag are shown as compliant in the report for up to 48 hours.

You can generate the report from your organization's management account in the `us-east-1` Region, provided that it has access to an Amazon S3 bucket. The bucket must have an attached bucket policy as shown in Amazon S3 Bucket Policy for Storing Report. To generate the report, run the following command:

```
$ aws resourcegroupstaggingapi get-compliance-summary --region us-east-1
{
    "SummaryList": [
        {
            "LastUpdated": "2020-06-09T18:40:46Z",
            "NonCompliantResources": 0
        }
    ]
}
```

You can generate one report at a time.

This report can take some time to complete. You can check the status by running the following command:

```
$ aws resourcegroupstaggingapi describe-report-creation --region us-east-1
{
    "Status": "SUCCEEDED"
}
```

After the above command returns `SUCCEEDED`, you can open the report from the Amazon S3 bucket.

Services and resource types that support enforcement

The following services and resource types support enforcement with tag policies:

Service name	Resource type	JSON syntax
Amazon API Gateway	• API keys • Domain names • REST API operations • Stages	• `"apigateway:apikeys"` • `"apigateway:domainnames"` • `"apigateway:restapis"` • `"apigateway:stages"`
AWS Amplify	• Component • Theme	• `"amplifyuibuilder:component"` • `"amplifyuibuilder:theme"`
AWS AppConfig	• Application • Configuration Profile • Deployment • Deployment Strategy • Environment	• `"appconfig:application"` • `"appconfig:configurationprofile"` • `"appconfig:deployment"` • `"appconfig:deploymentstrategy"` • `"appconfig:environment"`
AWS App Mesh	• All • Mesh • Router • Virtual node • Virtual router • Virtual service	• `"appmesh:*"` • `"appmesh:mesh"` • `"appmesh:route"` • `"appmesh:virtualNode"` • `"appmesh:virtualRouter"` • `"appmesh:virtualService"`
Amazon Athena	• All • Workgroup	• `"athena:*"` • `"athena:workgroup"`
AWS Audit Manager	• Assessment • Assessment Control Set • Assessment Framework • Control	• `"auditmanager:assessment"` • `"auditmanager:assessmentControlSet"` • `"auditmanager:assessmentFramework"` • `"auditmanager:control"`
AWS Backup	• All • Backup plan • Vault • Gateway • Hyper Visor • VM	• `"backup:*"` • `"backup:backupPlan"` • `"backup:backupVault"` • `"backup-gateway:gateway"` • `"backup-gateway:hypervisor"` • `"backup-gateway:vm"`
AWS Batch	• Job • Job Definition • Job Queue	• `"batch:job"` • `"batch:job-definition"` • `"batch:job-queue"`
AWS BugBust	• Event	• `"bugbust:event"`
AWS Certificate Manager	• All • Certificates	• `"acm:*"` • `"acm:certificate"`

Service name	Resource type	JSON syntax
	• Private Certificate Authority	• `"acm-pca:certificate-authority"`
Amazon CloudFront	• All • Distribution • Streaming distribution	• `"cloudfront:*"` • `"cloudfront:distribution"` • `"cloudfront:streaming-distribution"`
AWS CloudTrail	• All • Trail	• `"cloudtrail:*"` • `"cloudtrail:trail"`
Amazon CloudWatch	• All • Alarm	• `"cloudwatch:*"` • `"cloudwatch:alarm"`
Amazon CloudWatch Events	• All • Event bus • Rule	• `"events:*"` • `"events:event-bus"` • `"events:rule"`
Amazon CloudWatch Logs	• Log group	• `"logs:log-group"`
AWS CodeBuild	• All • Project	• `"codebuild:*"` • `"codebuild:project"`
AWS CodeCommit	• All • Repository	• `"codecommit:*"` • `"codecommit:repository"`
AWS CodePipeline	• All • Action type • Pipeline • Webhook	• `"codepipeline:*"` • `"codepipeline:actiontype"` • `"codepipeline:pipeline"` • `"codepipeline:webhook"`
Amazon Cognito Identity	• All • Identity pool	• `"cognito-identity:*"` • `"cognito-identity:identitypool"`
Amazon Cognito user pools	• All • User pool	• `"cognito-idp:*"` • `"cognito-idp:userpool"`
Amazon Comprehend	• All • Document classifier • Entity recognizer	• `"comprehend:*"` • `"comprehend:document-classifier"` • `"comprehend:entity-recognizer"`
AWS Config	• All • Aggregation authorization • Config aggregator • Config rule	• `"config:*"` • `"config:aggregation-authorization"` • `"config:config-aggregator"` • `"config:config-rule"`
CHMlong	• Application Instance • User Application Instance • Channel	• `"chime:app-instance"` • `"chime:app-instance-user"` • `"chime:channel"`

Service name	Resource type	JSON syntax
AWS Cloud9	• Environment	• `"cloud9:environment"`
Amazon CodeGuru Reviewer	• Association	• `"codeguru-reviewer:association"`
AWS CodeStar connections	• Connection • Host	• `"codestar-connections:connection"` • `"codestar-connections:host"`
Amazon Connect	• Contact Flow • Integration Association • Queue • Quick Connect • Routing Profile • User	• `"connect:contact-flow"` • `"connect:integration-association"` • `"connect:queue"` • `"connect:quick-connect"` • `"connect:routing-profile"` • `"connect:user"`
Amazon Connect Wisdom	• Assistant • Association • Content • Knowledge • Session	• `"wisdom:assistant"` • `"wisdom:association"` • `"wisdom:content"` • `"wisdom:knowledge"` • `"wisdom:session"`
AWS Database Migration Service	• All • Endpoint • ES • Rep • Subgrp • Task	• `"dms:*"` • `"dms:endpoint"` • `"dms:es"` • `"dms:rep"` • `"dms:subgrp"` • `"dms:task"`
Amazon Data Lifecycle Manager	• Policy	• `"dlm:policy"`
AWS Direct Connect	• All • Dxcon • Dxlag • Dxvif	• `"directconnect:*"` • `"directconnect:dxcon"` • `"directconnect:dxlag"` • `"directconnect:dxvif"`
Amazon DynamoDB	• All • Table	• `"dynamodb:*"` • `"dynamodb:table"`

Service name	Resource type	JSON syntax
Amazon EC2	• Capacity reservation • Client VPN endpoint • Customer gateway • DHCP options • Elastic IP • Fleet • FPGA image • Host reservation • Image • Instance • Internet gateway • Launch template • NAT gateway • Network ACL • Network interface • Reserved Instances • Route table • Security group • Snapshot • Spot Instance request • Subnet • Traffic mirror filter • Traffic mirror session • Traffic mirror target • Volume • VPC • VPC endpoint • VPC endpoint service • VPC peering connection • VPN connection • VPN gateway	• `"ec2:capacity-reservation"` • `"ec2:client-vpn-endpoint"` • `"ec2:customer-gateway"` • `"ec2:dhcp-options"` • `"ec2:elastic-ip"` • `"ec2:fleet"` • `"ec2:fpga-image"` • `"ec2:host-reservation"` • `"ec2:image"` • `"ec2:instance"` • `"ec2:internet-gateway"` • `"ec2:launch-template"` • `"ec2:natgateway"` • `"ec2:network-acl"` • `"ec2:network-interface"` • `"ec2:reserved-instances"` • `"ec2:route-table"` • `"ec2:security-group"` • `"ec2:snapshot"` • `"ec2:spot-instance-request"` • `"ec2:subnet"` • `"ec2:traffic-mirror-filter"` • `"ec2:traffic-mirror-session"` • `"ec2:traffic-mirror-target"` • `"ec2:volume"` • `"ec2:vpc"` • `"ec2:vpc-endpoint"` • `"ec2:vpc-endpoint-service"` • `"ec2:vpc-peering-connection"` • `"ec2:vpn-connection"` • `"ec2:vpn-gateway"`
Amazon EC2 Recycle Bin	• Rule	• `"rbin:rule"`
Amazon Elastic Container Registry	• Repository	• `"ecr:repository"`
AWS Elastic Beanstalk	• Application • Application version • Configuration template • Platform	• `"elasticbeanstalk:application"` • `"elasticbeanstalk:applicationversion"` • `"elasticbeanstalk:configurationtemplate"` • `"elasticbeanstalk:platform"`

Service name	Resource type	JSON syntax
Amazon Elastic Container Service	- Cluster - Service - Task set	- `"ecs:cluster"` - `"ecs:service"` - `"ecs:task-set"`
Amazon Elastic File System	- All - File system	- `"elasticfilesystem:*"` - `"elasticfilesystem:file-system"`
Amazon Elastic Inference	- Accelerator	- `"elastic-inference:accelerator"`
Amazon Elastic Kubernetes Service	- Cluster	- `"eks:cluster"`
Amazon Elastic Search	- Domain	- `"es:domain"`
Amazon EMR	- All - Cluster - Editor	- `"elasticmapreduce:*"` - `"elasticmapreduce:cluster"` - `"elasticmapreduce:editor"`
Amazon ElastiCache	- Cluster	- `"elasticache:cluster"`
Amazon Fraud Detector	- Detector - Detector version - Model - Rule - Variable	- `"frauddetector:detector"` - `"frauddetector:detector-version"` - `"frauddetector:model"` - `"frauddetector:rule"` - `"frauddetector:variable"`
Amazon Global Accelerator	- Accelerator	- `"globalaccelerator:accelerator"`
Elastic Load Balancing	- All - Load balancer - Target group	- `"elasticloadbalancing:*"` - `"elasticloadbalancing:loadbalancer"` - `"elasticloadbalancing:targetgroup"`
Amazon FSx	- All - Backup - File system	- `"fsx:*"` - `"fsx:backup"` - `"fsx:file-system"`
Amazon GuardDuty	- Detector - Filter - IP Set - Threat Intel Set	- `"guardduty:detector"` - `"guardduty:filter"` - `"guardduty:ipset"` - `"guardduty:threatintelset"`
Amazon HealthLake	- Datastore	- `"healthlake:datastore"`
Amazon Inspector	- Filter	- `"inspector2:filter"`

Service name	Resource type	JSON syntax
AWS Identity and Access Management	• Instance Profile • MFA • OIDC Provider • Policy • SAML Provider • Server Certificate	• `"iam:instance-profile"` • `"iam:mfa"` • `"iam:oidc-provider"` • `"iam:policy"` • `"iam:saml-provider"` • `"iam:server-certificate"`
AWS IoT Analytics	• All • Channel • Dataset • Datastore • Pipeline	• `"iotanalytics:*"` • `"iotanalytics:channel"` • `"iotanalytics:dataset"` • `"iotanalytics:datastore"` • `"iotanalytics:pipeline"`
AWS IoT Events	• All • Detector model • Input	• `"iotevents:*"` • `"iotevents:detectorModel"` • `"iotevents:input"`
AWS IoT Fleet Hub	• Application	• `"iotfleethub:application"`
AWS IoT SiteWise	• Asset • Asset Model	• `"iotsitewise:asset"` • `"iotsitewise:asset-model"`
AWS IoT Greengrass	• Bulk Deployment • Connector Definition • Core Definition • Device Definition • Function Definition • Logger Definition • Resource Definition • Subscription Definition	• `"greengrass:bulkDeployment"` • `"greengrass:connectorDefinition"` • `"greengrass:coreDefinition"` • `"greengrass:deviceDefinition"` • `"greengrass:functionDefinition"` • `"greengrass:loggerDefinition"` • `"greengrass:resourceDefinition"` • `"greengrass:subscriptionDefinition"`
AWS Key Management Service	• All • Key	• `"kms:*"` • `"kms:key"`
Amazon Kinesis	• All • Application	• `"kinesisanalytics:*"` • `"kinesisanalytics:application"`
Amazon Kinesis Data Firehose	• All • Delivery stream	• `"firehose:*"` • `"firehose:deliverystream"`
AWS Lambda	• All • Function	• `"lambda:*"` • `"lambda:function"`
Amazon Macie	• Custom Data Identifier	• `"macie2:custom-data-identifier"`
Amazon MediaStore	• Container	• `"mediastore:container"`
Amazon MQ	• Broker • Configuration	• `"mq:broker"` • `"mq:configuration"`

Service name	Resource type	JSON syntax
Amazon Network Firewall	• Firewall • Firewall Policy • Stateful Rule Group • Stateless Rule Group	• `"network-firewall:firewall"` • `"network-firewall:firewall-policy"` • `"network-firewall:stateful-rulegroup"` • `"network-firewall:stateless-rulegroup"`
AWS Organizations	• Account • Organizational Unit • Policy • Root	• `"organizations:account"` • `"organizations:ou"` • `"organizations:policy"` • `"organizations:root"`
Amazon RDS	• Cluster parameter group • Cluster endpoint • Event subscription • DB option group • DB parameter group • DB proxy • DB proxy endpoint • Reserved DB instance • DB security group • DB subnet group • Target group	• `"rds:cluster-pg"` • `"rds:cluster-endpoint"` • `"rds:es"` • `"rds:og"` • `"rds:pg"` • `"rds:db-proxy"` • `"rds:db-proxy-endpoint"` • `"rds:ri"` • `"rds:secgrp"` • `"rds:subgrp"` • `"rds:target-group"`
Amazon Redshift	• All • Cluster • DB group • DB name • DB user • Event subscription • HSM client certificate • HSM configuration • Parameter group • Snapshot • Snapshot copy grant • Snapshot schedule • Subnet group	• `"redshift:*"` • `"redshift:cluster"` • `"redshift:dbgroup"` • `"redshift:dbname"` • `"redshift:dbuser"` • `"redshift:eventsubscription"` • `"redshift:hsmclientcertificate"` • `"redshift:hsmconfiguration"` • `"redshift:parametergroup"` • `"redshift:snapshot"` • `"redshift:snapshotcopygrant"` • `"redshift:snapshotschedule"` • `"redshift:subnetgroup"`
AWS Resource Access Manager	• All • Resource share	• `"ram:*"` • `"ram:resource-share"`
AWS Resource Groups	• All • Group	• `"resource-groups:*"` • `"resource-groups:group"`
Amazon Route 53	• Hosted zone	• `"route53:hostedzone"`

Service name	Resource type	JSON syntax
Amazon Route 53 Resolver	• All • Resolver endpoint • Resolver rule	• `"route53resolver:*"` • `"route53resolver:resolver-endpoint"` • `"route53resolver:resolver-rule"`
Amazon S3	• Bucket	• `"s3:bucket"`
Amazon SageMaker	• App Image Config • Artifact • Context • Training job • Processing job • Model package group • Human task UI • Model Package • Action • Pipeline • Experiment • Flow Definition • Project	• `"sagemaker:app-image-config"` • `"sagemaker:artifact"` • `"sagemaker:context"` • `"sagemaker:training-job"` • `"sagemaker:processing-job "` • `"sagemaker:model-package-group"` • `"sagemaker:human-task-ui"` • `"sagemaker:model-package"` • `"sagemaker:action"` • `"sagemaker:pipeline"` • `"sagemaker:experiment"` • `"sagemaker:flow-definition"` • `"sagemaker:project"`
AWS Secrets Manager	• All • Secret	• `"secretsmanager:*"` • `"secretsmanager:secret"`
AWS Service Catalog	• Application • Attribute Group • Portfolio • Product	• `"servicecatalog:application"` • `"servicecatalog:attributeGroup"` • `"servicecatalog:portfolio"` • `"servicecatalog:product"`
Amazon Simple Notification Service (SNS)	• Topic	• `"sns:topic"`
Amazon Simple Queue Service (SQS)	• Queue	• `"sqs:queue"`
Amazon States Language	• All • Activity • State Machine	• `"states:*"` • `"states:activity"` • `"states:stateMachine"`
AWS Step Functions	• Activity	• `"states:activity"`
AWS Storage Gateway	• All • Gateway • Share • Tape • Volume	• `"storagegateway:*"` • `"storagegateway:gateway"` • `"storagegateway:share"` • `"storagegateway:tape"` • `"storagegateway:volume"`

Service name	Resource type	JSON syntax
AWS Systems Manager	• Automation execution • Document • Maintenance window task • Managed instance • Ops item • Patch baseline • Session • Contacts	• `"ssm:automation-execution"` • `"ssm:document"` • `"ssm:maintenancewindowtask"` • `"ssm:managed-instance"` • `"ssm:opsitem"` • `"ssm:patchbaseline"` • `"ssm:session"` • `"ssm-contacts:contact"`
AWS Transfer Family	• Server • User • Workflow	• `"transfer:server"` • `"transfer:user"` • `"transfer:workflow"`
Amazon Well-Architected	• Workload	• `"wellarchitected:workload"`
Amazon WorkSpaces	• All • Directory • WorkSpace • WorkSpaces bundle • WorkSpaces image • WorkSpaces IP group	• `"workspaces:*"` • `"workspaces:directory"` • `"workspaces:workspace"` • `"workspaces:workspacebundle"` • `"workspaces:workspaceimage"` • `"workspaces:workspaceipgroup"`
Amazon WorkLink	• Fleet	• `"worklink:fleet"`

Tag policy syntax and examples

This page describes tag policy syntax and provides examples.

Tag policy syntax

A tag policy is a plaintext file that is structured according to the rules of JSON. The syntax for tag policies follows the syntax for management policy types. For a complete discussion of that syntax, see Policy syntax and inheritance for management policy types (p. 94). This topic focuses on applying that general syntax to the specific requirements of the tag policy type.

The following tag policy shows basic tag policy syntax:

```
{
    "tags": {
        "costcenter": {
            "tag_key": {
                "@@assign": "CostCenter"
            },
            "tag_value": {
                "@@assign": [
                    "100",
                    "200"
                ]
            },
            "enforced_for": {
                "@@assign": [
                    "secretsmanager:*"
                ]
            }
        }
    }
}
```

Tag policy syntax includes the following elements:

- The `tags` field key name. Tag policies always start with this fixed key name. It's the top line in the example policy above.
- A *policy key* that uniquely identifies the policy statement. It must match the value for the *tag key*, except for the case treatment. Unlike the tag key (described next), the policy value *is not* case sensitive.

  In this example, `costcenter` is the policy key.
- At least one *tag key* that specifies the allowed tag key with the capitalization that you want resources to be compliant with. If case treatment isn't defined, lowercase is the default case treatment for tag keys. The value for the tag key must match the value for the policy key. But since the policy key value is case insensitive, the capitalization can be different.

  In this example, `CostCenter` is the tag key. This is the case treatment that is required for compliance with the tag policy. Resources with alternate case treatment for this tag key are noncompliant with the tag policy.

  You can define multiple tag keys in a tag policy.
- (Optional) A list of one or more acceptable *tag values* for the tag key. If the tag policy doesn't specify a tag value for a tag key, any value (including no value at all) is considered compliant.

  In this example, acceptable values for the `CostCenter` tag key are `100` and `200`.
- (Optional) An `enforced_for` option that indicates whether to prevent any noncompliant tagging operations on specified services and resources. In the console, this is the **Prevent noncompliant operations for this tag** option in the visual editor for creating tag policies. The default setting for this option is null.

  The example tag policy specifies that all AWS Secrets Manager resources must have this tag.

  > **Warning**
  > You should only change this option from the default if you are experienced with using tag policies. Otherwise, you could prevent users in your organization's accounts from creating the resources they need.
- *Operators* that specify how the tag policy merges with other tag policies within the organization tree to create an account's effective tag policy (p. 202). In this example, `@@assign` is used to assign strings to `tag_key`, `tag_value`, and `enforced_for`. For more information on operators, see Inheritance operators (p. 95).
- – You can use the `*` wildcard in tag values and `enforced_for` fields.
  - You can use only one wildcard per tag value. For example, `*@example.com` is allowed, but `*@*.com` is not.
  - For `enforced_for`, you can use `<service>:*` with some services to enable enforcement for all resources for that service. For a list of services and resource types that support `enforced_for`, see Services and resource types that support enforcement (p. 206).

    You can't use a wildcard to specify all services or to specify a resource for all services.

Tag policy examples

The example tag policies (p. 187) that follow are for information purposes only.

> **Note**
>
> Before you attempt to use these example tag policies in your organization, note the following:
>
> - Make sure that you've followed the recommended workflow (p. 190) for getting started with tag policies.
> - You should carefully review and customize these tag policies for your unique requirements.
> - All characters in your tag policy are subject to a maximum size (p. 340). The examples in this guide show tag policies formatted with extra white space to improve their readability. However, to save space if your policy size approaches the maximum size, you can delete any white space. Examples of white space include space characters and line breaks that are outside quotation marks.
> - Untagged resources don't appear as noncompliant in results.

Example 1: Define organization-wide tag key case

The following example shows a tag policy that only defines two tag keys and the capitalization that you want accounts in your organization to standardize on.

Policy A – organization root tag policy

```
{
    "tags": {
        "CostCenter": {
            "tag_key": {
                "@@assign": "CostCenter",
                "@@operators_allowed_for_child_policies": ["@@none"]
            }
        },
        "Project": {
            "tag_key": {
                "@@assign": "Project",
                "@@operators_allowed_for_child_policies": ["@@none"]
            }
        }
    }
}
```

This tag policy defines two tag keys: `CostCenter` and `Project`. Attaching this tag policy to the organization root has the following effects:

- All accounts in your organization inherit this tag policy.
- All accounts in your organization must use the defined case treatment for compliance. Resources with `CostCenter` and `Project` tags are compliant. Resources with alternate case treatment for the tag key (for example, `costcenter`, `Costcenter`, or `COSTCENTER`) are noncompliant.
- The `@@operators_allowed_for_child_policies": ["@@none"]` lines lock down the tag keys. Tag policies that are attached lower in the organization tree (child policies) can't use value-setting operators to change the tag key, including its case treatment.
- As with all tag policies, untagged resources or tags that aren't defined in the tag policy aren't evaluated for compliance with the tag policy.

AWS recommends that you use this example as a guide in creating a similar tag policy for tag keys that you want to use. Attach it to the organization root. Then create a tag policy similar to the next example, which only defines the acceptable values for the defined tag keys.

Next step: Define values

Assume that you attached the previous tag policy to the organization root. Next, you can create a tag policy like the following and attach it to an account. This policy defines acceptable values for the `CostCenter` and `Project` tag keys.

Policy B – account tag policy

```
{
    "tags": {
        "CostCenter": {
            "tag_value": {
                "@@assign": [
                    "Production",
                    "Test"
                ]
            }
        },
        "Project": {
            "tag_value": {
                "@@assign": [
                    "A",
                    "B"
                ]
            }
        }
    }
}
```

If you attach Policy A to the organization root and Policy B to an account, the policies combine to create the following effective tag policy for the account:

Policy A + Policy B = effective tag policy for account

```
{
    "tags": {
        "Project": {
            "tag_value": [
                "A",
                "B"
            ],
            "tag_key": "Project"
        },
        "CostCenter": {
            "tag_value": [
                "Production",
                "Test"
            ],
            "tag_key": "CostCenter"
        }
    }
}
```

For more information on policy inheritance, including examples of how the inheritance operators work and example effective tag policies, see Understanding policy inheritance (p. 91).

Example 2: Prevent use of a tag key

To prevent the use of a tag key, you can attach a tag policy like the following to an organization entity.

This example policy specifies that no values are acceptable for the `Color` tag key. It also specifies that no operators (p. 95) are allowed in child tag policies. Therefore, any `Color` tags on resources in affected accounts are considered non-compliant. However, the `enforced_for` option actually prevents affected accounts from tagging ***only*** Amazon DynamoDB tables with the `Color` tag.

```
{
    "tags": {
        "Color": {
            "tag_key": {
                "@@operators_allowed_for_child_policies": [
                    "@@none"
                ],
                "@@assign": "Color"
            },
            "tag_value": {
                "@@operators_allowed_for_child_policies": [
                    "@@none"
                ],
                "@@assign": []
            },
            "enforced_for": {
                "@@assign": [
                    "dynamodb:table"
                ]
            }
        }
    }
}
```

Supported Regions

Tag policy features are available in the following Regions:

Region name	Region parameter
US East (Ohio) Region	`us-east-2`
US East (N. Virginia) Region[1]	**`us-east-1`**
US West (N. California) Region	`us-west-1`
US West (Oregon) Region	`us-west-2`
Africa (Cape Town) Region[2]	`af-south-1`
Asia Pacific (Hong Kong) Region[2]	`ap-east-1`
Asia Pacific (Mumbai) Region	`ap-south-1`
Asia Pacific (Osaka) Region	`ap-northeast-3`
Asia Pacific (Seoul) Region	`ap-northeast-2`
Asia Pacific (Singapore) Region	`ap-southeast-1`
Asia Pacific (Sydney) Region	`ap-southeast-2`

Region name	Region parameter
Asia Pacific (Jakarta) Region	`ap-southeast-3`
Asia Pacific (Tokyo) Region	`ap-northeast-1`
Asia Pacific (Osaka) Region	`ap-northeast-3`
Canada (Central) Region	`ca-central-1`
Europe (Frankfurt) Region	`eu-central-1`
Europe (Milan) Region[2]	`eu-south-1`
Europe (Ireland) Region	`eu-west-1`
Europe (London) Region	`eu-west-2`
Europe (Paris) Region	`eu-west-3`
Europe (Stockholm) Region	`eu-north-1`
Middle East (Bahrain) Region[2]	`me-south-1`
South America (São Paulo) Region	`sa-east-1`

[1]You must specify the `us-east-1` Region when calling the following Organizations operations:

- DeletePolicy
- DisablePolicyType
- EnablePolicyType
- Any other operations on an organization root, such as ListRoots.

You must also specify the `us-east-1` Region when calling the following Resource Groups Tagging API operations that are part of the tag policies feature:

- DescribeReportCreation
- GetComplianceSummary
- GetResources
- StartReportCreation

> **Note**
> To evaluate organization-wide compliance with tag policies, you must also have access to an Amazon S3 bucket in the US East (N. Virginia) Region for report storage. For more information, see Amazon S3 Bucket Policy for Storing Report.

[2]These Regions must be manually enabled. To learn more about enabling and disabling AWS Regions, see Managing AWS Regions in the *AWS General Reference*. The Resource Groups console isn't available in these Regions.

Tagging AWS Organizations resources

A *tag* is a custom attribute label that you add to an AWS resource to make it easier to identify, organize, and search for resources. Each tag has two parts:

- A *tag key* (for example, `CostCenter`, `Environment`, or `Project`). Tag keys can be up to 128 characters in length and are case sensitive.
- A *tag value* (for example, `111122223333` or `Production`). Tag values can be up to 256 characters in length, and like tag keys, are case sensitive. You can set the value of a tag to an empty string, but you can't set the value of a tag to null. Omitting the tag value is the same as using an empty string.

For more information about what characters are allowed in a tag key or value, see the Tags parameter of the Tag API in the *Resource Groups Tagging API Reference*.

You can use tags to categorize resources by purpose, owner, environment, or other criteria. For more information, see AWS Tagging Strategies.

> **Tip**
> Use tag policies (p. 187) to help standardize your implementation of tags across the resources in your organization's accounts.

Currently, AWS Organizations supports the following tagging operations when you are logged in to the management account:

- You can add tags to the following organization resources:
  - AWS accounts
  - Organizational units
  - The organization's root
  - Policies

You can add tags at the following times:

- When you create the resource (p. 221) — Specify the tags in either the Organizations console, or use the `Tags` parameter with one of the `Create` API operations. This isn't applicable to the organization's root.
- After you create the resource (p. 221) — Use the Organizations console, or call the `TagResource` operation.

You can view the tags on any of the taggable resources in AWS Organizations by using the console or by calling the `ListTagsForResource` operation.

You can remove tags from a resource by specifying the keys to remove by using the console or by calling the `UntagResource` operation.

Using tags

Tags help you to organize your resources by enabling you to group them by things by whatever categories are useful to you. For example, you can assign a "Department" tag that tracks the owning department. You can assign an "Environment" tag to track whether a given resource is part of your alpha, beta, gamma, or production environments. Tags can help you to control who can access and manage the components that make up your organization (p. 327).

- You can enforce tagging standards on your resources by using tag policies (p. 187).
- Tags can help you to control who can access and manage the components that make up your organization (p. 327).

Adding, updating, and removing tags

When you sign in to your organization's management account, you can add tags to the resources in your organization.

Adding tags to a resource when you create it

Minimum permissions
To add tags to a resource when you create it, you need the following permissions:

- Permission to create a resource of the specified type
- `organizations:TagResource`
- `organizations:ListTagsForResource` – required only when using the Organizations console

You can include tag keys and values that are attached to the following resources as you create them.

- AWS account
  - Created account (p. 60)
  - Invited account (p. 52)
- Organizational unit (OU) (p. 78)
- Policy
  - AI services opt-out policy (p. 140)
  - Backup policy (p. 158)
  - Service control policy (p. 107)
  - Tag policy (p. 193)

The organization root is created when you initially create the organization, so you can only add tags to it as an existing resource.

Adding or updating tags for an existing resource

You can also add new tags or update the values of tags attached to existing resources.

Minimum permissions
To add or update tags to resources in your organization, you need the following permissions:

- `organizations:TagResource`
- `organizations:ListTagsForResource` – required only when using the Organizations console

To remove tags from resources in your organization, you need the following permissions:

- `organizations:UntagResource`

AWS Management Console

To add, update, or remove tags for an existing resource

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. Navigate to and choose the account, Root, OU, or policy, and click on its name to open its detail page.
3. On the **Tags** tab, choose **Manage tags**.
4. You can add new tags, modify the values of existing tags, or remove tags.

   To add a tag, choose **Add tag**, and then enter a **Key** and, optionally, a **Value** for the tag.

   To remove a tag, choose **Remove**.

   Tag keys and values are case sensitive. Use the capitalization that you want to standardize on. You must also comply with the requirements of any tag policies that apply.
5. Repeat the previous step as many times as you need.
6. Choose **Save changes**.

AWS CLI & AWS SDKs

To add or update tags to an existing resource

You can use one of the following commands to add tags to the taggable resources in your organization:

- AWS CLI: tag-resource
- AWS SDKs: TagResource

To delete tags from a resource in your organization

You can use one of the following commands to delete tags:

- AWS CLI: untag-resource
- AWS SDKs: UntagResource

Using AWS Organizations with other AWS services

You can use *trusted access* to enable a supported AWS service that you specify, called the *trusted service*, to perform tasks in your organization and its accounts on your behalf. This involves granting permissions to the trusted service but does *not* otherwise affect the permissions for IAM users or roles. When you enable access, the trusted service can create an IAM role called a *service-linked role* in every account in your organization whenever that role is needed. That role has a permissions policy that allows the trusted service to do the tasks that are described in that service's documentation. This enables you to specify settings and configuration details that you would like the trusted service to maintain in your organization's accounts on your behalf. The trusted service only creates service-linked roles when it needs to perform management actions on accounts, and not necessarily in all accounts of the organization.

> **Important**
> We ***strongly recommend*** that you enable and disable trusted access by using ***only*** the trusted service's console, or its AWS CLI or API operation equivalents. This lets the trusted service perform any required initialization when enabling trusted access, such as creating any required resources and any required clean up of resources when disabling trusted access.
> For information about how to enable or disable trusted service access to your organization using the trusted service, see the **Learn more** link under the **Supports Trusted Access** column at AWS services that you can use with AWS Organizations (p. 227).
> If you disable access by using the Organizations console, CLI commands, or API operations, it causes the following actions to occur:
>
> - The service can no longer create a service-linked role in the accounts in your organization. This means that the service can't perform operations on your behalf on any new accounts in your organization. The service can still perform operations in older accounts until the service completes its clean-up from AWS Organizations.
> - The service can no longer perform tasks in the member accounts in the organization, unless those operations are explicitly permitted by the IAM policies that are attached to your roles. This includes any data aggregation from the member accounts to the management account, or to a delegated administrator account, where relevant.
> - Some services detect this and clean up any remaining data or resources related to the integration, while other services stop accessing the organization but leave any historical data and configuration in place to support a possible re-enabling of the integration.
>
> Instead, using the other service's console or commands to disable the integration ensures that the other service can clean up any resources that are required only for the integration. How the service cleans up its resources in the organization's accounts depends on that service. For more information, see the documentation for the other AWS service.

Permissions required to enable trusted access

Trusted access requires permissions for two services: AWS Organizations and the trusted service. To enable trusted access, choose one of the following scenarios:

- If you have credentials with permissions in both AWS Organizations and the trusted service, enable access by using the tools (console or AWS CLI) provided by the trusted service. This allows the service to enable trusted access in AWS Organizations on your behalf and to create any resources required for the service to operate in your organization.

The minimum permissions for these credentials are the following:

  - `organizations:EnableAWSServiceAccess`. You can also use the `organizations:ServicePrincipal` condition key with this operation to limit requests that those operations make to a list of approved service principal names. For more information, see Condition keys (p. 320).
  - `organizations:ListAWSServiceAccessForOrganization` – Required if you use the AWS Organizations console.
  - The minimum permissions that are required by the trusted service depend on the service. For more information, see the trusted service's documentation.

- If one person has credentials with permissions in AWS Organizations but someone else has credentials with permissions in the trusted service, perform these steps in the following order:

  1. The person who has credentials with permissions in AWS Organizations should use the AWS Organizations console, the AWS CLI, or an AWS SDK to enable trusted access for the trusted service. This grants permission to the other service to perform its required configuration in the organization when the following step (step 2) is performed.

     The minimum AWS Organizations permissions are the following:

     - `organizations:EnableAWSServiceAccess`
     - `organizations:ListAWSServiceAccessForOrganization` – Required only if you use the AWS Organizations console

     For the steps to enable trusted access in AWS Organizations, see How to enable or disable trusted access (p. 225).

  2. The person who has credentials with permissions in the trusted service enables that service to work with AWS Organizations. This instructs the service to perform any required initialization, such as creating any resources that are required for the trusted service to operate in the organization. For more information, see the service-specific instructions at AWS services that you can use with AWS Organizations (p. 227).

Permissions required to disable trusted access

When you no longer want to allow the trusted service to operate on your organization or its accounts, choose one of the following scenarios.

> **Important**
> Disabling trusted service access does ***not*** prevent users and roles with appropriate permissions from using that service. To completely block users and roles from accessing an AWS service, you can remove the IAM permissions that grant that access, or you can use service control policies (SCPs) (p. 104) in AWS Organizations.
> You can apply SCPs to only member accounts. SCPs don't apply to the management account. We recommend that you don't run services in the management account. (p. 24) Instead, run them in member accounts where you can control the security by using SCPs.

- If you have credentials with permissions in both AWS Organizations and the trusted service, disable access by using the tools (console or AWS CLI) that are available for the trusted service. The service then cleans up by removing resources that are no longer required and by disabling trusted access for the service in AWS Organizations on your behalf.

  The minimum permissions for these credentials are the following:

  - `organizations:DisableAWSServiceAccess`. You can also use the `organizations:ServicePrincipal` condition key with this operation to limit requests that those operations make to a list of approved service principal names. For more information, see Condition keys (p. 320).

  - `organizations:ListAWSServiceAccessForOrganization` – Required if you use the AWS Organizations console.
  - The minimum permissions required by the trusted service depend on the service. For more information, see the trusted service's documentation.
- If the credentials with permissions in AWS Organizations aren't the credentials with permissions in the trusted service, perform these steps in the following order:
  1. The person with permissions in the trusted service first disables access using that service. This instructs the trusted service to clean up by removing the resources required for trusted access. For more information, see the service-specific instructions at AWS services that you can use with AWS Organizations (p. 227).
  2. The person with permissions in AWS Organizations can then use the AWS Organizations console, AWS CLI, or an AWS SDK to disable access for the trusted service. This removes the permissions for the trusted service from the organization and its accounts.

     The minimum AWS Organizations permissions are the following:
     - `organizations:DisableAWSServiceAccess`
     - `organizations:ListAWSServiceAccessForOrganization` – Required only if you use the AWS Organizations console

     For the steps to disable trusted access in AWS Organizations, see How to enable or disable trusted access (p. 225).

How to enable or disable trusted access

If you have permissions only for AWS Organizations and you want to enable or disable trusted access to your organization on behalf of the administrator of the other AWS service, use the following procedure.

Important

We ***strongly recommend*** that you enable and disable trusted access by using ***only*** the trusted service's console, or its AWS CLI or API operation equivalents. This lets the trusted service perform any required initialization when enabling trusted access, such as creating any required resources and any required clean up of resources when disabling trusted access.

For information about how to enable or disable trusted service access to your organization using the trusted service, see the **Learn more** link under the **Supports Trusted Access** column at AWS services that you can use with AWS Organizations (p. 227).

If you disable access by using the Organizations console, CLI commands, or API operations, it causes the following actions to occur:

- The service can no longer create a service-linked role in the accounts in your organization. This means that the service can't perform operations on your behalf on any new accounts in your organization. The service can still perform operations in older accounts until the service completes its clean-up from AWS Organizations.
- The service can no longer perform tasks in the member accounts in the organization, unless those operations are explicitly permitted by the IAM policies that are attached to your roles. This includes any data aggregation from the member accounts to the management account, or to a delegated administrator account, where relevant.
- Some services detect this and clean up any remaining data or resources related to the integration, while other services stop accessing the organization but leave any historical data and configuration in place to support a possible re-enabling of the integration.

Instead, using the other service's console or commands to disable the integration ensures that the other service can clean up any resources that are required only for the integration. How the service cleans up its resources in the organization's accounts depends on that service. For more information, see the documentation for the other AWS service.

AWS Management Console

To enable trusted service access

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for the service that you want to enable, and choose its name.
3. Choose **Enable trusted access**.
4. In the confirmation dialog box, check the box to **Show the option to enable trusted access**, enter **`enable`** in the box, and then choose **Enable trusted access**.
5. If you are *enabling* access, tell the administrator of the other AWS service that they can now enable the other service to work with AWS Organizations.

To disable trusted service access

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for the service that you want to disable, and choose its name.
3. Wait until the administrator of the other service tells you that the service is disabled and that its resources have been cleaned up.
4. In the confirmation dialog box, enter **`disable`** in the box, and then choose **Disable trusted access**.

AWS CLI, AWS API

To enable or disable trusted service access

You can use the following AWS CLI commands or API operations to enable or disable trusted service access:

- AWS CLI: AWS organizations enable-aws-service-access
- AWS CLI: AWS organizations disable-aws-service-access
- AWS API: EnableAWSServiceAccess
- AWS API: DisableAWSServiceAccess

AWS Organizations and service-linked roles

AWS Organizations uses IAM service-linked roles to enable trusted services to perform tasks on your behalf in your organization's member accounts. When you configure a trusted service and authorize it to integrate with your organization, that service can request that AWS Organizations create a service-linked role in its member account. The trusted service does this asynchronously as needed and not necessarily in all accounts in the organization at the same time. The service-linked role has predefined IAM permissions that allow the trusted service to perform only specific tasks within that account. In general, AWS manages all service-linked roles, which means that you typically can't alter the roles or the attached policies.

To make all of this possible, when you create an account in an organization or you accept an invitation to join your existing account to an organization, AWS Organizations provisions the member account with a service-linked role named `AWSServiceRoleForOrganizations`. Only the AWS Organizations service itself can assume this role. The role has permissions that allow AWS Organizations to create service-linked roles for other AWS services. This service-linked role is present in all organizations.

Although we don't recommend it, if your organization has only consolidated billing features (p. 7) enabled, the service-linked role named AWSServiceRoleForOrganizations is never used, and you can delete it. If you later want to enable all features (p. 7) in your organization, the role is required, and you must restore it. The following checks occur when you begin the process to enable all features:

- **For each member account that was *invited to join* the organization** – The account administrator receives a request to agree to enable all features. To successfully agree to the request, the administrator must have both `organizations:AcceptHandshake` *and* `iam:CreateServiceLinkedRole` permissions if the service-linked role (`AWSServiceRoleForOrganizations`) doesn't already exist. If the `AWSServiceRoleForOrganizations` role already exists, the administrator needs only the `organizations:AcceptHandshake` permission to agree to the request. When the administrator agrees to the request, AWS Organizations creates the service-linked role if it doesn't already exist.
- **For each member account that was *created* in the organization** – The account administrator receives a request to recreate the service-linked role. (The administrator of the member account doesn't receive a request to enable all features because the administrator of the management account (formerly known as the "master account") is considered the owner of the created member accounts.) AWS Organizations creates the service-linked role when the member account administrator agrees to the request. The administrator must have both `organizations:AcceptHandshake` *and* `iam:CreateServiceLinkedRole` permissions to successfully accept the handshake.

After you enable all features in your organization, you no longer can delete the `AWSServiceRoleForOrganizations` service-linked role from any account.

> **Important**
> AWS Organizations SCPs never affect service-linked roles. These roles are exempt from any SCP restrictions.

AWS services that you can use with AWS Organizations

With AWS Organizations you can perform account management activities at scale by consolidating multiple AWS accounts into a single organization. Consolidating accounts simplifies how you use other AWS services. You can leverage the multi-account management services available in AWS Organizations with select AWS services to perform tasks on all accounts that are members of your organization.

The following table lists AWS services that you can use with AWS Organizations, and the benefit of using each service on an organization-wide level.

Trusted Access – You can enable a compatible AWS service to perform operations across all of the AWS accounts in your organization. For more information, see Using AWS Organizations with other AWS services (p. 223).

Delegated Administrator – A compatible AWS service can register an AWS member account in the organization as an administrator for the organization's accounts in that service.

AWS service	Benefits of using with AWS Organizatio	Supports Trusted Access	Supports Delegated Administrator	
AWS Account Management (p. 248)	You can create, update,	✓Yes	✓Yes Learn more (p. 249)	

AWS service	Benefits of using with AWS Organizatio	Supports Trusted Access	Supports Delegated Administrator	
Manage the details and metadata for all of the AWS accounts for your organization.	and delete the alternate contact information for all of the accounts in your organization.	Learn more (p. 248)		
AWS Artifact (p. 250) Download AWS security compliance reports such as ISO and PCI reports.	You can accept agreements on behalf of all accounts within your organization.	✓Yes Learn more (p. 251)	⊗No	
AWS Audit Manager (p. 252) Automate the continuous collection of evidence to help you audit your use of cloud services.	Continuously audit your AWS use across multiple accounts in your organization to simplify how you assess risk and compliance.	✓Yes Learn more (p. 253)	✓Yes Learn more (p. 254)	

AWS service	Benefits of using with AWS Organizatio	Supports Trusted Access	Supports Delegated Administrator	
AWS Backup (p. 255) Manage and monitor backups across all of the accounts in your organization.	You can configure and manage backup plans for your entire organization, or for groups of accounts in your organization units (OUs). You can centrally monitor backups for all of your accounts.	✓Yes Learn more (p. 256)	⊗No	
AWS CloudFormation Stacksets (p. 256) Create, update, or delete stacks across multiple accounts and Regions with a single operation.	A user in the management account or a delegated administrator account can create a stack set with service-managed permissions that deploys stack instances to accounts in your organization.	✓Yes Learn more (p. 258)	✓Yes Learn more (p. 259)	

AWS service	Benefits of using with AWS Organizatio	Supports Trusted Access	Supports Delegated Administrator	
AWS CloudTrail (p. 259) Enable governance, compliance, and operational and risk auditing of your account.	A user in a management account can create an organization trail that logs all events for all accounts in that organization.	✓Yes Learn more (p. 260)	⊗No	
Amazon CloudWatch Events Monitor your AWS resources and the applications that you run on AWS in real time.	You can enable sharing of all CloudWatch Events across all accounts in your organization. For more information, see Sending and Receiving Events Between AWS accounts in the *Amazon CloudWatch Events User Guide*.	⊗No	⊗No	

AWS service	Benefits of using with AWS Organizatio	Supports Trusted Access	Supports Delegated Administrator	
AWS Compute Optimizer (p. 261) Get AWS compute optimization recommendations.	You can analyze all resources that are in your organization's accounts to get optimization recommendations. For more information, see Accounts Supported by Compute Optimizer in the *AWS Compute Optimizer User Guide*.	✓Yes Learn more (p. 263)	✓Yes Learn more (p. 263)	

AWS service	Benefits of using with AWS Organizatio	Supports Trusted Access	Supports Delegated Administrator	
AWS Config (p. 264) Assess, audit, and evaluate the configurations of your AWS resources.	You can get an organization-wide view of your compliance status. You can also use AWS Config API operations to manage AWS Config rules and conformance packs across all AWS accounts in your organization. You can use a delegated administrator account to aggregate resource configuration and compliance data from all member accounts of an organization in AWS Organizations. For more information, see Register a delegated administrator in the AWS Config Developer Guide.	✓Yes Learn more (p. 266)	✓Yes Learn more: Config rules Conformance packs Multi-account multi-region data aggregation	

AWS service	Benefits of using with AWS Organizatio	Supports Trusted Access	Supports Delegated Administrator	
AWS Control Tower (p. 266) Set up and govern a secure, compliant, multi-account AWS environment.	You can set up a landing zone, a multi-account environment for all of your AWS resources. This environment includes an organization and organization entities. You can use this environment to enforce compliance regulations on all of your AWS accounts. For more information, see How AWS Control Tower and Manage Accounts Through AWS Organizations in the *AWS Control Tower User Guide*.	✓Yes Learn more	⊗No	

AWS service	Benefits of using with AWS Organizatio	Supports Trusted Access	Supports Delegated Administrator	
Amazon Detective (p. 268) Generate visualizations from your log data to analyze, investigate, and quickly identify the root cause of security findings or suspicious activities.	You can integrate Amazon Detective with AWS Organizations to ensure that your Detective behavior graph provides visibility into the activity for all of your organization accounts.	✓Yes Learn more (p. 268)	✓Yes Learn more (p. 269)	
Amazon DevOps Guru (p. 270) Analyze operational data and application metrics and events to identify behaviors that deviate from normal operating patterns. Users are notified when DevOps Guru detects an operational issue or risk.	You can integrate with AWS Organizations to manage insights from all accounts across your entire organization. You delegate an administrator to view, sort, and filter insights from all accounts to obtain organization-wide health of all monitored applications.	✓Yes Learn more (p. 271)	✓Yes Learn more (p. 272)	

AWS service	Benefits of using with AWS Organizatio	Supports Trusted Access	Supports Delegated Administrator	
AWS Directory Service (p. 273) Set up and run directories in the AWS Cloud or connect your AWS resources with an existing on-premises Microsoft Active Directory.	You can integrate AWS Directory Service with AWS Organizations for seamless directory sharing across multiple accounts and any VPC in a Region.	✓Yes Learn more (p. 274)	⊗No	
AWS Firewall Manager (p. 274) Centrally configure and manage firewall rules for web applications across your accounts and applications.	You can centrally configure and manage AWS WAF rules across the accounts in your organization.	✓Yes Learn more (p. 275)	✓Yes Learn more (p. 277)	

AWS service	Benefits of using with AWS Organizatio	Supports Trusted Access	Supports Delegated Administrator	
Amazon GuardDuty (p. 277) GuardDuty is a continuous security monitoring service that analyzes and processes information from a variety of data sources. It uses threat intelligence feeds and machine learning to identify unexpected and potentially unauthorized and malicious activity within your AWS environment.	You can designate a member account to view and manage GuardDuty for all of the accounts in your organization. Adding member accounts automatically enables GuardDuty for those accounts in the selected AWS Region. You can also automate GuardDuty activation for new accounts added to your organization. For more information, see GuardDuty and Organizations in the *Amazon GuardDuty User Guide*.	✓Yes Learn more (p. 278)	✓Yes Learn more (p. 279)	

AWS service	Benefits of using with AWS Organizatio	Supports Trusted Access	Supports Delegated Administrator	
AWS Health (p. 279) Get visibility into events that might affect your resource performance or availability issues for AWS services.	You can aggregate AWS Health events across accounts in your organization.	✓Yes Learn more (p. 280)	✗No	

AWS service	Benefits of using with AWS Organization	Supports Trusted Access	Supports Delegated Administrator	
AWS Identity and Access Management Securely control access to AWS resources.	You can use service last accessed data in IAM to help you better understand AWS activity across your organization. You can use this data to create and update service control policies (SCPs) (p. 104) that restrict access to only the AWS services that your organization's accounts use. For an example, see Using Data to Refine Permissions for an Organizational Unit in the *IAM User Guide.*	⊗ No	⊗ No	

AWS service	Benefits of using with AWS Organizatio	Supports Trusted Access	Supports Delegated Administrator	
IAM Access Analyzer Analyze resource-based policies in your AWS environment to identify any policies that grant access to a principal outside of your zone of trust.	You can designate a member account to be an administrator for IAM Access Analyzer. For more information, see Enabling Access Analyzer in the *IAM User Guide*.	✓Yes Learn more	✓Yes Learn more	
Amazon Inspector (p. 281) Automatically scan your AWS workloads for vulnerabilities to discover Amazon EC2 instances and container images that reside in Amazon ECR for software vulnerabilities and unintended network exposure.	Delegate an administrator to enable or disable scans for member accounts, view aggregated finding data from the entire organization, create and manage suppression rules. For more information, see Managing multiple accounts with AWS Organizations in the *Amazon Inspector User Guide*.	✓Yes Learn more (p. 282)	✓Yes Learn more (p. 283)	

AWS service	Benefits of using with AWS Organizatio	Supports Trusted Access	Supports Delegated Administrator	
AWS License Manager (p. 284) Streamline the process of bringing software licenses to the cloud.	You can enable cross-account discovery of computing resources throughout your organization.	✓ Yes Learn more (p. 285)	✓ Yes Learn more (p. 286)	

AWS service	Benefits of using with AWS Organizatio	Supports Trusted Access	Supports Delegated Administrator	
Amazon Macie (p. 286) Discovers and classifies your business-critical content using machine learning to help you meet data security and privacy requirements. It continuously evaluates your content stored in Amazon S3 and notifies you of potential issues.	You can configure Amazon Macie for all of the accounts in your organization to get a consolidated view of all of your data in Amazon S3, across all accounts from a designated Macie administrator account. You can configure Macie to automatically protect resources in new accounts as your organization grows. You are alerted to remediate policy misconfigurations across S3 buckets throughout your organization.	✓Yes Learn more (p. 287)	✓Yes Learn more (p. 288)	

AWS service	Benefits of using with AWS Organizations	Supports Trusted Access	Supports Delegated Administrator	
AWS Marketplace (p. 288) A curated digital catalog that you can use to find, buy, deploy, and manage third-party software, data, and services that you need to build solutions and run your businesses.	You can share licenses for your AWS Marketplace subscriptions and purchases across the accounts in your organization.	✓Yes Learn more (p. 289)	⊗No	
AWS Network Manager (p. 290) Enables you to centrally manage your AWS Cloud WAN core network and your AWS Transit Gateway network across AWS accounts, Regions, and on-premises locations.	You can centrally manage and monitor your global networks with transit gateways and their attached resources in multiple AWS accounts within your organization.	✓Yes Learn more (p. 292)	✓Yes Learn more (p. 292)	

AWS service	Benefits of using with AWS Organizatio	Supports Trusted Access	Supports Delegated Administrator	
AWS Resource Access Manager (p. 292) Share specified AWS resources that you own with other accounts.	You can share resources within your organization without exchanging additional invitations. Resources you can share include Route 53 Resolver rules, on-demand capacity reservations, and more. For information about sharing capacity reservations, see the *Amazon EC2 User Guide for Linux Instances* or the *Amazon EC2 User Guide for Windows Instances*. For a list of shareable resources, see Shareable Resources in the *AWS RAM User Guide*.	✓Yes Learn more (p. 293)	⊗No	

AWS service	Benefits of using with AWS Organizatio	Supports Trusted Access	Supports Delegated Administrator	
AWS Security Hub (p. 295) View your security state in AWS and check your environment against security industry standards and best practices.	You can automatically enable Security Hub for all of your organization's accounts, including new accounts as they are added. This increases the coverage for Security Hub checks and findings, which provides a more accurate picture of your overall security posture.	✓Yes Learn more (p. 295)	✓Yes Learn more (p. 296)	
Amazon S3 Storage Lens (p. 296) Get visibility into your Amazon S3 storage usage and activity metrics with actionable recommendations to optimize storage.	Configure Amazon S3 Storage Lens to gain visibility into Amazon S3 storage usage and activity trends, and recommendations for all member accounts in your organization.	✓Yes Learn more (p. 297)	✓Yes Learn more (p. 298)	

AWS service	Benefits of using with AWS Organizatio	Supports Trusted Access	Supports Delegated Administrator	
AWS Service Catalog (p. 298) Create and manage catalogs of IT services that are approved for use on AWS.	You can share portfolios and copy products across accounts more easily, without sharing portfolio IDs.	✓Yes Learn more (p. 299)	✓Yes Learn more	
Service Quotas (p. 301) View and manage your service *quotas*, also referred to as *limits*, from a central location.	You can create a quota request template to automatically request a quota increase when accounts in your organization are created.	✓Yes Learn more (p. 302)	✗No	
AWS IAM Identity Center (successor to AWS Single Sign-On) (p. 302) Provide single sign-on access for all of your accounts and cloud applications.	Users can sign in to the AWS access portal with their corporate credentials and access resources in their assigned management account or member accounts.	✓Yes Learn more (p. 303)	✓Yes Learn more (p. 305)	

AWS service	Benefits of using with AWS Organizatio	Supports Trusted Access	Supports Delegated Administrator	
AWS Systems Manager (p. 305) Enable visibility and control of your AWS resources.	You can synchronize operations data across all AWS accounts in your organization by using Systems Manager Explorer. You can manage change templates, approvals and reporting for all member accounts in your organization from a delegated administrator account by using Systems Manager Change Manager.	✓Yes *(Systems Manager Explorer only)* Learn more (p. 306)	✓Yes Learn more (p. 308)	

AWS service	Benefits of using with AWS Organizatio	Supports Trusted Access	Supports Delegated Administrator	
Tag policies (p. 308) Use standardize tags across resources in your organization's accounts.	You can create tag policies to define tagging rules for specific resources and resource types and attach those policies to organization units and accounts to enforce those rules.	✓Yes Learn more (p. 308)	⊗No	
AWS Trusted Advisor (p. 309) Trusted Advisor inspects your AWS environment and makes recommendations when opportunities exist to save money, to improve system availability and performance, or to help close security gaps.	Run Trusted Advisor checks for all of the AWS accounts in your organization.	✓Yes Learn more (p. 310)	✓Yes Learn more (p. 311)	

AWS service	Benefits of using with AWS Organizatio	Supports Trusted Access	Supports Delegated Administrator	
AWS Well-Architected Tool (p. 311) The AWS Well-Architected Tool helps you document the state of your workloads and compares them to the latest AWS architectural best practices.	Enables both AWS WA Tool and Organizations customers to simplify the process of sharing AWS WA Tool resources with other members of their organization.	✓Yes Learn more (p. 312)	⊗No	
Amazon VPC IP Address Manager (IPAM) (p. 314) IPAM is a VPC feature that makes it easier for you to plan, track, and monitor IP addresses for your AWS workloads.	Monitor IP address usage throughout your organization and share IP address pools across member accounts.	✓Yes Learn more (p. 314)	✓Yes Learn more (p. 315)	

AWS Account Management and AWS Organizations

AWS Account Management helps you manage the account information and metadata for all of the AWS accounts in your organization. You can set, modify, or delete the alternate contact information for each of your organization's member accounts.

For more information, see Using AWS Account Management in your organization in the *AWS Account Management User Guide*.

Use the following information to help you integrate AWS Account Management with AWS Organizations.

To enable trusted access with Account Management

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

Account Management requires trusted access to AWS Organizations before you can designate a member account to be the delegated administrator for this service for your organization.

You can enable trusted access using only the Organizations tools.

You can enable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To enable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  You can run the following command to enable AWS Account Management as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal account.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

To disable trusted access with Account Management

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

Only an administrator in the AWS Organizations management account can disable trusted access with AWS Account Management.

You can disable trusted access using only the Organizations tools.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable AWS Account Management as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal account.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for Account Management

When you designate a member account to be a delegated administrator for the organization, users and roles from the designated account can manage the AWS account metadata for other member accounts in the organization. If you don't enable a delegated admin account, then these tasks can be performed only by the organization's management account. This helps you to separate management of the organization from management of your account details.

> **Minimum permissions**
> Only an IAM user or role in the Organizations management account can configure a member account as a delegated administrator for Account Management in the organization

For instruction about enabling a delegated administrator account for Account Management, see Enabling a delegated admin account for AWS Account Management in the *AWS Account Management Reference Guide*.

AWS CLI, AWS API

If you want to configure a delegated administrator account using the AWS CLI or one of the AWS SDKs, you can use the following commands:

- AWS CLI:

```
$  aws organizations register-delegated-administrator \
    --account-id 123456789012 \
    --service-principal account.amazonaws.com
```

- AWS SDK: Call the Organizations `RegisterDelegatedAdministrator` operation and the member account's ID number and identify the account service `principal account.amazonaws.com` as parameters.

AWS Artifact and AWS Organizations

AWS Artifact is a service that allows you to download AWS security compliance reports such as ISO and PCI reports. Using AWS Artifact, a user in the organization's management account can automatically accept agreements on behalf of all member accounts in an organization, even as new reports and accounts are added. Member account users can view and download agreements. For more information, see Managing an agreement for multiple accounts in AWS Artifact in the *AWS Artifact User Guide*.

Use the following information to help you integrate AWS Artifact with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows AWS Artifact to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between AWS Artifact and Organizations, or if you remove the member account from the organization.

Although you can delete or modify this role if you remove the member account from the organization, we do not recommend it.

Modifying the role is discouraged because it can lead to security issues such as the cross-service confused deputy. To learn more about protection against confused deputy, see Cross-service deputy prevention in the *AWS Artifact User Guide*.

- `AWSArtifactAccountSync`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by AWS Artifact grant access to the following service principals:

- `aws-artifact-account-sync.amazonaws.com`

Enabling trusted access with AWS Artifact

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

You can enable trusted access using only the Organizations tools.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for **AWS Artifact**, choose the service's name, and then choose **Enable trusted access**.
3. In the confirmation dialog box, enable **Show the option to enable trusted access**, enter **`enable`** in the box, and then choose **Enable trusted access**.
4. If you are the administrator of only AWS Organizations, tell the administrator of AWS Artifact that they can now enable that service using its console to work with AWS Organizations.

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

You can use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  You can run the following command to enable AWS Artifact as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal aws-artifact-account-sync.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access with AWS Artifact

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

Only an administrator in the AWS Organizations management account can disable trusted access with AWS Artifact.

You can disable trusted access using only the Organizations tools.

AWS Artifact requires trusted access with AWS Organizations to work with organization agreements. If you disable trusted access using AWS Organizations while you are using AWS Artifact for organization agreements, it stops functioning because it cannot access the organization. Any organization agreements that you accept in AWS Artifact remain, but can't be accessed by AWS Artifact. The AWS Artifact role that AWS Artifact creates remains. If you then re-enable trusted access, AWS Artifact continues to operate as before, without the need for you to reconfigure the service.

A standalone account that is removed from an organization no longer has access to any organization agreements.

You can disable trusted access by using either the AWS Organizations console, by running an Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for **AWS Artifact** and then choose the service's name.
3. Choose **Disable trusted access**.
4. In the confirmation dialog box, enter **`disable`** in the box, and then choose **Disable trusted access**.
5. If you are the administrator of only AWS Organizations, tell the administrator of AWS Artifact that they can now disable that service using its console or tools from working with AWS Organizations.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable AWS Artifact as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal aws-artifact-account-sync.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

AWS Audit Manager and AWS Organizations

AWS Audit Manager helps you continuously audit your AWS usage to simplify how you assess risk and compliance with regulations and industry standards. Audit Manager automates evidence collection to make it easier to assess if your policies, procedures, and activities are operating effectively. When it is time for an audit, Audit Manager helps you manage stakeholder reviews of your controls and helps you build audit-ready reports with much less manual effort.

When you integrate Audit Manager with AWS Organizations, you can gather evidence from a broader source by including multiple AWS accounts from your organization within the scope of your assessments.

For more information, see Enable AWS Organizations in the *Audit Manager User Guide*.

Use the following information to help you integrate AWS Audit Manager with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Audit Manager to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Audit Manager and Organizations, or if you remove the member account from the organization.

For more information about how Audit Manager uses this role, see Using service-linked roles in the *AWS Audit Manager Users Guide*.

- `AWSServiceRoleForAuditManager`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Audit Manager grant access to the following service principals:

- `auditmanager.amazonaws.com`

To enable trusted access with Audit Manager

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

Audit Manager requires trusted access to AWS Organizations before you can designate a member account to be the delegated administrator for your organization.

You can enable trusted access using either the AWS Audit Manager console or the AWS Organizations console.

> **Important**
> We strongly recommend that whenever possible, you use the AWS Audit Manager console or tools to enable integration with Organizations. This lets AWS Audit Manager perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can’t enable integration using the tools provided by AWS Audit Manager. For more information, see this note (p. 223).
> If you enable trusted access by using the AWS Audit Manager console or tools then you don’t need to complete these steps.

To enable trusted access using the Audit Manager console

For instructions about enabling trusted access, see Setting Up in the *AWS Audit Manager User Guide*.

> **Note**
> If you configure a delegated administrator using the AWS Audit Manager console, then AWS Audit Manager automatically enables trusted access for you.

You can enable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To enable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  You can run the following command to enable AWS Audit Manager as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal auditmanager.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

To disable trusted access with Audit Manager

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

Only an administrator in the AWS Organizations management account can disable trusted access with AWS Audit Manager.

You can disable trusted access using only the Organizations tools.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable AWS Audit Manager as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal auditmanager.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for Audit Manager

When you designate a member account to be a delegated administrator for the organization, users and roles from that account can perform administrative actions for Audit Manager that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of Audit Manager.

Minimum permissions
Only an IAM user or role in the Organizations management account with the following permission can configure a member account as a delegated administrator for Audit Manager in the organization:
`audit-manager:RegisterAccount`

For instruction about enabling a delegated administrator account for Audit Manager, see Setting Up in the *AWS Audit Manager User Guide*.

If you configure a delegated administrator using the AWS Audit Manager console, then Audit Manager automatically enables trusted access for you.

AWS CLI, AWS API

If you want to configure a delegated administrator account using the AWS CLI or one of the AWS SDKs, you can use the following commands:

- AWS CLI:

```
$  aws audit-manager register-account \
    --delegated-admin-account 123456789012
```

- AWS SDK: Call the `RegisterAccount` operation and provide `delegatedAdminAccount` as a parameter to delegate the administrator account.

AWS Backup and AWS Organizations

AWS Backup is a service that allows you to manage and monitor the AWS Backup jobs in your organization. Using AWS Backup, if you sign-in as a user in the organization's management account, you can enable organization-wide backup protection and monitoring. It helps you to achieve compliance by using backup policies (p. 154) to centrally apply AWS Backup plans to resources across all of the accounts in your organization. When you use both AWS Backup and AWS Organizations together, you can get the following benefits:

Protection

You can enable the backup policy type (p. 85) in your organization and then create backup policies (p. 158) to attach to the organization's root, OUs, or accounts. A backup policy combines an AWS Backup plan with the other details required to apply the plan automatically to your accounts.Policies that are directly attached to an account are merged with policies inherited (p. 94) from the organization's root and any parent OUs to create an *effective policy (p. 167)* that applies to the account. The policy includes the ID of an IAM role that has permissions to run AWS Backup on the resources in your accounts. AWS Backup uses the IAM role to perform the backup on your behalf as specified by the backup plan in the effective policy.

Monitoring

When you enable trusted access for AWS Backup (p. 225) in your organization, you can use the AWS Backup console to view details about the backup, restore, and copy jobs in any of the accounts in your organization. For more information, see Monitor your backup jobs in the *AWS Backup Developer Guide*.

For more information about AWS Backup, see the *AWS Backup Developer Guide*.

Use the following information to help you integrate AWS Backup with AWS Organizations.

Enabling trusted access with AWS Backup

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

You can enable trusted access using either the AWS Backup console or the AWS Organizations console.

> **Important**
> We strongly recommend that whenever possible, you use the AWS Backup console or tools to enable integration with Organizations. This lets AWS Backup perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can’t enable integration using the tools provided by AWS Backup. For more information, see this note (p. 223).
> If you enable trusted access by using the AWS Backup console or tools then you don’t need to complete these steps.

To enabled trusted access using AWS Backup, see Enabling backup in multiple AWS accounts in the *AWS Backup Developer Guide*.

Disabling trusted access with AWS Backup

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

AWS Backup requires trusted access with AWS Organizations to enable monitoring of backup, restore, and copy jobs across your organization's accounts. If you disable trusted access AWS Backup, you lose the ability to view jobs outside of the current account. The AWS Backup role that AWS Backup creates remains. If you later re-enable trusted access, AWS Backup continues to operate as before, without the need for you to reconfigure the service.

You can disable trusted access using only the Organizations tools.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable AWS Backup as a trusted service with Organizations.

```
$ aws organizations disable-aws-service-access \
    --service-principal backup.amazonaws.com
```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

AWS CloudFormation StackSets and AWS Organizations

AWS CloudFormation StackSets enables you to create, update, or delete stacks across multiple AWS accounts and AWS Regions with a single operation. StackSets integration with AWS Organizations enables you to create stack sets with service-managed permissions, using a service-linked role that has the relevant permission in each member account. This lets you deploy stack instances to member accounts in your organization. You don't have to create the necessary AWS Identity and Access Management roles; StackSets creates the IAM role in each member account on your behalf. You can also choose to enable automatic deployments to accounts that are added to your organization in the future.

With trusted access between StackSets and Organizations enabled, the management account has permissions to create and manage stack sets for your organization. The management account can register up to five member accounts as delegated administrators. With trusted access enabled, delegated administrators also have permissions to create and manage stack sets for your organization. Stack sets with service-managed permissions are created in the management account, including stack sets that are created by delegated administrators.

> **Important**
> Delegated administrators have full permissions to deploy to accounts in your organization. The management account cannot limit delegated administrator permissions to deploy to specific OUs or to perform specific stack set operations.

For more information about integrating StackSets with Organizations, see Working with AWS CloudFormation StackSets in the *AWS CloudFormation User Guide*.

Use the following information to help you integrate AWS CloudFormation StackSets with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows AWS CloudFormation Stacksets to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between AWS CloudFormation Stacksets and Organizations, or if you remove the member account from the organization.

- Management account: `AWSServiceRoleForCloudFormationStackSetsOrgAdmin`

To create the service-linked role `AWSServiceRoleForCloudFormationStackSetsOrgMember` for the member accounts in your organization, you need to create a stack set in the management account first. This creates a stack set instance, which then creates the role in the member account.

- Member accounts: `AWSServiceRoleForCloudFormationStackSetsOrgMember`

For more details about creating stack sets, see Working with AWS CloudFormation StackSets in the *AWS CloudFormation User Guide*.

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by AWS CloudFormation Stacksets grant access to the following service principals:

- Management account: `stacksets.cloudformation.amazonaws.com`

  You can modify or delete this role only if you disabled trusted access between StackSets and Organizations.
- Member accounts: `member.org.stacksets.cloudformation.amazonaws.com`

You can modify or delete this role from an account only if you first disable trusted access between StackSets and Organizations, or if you first remove the account from the target organization or organizational unit (OU).

Enabling trusted access with AWS CloudFormation Stacksets

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

Only an administrator in the Organizations management account has permissions to enable trusted access with another AWS service. You can enable trusted access using either the AWS CloudFormation console or the Organizations console.

You can enable trusted access using only AWS CloudFormation StackSets.

To enable trusted access using the AWS CloudFormation Stacksets console, see Enable Trusted Access with AWS Organizations in the AWS CloudFormation User Guide.

Disabling trusted access with AWS CloudFormation Stacksets

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

Only an administrator in an Organizations management account has permissions to disable trusted access with another AWS service. You can disable trusted access only by using the Organizations console. If you disable trusted access with Organizations while you are using StackSets, all previously created stack instances are retained. However, stack sets deployed using the service-linked role's permissions can no longer perform deployments to accounts managed by Organizations.

You can disable trusted access using either the AWS CloudFormation console or the Organizations console.

> **Important**
> If you disable trusted access programmatically (e.g with AWS CLI or with an API), be aware that this will remove the permission. It is better to disable trusted access with the AWS CloudFormation console.

You can disable trusted access by using either the AWS Organizations console, by running an Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **Services** page, find the row for **AWS CloudFormation StackSets** and then choose the service’s name.
3. Choose **Disable trusted access**.
4. In the confirmation dialog box, enter **`disable`** in the box, and then choose **Disable trusted access**.
5. If you are the administrator of only AWS Organizations, tell the administrator of AWS CloudFormation StackSets that they can now disable that service using its console or tools from working with AWS Organizations.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable AWS CloudFormation StackSets as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal stacksets.cloudformation.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for AWS CloudFormation Stacksets

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for AWS CloudFormation Stacksets that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of AWS CloudFormation Stacksets.

For instructions on how to designate a member account as a delegated administrator of AWS CloudFormation Stacksets in the organization, see Register a delegated administrator in the *AWS CloudFormation User Guide*.

AWS CloudTrail and AWS Organizations

AWS CloudTrail is an AWS service that helps you enable governance, compliance, and operational and risk auditing of your AWS account. Using AWS CloudTrail, a user in a management account can create an organization trail that logs all events for all AWS accounts in that organization. Organization trails are automatically applied to all member accounts in the organization. Member accounts can see the organization trail, but can't modify or delete it. By default, member accounts don't have access to the log files for the organization trail in the Amazon S3 bucket. This helps you uniformly apply and enforce your event logging strategy across the accounts in your organization.

For more information, see Creating a Trail for an Organization in the *AWS CloudTrail User Guide*.

Use the following information to help you integrate AWS CloudTrail with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows CloudTrail to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between CloudTrail and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForCloudTrail`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by CloudTrail grant access to the following service principals:

- `cloudtrail.amazonaws.com`

Enabling trusted access with CloudTrail

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

You can enable trusted access using either the AWS CloudTrail console or the AWS Organizations console.

> **Important**
> We strongly recommend that whenever possible, you use the AWS CloudTrail console or tools to enable integration with Organizations. This lets AWS CloudTrail perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by AWS CloudTrail. For more information, see this note (p. 223).
> If you enable trusted access by using the AWS CloudTrail console or tools then you don't need to complete these steps.

You must sign in with your AWS Organizations management account to create an organization trail. If you create the trail from the AWS CloudTrail console, trusted access is configured automatically for you. If you choose to create an organization trail using the AWS CLI or the AWS API, you must manually configure trusted access. For more information, see Enabling CloudTrail as a trusted service in AWS Organizations in the *AWS CloudTrail User Guide.*

You can enable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To enable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  You can run the following command to enable AWS CloudTrail as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal cloudtrail.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access with CloudTrail

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

AWS CloudTrail requires trusted access with AWS Organizations to work with organization trails. If you disable trusted access using AWS Organizations while you're using AWS CloudTrail for organization trails, the trails stop functioning for member accounts because CloudTrail can't access the organization. The organization trails remain, as does the `AWSServiceRoleForCloudTrail` role created for integration between CloudTrail and AWS Organizations. If you re-enable trusted access, CloudTrail continues to operate as before, without the need for you to reconfigure the trails.

Only an administrator in the AWS Organizations management account can disable trusted access with AWS CloudTrail.

You can disable trusted access using only the Organizations tools.

You can disable trusted access by using either the AWS Organizations console, by running an Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for **AWS CloudTrail** and then choose the service's name.
3. Choose **Disable trusted access**.
4. In the confirmation dialog box, enter **`disable`** in the box, and then choose **Disable trusted access**.
5. If you are the administrator of only AWS Organizations, tell the administrator of AWS CloudTrail that they can now disable that service using its console or tools from working with AWS Organizations.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable AWS CloudTrail as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal cloudtrail.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

AWS Compute Optimizer and AWS Organizations

AWS Compute Optimizer is a service that analyzes the configuration and utilization metrics of your AWS resources. Resource examples include Amazon Elastic Compute Cloud (Amazon EC2) instances and Auto Scaling groups. Compute Optimizer reports whether your resources are optimal and generates optimization recommendations to reduce the cost and improve the performance of your workloads. For more information about Compute Optimizer, see the AWS Compute Optimizer User Guide.

Use the following information to help you integrate AWS Compute Optimizer with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Compute Optimizer to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Compute Optimizer and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForComputeOptimizer`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Compute Optimizer grant access to the following service principals:

- `compute-optimizer.amazonaws.com`

Enabling trusted access with Compute Optimizer

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

You can enable trusted access using either the AWS Compute Optimizer console or the AWS Organizations console.

> **Important**
> We strongly recommend that whenever possible, you use the AWS Compute Optimizer console or tools to enable integration with Organizations. This lets AWS Compute Optimizer perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can’t enable integration using the tools provided by AWS Compute Optimizer. For more information, see this note (p. 223).
> If you enable trusted access by using the AWS Compute Optimizer console or tools then you don’t need to complete these steps.

To enable trusted access using the Compute Optimizer console

You must sign in to the Compute Optimizer console using your organization's management account. Opt-in on behalf of your organization by following the instructions at Opting in your Account in the *AWS Compute Optimizer User Guide*.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **Services** page, find the row for **AWS Compute Optimizer**, choose the service’s name, and then choose **Enable trusted access**.
3. In the confirmation dialog box, enable **Show the option to enable trusted access**, enter **`enable`** in the box, and then choose **Enable trusted access**.

4. If you are the administrator of only AWS Organizations, tell the administrator of AWS Compute Optimizer that they can now enable that service using its console to work with AWS Organizations.

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

You can use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  You can run the following command to enable AWS Compute Optimizer as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal compute-optimizer.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access with Compute Optimizer

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

Only an administrator in the AWS Organizations management account can disable trusted access with AWS Compute Optimizer.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable AWS Compute Optimizer as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal compute-optimizer.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for Compute Optimizer

When you designate a member account to be a delegated administrator for the organization, users and roles from the designated account can manage the AWS account metadata for other member accounts in the organization. If you don't enable a delegated admin account, then these tasks can be performed only by the organization's management account. This helps you to separate management of the organization from management of your account details.

> **Minimum permissions**
> Only an IAM user or role in the Organizations management account can configure a member account as a delegated administrator for Compute Optimizer in the organization

For instructions about enabling a delegated administrator account for Compute Optimizer, see https://docs.aws.amazon.com/compute-optimizer/latest/ug/delegate-administrator-account.html in the *AWS Compute Optimizer User Guide*.

AWS CLI, AWS API

If you want to configure a delegated administrator account using the AWS CLI or one of the AWS SDKs, you can use the following commands:

- AWS CLI:

```
$  aws organizations register-delegated-administrator \
    --account-id 123456789012 \
    --service-principal compute-optimizer.amazonaws.com
```

- AWS SDK: Call the Organizations `RegisterDelegatedAdministrator` operation and the member account's ID number and identify the account service `principal account.amazonaws.com` as parameters.

Disabling a delegated administrator for Compute Optimizer

Only an administrator in the organization management account can configure a delegated administrator for Compute Optimizer.

To disable the delegated admin Compute Optimizer account using the Compute Optimizer console, see https://docs.aws.amazon.com/compute-optimizer/latest/ug/delegate-administrator-account.html in the *AWS Compute Optimizer User Guide*.

To remove a delegated administrator using the AWS AWS CLI, see deregister-delegated-administrator in the *AWS AWS CLI Command Reference*.

AWS Config and AWS Organizations

Multi-account, multi-region data aggregation in AWS Config enables you to aggregate AWS Config data from multiple accounts and AWS Regions into a single account. Multi-account, multi-region data aggregation is useful for central IT administrators to monitor compliance for multiple AWS accounts in the enterprise. An aggregator is a resource type in AWS Config that collects AWS Config data from multiple source accounts and Regions. Create an aggregator in the Region where you want to see the aggregated AWS Config data. While creating an aggregator, you can choose to add either individual account IDs or your organization. For more information about AWS Config, see the AWS Config Developer Guide.

You can also use AWS Config APIs to manage AWS Config rules across all AWS accounts in your organization. For more information, see Enabling AWS Config Rules Across All Accounts in Your Organization in the *AWS Config Developer Guide*.

Use the following information to help you integrate AWS Config with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is created in your organization's accounts when you enable trusted access. This role allows AWS Config to perform supported operations within the accounts in your organization.

- `AWSServiceRoleForConfig`

This role is created when you enable AWS Config in your organization by creating a multi-account aggregator. AWS Config asks you to select or create a role and for you to provide the name. There is no automatically generated name.

You can delete or modify this role only if you disable trusted access between AWS Config and Organizations, or if you remove the member account from the organization.

Enabling trusted access with AWS Config

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

You can enable trusted access using either the AWS Config console or the AWS Organizations console.

> **Important**
> We strongly recommend that whenever possible, you use the AWS Config console or tools to enable integration with Organizations. This lets AWS Config perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by AWS Config. For more information, see this note (p. 223).
> If you enable trusted access by using the AWS Config console or tools then you don't need to complete these steps.

To enable trusted access using the AWS Config console

To enable trusted access to AWS Organizations using AWS Config, create a multi-account aggregator and add the organization. For information on how to configure a multi-account aggregator, see Setting up an aggregator using the console in the *AWS Config Developer Guide*.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for **AWS Config**, choose the service's name, and then choose **Enable trusted access**.
3. In the confirmation dialog box, enable **Show the option to enable trusted access**, enter **`enable`** in the box, and then choose **Enable trusted access**.
4. If you are the administrator of only AWS Organizations, tell the administrator of AWS Config that they can now enable that service using its console to work with AWS Organizations.

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

You can use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  You can run the following command to enable AWS Config as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal config.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access with AWS Config

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

You can disable trusted access using only the Organizations tools.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable AWS Config as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal config.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

AWS Control Tower and AWS Organizations

AWS Control Tower offers a straightforward way to set up and govern an AWS multi-account environment, following prescriptive best practices. AWS Control Tower orchestration extends the capabilities of AWS Organizations. AWS Control Tower applies preventive and detective controls (guardrails) to help keep your organizations and accounts from divergence from best practices (drift).

AWS Control Tower orchestration extends the capabilities of AWS Organizations.

For more information, see the *AWS Control Tower user guide*.

Use the following information to help you integrate AWS Control Tower with AWS Organizations.

Roles needed for integration

The `AWSControlTowerExecution` role must be present in all enrolled accounts. It allows AWS Control Tower to manage your individual accounts and report information about them to your Audit and Log Archive accounts.

To learn more about roles used by AWS Control Tower, see How AWS Control Tower works with roles to create and manage accounts and Using Identity-Based Policies (IAM Policies) for AWS Control Tower.

Service principals used by AWS Control Tower

AWS Control Tower uses the `controltower.amazonaws.com` service principal.

Enabling trusted access with AWS Control Tower

AWS Control Tower uses trusted access to detect drift for preventive controls, and to track account and OU changes that cause drift.

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

You can enable trusted access using only the Organizations tools.

To enable trusted access from the Organizations console, choose **`Enable access`** next to **AWS Control Tower**.

You can enable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To enable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  You can run the following command to enable AWS Control Tower as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal controltower.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access with AWS Control Tower

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

You can disable trusted access using only the Organizations tools.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable AWS Control Tower as a trusted service with Organizations.

```
$ aws organizations disable-aws-service-access \
    --service-principal controltower.amazonaws.com
```

This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Amazon Detective and AWS Organizations

Amazon Detective uses your log data to generate visualizations that allow you to analyze, investigate, and identify the root cause of security findings or suspicious activity.

Using AWS Organizations allows you to ensure that your Detective behavior graph provides visibility into the activity for all of your organization accounts.

When you grant trusted access to Detective, the Detective service can react automatically to changes in the organization membership. The delegated administrator can enable any organization account as a member account in the behavior graph. Detective also can automatically enable new organization accounts as member accounts. Organization accounts cannot disassociate themselves from the behavior graph.

For more information, see Using Amazon Detective in your organization in the *Amazon Detective Administration Guide*.

Use the following information to help you integrate Amazon Detective with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Detective to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Detective and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForDetective`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Detective grant access to the following service principals:

- `detective.amazonaws.com`

To enable trusted access with Detective

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

> **Note**
> When you designate a delegated administrator for Amazon Detective, Detective automatically enables trusted access for Detective for your organization.

Detective requires trusted access to AWS Organizations before you can designate a member account to be the delegated administrator for this service for your organization.

You can enable trusted access using only the Organizations tools.

You can enable trusted access by using the AWS Organizations console.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for **Amazon Detective**, choose the service's name, and then choose **Enable trusted access**.
3. In the confirmation dialog box, enable **Show the option to enable trusted access**, enter **`enable`** in the box, and then choose **Enable trusted access**.
4. If you are the administrator of only AWS Organizations, tell the administrator of Amazon Detective that they can now enable that service using its console to work with AWS Organizations.

To disable trusted access with Detective

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

Only an administrator in the AWS Organizations management account can disable trusted access with Amazon Detective.

You can disable trusted access using only the Organizations tools.

You can disable trusted access by using the AWS Organizations console.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for **Amazon Detective** and then choose the service's name.
3. Choose **Disable trusted access**.
4. In the confirmation dialog box, enter **`disable`** in the box, and then choose **Disable trusted access**.
5. If you are the administrator of only AWS Organizations, tell the administrator of Amazon Detective that they can now disable that service using its console or tools from working with AWS Organizations.

Enabling a delegated administrator account for Detective

The delegated administrator account for Detective is the administrator account for a Detective behavior graph. The delegated administrator determines which organization accounts to enable and disable as member accounts in that behavior graph. The delegated administrator can configure Detective to automatically enable new organization accounts as member accounts as they are added to the organization. For information on how a delegated administrator manages organization accounts, see Managing organization accounts as member accounts in the *Amazon Detective Administration Guide*.

Only an administrator in the organization management account can configure a delegated administrator for Detective.

You can specify a delegated administrator account from the Detective console or API, or by using the Organizations CLI or SDK operation.

Minimum permissions

Only an IAM user or role in the Organizations management account can configure a member account as a delegated administrator for Detective in the organization

To configure a delegated administrator using the Detective console or API, see Designating a Detective administrator account for an organization in the *Amazon Detective Administration Guide*.

AWS CLI, AWS API

If you want to configure a delegated administrator account using the AWS CLI or one of the AWS SDKs, you can use the following commands:

- AWS CLI:

```
$ aws organizations register-delegated-administrator \
    --account-id 123456789012 \
    --service-principal detective.amazonaws.com
```

- AWS SDK: Call the Organizations `RegisterDelegatedAdministrator` operation and the member account's ID number and identify the account service `principal account.amazonaws.com` as parameters.

Disabling a delegated administrator for Detective

You can remove the delegated administrator using either the Detective console or API, or by using the Organizations `DeregisterDelegatedAdministrator` CLI or SDK operation. For information on how to remove a delegated administrator using the Detective console or API, or the Organizations API, see Designating a Detective administrator account for an organization in the *Amazon Detective Administration Guide*.

Amazon DevOps Guru and AWS Organizations

Amazon DevOps Guru analyzes operational data and application metrics and events to identify behaviors that deviate from normal operating patterns. Users are notified when DevOps Guru detects an operational issue or risk.

Using DevOps Guru enables multi-account support with AWS Organizations, so you can designate a member account to manage insights across your entire organization. This delegated administrator can then view, sort, and filter insights from all accounts within your organization to develop a holistic view of the health of all monitored applications within your organization without the need for any additional customization.

For more information, see Monitor accounts across your organization in the *Amazon DevOps Guru User Guide*.

Use the following information to help you integrate Amazon DevOps Guru with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows DevOps Guru to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between DevOps Guru and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForDevOpsGuru`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by DevOps Guru grant access to the following service principals:

- `devops-guru.amazonaws.com`

For more information, see Using service-linked roles for DevOps Guru in the *Amazon DevOps Guru User Guide*.

To enable trusted access with DevOps Guru

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

> **Note**
> When you designate a delegated administrator for Amazon DevOps Guru, DevOps Guru automatically enables trusted access for DevOps Guru for your organization.
> DevOps Guru requires trusted access to AWS Organizations before you can designate a member account to be the delegated administrator for this service for your organization.

> **Important**
> We strongly recommend that whenever possible, you use the Amazon DevOps Guru console or tools to enable integration with Organizations. This lets Amazon DevOps Guru perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by Amazon DevOps Guru. For more information, see this note (p. 223).

You can enable trusted access by using either the AWS Organizations console or the DevOps Guru console.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for **Amazon DevOps Guru**, choose the service's name, and then choose **Enable trusted access**.
3. In the confirmation dialog box, enable **Show the option to enable trusted access**, enter **`enable`** in the box, and then choose **Enable trusted access**.
4. If you are the administrator of only AWS Organizations, tell the administrator of Amazon DevOps Guru that they can now enable that service using its console to work with AWS Organizations.

DevOps Guru console

To enable trusted service access using the DevOps Guru console

1. Sign in as administrator in the management account and open DevOps Guru console: Amazon DevOps Guru console

2. Choose **Enable trusted access**.

To disable trusted access with DevOps Guru

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

Only an administrator in the AWS Organizations management account can disable trusted access with Amazon DevOps Guru.

You can disable trusted access using only the Organizations tools.

You can disable trusted access by using the AWS Organizations console.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for **Amazon DevOps Guru** and then choose the service's name.
3. Choose **Disable trusted access**.
4. In the confirmation dialog box, enter **`disable`** in the box, and then choose **Disable trusted access**.
5. If you are the administrator of only AWS Organizations, tell the administrator of Amazon DevOps Guru that they can now disable that service using its console or tools from working with AWS Organizations.

Enabling a delegated administrator account for DevOps Guru

The delegated administrator account for DevOps Guru can see the insights data from all the member accounts which are onboarded to DevOps Guru from the organization. For information on how a delegated administrator manages organization accounts, see Monitor accounts across your organization in the *Amazon DevOps Guru User Guide*.

Only an administrator in the organization management account can configure a delegated administrator for DevOps Guru.

You can specify a delegated administrator account from the DevOps Guru console, or by using the Organizations `RegisterDelegatedAdministrator` CLI or SDK operation.

Minimum permissions
Only an IAM user or role in the Organizations management account can configure a member account as a delegated administrator for DevOps Guru in the organization

DevOps Guru console

To configure a delegated administrator in the DevOps Guru console

1. Sign in as administrator in the management account and open DevOps Guru console: Amazon DevOps Guru console
2. Choose **Register delegated administrator**. You can choose either Management account or any member account as the delegated admin.

AWS CLI, AWS API

If you want to configure a delegated administrator account using the AWS CLI or one of the AWS SDKs, you can use the following commands:

- AWS CLI:

```
$ aws organizations register-delegated-administrator \
    --account-id 123456789012 \
    --service-principal devops-guru.amazonaws.com
```

- AWS SDK: Call the Organizations `RegisterDelegatedAdministrator` operation and the member account's ID number and identify the account service `principal account.amazonaws.com` as parameters.

Disabling a delegated administrator for DevOps Guru

You can remove the delegated administrator using either the DevOps Guru console, or by using the Organizations `DeregisterDelegatedAdministrator` CLI or SDK operation. For information on how to remove a delegated administrator using the DevOps Guru console, see Monitor accounts across your organization in the *Amazon DevOps Guru User Guide*.

AWS Directory Service and AWS Organizations

AWS Directory Service for Microsoft Active Directory, or AWS Managed Microsoft AD, lets you run Microsoft Active Directory (AD) as a managed service. AWS Directory Service makes it easy to set up and run directories in the AWS Cloud or connect your AWS resources with an existing on-premises Microsoft Active Directory. AWS Managed Microsoft AD also integrates tightly with AWS Organizations to allow seamless directory sharing across multiple AWS accounts and any VPC in a Region. For more information, see the AWS Directory Service Administration Guide.

Use the following information to help you integrate AWS Directory Service with AWS Organizations.

Enabling trusted access with AWS Directory Service

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

You can enable trusted access using either the AWS Directory Service console or the AWS Organizations console.

> **Important**
> We strongly recommend that whenever possible, you use the AWS Directory Service console or tools to enable integration with Organizations. This lets AWS Directory Service perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can’t enable integration using the tools provided by AWS Directory Service. For more information, see this note (p. 223).
> If you enable trusted access by using the AWS Directory Service console or tools then you don’t need to complete these steps.

To enable trusted access using the AWS Directory Service console

To share a directory, which automatically enables trusted access, see Share Your Directory in the *AWS Directory Service Administration Guide*. For step-by-step instructions, see Tutorial: Sharing Your AWS Managed Microsoft AD Directory.

You can enable trusted access by using the AWS Organizations console.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for **AWS Directory Service**, choose the service's name, and then choose **Enable trusted access**.
3. In the confirmation dialog box, enable **Show the option to enable trusted access**, enter **`enable`** in the box, and then choose **Enable trusted access**.
4. If you are the administrator of only AWS Organizations, tell the administrator of AWS Directory Service that they can now enable that service using its console to work with AWS Organizations.

Disabling trusted access with AWS Directory Service

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

If you disable trusted access using AWS Organizations while you are using AWS Directory Service, all previously shared directories continue to operate as normal. However, you can no longer share new directories within the organization until you enable trusted access again.

You can disable trusted access using only the Organizations tools.

You can disable trusted access by using the AWS Organizations console.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for **AWS Directory Service** and then choose the service's name.
3. Choose **Disable trusted access**.
4. In the confirmation dialog box, enter **`disable`** in the box, and then choose **Disable trusted access**.
5. If you are the administrator of only AWS Organizations, tell the administrator of AWS Directory Service that they can now disable that service using its console or tools from working with AWS Organizations.

AWS Firewall Manager and AWS Organizations

AWS Firewall Manager is a security management service you use to centrally configure and manage firewall rules and other protections across the AWS accounts and applications in your organization. Using Firewall Manager, you can roll out AWS WAF rules, create AWS Shield Advanced protections, configure and audit Amazon Virtual Private Cloud (Amazon VPC) security groups, and deploy AWS Network Firewalls. Use Firewall Manager to set up your protections just once and have them automatically applied across all accounts and resources within your organization, even as new resources and accounts are added. For more information about AWS Firewall Manager, see the *AWS Firewall Manager Developer Guide*.

Use the following information to help you integrate AWS Firewall Manager with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Firewall Manager to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Firewall Manager and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForFMS`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Firewall Manager grant access to the following service principals:

- `fms.amazonaws.com`

Enabling trusted access with Firewall Manager

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

You can enable trusted access using either the AWS Firewall Manager console or the AWS Organizations console.

> **Important**
> We strongly recommend that whenever possible, you use the AWS Firewall Manager console or tools to enable integration with Organizations. This lets AWS Firewall Manager perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by AWS Firewall Manager. For more information, see this note (p. 223).
> If you enable trusted access by using the AWS Firewall Manager console or tools then you don't need to complete these steps.

You must sign in with your AWS Organizations management account and configure an account within the organization as the AWS Firewall Manager administrator account. For more information, see Set the AWS Firewall Manager Administrator Account in the *AWS Firewall Manager Developer Guide*.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for **AWS Firewall Manager**, choose the service's name, and then choose **Enable trusted access**.
3. In the confirmation dialog box, enable **Show the option to enable trusted access**, enter **`enable`** in the box, and then choose **Enable trusted access**.

4. If you are the administrator of only AWS Organizations, tell the administrator of AWS Firewall Manager that they can now enable that service using its console to work with AWS Organizations.

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

You can use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  You can run the following command to enable AWS Firewall Manager as a trusted service with Organizations.

```
$ aws organizations enable-aws-service-access \
    --service-principal fms.amazonaws.com
```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access with Firewall Manager

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

You can disable trusted access using either the AWS Firewall Manager or AWS Organizations tools.

> **Important**
> We strongly recommend that whenever possible, you use the AWS Firewall Manager console or tools to disable integration with Organizations. This lets AWS Firewall Manager perform any clean up that it requires, such as deleting resources or access roles that are no longer needed by the service. Proceed with these steps only if you can't disable integration using the tools provided by AWS Firewall Manager.
> If you disable trusted access by using the AWS Firewall Manager console or tools then you don't need to complete these steps.

To disable trusted access using the Firewall Manager console

You can change or revoke the AWS Firewall Manager administrator account by following the instructions in Designating a Different Account as the AWS Firewall Manager Administrator Account in the *AWS Firewall Manager Developer Guide*.

If you revoke the administrator account, you must sign in to the AWS Organizations management account and set a new administrator account for AWS Firewall Manager.

You can disable trusted access by using either the AWS Organizations console, by running an Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for **AWS Firewall Manager** and then choose the service's name.

3. Choose **Disable trusted access**.
4. In the confirmation dialog box, enter **`disable`** in the box, and then choose **Disable trusted access**.
5. If you are the administrator of only AWS Organizations, tell the administrator of AWS Firewall Manager that they can now disable that service using its console or tools from working with AWS Organizations.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable AWS Firewall Manager as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal fms.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for Firewall Manager

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for Firewall Manager that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of Firewall Manager.

Minimum permissions
Only an IAM user or role in the Organizations management account can configure a member account as a delegated administrator for Firewall Manager in the organization.

For instructions on how to designate a member account as the Firewall Manager administrator for the organization, see Set the AWS Firewall Manager Administrator Account in the *AWS Firewall Manager Developer Guide*.

Amazon GuardDuty and AWS Organizations

Amazon GuardDuty is a continuous security monitoring service that analyzes and processes a variety data sources, using threat intelligence feeds and machine learning to identify unexpected and potentially unauthorized and malicious activity within your AWS environment. This can include issues like escalations of privileges, uses of exposed credentials, communication with malicious IP addresses, URLs, or domains, or presence of malware on your Amazon Elastic Compute Cloud instances and container workloads.

You can help simplify management of GuardDuty by using Organizations to manage GuardDuty across all of the accounts in your organization.

For more information, see Managing GuardDuty accounts with AWS Organizations in the *Amazon GuardDuty User Guide*

Use the following information to help you integrate Amazon GuardDuty with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked roles are automatically created in your organization's management account when you enable trusted access. These roles allow GuardDuty to perform supported operations within your organization's accounts in your organization. You can delete a role only if you disable trusted access between GuardDuty and Organizations, or if you remove the member account from the organization.

- The `AWSServiceRoleForAmazonGuardDuty` service-linked role is automatically created in accounts that have integrated GuardDuty with Organizations. For more information, see Managing GuardDuty accounts with Organizations in the *Amazon GuardDuty User Guide*
- The `AmazonGuardDutyMalwareProtectionServiceRolePolicy` service-linked role is automatically created in accounts that have enabled GuardDuty Malware Protection. For more information, see Service-linked role permissions for GuardDuty Malware Protection in the *Amazon GuardDuty User Guide*

Service principals used by the service-linked roles

- `guardduty.amazonaws.com`, used by the `AWSServiceRoleForAmazonGuardDuty` service-linked role.
- `malware-protection.guardduty.amazonaws.com`, used by the `AmazonGuardDutyMalwareProtectionServiceRolePolicy` service-linked role.

Enabling trusted access with GuardDuty

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

You can enable trusted access using only Amazon GuardDuty.

Amazon GuardDuty requires trusted access to AWS Organizations before you can designate a member account to be the GuardDuty administrator for your organization. If you configure a delegated administrator using the GuardDuty console, then GuardDuty automatically enables trusted access for you.

However, if you want to configure a delegated administrator account using the AWS CLI or one of the AWS SDKs, then you must explicitly call the EnableAWSServiceAccess operation and provide the service principal as a parameter. Then you can call EnableOrganizationAdminAccount to delegate the GuardDuty administrator account.

Disabling trusted access with GuardDuty

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

You can disable trusted access using only the Organizations tools.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable Amazon GuardDuty as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal guardduty.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for GuardDuty

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for GuardDuty that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of GuardDuty.

> **Minimum permissions**
> For information about the permissions required to designate a member account as a delegated administrator, see Permissions required to designate a delegated administrator in the *Amazon GuardDuty User Guide*

To designate a member account as a delegated administrator for GuardDuty

See Designate a delegated administrator and add member accounts (console) and Designate a delegated administrator and add member accounts (API)

AWS Health and AWS Organizations

AWS Health provides ongoing visibility into your resource performance and the availability of your AWS services and accounts. AWS Health delivers events when your AWS resources and services are impacted by an issue or will be affected by upcoming changes. After you enable organizational view, a user in the organization’s management account can aggregate AWS Health events across all accounts in the organization. Organizational view only shows AWS Health events delivered after the feature is enabled and retains them for 90 days.

You can enable organizational view by using the AWS Health console, the AWS Command Line Interface (AWS CLI), or the AWS Health API.

For more information, see Aggregating AWS Health events in the *AWS Health User Guide*.

Use the following information to help you integrate AWS Health with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows AWS Health to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between AWS Health and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForHealth_Organizations`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by AWS Health grant access to the following service principals:

- `health.amazonaws.com`

Enabling trusted access with AWS Health

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

When you the enable organizational view feature for AWS Health, trusted access is also enabled for you automatically.

You can enable trusted access using either the AWS Health console or the AWS Organizations console.

> **Important**
> We strongly recommend that whenever possible, you use the AWS Health console or tools to enable integration with Organizations. This lets AWS Health perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by AWS Health. For more information, see this note (p. 223).
> If you enable trusted access by using the AWS Health console or tools then you don't need to complete these steps.

To enable trusted access using the AWS Health console

You can enable trusted access by using AWS Health and one of the following options:

- Use the AWS Health console. For more information, see Organizational view (console) in the *AWS Health User Guide*.
- Use the AWS CLI. For more information, see Organizational view (CLI) in the *AWS Health User Guide*.
- Call the EnableHealthServiceAccessForOrganization API operation.

You can enable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To enable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  You can run the following command to enable AWS Health as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal health.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access with AWS Health

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

After you disable the organizational view feature, AWS Health stops aggregating events for all other accounts in your organization. This also disables trusted access for you automatically.

You can disable trusted access using either the AWS Health or AWS Organizations tools.

> **Important**
> We strongly recommend that whenever possible, you use the AWS Health console or tools to disable integration with Organizations. This lets AWS Health perform any clean up that it requires, such as deleting resources or access roles that are no longer needed by the service. Proceed with these steps only if you can't disable integration using the tools provided by AWS Health.
> If you disable trusted access by using the AWS Health console or tools then you don't need to complete these steps.

To disable trusted access using the AWS Health console

You can disable trusted access with one of the following options:

- Use the AWS Health console. For more information, see Disabling organizational view (console) in the *AWS Health User Guide*.
- Use the AWS CLI. For more information, see Disabling organizational view (CLI) in the *AWS Health User Guide*.
- Call the DisableHealthServiceAccessForOrganization API operation.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable AWS Health as a trusted service with Organizations.

```
$ aws organizations disable-aws-service-access \
    --service-principal health.amazonaws.com
```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

Amazon Inspector and AWS Organizations

Amazon Inspector is an automated vulnerability management service that continually scans Amazon EC2 and container workloads for software vulnerabilities and unintended network exposure.

Using Amazon Inspector you can manage multiple accounts that are associated through AWS Organizations by simply delegating an administrator account for Amazon Inspector. The delegated administrator manages Amazon Inspector for the organization and is granted special permissions to perform tasks on behalf of your organization such as:

- Enable or disable scans for member accounts
- View aggregated finding data from the entire organization
- Create and manage suppression rules

For more information, see Managing multiple accounts with AWS Organizations in the *Amazon Inspector User Guide*.

Use the following information to help you integrate Amazon Inspector with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Amazon Inspector to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Amazon Inspector and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForAmazonInspector2`

For more information, see Using service-linked roles with Amazon Inspector in the *Amazon Inspector User Guide*.

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Amazon Inspector grant access to the following service principals:

- `inspector2.amazonaws.com`

To enable trusted access with Amazon Inspector

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

Amazon Inspector requires trusted access to AWS Organizations before you can designate a member account to be the delegated administrator for this service for your organization.

When you designate a delegated administrator for Amazon Inspector, Amazon Inspector automatically enables trusted access for Amazon Inspector for your organization.

However, if you want to configure a delegated administrator account using the AWS CLI or one of the AWS SDKs, then you must explicitly call the `EnableAWSServiceAccess` operation and provide the service principal as a parameter. Then you can call `EnableDelegatedAdminAccount` to delegate the Inspector administrator account.

You can enable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To enable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  You can run the following command to enable Amazon Inspector as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal inspector2.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Note
If you are using the `EnableAWSServiceAccess` API, you need to also call `EnableDelegatedAdminAccount` to delegate the Inspector administrator account.

To disable trusted access with Amazon Inspector

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

Only an administrator in the AWS Organizations management account can disable trusted access with Amazon Inspector.

You can disable trusted access using only the Organizations tools.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable Amazon Inspector as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal inspector2.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for Amazon Inspector

With Amazon Inspector you can manage multiple accounts in an organization using a delegated administrator with AWS Organizations service.

The AWS Organizations management account designates an account within the organization as the delegated administrator account for Amazon Inspector. The delegated administrator manages Amazon Inspector for the organization and is granted special permissions to perform tasks on behalf of your organization such as: enable or disable scans for member accounts, view aggregated finding data from the entire organization, and create and manage suppression rules

For information on how a delegated administrator manages organization accounts, see Understanding the relationship between administrator and member accounts in the *Amazon Inspector User Guide*.

Only an administrator in the organization management account can configure a delegated administrator for Amazon Inspector.

You can specify a delegated administrator account from the Amazon Inspector console or API, or by using the Organizations CLI or SDK operation.

> **Minimum permissions**
> Only an IAM user or role in the Organizations management account can configure a member account as a delegated administrator for Amazon Inspector in the organization

To configure a delegated administrator using the Amazon Inspector console, see Step 1: Enable Amazon Inspector - Multi-account environment in the *Amazon Inspector User Guide*.

> **Note**
> You must call `inspector2:enableDelegatedAdminAccount` in each region where you use Amazon Inspector.

AWS CLI, AWS API

If you want to configure a delegated administrator account using the AWS CLI or one of the AWS SDKs, you can use the following commands:

- AWS CLI:

```
$ aws organizations register-delegated-administrator \
    --account-id 123456789012 \
    --service-principal inspector2.amazonaws.com
```

- AWS SDK: Call the Organizations `RegisterDelegatedAdministrator` operation and the member account's ID number and identify the account service `principal account.amazonaws.com` as parameters.

Disabling a delegated administrator for Amazon Inspector

Only an administrator in the AWS Organizations management account can remove a delegated administrator account from the organization.

You can remove the delegated administrator using either the Amazon Inspector console or API, or by using the Organizations `DeregisterDelegatedAdministrator` CLI or SDK operation. To remove a delegated administrator using the Amazon Inspector console, see Removing a delegated administrator in the *Amazon Inspector User Guide*.

AWS License Manager and AWS Organizations

AWS License Manager streamlines the process of bringing software vendor licenses to the cloud. As you build out cloud infrastructure on AWS, you can save costs by using bring-your-own-license (BYOL) opportunities—that is, by repurposing your existing license inventory for use with cloud resources. With rule-based controls on the consumption of licenses, administrators can set hard or soft limits on new and existing cloud deployments, stopping noncompliant server usage before it happens.

By linking License Manager with AWS Organizations, you can enable cross-account discovery of computing resources throughout your organization.

For more information about License Manager, see the License Manager User Guide.

Use the following information to help you integrate AWS License Manager with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows License Manager to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between License Manager and Organizations, or if you remove the member account from the organization.

- `AWSLicenseManagerMasterAccountRole`
- `AWSLicenseManagerMemberAccountRole`
- `AWSServiceRoleForAWSLicenseManagerRole`

For more information, see Using the License Manager–management account role and Using the License Manager–member account role.

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by License Manager grant access to the following service principals:

- `license-manager.amazonaws.com`
- `license-manager.member-account.amazonaws.com`

Enabling trusted access with License Manager

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

You can enable trusted access using only AWS License Manager.

To enable trusted access with License Manager

You must sign in to the License Manager console using your AWS Organizations management account and associate it with your License Manager account. For more information, see Configuring AWS License Manager Guide Settings. It is also summarized here for your convenience.

> **Important**
> This procedure is a one-way door. You can't undo this.

To enable trusted access between Organizations and License Manager

1. Sign in to the AWS Management Console using your organization's management account.
2. Navigate to the License Manager console and choose **Settings**.
3. Choose **Edit**.
4. Choose **Link AWS Organizations accounts**.

Disabling trusted access with License Manager

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

You can disable trusted access using only the Organizations tools.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable AWS License Manager as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal license-manager.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for License Manager

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for License Manager that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of License Manager.

To delegate a member account as an administrator for License Manager, follow the steps at Register a delegated administrator in the *License Manager User Guide*. They are also summarized here for your convenience.

To register a delegated administrator account for License Manager

1. Sign in to the AWS Management Console using your organization's management account.
2. Navigate to the License Manager console and choose **Settings**.
3. Under **Delegated administrator**, choose **Delegate administrator**.
4. Enter the account ID number for the AWS account that you want to assign, and then choose **Delegate**. You can't use the ID for the management account. It must be a member account.

Amazon Macie and AWS Organizations

Amazon Macie is a fully managed data security and data privacy service that uses machine learning and pattern matching to discover, monitor, and help you protect your sensitive data in Amazon Simple Storage Service (Amazon S3). Macie automates the discovery of sensitive data, such as personally identifiable information (PII) and intellectual property, to provide you with a better understanding of the data that your organization stores in Amazon S3.

For more information, see Managing Amazon Macie accounts with AWS Organizations in the *Amazon Macie User Guide*.

Use the following information to help you integrate Amazon Macie with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created for your organization's delegated Macie administrator account when you enable trusted access. This role allows Macie to perform supported operations for the accounts in your organization.

You can delete this role only if you disable trusted access between Macie and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleRorAmazonMacie`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Macie grant access to the following service principals:

- `macie.amazonaws.com`

Enabling trusted access with Macie

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

You can enable trusted access using either the Amazon Macie console or the AWS Organizations console.

> **Important**
> We strongly recommend that whenever possible, you use the Amazon Macie console or tools to enable integration with Organizations. This lets Amazon Macie perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can’t enable integration using the tools provided by Amazon Macie. For more information, see this note (p. 223).
> If you enable trusted access by using the Amazon Macie console or tools then you don’t need to complete these steps.

To enable trusted access using the Macie console

Amazon Macie requires trusted access to AWS Organizations to designate a member account to be the Macie administrator for your organization. If you configure a delegated administrator using the Macie management console, then Macie automatically enables trusted access for you.

For more information, see Integrating and configuring an organization in Amazon Macie in the *Amazon Macie User Guide*.

You can enable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To enable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  You can run the following command to enable Amazon Macie as a trusted service with Organizations.

```
$ aws organizations enable-aws-service-access \
    --service-principal macie.amazonaws.com
```

This command produces no output when successful.

- AWS API: EnableAWSServiceAccess

Enabling a delegated administrator account for Macie

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for Macie that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of Macie.

Minimum permissions
Only an IAM user or role in the Organizations management account with the following permissions can configure a member account as a delegated administrator for Macie in the organization:

- `organizations:EnableAWSServiceAccess`
- `macie:EnableOrganizationAdminAccount`

To designate a member account as a delegated administrator for Macie

Amazon Macie requires trusted access to AWS Organizations to designate a member account to be the Macie administrator for your organization. If you configure a delegated administrator using the Macie management console, then Macie automatically enables trusted access for you.

For more information, see https://docs.aws.amazon.com/macie/latest/user/macie-organizations.html#register-delegated-admin

AWS Marketplace and AWS Organizations

AWS Marketplace is a curated digital catalog that you can use to find, buy, deploy, and manage third-party software, data, and services that you need to build solutions and run your businesses.

AWS Marketplace creates and manages licenses using AWS License Manager for your purchases in AWS Marketplace. When you share (grant access to) your licenses with other accounts in your organization, AWS Marketplace creates and manages new licenses for those accounts.

For more information, see Service-linked roles for AWS Marketplace in the *AWS Marketplace Buyer Guide*.

Use the following information to help you integrate AWS Marketplace with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows AWS Marketplace to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between AWS Marketplace and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForMarketplaceLicenseManagement`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by AWS Marketplace grant access to the following service principals:

- `license-management.marketplace.amazonaws.com`

Enabling trusted access with AWS Marketplace

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

You can enable trusted access using either the AWS Marketplace console or the AWS Organizations console.

> **Important**
> We strongly recommend that whenever possible, you use the AWS Marketplace console or tools to enable integration with Organizations. This lets AWS Marketplace perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can’t enable integration using the tools provided by AWS Marketplace. For more information, see this note (p. 223).
> If you enable trusted access by using the AWS Marketplace console or tools then you don’t need to complete these steps.

To enable trusted access using the AWS Marketplace console

see Creating a service-linked role for AWS Marketplace in the *AWS Marketplace Buyer Guide*.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **Services** page, find the row for **AWS Marketplace**, choose the service’s name, and then choose **Enable trusted access**.
3. In the confirmation dialog box, enable **Show the option to enable trusted access**, enter **`enable`** in the box, and then choose **Enable trusted access**.
4. If you are the administrator of only AWS Organizations, tell the administrator of AWS Marketplace that they can now enable that service using its console to work with AWS Organizations.

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

You can use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  You can run the following command to enable AWS Marketplace as a trusted service with Organizations.

```
$ aws organizations enable-aws-service-access \
    --service-principal license-management.marketplace.amazonaws.com
```

This command produces no output when successful.

- AWS API: EnableAWSServiceAccess

Disabling trusted access with AWS Marketplace

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

You can enable trusted access using only the Organizations tools.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable AWS Marketplace as a trusted service with Organizations.

```
$ aws organizations disable-aws-service-access \
    --service-principal license-management.marketplace.amazonaws.com
```

  This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

AWS Network Manager and AWS Organizations

Network Manager enables you to centrally manage your AWS Cloud WAN core network and your AWS Transit Gateway network across AWS accounts, Regions, and on-premises locations. With multi-account support you can create a single global network for any of your AWS accounts, and register transit gateways from multiple accounts to the global network using the Network Manager console.

With trusted access between Network Manager and Organizations enabled, the registered delegated administrators and the management accounts can leverage the service-linked role deployed in the member accounts to describe resources attached to your global networks. From the Network Manager console the registered delegated administrators and the management accounts can assume the custom IAM roles deployed in the member accounts: `CloudWatch-CrossAccountSharingRole` for multi-account monitoring and eventing, and `IAMRoleForAWSNetworkManagerCrossAccountResourceAccess` for the console switch role access for viewing and managing multi-account resources)

Important

- We strongly recommend using the Network Manager console to manage multi-account settings (enable/disable trusted access and register/deregister delegated administrators). Managing these settings from the console automatically deploys and manages all required

service-linked roles and custom IAM roles to the member accounts needed for multi-account access.
- When you enable trusted access for Network Manager in the Network Manager console, the console also enables AWS CloudFormation StackSets service. Network Manager uses StackSets to deploy custom IAM roles needed for multi-account management.

For more information about integrating Network Manager with Organizations, see Manage multiple accounts in Network Manager with AWS Organizations in the *Amazon VPC User Guide*.

Use the following information to help you integrate AWS Network Manager with AWS Organizations.

Service-linked roles created when you enable integration

When you enable trusted access, the following service-linked roles are automatically created in the listed organization accounts. These roles allow Network Manager to perform supported operations within the accounts in your organization. If you disable trusted access, Network Manager will not delete these roles from accounts in your organization. You can manually delete them using the IAM console.

Management account

- `AWSServiceRoleForNetworkManager`
- `AWSServiceRoleForCloudFormationStackSetsOrgAdmin`
- `AWSServiceRoleForCloudWatchCrossAccount`

Member accounts

- `AWSServiceRoleForNetworkManager`
- `AWSServiceRoleForCloudFormationStackSetsOrgMember`

When you register a member account as a delegated administrator, the following additional role is automatically created in the delegated administrator account:

- `AWSServiceRoleForCloudWatchCrossAccount`

Service principals used by the service-linked roles

The service-linked roles can only be assumed by the service principals authorized by the trust relationships defined for the role.

- For the `AWSServiceRoleForNetworkManager service-linked` role, `networkmanager.amazon.aws.com` is the only service principal that has access.
- For the `AWSServiceRoleForCloudFormationStackSetsOrgMember` service-linked role, `member.org.stacksets.cloudformation.amazonaws.com` is the only service principal that has access.
- For the `AWSServiceRoleForCloudFormationStackSetsOrgAdmin` service-linked role, `stacksets.cloudformation.amazonaws.com` is the only service principal that has access.
- For the `AWSServiceRoleForCloudWatchCrossAccount` service-linked role, `cloudwatch-crossaccount.amazonaws.com` is the only service principal that has access.

Deleting these roles will impair multi-account functionality for Network Manager.

Enabling trusted access with Network Manager

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

Only an administrator in the Organizations management account has permissions to enable trusted access with another AWS service. Be sure to use the Network Manager *console* to enable trusted access, to avoid permissions issues. For more information, see Manage multiple accounts in Network Manager with AWS Organizations in the *Amazon VPC User Guide*.

Disabling trusted access with Network Manager

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

Only an administrator in an Organizations management account has permissions to disable trusted access with another AWS service.

> **Important**
> We strongly recommend using the Network Manager console to disable trusted access. If you disable trusted access in any other way, such as using AWS CLI, with an API, or with the AWS CloudFormation console, deployed AWS CloudFormation StackSets and custom IAM roles may not be properly cleaned up. To disable trusted service access, sign in to the Network Manager console.

Enabling a delegated administrator account for Network Manager

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for Network Manager that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of Network Manager.

For instructions on how to designate a member account as a delegated administrator of Network Manager in the organization, see Register a delegated administrator in the *Amazon VPC User Guide*.

AWS Resource Access Manager and AWS Organizations

AWS Resource Access Manager (AWS RAM) enables you to share specified AWS resources that you own with other AWS accounts. It's a centralized service that provides a consistent experience for sharing different types of AWS resources across multiple accounts.

For more information about AWS RAM, see the *AWS RAM User Guide*.

Use the following information to help you integrate AWS Resource Access Manager with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows AWS RAM to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between AWS RAM and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForResourceAccessManager`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by AWS RAM grant access to the following service principals:

- `ram.amazonaws.com`

Enabling trusted access with AWS RAM

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

You can enable trusted access using either the AWS Resource Access Manager console or the AWS Organizations console.

> **Important**
> We strongly recommend that whenever possible, you use the AWS Resource Access Manager console or tools to enable integration with Organizations. This lets AWS Resource Access Manager perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by AWS Resource Access Manager. For more information, see this note (p. 223).
> If you enable trusted access by using the AWS Resource Access Manager console or tools then you don't need to complete these steps.

To enable trusted access using the AWS RAM console or CLI

See Enable Sharing with AWS Organizations in the *AWS RAM User Guide*.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for **AWS Resource Access Manager**, choose the service's name, and then choose **Enable trusted access**.
3. In the confirmation dialog box, enable **Show the option to enable trusted access**, enter **`enable`** in the box, and then choose **Enable trusted access**.
4. If you are the administrator of only AWS Organizations, tell the administrator of AWS Resource Access Manager that they can now enable that service using its console to work with AWS Organizations.

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

You can use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  You can run the following command to enable AWS Resource Access Manager as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal ram.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access with AWS RAM

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

You can disable trusted access using either the AWS Resource Access Manager or AWS Organizations tools.

> **Important**
> We strongly recommend that whenever possible, you use the AWS Resource Access Manager console or tools to disable integration with Organizations. This lets AWS Resource Access Manager perform any clean up that it requires, such as deleting resources or access roles that are no longer needed by the service. Proceed with these steps only if you can't disable integration using the tools provided by AWS Resource Access Manager.
> If you disable trusted access by using the AWS Resource Access Manager console or tools then you don't need to complete these steps.

To enable trusted access using the AWS Resource Access Manager console or CLI

See Enable Sharing with AWS Organizations in the *AWS RAM User Guide*.

You can disable trusted access by using either the AWS Organizations console, by running an Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for **AWS Resource Access Manager** and then choose the service's name.
3. Choose **Disable trusted access**.
4. In the confirmation dialog box, enter `disable` in the box, and then choose **Disable trusted access**.
5. If you are the administrator of only AWS Organizations, tell the administrator of AWS Resource Access Manager that they can now disable that service using its console or tools from working with AWS Organizations.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable AWS Resource Access Manager as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal ram.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

AWS Security Hub and AWS Organizations

AWS Security Hub provides you with a comprehensive view of your security state in AWS and helps you check your environment against security industry standards and best practices.

Security Hub collects security data from across your AWS accounts, the AWS services you use, and supported third-party partner products. It helps you to analyze your security trends and identify the highest priority security issues.

When you use both Security Hub and AWS Organizations together, you can automatically enable Security Hub for all of your accounts, including new accounts as they are added. This increases the coverage for Security Hub checks and findings, which provides a more comprehensive and accurate picture of your overall security posture.

For more information about Security Hub, see the *AWS Security Hub User Guide*.

Use the following information to help you integrate AWS Security Hub with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Security Hub to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Security Hub and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForSecurityHub`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Security Hub grant access to the following service principals:

- `securityhub.amazonaws.com`

Enabling trusted access with Security Hub

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

When you designate a delegated administrator for Security Hub, Security Hub automatically enables trusted access for Security Hub in your organization.

Enabling a delegated administrator account for Security Hub

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for Security Hub that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of Security Hub.

For information, see Designating a Security Hub administrator account in the *AWS Security Hub User Guide*.

To designate a member account as a delegated administrator for Security Hub

1. Sign in with your Organizations management account.
2. Perform one of the following:
   - If your management account does not have Security Hub enabled, then on the Security Hub console, choose **Go to Security Hub**.
   - If your management account does have Security Hub enabled, then on the Security Hub console, choose **Settings**.
3. Under **Delegated Administrator**, enter the account ID.

Amazon S3 Storage Lens and AWS Organizations

By giving Amazon S3 Storage Lens trusted access to your organization, you allow it to collect and aggregate metrics across all of the AWS accounts in your organization. S3 Storage Lens does this by accessing the list of accounts that belong to your organization and collects and analyzes the storage and usage and activity metrics for all of them.

For more information, see . For more information, see the Using service-linked roles for Amazon S3 Storage Lens in the *Amazon S3 Storage Lens User Guide*.

Use the following information to help you integrate Amazon S3 Storage Lens with AWS Organizations.

Service-linked role created when you enable integration

The following service-linked role is automatically created in your organization's delegated administrator account when you enable trusted access and the Storage Lens configuration has been applied to your organization. This role allows Amazon S3 Storage Lens to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Amazon S3 Storage Lens and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForS3StorageLens`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Amazon S3 Storage Lens grant access to the following service principals:

- `storage-lens.s3.amazonaws.com`

Enabling trusted access with Amazon S3 Storage Lens

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

You can enable trusted access using either the Amazon S3 Storage Lens console or the AWS Organizations console.

> **Important**
> We strongly recommend that whenever possible, you use the Amazon S3 Storage Lens console or tools to enable integration with Organizations. This lets Amazon S3 Storage Lens perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by Amazon S3 Storage Lens. For more information, see this note (p. 223).
> If you enable trusted access by using the Amazon S3 Storage Lens console or tools then you don't need to complete these steps.

To enable trusted access using the Amazon S3 console

See How to enable Trusted Access in the *Amazon Simple Storage Service User Guide*.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for **Amazon S3 Storage Lens**, choose the service's name, and then choose **Enable trusted access**.
3. In the confirmation dialog box, enable **Show the option to enable trusted access**, enter **`enable`** in the box, and then choose **Enable trusted access**.
4. If you are the administrator of only AWS Organizations, tell the administrator of Amazon S3 Storage Lens that they can now enable that service using its console to work with AWS Organizations.

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

You can use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  You can run the following command to enable Amazon S3 Storage Lens as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal storage-lens.s3.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access with Amazon S3 Storage Lens

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

You can disable trusted access using only the Amazon S3 Storage Lens tools.

You can disable trusted access using the Amazon S3 console, the AWS CLI or any of the AWS SDKs.

To disable trusted access using the Amazon S3 console

See How to disable Trusted Access in the *Amazon Simple Storage Service User Guide*.

Enabling a delegated administrator account for Amazon S3 Storage Lens

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for Amazon S3 Storage Lens that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of Amazon S3 Storage Lens.

> **Minimum permissions**
> Only an IAM user or role in the Organizations management account with the following permission can configure a member account as a delegated administrator for Amazon S3 Storage Lens in the organization:
> `organizations:RegisterDelegatedAdministrator`
> `organizations:DeregisterDelegatedAdministrator`

Amazon S3 Storage Lens supports a maximum of 5 delegated administrator accounts in your organization.

To designate a member account as a delegated administrator for Amazon S3 Storage Lens

You can register a delegated administrator using the Amazon S3 console, the AWS CLI or any of the AWS SDKs. To register a member account as a delegated administrator account for your organization using the Amazon S3 console, see How to register a Delegated Administrator in the *Amazon Simple Storage Service User Guide*.

To deregister a delegated administrator for Amazon S3 Storage Lens

You can deregister a delegated administrator using the Amazon S3 console, the AWS CLI or any of the AWS SDKs. To deregister a delegated administrator using the Amazon S3 console, see How to unregister a Delegated Administrator in the *Amazon Simple Storage Service User Guide*.

AWS Service Catalog and AWS Organizations

AWS Service Catalog enables you to create and manage catalogs of IT services that are approved for use on AWS.

The integration of AWS Service Catalog with AWS Organizations simplifies the sharing of portfolios and copying of products across an organization. AWS Service Catalog administrators can reference an existing organization in AWS Organizations when sharing a portfolio, and they can share the portfolio with any trusted organizational unit (OU) in the organization's tree structure. This eliminates the need to share portfolio IDs, and for the receiving account to manually reference the portfolio ID when importing the portfolio. Portfolios shared via this mechanism are listed in the shared-to account in the administrator’s **Imported Portfolio** view in AWS Service Catalog.

For more information about AWS Service Catalog, see the *AWS Service Catalog Administrator Guide*.

Use the following information to help you integrate AWS Service Catalog with AWS Organizations.

Service-linked roles created when you enable integration

AWS Service Catalog doesn't create any service-linked roles as part of enabling trusted access.

Service principals used to grant permissions

To enable trusted access, you must specify the following service principal:

- `servicecatalog.amazonaws.com`

Enabling trusted access with AWS Service Catalog

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

You can enable trusted access using either the AWS Service Catalog console or the AWS Organizations console.

> **Important**
> We strongly recommend that whenever possible, you use the AWS Service Catalog console or tools to enable integration with Organizations. This lets AWS Service Catalog perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can’t enable integration using the tools provided by AWS Service Catalog. For more information, see this note (p. 223).
> If you enable trusted access by using the AWS Service Catalog console or tools then you don’t need to complete these steps.

To enable trusted access using the AWS Service Catalog CLI or AWS SDK

Call one of the following commands or operations:

- AWS CLI: aws servicecatalog enable-aws-organizations-access
- AWS SDKs: AWSServiceCatalog::EnableAWSOrganizationsAccess

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **Services** page, find the row for **AWS Service Catalog**, choose the service’s name, and then choose **Enable trusted access**.
3. In the confirmation dialog box, enable **Show the option to enable trusted access**, enter **`enable`** in the box, and then choose **Enable trusted access**.
4. If you are the administrator of only AWS Organizations, tell the administrator of AWS Service Catalog that they can now enable that service using its console to work with AWS Organizations.

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

You can use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  You can run the following command to enable AWS Service Catalog as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal servicecatalog.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access with AWS Service Catalog

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

If you disable trusted access using AWS Organizations while you are using AWS Service Catalog, it doesn't delete your current shares, but it prevents you from creating new shares throughout your organization. Current shares won't be in sync with your organization structure if it changes after you call this action.

You can disable trusted access using either the AWS Service Catalog or AWS Organizations tools.

> **Important**
>
> We strongly recommend that whenever possible, you use the AWS Service Catalog console or tools to disable integration with Organizations. This lets AWS Service Catalog perform any clean up that it requires, such as deleting resources or access roles that are no longer needed by the service. Proceed with these steps only if you can’t disable integration using the tools provided by AWS Service Catalog.
>
> If you disable trusted access by using the AWS Service Catalog console or tools then you don’t need to complete these steps.

To disable trusted access using the AWS Service Catalog CLI or AWS SDK

Call one of the following commands or operations:

- AWS CLI: aws servicecatalog enable-aws-organizations-access
- AWS SDKs: DisableAWSOrganizationsAccess

You can disable trusted access by using either the AWS Organizations console, by running an Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **Services** page, find the row for **AWS Service Catalog** and then choose the service’s name.
3. Choose **Disable trusted access**.

4. In the confirmation dialog box, enter **`disable`** in the box, and then choose **Disable trusted access**.
5. If you are the administrator of only AWS Organizations, tell the administrator of AWS Service Catalog that they can now disable that service using its console or tools from working with AWS Organizations.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable AWS Service Catalog as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal servicecatalog.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

Service Quotas and AWS Organizations

Service Quotas is an AWS service that enables you to view and manage your quotas from a central location. Quotas, also referred to as limits, are the maximum value for your resources, actions, and items in your AWS account.

When Service Quotas is associated with AWS Organizations, you can create a quota request template to automatically request quota increases when accounts are created.

For more information about Service Quotas, see the Service Quotas User Guide.

Use the following information to help you integrate Service Quotas with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Service Quotas to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Service Quotas and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForServiceQuotas`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Service Quotas grant access to the following service principals:

- `servicequotas.amazonaws.com`

Enabling trusted access with Service Quotas

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

You can enable trusted access using only Service Quotas.

You can enable trusted access using the Service Quotas console

To enable trusted access using the Service Quotas console

Sign in with your AWS Organizations management account and then configure the template on the Service Quotas console. For more information, see Using the Service Quota Template in the *Service Quotas User Guide*.

To enable trusted access using the Service Quotas AWS CLI or SDK

Call the following command or operation:

- AWS CLI: aws service-quotas associate-service-quota-template
- AWS SDKs: AssociateServiceQuotaTemplate

AWS IAM Identity Center (successor to AWS Single Sign-On) and AWS Organizations

AWS IAM Identity Center (successor to AWS Single Sign-On) provides single sign-on access for all of your AWS accounts and cloud applications. It connects with Microsoft Active Directory through AWS Directory Service to allow users in that directory to sign in to a personalized AWS access portal using their existing Active Directory user names and passwords. From the AWS access portal, users have access to all the AWS accounts and cloud applications that they have permissions for.

For more information about IAM Identity Center, see the AWS IAM Identity Center (successor to AWS Single Sign-On) User Guide.

Use the following information to help you integrate AWS IAM Identity Center (successor to AWS Single Sign-On) with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows IAM Identity Center to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between IAM Identity Center and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForSSO`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by IAM Identity Center grant access to the following service principals:

- `sso.amazonaws.com`

Enabling trusted access with IAM Identity Center

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

You can enable trusted access using either the AWS IAM Identity Center (successor to AWS Single Sign-On) console or the AWS Organizations console.

> **Important**
> We strongly recommend that whenever possible, you use the AWS IAM Identity Center (successor to AWS Single Sign-On) console or tools to enable integration with Organizations. This lets AWS IAM Identity Center (successor to AWS Single Sign-On) perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by AWS IAM Identity Center (successor to AWS Single Sign-On). For more information, see this note (p. 223).
> If you enable trusted access by using the AWS IAM Identity Center (successor to AWS Single Sign-On) console or tools then you don't need to complete these steps.

IAM Identity Center requires trusted access with AWS Organizations to function. Trusted access is enabled when you set up IAM Identity Center. For more information, see Getting Started - Step 1: Enable AWS IAM Identity Center (successor to AWS Single Sign-On) in the *AWS IAM Identity Center (successor to AWS Single Sign-On) User Guide*.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for **AWS IAM Identity Center (successor to AWS Single Sign-On)**, choose the service's name, and then choose **Enable trusted access**.
3. In the confirmation dialog box, enable **Show the option to enable trusted access**, enter **`enable`** in the box, and then choose **Enable trusted access**.
4. If you are the administrator of only AWS Organizations, tell the administrator of AWS IAM Identity Center (successor to AWS Single Sign-On) that they can now enable that service using its console to work with AWS Organizations.

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

You can use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  You can run the following command to enable AWS IAM Identity Center (successor to AWS Single Sign-On) as a trusted service with Organizations.

```
$ aws organizations enable-aws-service-access \
    --service-principal sso.amazonaws.com
```

This command produces no output when successful.

- AWS API: EnableAWSServiceAccess

Disabling trusted access with IAM Identity Center

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

IAM Identity Center requires trusted access with AWS Organizations to operate. If you disable trusted access using AWS Organizations while you are using IAM Identity Center, it stops functioning because it can't access the organization. Users can't use IAM Identity Center to access accounts. Any roles that IAM Identity Center creates remain, but the IAM Identity Center service can't access them. The IAM Identity Center service-linked roles remain. If you reenable trusted access, IAM Identity Center continues to operate as before, without the need for you to reconfigure the service.

If you remove an account from your organization, IAM Identity Center automatically cleans up any metadata and resources, such as its service-linked role. A standalone account that is removed from an organization no longer works with IAM Identity Center.

You can disable trusted access using only the Organizations tools.

You can disable trusted access by using either the AWS Organizations console, by running an Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **Services** page, find the row for **AWS IAM Identity Center (successor to AWS Single Sign-On)** and then choose the service’s name.
3. Choose **Disable trusted access**.
4. In the confirmation dialog box, enter **`disable`** in the box, and then choose **Disable trusted access**.
5. If you are the administrator of only AWS Organizations, tell the administrator of AWS IAM Identity Center (successor to AWS Single Sign-On) that they can now disable that service using its console or tools from working with AWS Organizations.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable AWS IAM Identity Center (successor to AWS Single Sign-On) as a trusted service with Organizations.

```
$ aws organizations disable-aws-service-access \
    --service-principal sso.amazonaws.com
```

  This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for IAM Identity Center

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for IAM Identity Center that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of IAM Identity Center.

Minimum permissions
Only an IAM user or role in the Organizations management account can configure a member account as a delegated administrator for IAM Identity Center in the organization.

For instructions about how to enable a delegated administrator account for IAM Identity Center, see Delegated administration in the *AWS IAM Identity Center (successor to AWS Single Sign-On) User Guide*.

AWS Systems Manager and AWS Organizations

AWS Systems Manager is a collection of capabilities that enable visibility and control of your AWS resources. Two of the features that are part of Systems Manager can work with Organizations to work across all of the AWS accounts in your organization.

- Systems Manager Explorer, is a customizable operations dashboard that reports information about your AWS resources. You can synchronize operations data across all AWS accounts in your organization by using Organizations and Systems Manager Explorer. For more information, see Systems Manager Explorer in the *AWS Systems Manager User Guide*.
- Systems Manager Change Manager is an enterprise change management framework for requesting, approving, implementing, and reporting on operational changes to your application configuration and infrastructure. For more information, see AWS Systems Manager Change Manager in the *AWS Systems Manager User Guide*.

Use the following information to help you integrate AWS Systems Manager with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Systems Manager to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Systems Manager and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForAmazonSSM_AccountDiscovery`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Systems Manager grant access to the following service principals:

- `ssm.amazonaws.com`

Enabling trusted access with Systems Manager

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

You can enable trusted access using either the AWS Systems Manager console or the AWS Organizations console.

> **Important**
> We strongly recommend that whenever possible, you use the AWS Systems Manager console or tools to enable integration with Organizations. This lets AWS Systems Manager perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by AWS Systems Manager. For more information, see this note (p. 223).
> If you enable trusted access by using the AWS Systems Manager console or tools then you don't need to complete these steps.

To enable trusted access by using the Systems Manager console

You must sign in with your AWS Organizations management account and create a Resource Data Sync. For information, see Setting Up Explorer to Display Data from Multiple Accounts and Regions in the *AWS Systems Manager User Guide*.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for **AWS Systems Manager**, choose the service's name, and then choose **Enable trusted access**.
3. In the confirmation dialog box, enable **Show the option to enable trusted access**, enter **`enable`** in the box, and then choose **Enable trusted access**.
4. If you are the administrator of only AWS Organizations, tell the administrator of AWS Systems Manager that they can now enable that service using its console to work with AWS Organizations.

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

You can use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  You can run the following command to enable AWS Systems Manager as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal ssm.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access with Systems Manager

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

Systems Manager requires trusted access with AWS Organizations to synchronize operations data across AWS accounts in your organization. If you disable trusted access, then Systems Manager fails to synchronize operations data and reports an error.

You can disable trusted access using either the AWS Systems Manager or AWS Organizations tools.

> **Important**
>
> We strongly recommend that whenever possible, you use the AWS Systems Manager console or tools to disable integration with Organizations. This lets AWS Systems Manager perform any clean up that it requires, such as deleting resources or access roles that are no longer needed by the service. Proceed with these steps only if you can't disable integration using the tools provided by AWS Systems Manager.
>
> If you disable trusted access by using the AWS Systems Manager console or tools then you don't need to complete these steps.

To disable trusted access using the Systems Manager console

See Deleting a Systems Manager Explorer Resource Data Sync in the *AWS Systems Manager User Guide*. To reenable trusted access, you must create a new Resource Data Sync for Systems Manager Explorer.

You can disable trusted access by using either the AWS Organizations console, by running an Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for **AWS Systems Manager** and then choose the service's name.
3. Choose **Disable trusted access**.
4. In the confirmation dialog box, enter **`disable`** in the box, and then choose **Disable trusted access**.
5. If you are the administrator of only AWS Organizations, tell the administrator of AWS Systems Manager that they can now disable that service using its console or tools from working with AWS Organizations.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable AWS Systems Manager as a trusted service with Organizations.

```
$ aws organizations disable-aws-service-access \
    --service-principal ssm.amazonaws.com
```

This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for Systems Manager

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for Systems Manager that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of Systems Manager.

If you use Change Manager across an organization, you use a delegated administrator account. This is the AWS account that has been designated as the account for managing change templates, change requests, change runbooks and approval workflows in Change Manager. The delegated account manages change activities across your organization. When you set up your organization for use with Change Manager, you specify which of your accounts serves in this role. It does not have to be the organization's management account. The delegated administrator account is not required if you use Change Manager with a single account only."

To designate a member account as a delegated administrator for Systems Manager

For Systems Manager Explorer, see Configuring a Delegated Administrator in the *AWS Systems Manager User Guide*.

For Systems Manager Change Manager, see Setting up an organization and delegated account for Change Manager in the *AWS Systems Manager User Guide*.

Tag policies and AWS Organizations

Tag policies are a type of policy in AWS Organizations that can help you standardize tags across resources in your organization's accounts. For more information about tag policies, see Tag policies (p. 187).

Use the following information to help you integrate tag policies with AWS Organizations.

Service principals used by the service-linked roles

Organizations interacts with the tags attached to your resources using the following service principal.

- `tagpolicies.tag.amazonaws.com`

Enabling trusted access for tag policies

You can enable trusted access either by enabling tag policies in the organization, or by using the AWS Organizations console.

> **Important**
> We strongly recommend that you enable trusted access by enabling tag policies. This enables Organizations to perform required setup tasks.

You can enable trusted access for tag policies by enabling the tag policy type in the AWS Organizations console. For more information, see Enabling a policy type (p. 85).

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for **tag policies**, choose the service's name, and then choose **Enable trusted access**.
3. In the confirmation dialog box, enable **Show the option to enable trusted access**, enter **`enable`** in the box, and then choose **Enable trusted access**.
4. If you are the administrator of only AWS Organizations, tell the administrator of tag policies that they can now enable that service using its console to work with AWS Organizations.

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

You can use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  You can run the following command to enable tag policies as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal tagpolicies.tag.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access with tag policies

You can disable trusted access for tag policies by disabling the tag policy type in the AWS Organizations console. For more information, see Disabling a policy type (p. 86).

AWS Trusted Advisor and AWS Organizations

AWS Trusted Advisor inspects your AWS environment and makes recommendations when opportunities exist to save money, to improve system availability and performance, or to help close security gaps. When integrated with Organizations, you can receive Trusted Advisor check results for all of the accounts in your organization and download reports to view the summaries of your checks and any affected resources.

For more information, see Organizational view for AWS Trusted Advisor in the *AWS Support User Guide*.

Use the following information to help you integrate AWS Trusted Advisor with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Trusted Advisor to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Trusted Advisor and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForTrustedAdvisorReporting`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Trusted Advisor grant access to the following service principals:

- `reporting.trustedadvisor.amazonaws.com`

Enabling trusted access with Trusted Advisor

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

You can enable trusted access using only AWS Trusted Advisor.

To enable trusted access using the Trusted Advisor console

See Enable organizational view in the *AWS Support User Guide*.

Disabling trusted access with Trusted Advisor

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

After you disable this feature, Trusted Advisor stops recording check information for all other accounts in your organization. You can't view or download existing reports or create new reports.

You can disable trusted access using either the AWS Trusted Advisor or AWS Organizations tools.

> **Important**
> We strongly recommend that whenever possible, you use the AWS Trusted Advisor console or tools to disable integration with Organizations. This lets AWS Trusted Advisor perform any clean up that it requires, such as deleting resources or access roles that are no longer needed by the service. Proceed with these steps only if you can't disable integration using the tools provided by AWS Trusted Advisor.
> If you disable trusted access by using the AWS Trusted Advisor console or tools then you don't need to complete these steps.

To disable trusted access using the Trusted Advisor console

See Disable organizational view in the *AWS Support User Guide*.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable AWS Trusted Advisor as a trusted service with Organizations.

```
$ aws organizations disable-aws-service-access \
    --service-principal reporting.trustedadvisor.amazonaws.com
```

This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for Trusted Advisor

When you designate a member account to be a delegated administrator for the organization, users and roles from the designated account can manage the AWS account metadata for other member accounts in the organization. If you don't enable a delegated admin account, then these tasks can be performed only by the organization's management account. This helps you to separate management of the organization from management of your account details.

Minimum permissions

Only an IAM user or role in the Organizations management account can configure a member account as a delegated administrator for Trusted Advisor in the organization

For instruction about enabling a delegated administrator account for Trusted Advisor, see Register delegated administrators in the *AWS Support User Guide*.

AWS CLI, AWS API

If you want to configure a delegated administrator account using the AWS CLI or one of the AWS SDKs, you can use the following commands:

- AWS CLI:

```
$  aws organizations register-delegated-administrator \
    --account-id 123456789012 \
    --service-principal reporting.trustedadvisor.amazonaws.com
```

- AWS SDK: Call the Organizations `RegisterDelegatedAdministrator` operation and the member account's ID number and identify the account service `principal account.amazonaws.com` as parameters.

Disabling a delegated administrator for Trusted Advisor

You can remove the delegated administrator using either the Trusted Advisor console, or by using the the Organizations `DeregisterDelegatedAdministrator` CLI or SDK operation. For information on how to disable the delegated admin Trusted Advisor account using the Trusted Advisor console, see Deregister delegated administrators in the *AWS Support user guide*.

AWS Well-Architected Tool and AWS Organizations

The AWS Well-Architected Tool helps you document the state of your workloads and compares them to the latest AWS architectural best practices.

Using AWS Well-Architected Tool with Organizations enables both AWS Well-Architected Tool and Organizations customers to simplify the process of sharing AWS Well-Architected Tool resources with other members of their organization.

For more information, see Sharing your AWS Well-Architected Tool resources in the *AWS Well-Architected Tool User Guide*.

Use the following information to help you integrate AWS Well-Architected Tool with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows AWS WA Tool to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between AWS WA Tool and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForWellArchitected`

The service role policy is `AWSWellArchitectedOrganizationsServiceRolePolicy`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by AWS WA Tool grant access to the following service principals:

- `wellarchitected.amazonaws.com`

Enabling trusted access with AWS WA Tool

Allows the updating of AWS WA Tool to reflect hierarchical changes in an organization.

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

You can enable trusted access using either the AWS Well-Architected Tool console or the AWS Organizations console.

> **Important**
> We strongly recommend that whenever possible, you use the AWS Well-Architected Tool console or tools to enable integration with Organizations. This lets AWS Well-Architected Tool perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can’t enable integration using the tools provided by AWS Well-Architected Tool. For more information, see this note (p. 223).
> If you enable trusted access by using the AWS Well-Architected Tool console or tools then you don’t need to complete these steps.

To enable trusted access using the AWS WA Tool console

See Sharing your AWS Well-Architected Tool resources in the *AWS Well-Architected Tool User Guide*.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization’s management account.
2. On the **Services** page, find the row for **AWS Well-Architected Tool**, choose the service’s name, and then choose **Enable trusted access**.

3. In the confirmation dialog box, enable **Show the option to enable trusted access**, enter `enable` in the box, and then choose **Enable trusted access**.
4. If you are the administrator of only AWS Organizations, tell the administrator of AWS Well-Architected Tool that they can now enable that service using its console to work with AWS Organizations.

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

You can use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  You can run the following command to enable AWS Well-Architected Tool as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal wellarchitected.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access with AWS WA Tool

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

You can disable trusted access using either the AWS Well-Architected Tool or AWS Organizations tools.

> **Important**
> We strongly recommend that whenever possible, you use the AWS Well-Architected Tool console or tools to disable integration with Organizations. This lets AWS Well-Architected Tool perform any clean up that it requires, such as deleting resources or access roles that are no longer needed by the service. Proceed with these steps only if you can't disable integration using the tools provided by AWS Well-Architected Tool.
> If you disable trusted access by using the AWS Well-Architected Tool console or tools then you don't need to complete these steps.

To disable trusted access using the AWS WA Tool console

See Sharing your AWS Well-Architected Tool resources in the *AWS Well-Architected Tool User Guide*.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable AWS Well-Architected Tool as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
  ```

```
    --service-principal wellarchitected.amazonaws.com
```

This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Amazon VPC IP Address Manager (IPAM) and AWS Organizations

Amazon VPC IP Address Manager (IPAM) is a VPC feature that makes it easier for you to plan, track, and monitor IP addresses for your AWS workloads.

Using AWS Organizations allows you to monitor IP address usage throughout your organization and share IP address pools across member accounts.

For more information, see Integrate IPAM with AWS Organizations in the *Amazon VPC IPAM User Guide*.

Use the following information to help you integrate Amazon VPC IP Address Manager (IPAM) with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account and each member account when you integrate IPAM with AWS Organizations either by using the IPAM console or using IPAM's `EnableIpamOrganizationAdminAccount` API.

- `AWSServiceRoleForIPAM`

For more information, see Service-linked roles for IPAM in the *Amazon VPC IPAM User Guide*.

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by IPAM grant access to the following service principals:

- `ipam.amazonaws.com`

To enable trusted access with IPAM

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access (p. 223).

> **Note**
> When you designate a delegated administrator for IPAM it automatically enables trusted access for IPAM for your organization.
> IPAM requires trusted access to AWS Organizations before you can designate a member account to be the delegated administrator for this service for your organization.

You can enable trusted access using only Amazon VPC IP Address Manager (IPAM) tools.

If you integrate IPAM with AWS Organizations using the IPAM console or using the IPAM `EnableIpamOrganizationAdminAccount` API, you automatically grant trusted access to IPAM.

Granting trusted access creates the service-linked role `AWSServiceRoleForIPAM` in the management account and in all of the member accounts in the organization. IPAM uses the service-linked role to monitor CIDRs associated with EC2 networking resources in your organization and to store metrics related to IPAM in Amazon CloudWatch. For more information, see Service-linked roles for IPAM in the *Amazon VPC IPAM User Guide*.

For instructions about enabling trusted access, see Integrate IPAM with AWS Organizations in the *Amazon VPC IPAM User Guide*.

> **Note**
> You can't enable trusted access with IPAM using the AWS Organizations console or with the `enable-aws-service-access` API.

To disable trusted access with IPAM

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access (p. 224).

Only an administrator in the AWS Organizations management account can disable trusted access with IPAM using the AWS Organizations `disable-aws-service-access` API.

For information about disabling IPAM account permissions and deleting the service-linked role, see Service-linked roles for IPAM in the *Amazon VPC IPAM User Guide*.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable Amazon VPC IP Address Manager (IPAM) as a trusted service with Organizations.

```
$ aws organizations disable-aws-service-access \
    --service-principal ipam.amazonaws.com
```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for IPAM

The delegated administrator account for IPAM is responsible for creating the IPAM and IP address pools, managing and monitoring IP address usage in the organization, and sharing IP address pools across member accounts. For more information, see Integrate IPAM with AWS Organizations in the *Amazon VPC IPAM User Guide*.

Only an administrator in the organization management account can configure a delegated administrator for IPAM.

You can specify a delegated administrator account from the IPAM console, or by using the `enable-ipam-organization-admin-account` API. For more information, see enable-ipam-organization-admin-account in the *AWS AWS CLI Command Reference*.

Minimum permissions
Only an IAM user or role in the Organizations management account can configure a member account as a delegated administrator for IPAM in the organization

To configure a delegated administrator using the IPAM console, see Integrate IPAM with AWS Organizations in the *Amazon VPC IPAM User Guide*.

Disabling a delegated administrator for IPAM

Only an administrator in the organization management account can configure a delegated administrator for IPAM.

To remove a delegated administrator using the AWS AWS CLI, see disable-ipam-organization-admin-account in the *AWS AWS CLI Command Reference*.

To disable the delegated admin IPAM account using the IPAM console, see Integrate IPAM with AWS Organizations in the *Amazon VPC IPAM User Guide*.

Security in AWS Organizations

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

Topics

AWS Identity and Access Management and AWS Organizations

Access to AWS Organizations requires credentials. Those credentials must have permissions to access AWS resources, such as an Amazon Simple Storage Service (Amazon S3) bucket, an Amazon Elastic Compute Cloud (Amazon EC2) instance, or an AWS Organizations organizational unit (OU). The following sections provide details on how you can use AWS Identity and Access Management (IAM) to help secure access to your organization and control who can administer it.

To determine who can administer which parts of your organization, AWS Organizations uses the same IAM-based permissions model as other AWS services. As an administrator in the management account of an organization, you can grant IAM-based permissions to perform AWS Organizations tasks by attaching policies to users, groups, and roles in the management account. Those policies specify the actions that those principals can perform. You attach an IAM permissions policy to a group that the user is a member of or directly to a user or role. As a best practice, we recommend that you attach policies to groups instead of users. You also have the option to grant full administrator permissions to others.

For most administrator operations for AWS Organizations, you need to attach permissions to users or groups in the management account. If a user in a member account needs to perform administrator operations for your organization, you need to grant the AWS Organizations permissions to an *IAM role* in the management account and enable the user in the member account to assume the role. For general information about IAM permissions policies, see Overview of IAM Policies in the *IAM User Guide*.

Topics

Authentication

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

- **AWS account root user** – When you sign up for AWS, you provide an email address and password that is associated with your AWS account. These are your *root credentials*, and they provide complete access to all of your AWS resources.

  Important

  For security reasons, we recommend that you use the root user credentials only to create an *administrator user*, which is an *IAM user* with full permissions to your AWS account. Then you can use this administrator user to create other IAM users and roles with limited permissions. For more information, see IAM Best Practices and Creating Your First IAM Admin User and Group in the *IAM User Guide*.
- **IAM user** – An IAM user is simply an identity within your AWS account that has specific custom permissions (for example, permissions to create a file system in Amazon Elastic File System). 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 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 (AWS CLI). The SDK and AWS CLI tools use the access keys to cryptographically sign your request. If you don't use the AWS tools, you must sign the request yourself. AWS Organizations 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 another IAM identity you can create in your account that has specific permissions. It is similar to an IAM user, but it isn't associated with a specific person. An IAM role allows you to obtain temporary access keys that can 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 preexisting 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*.
  - **Cross-account access** – You can use an IAM role in your account to grant another AWS account permissions to access your account's resources. For an example, see Tutorial: Delegate Access Across AWS accounts Using IAM 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 stored in the 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** – Instead of storing access keys in the EC2 instance for use by applications running on the instance and making AWS API requests, you can use an IAM role to manage temporary credentials for these applications. To assign an AWS role to an EC2 instance and make it available to all of its applications, you can create an instance profile that is attached to the

instance. An instance profile contains the role and enables programs running on the EC2 instance to get temporary credentials. For more information, see Using Roles for Applications on Amazon EC2 in the *IAM User Guide*.

Access control

You can have valid credentials to authenticate your requests, but unless you have permissions, you can't administer or access AWS Organizations resources. For example, you must have permissions to create an OU or to attach a service control policy (SCP) (p. 104) to an account.

The following sections describe how to manage permissions for AWS Organizations.

- Managing access permissions for your AWS organization (p. 319)
- Using identity-based policies (IAM policies) for AWS Organizations (p. 324)
- Attribute-based access control with tags and AWS Organizations (p. 327)

Managing access permissions for your AWS organization

All AWS resources, including the roots, OUs, accounts, and policies in an organization, are owned by an AWS account, and permissions to create or access a resource are governed by permissions policies. For an organization, its management account owns all resources. An account administrator can control access to AWS resources by attaching permissions policies to IAM identities (users, groups, and roles).

> **Note**
> An *account administrator* (or administrator user) is a user with administrator permissions. For more information, see IAM Best Practices in the *IAM User Guide*.

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

By default, IAM users, groups, and roles have no permissions. As an administrator in the management account of an organization, you can perform administrative tasks or delegate administrator permissions to other IAM users or roles in the management account. To do this, you attach an IAM permissions policy to an IAM user, group, or role. By default, a user has no permissions at all; this is sometimes called an *implicit deny*. The policy overrides the implicit deny with an *explicit allow* that specifies which actions the user can perform, and which resources they can perform the actions on. If the permissions are granted to a role, users in other accounts in the organization can assume that role.

AWS Organizations resources and operations

This section discusses how AWS Organizations concepts map to their IAM-equivalent concepts.

Resources

In AWS Organizations, you can control access to the following resources:

- The root and the OUs that make up the hierarchical structure of an organization
- The accounts that are members of the organization
- The policies that you attach to the entities in the organization
- The handshakes that you use to change the state of the organization

Each of these resources has a unique Amazon Resource Name (ARN) associated with it. You control access to a resource by specifying its ARN in the `Resource` element of an IAM permission policy. For a complete list of the ARN formats for resources that are used in AWS Organizations, see Resources Defined by AWS Organizations in the *IAM User Guide*.

Operations

AWS provides a set of operations to work with the resources in an organization. They enable you to do things like create, list, modify, access the contents of, and delete resources. Most operations can be referenced in the `Action` element of an IAM policy to control who can use that operation. For a list of AWS Organizations operations that can be used as permissions in an IAM policy, see API Action Permissions Defined by AWS Organizations in the *IAM User Guide*.

When you combine an `Action` and a `Resource` in a single permission policy `Statement`, you control exactly which resources that particular set of actions can be used on.

Condition keys

AWS provides condition keys that you can query to provide more granular control over certain actions. You can reference these condition keys in the `Condition` element of an IAM policy to specify the additional circumstances that must be met for the statement to be considered a match.

The following condition keys are especially useful with AWS Organizations:

- `aws:PrincipalOrgID` – Simplifies specifying the `Principal` element in a resource-based policy. This global key provides an alternative to listing all the account IDs for all AWS accounts in an organization. Instead of listing all of the accounts that are members of an organization, you can specify the organization ID (p. 42) in the `Condition` element.

  > **Note**
  > This global condition also applies to the management account of an organization.

  For more information, see the description of `PrincipalOrgID` in AWS Global Condition Context Keys in the *IAM User Guide*.
- `aws:PrincipalOrgPaths` – Use this condition key to match members of a specific organization root, an OU, or its children. The `aws:PrincipalOrgPaths` condition key returns true when the principal (root user, IAM user, or role) making the request is in the specified organization path. A path is a text representation of the structure of an AWS Organizations entity. For more information about paths, see Understanding the AWS Organizations Entity Path in the *IAM User Guide*. For more information about using this condition key, see aws:PrincipalOrgPaths in the *IAM User Guide*.

  For example, the following condition element matches for members of either of two OUs in the same organization.

```
            "Condition": {
                "ForAnyValue:StringLike": {
                    "aws:PrincipalOrgPaths": [
                        "o-a1b2c3d4e5/r-f6g7h8i9j0example/ou-def0-awsbbbbb/",
                        "o-a1b2c3d4e5/r-f6g7h8i9j0example/ou-jkl0-awsddddd/"
                    ]
                }
            }
```

- `organizations:PolicyType` – You can use this condition key to restrict the Organizations policy-related API operations to work on only Organizations policies of the specified type. You can apply this condition key to any policy statement that includes an action that interacts with Organizations policies.

  You can use the following values with this condition key:
  - `AISERVICES_OPT_OUT_POLICY`
  - `BACKUP_POLICY`

  - `SERVICE_CONTROL_POLICY`
  - `TAG_POLICY`

  For example, the following example policy allows the user to perform any Organizations operation. However, if the user performs an operation that takes a policy argument, the operation is allowed only if the specified policy is a tagging policy. The operation fails if the user specifies any other type of policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "IfTaggingAPIThenAllowOnOnlyTaggingPolicies",
            "Effect": "Allow",
            "Action": "organizations:*",
            "Resource": "*",
            "Condition": {
                "StringLikeIfExists": {
                    "organizations:PolicyType": [ "TAG_POLICY" ]
                }
            }
        }
    ]
}
```

- `organizations:ServicePrincipal` – Available as a condition if you use the EnableAWSServiceAccess or DisableAWSServiceAccess operations to enable or disable trusted access (p. 223) with other AWS services. You can use `organizations:ServicePrincipal` to restrict requests that those operations make to a list of approved service principal names.

  For example, the following policy allows the user to specify only AWS Firewall Manager when enabling and disabling trusted access with AWS Organizations.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowOnlyAWSFirewallIntegration",
            "Effect": "Allow",
            "Action": [
                "organizations:EnableAWSServiceAccess",
                "organizations:DisableAWSServiceAccess"
            ],
            "Resource": "*",
            "Condition": {
                "StringLikeIfExists": {
                    "organizations:ServicePrincipal": [ "fms.amazonaws.com" ]
                }
            }
        }
    ]
}
```

For a list of all of the AWS Organizations–specific condition keys that can be used as permissions in an IAM policy, see Condition Context Keys for AWS Organizations in the *IAM User Guide*.

Understanding resource ownership

The AWS account owns the resources that are created in the account, regardless of who created the resources. Specifically, the resource owner is the AWS account of the principal entity (that is, the root account, an IAM user, or an IAM role) that authenticates the resource creation request. For an AWS organization, that is ***always*** the management account. You can't call most operations that create or access organization resources from the member accounts. The following examples illustrate how this works:

- If you use the root account credentials of your management account to create an OU, your management account is the owner of the resource. (In AWS Organizations, the resource is the OU.)
- If you create an IAM user in your management account and grant permissions to create an OU to that user, the user can create an OU. However, the management account, to which the user belongs, owns the OU resource.
- If you create an IAM role in your management account with permissions to create an OU, anyone who can assume the role can create an OU. The management account, to which the role (not the assuming user) belongs, owns the OU resource.

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 Organizations. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see the IAM User Guide. For information about IAM policy syntax and descriptions, see the IAM JSON 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 Organizations supports only identity-based policies (IAM policies).

Topics

Identity-based permission policies (IAM policies)

You can attach policies to IAM identities to allow those identities to perform operations on AWS resources. For example, you can do the following:

- **Attach a permissions policy to a user or a group in your account** – To grant a user permissions to create an AWS Organizations resource, such as a service control policy (SCP) (p. 104) or an OU, you can attach a permissions policy to a user or a group that the user belongs to. The user or group must be in the organization's management account.
- **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 access to an organization. For example, the administrator in the management account can create a role to grant cross-account permissions to a user in a member account as follows:
  1. The management account administrator creates an IAM role and attaches a permissions policy to the role that grants permissions to the organization's resources.
  2. The management account administrator attaches a trust policy to the role that identifies the member account ID as the `Principal` who can assume the role.
  3. The member account administrator can then delegate permissions to assume the role to any users in the member account. Doing this allows users in the member account to create or access resources in the management account and the organization. The principal in the trust policy can also be an AWS service principal if you want to grant permissions to an AWS service to assume the role.

For more information about using IAM to delegate permissions, see Access Management in the *IAM User Guide*.

The following are examples of policies that allows a user to perform the `CreateAccount` action in your organization.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"Stmt1OrgPermissions",
         "Effect":"Allow",
         "Action":[
            "organizations:CreateAccount"
         ],
         "Resource":"*"
      }
   ]
}
```

You can also provide a partial ARN in the `Resource` element of the policy to indicate the type of resource.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowCreatingAccountsOnResource",
         "Effect":"Allow",
         "Action":"organizations:CreateAccount",
         "Resource":"arn:aws:organizations::*:account/*"
      }
   ]
}
```

You can also deny the creation of accounts that do not include specific tags to the account being created.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"DenyCreatingAccountsOnResourceBasedOnTag",
         "Effect":"Deny",
         "Action":"organizations:CreateAccount",
         "Resource":"*",
         "Condition":{
            "StringEquals":{
               "aws:ResourceTag/key":"value"
            }
         }
      }
   ]
}
```

For more information about users, groups, roles, and permissions, see Identities (Users, Groups, and Roles) in the *IAM User Guide*.

Resource-based policies

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

Specifying policy elements: Actions, conditions, effects, and resources

For each AWS Organizations resource, the service defines a set of API operations, or actions, that can interact with or manipulate that resource in some way. To grant permissions for these operations, AWS Organizations defines a set of actions that you can specify in a policy. For example, for the OU resource, AWS Organizations defines actions like the following:

- `AttachPolicy` and `DetachPolicy`
- `CreateOrganizationalUnit` and `DeleteOrganizationalUnit`
- `ListOrganizationalUnits` and `DescribeOrganizationalUnit`

In some cases, performing an API operation might require permissions to more than one action and might require permissions to more than one resource.

The following are the most basic elements that you can use in an IAM permission policy:

- **Action** – Use this keyword to identify the operations (actions) that you want to allow or deny. For example, depending on the specified `Effect`, `organizations:CreateAccount` allows or denies the user permissions to perform the AWS Organizations `CreateAccount` operation. For more information, see IAM JSON Policy Elements: Action in the *IAM User Guide*.
- **Resource** – Use this keyword to specify the ARN of the resource that the policy statement applies to. For more information, see IAM JSON Policy Elements: Resource in the *IAM User Guide*.
- **Condition** – Use this keyword to specify a condition that must be met for the policy statement to apply. `Condition` usually specifies additional circumstances that must be true for the policy to match. For more information, see IAM JSON Policy Elements: Condition in the *IAM User Guide*.
- **Effect** – Use this keyword to specify whether the policy statement allows or denies the action on the resource. If you don't explicitly grant access to (or allow) a resource, access is implicitly denied. You also can explicitly deny access to a resource, which you might do to ensure that a user can't perform the specified action on the specified resource, even if a different policy grants access. For more information, see IAM JSON Policy Elements: Effect in the *IAM User Guide*.
- **Principal** – In identity-based policies (IAM policies), the user that the policy is attached to is automatically and implicitly the 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 Organizations currently supports only identity-based policies, not resource-based policies.

To learn more about IAM policy syntax and descriptions, see the IAM JSON Policy Reference in the *IAM User Guide*.

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

As an administrator of the management account of an organization, you can control access to AWS resources by attaching permissions policies to AWS Identity and Access Management (IAM) identities (users, groups, and roles) within the organization. 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. If the permissions are granted to a role, that role can be assumed by users in other accounts in the organization.

By default, a user has no permissions of any kind. All permissions must be explicitly granted by a policy. If a permission isn't explicitly granted, it's implicitly denied. If a permission is explicitly denied, that overrules any other policy that might have allowed it. In other words, a user has only those permissions that are explicitly granted and that aren't explicitly denied.

In addition to the basic techniques described in this topic, you can control access to your organization by using the tags applied to the resources in your organization: the organization root, organizational units (OU), accounts, and policies. For more information, see Attribute-based access control with tags and AWS Organizations (p. 327).

Granting full admin permissions to a user

You can create an IAM policy that grants full AWS Organizations administrator permissions to an IAM user in your organization. You can do this using the JSON policy editor in the IAM console.

To use the JSON policy editor to create a policy

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation column on the left, choose **Policies**.

   If this is your first time choosing **Policies**, the **Welcome to Managed Policies** page appears. Choose **Get Started**.
3. At the top of the page, choose **Create policy**.
4. Choose the **JSON** tab.
5. Enter the following JSON policy document:

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "organizations:*",
        "Resource": "*"
    }
}
```

6. Choose **Review policy**.

   Note
   You can switch between the **Visual editor** and **JSON** tabs any time. However, if you make changes or choose **Review policy** in the **Visual editor** tab, IAM might restructure your policy to optimize it for the visual editor. For more information, see Policy restructuring in the *IAM User Guide*.
7. On the **Review policy** page, enter a **Name** and an optional **Description** for the policy that you are creating. Review the policy **Summary** to see the permissions that are granted by your policy. Then choose **Create policy** to save your work.

To learn more about creating an IAM policy, see Creating IAM Policies in the *IAM User Guide*.

Granting limited access by actions

If you want to grant limited permissions instead of full permissions, you can create a policy that lists individual permissions that you want to allow in the `Action` element of the IAM permissions policy. As shown in the following example, you can use wildcard (*) characters to grant only the `Describe*` and `List*` permissions, essentially providing read-only access to the organization.

> **Note**
>
> In a service control policy (SCP), the wildcard (*) character in an `Action` element can be used only by itself or at the end of the string. It can't appear at the beginning or middle of the string. Therefore, `"servicename:action*"` is valid, but `"servicename:*action"` and `"servicename:some*action"` are both invalid in SCPs.

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

For a list of all the permissions that are available to assign in an IAM policy, see Actions Defined by AWS Organizations in the *IAM User Guide*.

Granting access to specific resources

In addition to restricting access to specific actions, you can restrict access to specific entities in your organization. The `Resource` elements in the examples in the preceding sections both specify the wildcard character ("*"), which means "any resource that the action can access." Instead, you can replace the "*" with the Amazon Resource Name (ARN) of specific entities to which you want to allow access.

Example: Granting permissions to a single OU

The first statement of the following policy allows an IAM user read access to the entire organization, but the second statement allows the user to perform AWS Organizations administrative actions only within a single, specified organizational unit (OU). This does not extend to any child OUs. No billing access is granted. Note that this doesn't give you administrative access to the AWS accounts in the OU. It grants only permissions to perform AWS Organizations operations on the accounts within the specified OU:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "organizations:Describe*",
        "organizations:List*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "organizations:*",
      "Resource": "arn:aws:organizations::<masterAccountId>:ou/o-<organizationId>/ou-
<organizationalUnitId>"
    }
  ]
}
```

You get the IDs for the OU and the organization from the AWS Organizations console or by calling the `List*` APIs. The user or group that you apply this policy to can perform any action (`"organizations:*"`) on any entity that is directly contained in the specified OU. The OU is identified by the Amazon Resource Name (ARN).

For more information about the ARNs for various resources, see Resources Defined by AWS Organizations in the *IAM User Guide*.

Granting the ability to enable trusted access to limited service principals

You can use the `Condition` element of a policy statement to further limit the circumstances where the policy statement matches.

Example: Granting permissions to enable trusted access to one specified service

The following statement shows how you can restrict the ability to enable trusted access to only those services that you specify. If the user tries to call the API with a different service principal than the one for AWS IAM Identity Center (successor to AWS Single Sign-On), this policy doesn't match and the request is denied:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "organizations:EnableAWSServiceAccess",
            "Resource": "*",
            "Condition": {
                "StringEquals" : {
                    "organizations:ServicePrincipal" : "sso.amazonaws.com"
                }
            }
        }
    ]
}
```

For more information about the ARNs for various resources, see Resources Defined by AWS Organizations in the *IAM User Guide*.

Attribute-based access control with tags and AWS Organizations

Attribute-based access control let you use administrator-managed attributes such as tags attached to both AWS resources and AWS identities to control access to those resources. For example, you can specify that a user can access a resource when both the user and the resource have the same value for a certain tag.

AWS Organizations taggable resources include AWS accounts, the organization's root, organizational units (OUs), or policies. When you attach tags to Organizations resources, you can then use those tags to control who can access those resources. You do this by adding `Condition` elements to your AWS Identity and Access Management (IAM) permissions policy statements that check whether certain tag keys and values are present before allowing the action. This enables you to create an IAM policy that effectively says "Allow the user to manage only those OUs that have a tag with a key `X` and a value `Y`" or "Allow the user to manage only those OUs that are tagged with a key `Z` that has the same value as the user's attached tag key `Z`."

You can base your `Condition` tests on different types of tag references in an IAM policy.

- Checking the tags that are attached to resources specified in the request (p. 328)
- Checking the tags that are attached to the IAM user or role who is making the request (p. 328)
- Check the tags that are included as parameters in the request (p. 329)

For more information about using tags for access control in policies, see Controlling access to and for IAM users and roles using resource tags. For complete syntax of IAM permission policies, see the IAM JSON Policy Reference

Checking the tags that are attached to resources specified in the request

When you make a request by using the AWS Management Console, the AWS Command Line Interface (AWS CLI), or one of the AWS SDKs, you specify what resources you want to access with that request. Whether you are trying to list available resources of a given type, read a resource, or write to, modify, or update a resource, you specify the resource to access as a parameter in the request. Such requests are controlled by IAM permissions policies that you attach to your users and roles. In these policies, you can compare the tags attached to the requested resource and choose to allow or deny access based on the keys and values of those tags.

To check a tag that is attached to the resource, you reference the tag in a `Condition` element by prefacing the tag key name with the following string: `aws:ResourceTag/`

For example, the following sample policy allows the user or role to perform any AWS Organizations operation ***unless*** that resource has a tag with the key `department` and the value `security`. If that key and value is present, then the policy explicitly denies the `UntagResource` operation.

```
{
    "Version" : "2012-10-17",
    "Statement" : [
        {
            "Effect" : "Allow",
            "Action" : "organizations:*",
            "Resource" : "*"

        },
        {
            "Effect" : "Deny",
            "Action" : "organizations:UntagResource",
            "Resource" : "*",
            "Condition" : {
                "StringEquals" : {
                    "aws:ResourceTag/department" : "security"
                }
            }
        }
    ]
}
```

For more information about how to use this element, see Controlling access to resource and aws:ResourceTag in the *IAM User Guide*.

Checking the tags that are attached to the IAM user or role who is making the request

You can control what the person making the request (the principal) is allowed to do based on the tags that are attached to that person's IAM user or role. To do this, use the `aws:PrincipalTag/key-name` condition key to specify which tag and value must be attached to the calling user or role.

The following example shows how to allow an action only when the specified tag (`cost-center`) has the same value on both the principal calling the operation, and the resource being accessed by the operation. In this example, the calling user can start and stop an Amazon EC2 instance only if the instance is tagged with the same `cost-center` value as the user.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "ec2:startInstances",
            "ec2:stopInstances"
        ],
        "Resource": "*",
        "Condition": {"StringEquals":
            {"ec2:ResourceTag/cost-center": "${aws:PrincipalTag/cost-center}"}}
    }
}
```

For more information about how to use this element, see Controlling access for IAM principals and `aws:PrincipalTag` in the *IAM User Guide*.

Check the tags that are included as parameters in the request

Several operations enable you to specify tags as part of the request. For example, when you create a resource you can specify the tags that are attached to the new resource. You can specify a `Condition` element that uses `aws:TagKeys` to allow or deny the operation based on whether a specific tag key, or a set of keys, is included in the request. This comparison operator doesn't care what value the tag contains. It only checks whether a tag with the specified key is present.

To check the tag key, or a list of keys, specify a `Condition` element with the following syntax:

```
"aws:TagKeys": [ "tag-key-1", "tag-key-2", ... , "tag-key-n" ]
```

You can use `ForAllValues:` to preface the comparison operator to ensure that all of the keys in the request must match one of the keys specified in the policy. For example, the following sample policy allows any Organizations operation only if ***all three*** of the specified tag keys are present in the request.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "organizations:*",
        "Resource": "*",
        "Condition": {
            "ForAllValues:StringEquals": {
                "aws:TagKeys": [
                    "department",
                    "costcenter",
                    "manager"
                ]
            }
        }
    }
}
```

Alternatively, you can use `ForAnyValue:` to preface a comparison operator to ensure that at least one of the keys in the request must match one of the keys specified in the policy. For example, the following policy allows an Organizations operation only if ***at least one*** of the specified tag keys is present in the request.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "organizations:*",
        "Resource": "*",
        "Condition": {
            "ForAnyValue:StringEquals": {
                "aws:TagKeys": [
                    "stage",
                    "region",
                    "domain"
                ]
            }
        }
    }
}
```

Several operations enable you to specify tags in the request. For example, when you create a resource you can specify the tags that are attached to the new resource. You can compare a tag key-value pair in the policy with a key-value pair that is included with the request. To do this, reference the tag in a `Condition` element by prefacing the tag key name with the following string: `aws:RequestTag/key-name` and then specify the tag value that must be present.

For example, the following sample policy denies any request by the user or role to create an AWS account where the request is either missing the `costcenter` tag, or provides that tag with a value other than `1`, `2`, or `3`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "organizations:CreateAccount",
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:RequestTag/costcenter": "true"
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": "organizations:CreateAccount",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringNotEquals": {
                    "aws:RequestTag/costcenter": [
                        "1",
                        "2",
                        "3"
                    ]
                }
            }
        }
    ]
}
```

For more information about how to use these elements, see aws:TagKeys and aws:RequestTag in the *IAM User Guide*.

Logging and monitoring in AWS Organizations

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 Organizations currently supports two AWS services that enable you to monitor your organization and the activity that happens within it.

Topics

Logging AWS Organizations API calls with AWS CloudTrail

AWS Organizations is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS Organizations. CloudTrail captures all API calls for AWS Organizations as events, including calls from the AWS Organizations console and from code calls to the AWS Organizations APIs. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS Organizations. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to AWS Organizations, the IP address it was made from, who made it, when it was made, and additional details.

To learn more about CloudTrail, see the *AWS CloudTrail User Guide*.

> **Important**
> You can view all CloudTrail information for AWS Organizations only in the US East (N. Virginia) Region. If you don't see your AWS Organizations activity in the CloudTrail console, set your console to **US East (N. Virginia)** using the menu in the upper-right corner. If you query CloudTrail with the AWS CLI or SDK tools, direct your query to the US East (N. Virginia) endpoint.

AWS Organizations information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in AWS Organizations, 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 Organizations, create a trail. A trail enables CloudTrail to deliver log files to an Amazon S3 bucket. When CloudTrail logging is enabled in your AWS account, API calls made to AWS Organizations actions are tracked in CloudTrail log files, where they are written with other AWS service records. You can configure other AWS services to further analyze and act on 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

All AWS Organizations actions are logged by CloudTrail and are documented in the AWS Organizations API Reference. For example, calls to `CreateAccount` (including the `CreateAccountResult` event), `ListHandshakesForAccount`, `CreatePolicy`, and `InviteAccountToOrganization` generate entries in the CloudTrail log files.

Every log entry contains information about who generated the request. The user identity information in the log entry helps you determine the following:

- Whether the request was made with root or IAM user credentials
- Whether the request was made with temporary security credentials for an IAM role or a federated user

- Whether the request was made by another AWS service

For more information, see the CloudTrail userIdentity Element.

Understanding AWS Organizations 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 aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

Example log entries: CloseAccount

The following example shows a CloudTrail log entry for a sample `CloseAccount` call that is generated when the API is called and the workflow to close the account starts processing in the background.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAMVNPBQA3EXAMPLE:my-admin-role",
        "arn": "arn:aws:sts::111122223333:assumed-role/my-admin-role/my-session-id",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDAMVNPBQA3EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/my-admin-role",
                "accountId": "111122223333",
                "userName": "my-session-id"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2022-03-18T18:17:06Z"
            }
        }
    },
    "eventTime": "2022-03-18T18:17:06Z",
    "eventSource": "organizations.amazonaws.com",
    "eventName": "CloseAccount",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.168.0.1",
    "userAgent":  "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": {
        "accountId": "555555555555"
    },
    "responseElements": null,
    "requestID": "e28932f8-d5da-4d7a-8238-ef74f3d5c09a",
    "eventID": "19fe4c10-f57e-4cb7-a2bc-6b5c30233592",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

The following example shows a CloudTrail log entry for a `CloseAccountResult` call after the background workflow to close the account successfully completes.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "accountId": "111122223333",
    "invokedBy": "organizations.amazonaws.com"
  },
  "eventTime": "2022-03-18T18:17:06Z",
  "eventSource": "organizations.amazonaws.com",
  "eventName": "CloseAccountResult",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "organizations.amazonaws.com",
  "userAgent": "organizations.amazonaws.com",
  "requestParameters": null,
  "responseElements": null,
  "eventID": "EXAMPLE8-90ab-cdef-fedc-ba987EXAMPLE",
  "readOnly": false,
  "eventType": "AwsServiceEvent",
  "readOnly": false,
  "eventType": "AwsServiceEvent",
  "managementEvent": true,
  "recipientAccountId": "111122223333",
  "serviceEventDetails": {
    "closeAccountStatus": {
      "accountId": "555555555555",
      "state": "SUCCEEDED",
      "requestedTimestamp": "Mar 18, 2022 6:16:58 PM",
      "completedTimestamp": "Mar 18, 2022 6:16:58 PM"
    }
   },
   "eventCategory": "Management"
}
```

Example log entries: CreateAccount

The following example shows a CloudTrail log entry for a sample `CreateAccount` call that is generated when the API is called and the workflow to create the account starts processing in the background.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAMVNPBQA3EXAMPLE:my-admin-role",
        "arn": "arn:aws:sts::111122223333:assumed-role/my-admin-role/my-session-id",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDAMVNPBQA3EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/my-admin-role",
                "accountId": "111122223333",
                "userName": "my-session-id"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-09-16T21:16:45Z"
            }
        }
    },
    "eventTime": "2018-06-21T22:06:27Z",
    "eventSource": "organizations.amazonaws.com",
    "eventName": "CreateAccount",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.168.0.1",
    "userAgent":  "Mozilla/5.0 (Windows NT 10.0; Win64; x64)...",
    "requestParameters": {
        "tags": [],
        "email": "****",
        "accountName": "****"
    },
    "responseElements": {
        "createAccountStatus": {
            "accountName": "****",
            "state": "IN_PROGRESS",
            "id": "car-examplecreateaccountrequestid111",
            "requestedTimestamp": "Sep 16, 2020 9:20:50 PM"
        }
    },
    "requestID": "EXAMPLE8-90ab-cdef-fedc-ba987EXAMPLE",
    "eventID": "EXAMPLE8-90ab-cdef-fedc-ba987EXAMPLE",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111111111111"
}
```

The following example shows a CloudTrail log entry for a `CreateAccount` call after the background workflow to create the account successfully completes.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "accountId": "111122223333",
    "invokedBy": "..."
  },
  "eventTime": "2020-09-16T21:20:53Z",
  "eventSource": "organizations.amazonaws.com",
  "eventName": "CreateAccountResult",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "....",
  "requestParameters": null,
  "responseElements": null,
  "eventID": "EXAMPLE8-90ab-cdef-fedc-ba987EXAMPLE",
  "readOnly": false,
  "eventType": "AwsServiceEvent",
  "recipientAccountId": "111122223333",
  "serviceEventDetails": {
    "createAccountStatus": {
      "id": "car-examplecreateaccountrequestid111",
      "state": "SUCCEEDED",
      "accountName": "****",
      "accountId": "444455556666",
      "requestedTimestamp": "Sep 16, 2020 9:20:50 PM",
      "completedTimestamp": "Sep 16, 2020 9:20:53 PM"
    }
  }
}
```

The following example shows a CloudTrail log entry that is generated after a `CreateAccount` background workflow fails to create the account.

```

  {
  "eventVersion": "1.06",
  "userIdentity": {
    "accountId": "111122223333",
    "invokedBy": "AWS Internal"
  },
  "eventTime": "2018-06-21T22:06:27Z",
  "eventSource": "organizations.amazonaws.com",
  "eventName": "CreateAccountResult",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "AWS Internal",
  "userAgent": "AWS Internal",
  "requestParameters": null,
  "responseElements": null,
  "eventID": "EXAMPLE8-90ab-cdef-fedc-ba987EXAMPLE",
  "readOnly": false,
  "eventType": "AwsServiceEvent",
  "recipientAccountId": "111122223333",
  "serviceEventDetails": {
    "createAccountStatus": {
      "id": "car-examplecreateaccountrequestid111",
      "state": "FAILED",
      "accountName": "****",
      "failureReason": "EMAIL_ALREADY_EXISTS",
      "requestedTimestamp": Jun 21, 2018 10:06:27 PM,
      "completedTimestamp": Jun 21, 2018 10:07:15 PM
    }
  }
}
```

Example log entry: CreateOrganizationalUnit

The following example shows a CloudTrail log entry for a sample `CreateOrganizationalUnit` call.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAMVNPBQA3EXAMPLE",
        "arn": "arn:aws:iam::111111111111:user/diego",
        "accountId": "111111111111",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "diego"
    },
    "eventTime": "2017-01-18T21:40:11Z",
    "eventSource": "organizations.amazonaws.com",
    "eventName": "CreateOrganizationalUnit",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/55.0.2883.95 Safari/537.36",
    "requestParameters": {
        "name": "OU-Developers-1",
        "parentId": "r-a1b2"
    },
    "responseElements": {
        "organizationalUnit": {
            "arn": "arn:aws:organizations::111111111111:ou/o-aa111bb222/ou-
examplerootid111-exampleouid111",
            "id": "ou-examplerootid111-exampleouid111",
            "name": "test-cloud-trail"
        }
    },
    "requestID": "EXAMPLE8-90ab-cdef-fedc-ba987EXAMPLE",
    "eventID": "EXAMPLE8-90ab-cdef-fedc-ba987EXAMPLE",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111111111111"
}
```

Example log entry: InviteAccountToOrganization

The following example shows a CloudTrail log entry for a sample `InviteAccountToOrganization` call.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAMVNPBQA3EXAMPLE",
        "arn": "arn:aws:iam::111111111111:user/diego",
        "accountId": "111111111111",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "diego"
    },
    "eventTime": "2017-01-18T21:41:17Z",
    "eventSource": "organizations.amazonaws.com",
    "eventName": "InviteAccountToOrganization",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/55.0.2883.95 Safari/537.36",
    "requestParameters": {
        "notes": "This is a request for Mary's account to join Diego's organization.",
        "target": {
            "type": "ACCOUNT",
            "id": "111111111111"
        }
    },
    "responseElements": {
        "handshake": {
            "requestedTimestamp": "Jan 18, 2017 9:41:16 PM",
            "state": "OPEN",
            "arn": "arn:aws:organizations::111111111111:handshake/o-aa111bb222/invite/h-
examplehandshakeid111",
            "id": "h-examplehandshakeid111",
            "parties": [
                {
                    "type": "ORGANIZATION",
                    "id": "o-aa111bb222"
                },
                {
                    "type": "ACCOUNT",
                    "id": "222222222222"
                }
            ],
            "action": "invite",
            "expirationTimestamp": "Feb 2, 2017 9:41:16 PM",
            "resources": [
                {
                    "resources": [
                        {
                            "type": "MASTER_EMAIL",
                            "value": "diego@example.com"
                        },
                        {
                            "type": "MASTER_NAME",
                            "value": "Management account for organization"
                        },
                        {
                            "type": "ORGANIZATION_FEATURE_SET",
                            "value": "ALL"
                        }
                    ],
                    "type": "ORGANIZATION",
                    "value": "o-aa111bb222"
                },
                {
                    "type": "ACCOUNT",
                    "value": "222222222222"
                },
                {
                    "type": "NOTES",
                    "value": "This is a request for Mary's account to join Diego's
 organization."
                }
            ]
        }
    },
    "requestID": "EXAMPLE8-90ab-cdef-fedc-ba987EXAMPLE",
    "eventID": "EXAMPLE8-90ab-cdef-fedc-ba987EXAMPLE",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111111111111"
}
```

Example log entry: AttachPolicy

The following example shows a CloudTrail log entry for a sample `AttachPolicy` call. The response indicates that the call failed because the requested policy type isn't enabled in the root where the request to attach was attempted.

```
{
    "eventVersion": "1.06",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAMVNPBQA3EXAMPLE",
        "arn": "arn:aws:iam::111111111111:user/diego",
        "accountId": "111111111111",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "diego"
    },
    "eventTime": "2017-01-18T21:42:44Z",
    "eventSource": "organizations.amazonaws.com",
    "eventName": "AttachPolicy",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/55.0.2883.95 Safari/537.36",
    "errorCode": "PolicyTypeNotEnabledException",
    "errorMessage": "The given policy type ServiceControlPolicy is not enabled on the
 current view",
    "requestParameters": {
        "policyId": "p-examplepolicyid111",
        "targetId": "ou-examplerootid111-exampleouid111"
    },
    "responseElements": null,
    "requestID": "EXAMPLE8-90ab-cdef-fedc-ba987EXAMPLE",
    "eventID": "EXAMPLE8-90ab-cdef-fedc-ba987EXAMPLE",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111111111111"
}
```

Amazon CloudWatch Events

AWS Organizations 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.

For a tutorial that shows how to enable CloudWatch Events to monitor key activity in your organization, see Tutorial: Monitor important changes to your organization with CloudWatch Events (p. 18).

To learn more about CloudWatch Events, including how to configure and enable it, see the *Amazon CloudWatch Events User Guide*.

Compliance validation for AWS Organizations

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 Organizations or other AWS services are within the 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 on Amazon Web Services – This whitepaper describes how companies can use AWS to create HIPAA-eligible applications.

  Note
  Not all AWS services are HIPAA eligible. For more information, see the HIPAA Eligible Services Reference.
- 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 Organizations

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 Organizations

As a managed service, AWS Organizations 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 Organizations through the network. Clients must support Transport Layer Security (TLS). 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.

If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

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.

AWS Organizations reference

Use the topics in this section to find detailed reference information for various aspects of AWS Organizations.

Topics

Quotas for AWS Organizations

This section specifies quotas that affect AWS Organizations.

Naming guidelines

The following are guidelines for names that you create in AWS Organizations, including names of accounts, organizational units (OUs), roots, and policies:

- They must be composed of Unicode characters
- Maximum string length for names vary by the object. To see actual limit for each, see the AWS Organizations API Reference and find the API operation that creates the object. Look at the details for that operation's `Name` parameter. For example: Account name, or OU name.

Maximum and minimum values

The following are the ***default*** maximums for entities in AWS Organizations.

> **Note**
> You can request increases for some of these values by using the Service Quotas console. Organizations is a global service that is physically hosted in the US East (N. Virginia) Region (`us-east-1`). Therefore, you must use `us-east-1` to access Organizations quotas when using the Service Quotas console, the AWS CLI, or an AWS SDK.

Number of AWS accounts in an organization	10 — The default maximum number of accounts allowed in an organization. If you need more, you can request an increase by using the Service Quotas console. An invitation sent to an account counts against this quota. The count is returned if the invited account declines, the management account cancels the invitation, or the invitation expires.
Number of roots in an organization	1
Number of OUs in an organization	1000
Number of policies of each type in an organization	1000 per policy type
Maximum size of a policy document	Service control policies: 5120 bytes *(not characters)*

	AI services opt-out policies: 2500 characters Backup policies: 10,000 characters Tag policies: 10,000 characters **Note:** If you save the policy by using the AWS Management Console, extra white space (such as spaces and line breaks) between JSON elements and outside of quotation marks, is removed and not counted. If you save the policy using an SDK operation or the AWS CLI, then the policy is saved exactly as you provided and no automatic removal of characters occurs.
OU maximum nesting in a root	Five levels of OUs deep under a root.
Maximum number of invitation attempts you can perform in a 24-hour period	Either 20 or the maximum number of accounts allowed in your organization, whichever is greater. Accepted invitations don't count against this quota. As soon as one invitation is accepted, you can send another invitation that same day. If the maximum number of accounts allowed in your organization is less than 20, then you get an "account limit exceeded" exception if you attempt to invite more accounts than your organization can contain. However, you can cancel invitations and send new ones up to the maximum of 20 attempts in one day.
Number of member accounts you can create concurrently	5 — As soon as one finishes, you can start another, but only five can be in progress at a time.
Number of member accounts you can close in a 30 day period	10% of active member accounts in an organization can be closed within a 30 day period. The maximum account closure is 200, even if 10% active accounts exceeds 200.
Number of member accounts you can close concurrently	3 — Only three account closures can be in progress at the same time. As soon as one finishes, you can close another account.
Number of entities to which you can attach a policy	Unlimited
Number of tags that you can attach to a root, OU, or account	50

Expiration times for handshakes

The following are the timeouts for handshakes in AWS Organizations.

Invitation to join an organization	15 days
Request to enable all features in an organization	90 days

Handshake is deleted and no longer appears in lists	30 days after the handshake is completed

Number of policies that you can attach to an entity

The minimum and maximum depend on the policy type and the entity that you're attaching the policy to. The following table shows each policy type and the number of entities that you can attach each type to.

> **Note**
> These numbers apply to only those policies that are directly attached to an OU or an account. Policies that affect an OU or account by inheritance do ***not*** count against these limits.

Policy type	Minimum attached to an entity	Maximum attached to root	Maximum attached per OU	Maximum attached per account
Service control policy	1 — Every entity must have *at least* one SCP attached at all times. You can't remove the last SCP from an entity.	5	5	5
AI services opt-out policy	0	5	5	5
Backup policy	0	10	10	10
Tag policy	0	10	10	10

> **Note**
> Currently, you can have only one root in an organization.

AWS managed policies available for use with AWS Organizations

This section identifies the AWS-managed policies provided for your use to manage your organization. You can't modify or delete an AWS managed policy, but you can attach or detach them to entities in your organization as needed.

AWS Organizations managed policies for use with AWS Identity and Access Management (IAM)

An IAM managed policy is provided and maintained by AWS. A managed policy provides permissions for common tasks that you can assign to your users by attaching the managed policy to the appropriate IAM user or role object. You don't have to write the policy yourself, and when AWS updates the policy as appropriate to support new services, you automatically and immediately get the benefit of the update. You can see the list of AWS managed policies in Policies page on the IAM console. Use the **Filter policies** drop-down to select **AWS managed**.

You can use the following managed policies to grant permissions to users in your organization.

<table>
<tr><th>Policy name</th><th>Description</th><th>ARN</th></tr>
<tr><td>AWSOrganizationsFullAccess</td><td>Provides all of the permissions required to create and fully administer an organization. The content of this policy statement is shown in the following snippet:
<pre>
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",

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

 "account:PutAlternateContact",

 "account:DeleteAlternateContact",

 "account:GetAlternateContact"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",

 "Action":"iam:CreateServiceLinkedRole",
         "Resource":"*",
         "Condition":{
            "StringEquals":{

 "iam:AWSServiceName":"organizations.amazonaws.com"
            }
         }
      }
   ]
}
</pre></td><td>arn:aws:iam::aws:policy/
AWSOrganizationsFullAccess</td></tr>
<tr><td>AWSOrganizationsReadOnlyAccess</td><td>Provides read only access to information about the organization. It doesn't permit the user to make any changes. The content of this policy statement is shown in the following snippet:
<pre>
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[

 "organizations:Describe*",
</pre></td><td>arn:aws:iam::aws:policy/
AWSOrganizationsReadOnlyAccess</td></tr>
</table>

Policy name	Description	ARN

	"organizations:List*"
	],
	"Resource":"*"
	},
	{
	"Effect":"Allow",
	"Action":[
	"account:GetAlternateContact"
	],
	"Resource":"*"
	}
	]
	}	

Updates to Organizations AWS managed policies

The following table details updates to AWS managed policies since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the AWS Organizations Document History page (p. 353).

Change	Description	Date
AWSOrganizationsFullAccess – updated to allow creating an organization.	Organizations added the `CreateServiceLinkedRole` permission to the policy to enable creating the service linked role required to create an organization. The permission is restricted to creating a role that can be used only by the `organizations.amazonaws.com` service.	August 24, 2022
AWSOrganizationsFullAccess – updated to allow account API permissions required to add, edit, or delete account alternate contacts via the Organizations console.	Organizations added the `account:GetAlternateContact`, `account:DeleteAlternateContact`, `account:PutAlternateContact` actions to the policy to enable write access to modify alternate contacts for an account.	February 7, 2022
AWSOrganizationsReadOnlyAccess – updated to allow account API permissions required to view account alternate contacts via the Organizations console.	Organizations added the `account:GetAlternateContact` action to the policy to enable access to view alternate contacts for an account.	February 7, 2022

AWS Organizations managed service control policies

Service control policies (SCPs) (p. 104) are similar to IAM permission policies, but are a feature of AWS Organizations rather than IAM. You use SCPs to specify maximum permissions for affected entities. You can attach SCPs to roots, organizational units (OUs), or accounts in your organization. You can create your own, or you can use the policies that IAM defines. You can see the list of policies in your organization on the Policies page on the Organizations console.

Important
Every root, OU, and account must have at least one SCP attached at all times.

Policy name	Description	ARN
FullAWSAccess	Provides AWS Organizations management account access to member accounts.	arn:aws:organizations::aws:policy/service_control_policy/p-FullAWSAccess

Troubleshooting AWS Organizations

If you encounter issues when working with AWS Organizations, consult the topics in this section.

Topics

Troubleshooting general issues

Use the information here to help you diagnose and fix access-denied or other common issues that you might encounter when working with AWS Organizations.

Topics

I get an "access denied" message when I make a request to AWS Organizations

- Verify that you have permissions to call the action and resource that you have requested. An administrator must grant permissions by attaching an IAM policy to your IAM user or to a group that you're a member of. If the policy statements that grant those permissions include any conditions, such as time-of-day or IP address restrictions, you also must meet those requirements when you send the request. For information about viewing or modifying policies for an IAM user, group, or role, see Working with Policies in the *IAM User Guide*.
- If you are signing API requests manually (without using the AWS SDKs), verify that you have correctly signed the request.

I get an "access denied" message when I make a request with temporary security credentials

- Verify that the IAM user or role that you are using to make the request has the correct permissions. Permissions for temporary security credentials are derived from an IAM user or role, so the permissions are limited to those granted to the IAM user or role. For more information about how permissions for temporary security credentials are determined, see Controlling Permissions for Temporary Security Credentials in the *IAM User Guide*.

- Verify that your requests are being signed correctly and that the request is well formed. For details, see the toolkit documentation for your chosen SDK or Using Temporary Security Credentials to Request Access to AWS Resources in the *IAM User Guide*.
- Verify that your temporary security credentials haven't expired. For more information, see Requesting Temporary Security Credentials in the *IAM User Guide*.

I get an "access denied" message when I try to leave an organization as a member account or remove a member account as the management account

- You can remove a member account only after you enable IAM user access to billing in the member account. For more information, see Activating Access to the Billing and Cost Management Console in the *AWS Billing User Guide*.
- You can remove an account from your organization only if the account has the information required for it to operate as a standalone account. When you create an account in an organization using the AWS Organizations console, API, or AWS CLI commands, that information isn't automatically collected. For an account that you want to make standalone, you must accept the AWS Customer Agreement, choose a support plan, provide and verify the required contact information, and provide a current payment method. AWS uses the payment method to charge for any billable (not AWS Free Tier) AWS activity that occurs while the account isn't attached to an organization. For more information, see Leaving an organization as a member account (p. 72).

I get a "quota exceeded" message when I try to add an account to my organization

There is a maximum number of accounts that you can have in an organization. Deleted or closed accounts continue to count against this quota.

An invitation to join counts against the maximum number of accounts in your organization. The count is returned if the invited account declines, the management account cancels the invitation, or the invitation expires.

- Before you close or delete an AWS account, remove it from your organization (p. 69) so that it doesn't continue to count against your quota.
- See Maximum and minimum values (p. 340) for information about how to request a quota increase.

I get a "this operation requires a wait period" message while adding or removing accounts

Some actions require a wait period. For example, you can't immediately remove newly created accounts. Try the action again in a few days. If you experience issues with account quotas while adding and removing accounts, see Maximum and minimum values (p. 340) for information about how to request a quota increase.

I get an "organization is still initializing" message when I try to add an account to my organization

If you receive this error and it's been over an hour since you created the organization, contact AWS Support.

I get an "Invitations are disabled" message when I try to invite an account to my organization.

This happens when you enable all features in your organization (p. 35). This operation can take some time and requires that all member accounts respond. Until the operation is completed, you can't invite new accounts to join the organization.

Changes that I make aren't always immediately visible

As a service that is accessed through computers in data centers around the world, AWS Organizations uses a distributed computing model called eventual consistency. Any change that you make in AWS Organizations takes time to become visible from all possible endpoints. Some of the delay results from the time it takes to send the data from server to server or from replication zone to replication zone. AWS Organizations also uses caching to improve performance, but in some cases this can add time. The change might not be visible until the previously cached data times out.

Design your global applications to account for these potential delays and ensure that they work as expected, even when a change made in one location isn't instantly visible at another.

For more information about how some other AWS services are affected by this, consult the following resources:

- Managing Data Consistency in the *Amazon Redshift Database Developer Guide*
- Amazon S3 Data Consistency Model in the *Amazon Simple Storage Service User Guide*
- Ensuring Consistency When Using Amazon S3 and Amazon Elastic MapReduce for ETL Workflows in the AWS Big Data Blog
- EC2 Eventual Consistency in the *Amazon EC2 API Reference*.

Troubleshooting AWS Organizations policies

Use the information here to help you diagnose and fix common errors found in AWS Organizations policies.

Service control policies

Service control policies (SCPs) in AWS Organizations are similar to IAM policies and share a common syntax. This syntax begins with the rules of JavaScript Object Notation (JSON). JSON describes an *object* with name and value pairs that make up the object. The IAM policy grammar builds on that by defining what names and values have meaning for, and are understood by, the AWS services that use policies to grant permissions.

AWS Organizations uses a subset of the IAM syntax and grammar. For details, see SCP syntax (p. 119).

Common policy errors

- More than one policy object (p. 349)
- More than one statement element (p. 349)
- Policy document exceeds maximum size (p. 350)

More than one policy object

An SCP must consist of one and only one JSON object. You denote an object by placing { } braces around it. Although you can nest other objects within a JSON object by embedding additional { } braces within the outer pair, a policy can contain only one outermost pair of { } braces. The following example is ***incorrect*** because it contains two objects at the top level (called out in *red*):

```
{
  "Version": "2012-10-17",
  "Statement":
  {
     "Effect":"Allow",
     "Action":"ec2:Describe*",
     "Resource":"*"
  }
}
{
  "Statement": {
     "Effect": "Deny",
     "Action": "s3:*",
     "Resource": "*"
  }
}
```

You can, however, meet the intention of the preceding example with the use of correct policy grammar. Instead of including two complete policy objects, each with its own `Statement` element, you can combine the two blocks into a single `Statement` element. The `Statement` element has an array of two objects as its value, as shown in the following example:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:Describe*",
      "Resource":" *"
    },
    {
      "Effect": "Deny",
      "Action": "s3:*",
      "Resource": "*"
    }
  ]
}
```

This example cannot be further compressed into a `Statement` with one element because the two elements have different effects. Generally, you can combine statements only when the `Effect` and `Resource` elements in each statement are identical.

More than one statement element

This error might at first appear to be a variation on the error in the preceding section. However, syntactically it's a different type of error. In the following example, there is only one policy object as denoted by a single pair of { } braces at the top level. However, that object contains two `Statement` elements within it.

An SCP must contain only one `Statement` element, consisting of the name (`Statement`) appearing to the left of a colon, followed by its value on the right. The value of a `Statement` element must be an object, denoted by { } braces, containing one `Effect` element, one `Action` element, and one `Resource` element. The following example is ***incorrect*** because it contains two `Statement` elements in the policy object:

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "ec2:Describe*",
    "Resource": "*"
  },
  "Statement": {
    "Effect": "Deny",
    "Action": "s3:*",
    "Resource": "*"
  }
}
```

Because a value object can be an array of multiple value objects, you can solve this problem by combining the two `Statement` elements into one element with an object array, as shown in the following example:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:Describe*",
      "Resource":"*"
    },
    {
      "Effect": "Deny",
      "Action": "s3:*",
      "Resource": "*"
    }
 ]
}
```

The value of the `Statement` element is an object array. The array in the example consists of two objects, each of which is a correct value for a `Statement` element. Each object in the array is separated by commas.

Policy document exceeds maximum size

The maximum size of an SCP document is 5,120 bytes. This maximum size includes all characters, including white space. To reduce the size of your SCP, you can remove all white space characters (such as spaces and line breaks) that are outside quotation marks.

Calling the API by making HTTP Query requests

This section contains general information about using the Query API for AWS Organizations. For details about the API operations and errors, see the AWS Organizations API Reference.

> **Note**
> Instead of making direct calls to the AWS Organizations Query API, you can use one of the AWS SDKs. The AWS SDKs consist of libraries and sample code for various programming languages and platforms (Java, Ruby, .NET, iOS, Android, and more). The SDKs provide a convenient way to create programmatic access to AWS Organizations and AWS. For example, the SDKs take care of tasks such as cryptographically signing requests, managing errors, and retrying requests automatically. For information about the AWS SDKs, including how to download and install them, see Tools for Amazon Web Services.

The Query API for AWS Organizations lets you call service actions. Query API requests are HTTPS requests that must contain an `Action` parameter to indicate the operation to be performed. AWS Organizations supports GET and POST requests for all operations. That is, the API doesn't require you to use GET for some actions and POST for others. However, GET requests are subject to the limitation size of a URL. Although this limit is browser dependent, a typical limit is 2048 bytes. Therefore, for Query API requests that require larger sizes, you must use a POST request.

The response is an XML document. For details about the response, see the individual action pages in the AWS Organizations API Reference.

Topics

Endpoints

AWS Organizations has a single global API endpoint that is hosted in the US East (N. Virginia) Region.

For more information about AWS endpoints and regions for all services, see Regions and Endpoints in the *AWS General Reference*.

HTTPS required

Because the Query API returns sensitive information such as security credentials, you must use HTTPS to encrypt all API requests.

Signing AWS Organizations API requests

Requests must be signed using an access key ID and a secret access key. We strongly recommend that you don't use your AWS root account credentials for everyday work with AWS Organizations. You can use the credentials for an IAM user or temporary credentials such as you use with an IAM role.

To sign your API requests, you must use AWS Signature Version 4. For information about using Signature Version 4, see Signature Version 4 Signing Process in the *AWS General Reference*.

AWS Organizations doesn't support earlier versions, such as Signature Version 2.

For more information, see the following:

- AWS Security Credentials – Provides general information about the types of credentials that you can use to access AWS
- IAM Best Practices – Offers suggestions for using the IAM service to help secure your AWS resources, including those in AWS Organizations
- Temporary Credentials – Describes how to create and use temporary security credentials

Document history for AWS Organizations

The following table describes major documentation updates for AWS Organizations.

- **API version: 2016-11-28**

Change	Description	Date
Updated the AWSOrganizationsFullAccess managed policy to enable the creation of an organization.	The managed policy was updated to allow creating an organization by adding the permission required to create the service linked role needed by a new organization.	August 24, 2022
Organizations close account capability from the AWS Organizations console.	Principals in the management account can close member accounts from the AWS Organizations console, and protect member accounts from accidental closure by using IAM policies.	March 29, 2022
Updated announcement to update alternate contacts with AWS Organizations console.	Organizations now provides ability to update alternate contacts for accounts within your organization using the AWS Organizations console. Announce new capability and points to Account Management Reference for instructions.	February 8, 2022
Organizations managed policy updates - Update to an existing policy	Updated the AWSOrganizationsFullAccess and AWSOrganizationsReadOnlyAccess managed policies to allow account API permissions required to update or view account alternate contacts via the AWS Organizations console.	February 7, 2022
Organizations integration with Amazon DevOps Guru.	You can integrate Amazon DevOps Guru with AWS Organizations to monitor application health holistically across all of your organization accounts and gain insights.	January 3, 2022
Organizations integration with Amazon Detective.	You can integrate Amazon Detective with AWS Organizations to ensure	December 16, 2021

	that your Detective behavior graph provides visibility into the activity for all of your organization accounts.	
Organizations integration with AWS Config now supports multi-account multi-region data aggregation.	You can use a delegated administrator account to aggregate resource configuration and compliance data from all of the member accounts your organization. For more information, see Multi-account multi-region data aggregationin the *AWS Config Developer Guide*.	June 16, 2021
Organizations integration with AWS Firewall Manager now includes support for a delegated administrator.	You can now designate a member account in your organization to be the Firewall Manager administrator for the entire organization. This allows for better separation of permissions from the organization's management account.	April 30, 2021
Organizations backup policies now support continuous backup.	You can use the AWS Backup continuous backups feature with your organization's backup policies.	March 10, 2021
Organizations integration with AWS CloudFormation StackSets now includes support for a delegated administrator.	You can now designate a member account in your organization to be the AWS CloudFormation StackSets administrator for the entire organization. This allows for better separation of permissions from the organization's management account.	February 18, 2021
Continue inviting accounts while you enable all features	AWS updated the process to enable all features in an organization. You can now continue to invite new accounts to join your organization while you wait for existing accounts to respond to their invitations.	February 3, 2021
Introduces version 2.0 of the AWS Organizations console (p. 353)	AWS introduced a new version of the AWS console. All of the documentation has been updated to reflect the new way of performing tasks.	January 21, 2021

Organizations now supports integration with AWS Marketplace	You can now enable AWS Marketplace to more easily share your software licenses across all of the accounts in your organization.	December 3, 2020
Organizations now supports integration with Amazon S3 Lens	Amazon S3 Lens supports both trusted access and delegated administrator with Organizations. For details, see Amazon S3 Storage Lens in the *Amazon Simple Storage Service User Guide*.	November 18, 2020
Cross-account backup copies	When you use backup policies to backup the resources in your organization, you can now store copies of your backup in other AWS accounts in the organization.	November 18, 2020
AWS Regions in China now support AWS Resource Access Manager as an Organizations trusted service. (p. 353)	You can now use AWS RAM features that integrate with Organizations as a trusted service when you use Organizations and AWS RAM in China.	November 18, 2020
Organizations now supports integration with AWS Security Hub	You can enable Security Hub across all of the accounts in your organization, and designate one of your organization's member accounts as the delegated administrator account for Security Hub.	November 12, 2020
Renamed the master account (p. 353)	AWS Organizations changed the name of the “master account” to “management account”. This is a name change only, and there is no change in functionality.	October 20, 2020
New Best Practices section and topics	Added a new section for best practices for AWS Organizations. The new section includes topics that discuss best practices for the management account and member account root users and password management.	October 6, 2020
Added new best practices section and first two pages	There is a new section for topics that describe best practices for AWS Organizations. This update includes a topic for best practices for an organization's management account and a topic for best practices for member accounts.	October 2, 2020

Organizations backup policies now support application-consistent backups on Windows EC2 instances by using VSS (Volume Shadow Copy Service).	Backup policies support a new `advanced_backup_settings`" section. The first entry in this new section is an `ec2` setting called `WindowsVSS` that you can enable or disable. For details, see Creating a VSS-Enabled Windows Backup in the *AWS Backup Developer Guide*.	September 24, 2020
Organizations supports tag-on-create and tag-based access control	You can add tags to Organizations resources when you create them. You can use tag policies to standardize tag usage on Organizations resources. You can use IAM policies to restrict access to only resources that have specified tag keys and values.	September 15, 2020
Added AWS Health as a trusted service.	You can aggregate AWS Health events across accounts in your organization.	August 4, 2020
Artificial Intelligence (AI) services opt-out policies	You can use AI services opt-out policies to control whether AWS AI services may store and use customer content processed by those services (AI content) for the development and continuous improvement of AWS AI services and technologies.	July 8, 2020
Added backup policies and integration with AWS Backup.	You can use backup policies to create and enforce backup policies across all of the accounts in your organization.	June 24, 2020
Support delegated administration for IAM Access Analyzer.	Enables you to delegate administrative access for Access Analyzer in your organization to a designated member account.	March 30, 2020
Integration with AWS CloudFormation StackSets	You can create a service-managed stack set to deploy stack instances to accounts managed by AWS Organizations.	February 11, 2020
Integration with Compute Optimizer	Compute Optimizer was added as a service that can work with accounts in your organization.	February 4, 2020
Tag policies	You can use tag policies to help standardize tags across resources in your organization's accounts.	November 26, 2019

Integration with Systems Manager	You can synchronize operations data across all AWS accounts in your organization in Systems Manager Explorer.	November 26, 2019
aws:PrincipalOrgPaths	New global condition key checks the AWS Organizations path for the IAM user, IAM role, or AWS account root user who is making the request.	November 20, 2019
Integration with AWS Config rules	You can use AWS Config API operations to manage AWS Config rules across all AWS accounts in your organization.	July 8, 2019
New service for trusted access	Service Quotas added as a service that can work with the accounts in your organization.	June 24, 2019
Integration with AWS Control Tower	AWS Control Tower added as a service that can work with the accounts in your organization.	June 24, 2019
Integration with AWS Identity and Access Management	IAM provides service last accessed data for your organization's entities (the organization root, OUs, and accounts). You can use this data to restrict access to only the AWS services that you need.	June 20, 2019
Tagging accounts	You can tag and untag accounts in your organization and view tags on an account in your organization.	June 6, 2019
Resources, conditions, and the `NotAction` element in service control policies (SCPs)	You can now specify resources, conditions, and the `NotAction` element in SCPs to deny access across accounts in your organization or organizational unit (OU).	March 25, 2019
New services for trusted access	AWS License Manager and AWS Service Catalog added as services that can work with the accounts in your organization.	December 21, 2018
New services for trusted access	AWS CloudTrail and AWS RAM added as services that can work with the accounts in your organization.	December 4, 2018
New service for trusted access	AWS Directory Service added as a service that can work with the accounts in your organization.	September 25, 2018

Email address verification	You must verify that you own the email address that is associated with the management account before you can invite existing accounts to your organization.	September 20, 2018
CreateAccount notifications	`CreateAccount` notifications are published to the management account's CloudTrail logs.	June 28, 2018
New service for trusted access	AWS Artifact added as a service that can work with the accounts in your organization.	June 20, 2018
New services for trusted access	AWS Config and AWS Firewall Manager added as services that can work with the accounts in your organization.	April 18, 2018
Trusted service access	You can now enable or disable access for select AWS services to work in the accounts in your organization. IAM Identity Center is the initial supported trusted service.	March 29, 2018
Account removal is now self-service	You can now remove accounts that were created from within AWS Organizations without contacting AWS Support.	December 19, 2017
Added support for new service AWS IAM Identity Center (successor to AWS Single Sign-On)	AWS Organizations now supports integration with AWS IAM Identity Center (successor to AWS Single Sign-On) (IAM Identity Center).	December 7, 2017
AWS added a service-linked role to all organization accounts	A service-linked role named `AWSServiceRoleForOrganizations` is added to all accounts in an organization to enable integration between AWS Organizations and other AWS services.	October 11, 2017
You can now remove created accounts (p. 353)	Customers can now remove created accounts from their organization, with help from AWS Support.	June 15, 2017
Service launch	Initial version of the AWS Organizations documentation that accompanied the launch of the new service.	February 17, 2017

AWS glossary

For the latest AWS terminology, see the AWS glossary in the *AWS General Reference*.