AWS Organizations

User Guide

AWS Organizations: User Guide

Table of Contents

What Is AWS Organizations?

AWS Organizations is an account 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 master account of your organization to consolidate and pay for all member accounts.

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.

Control over the AWS services and API actions that each account can access

As an administrator of the master 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 (p. 60).

Help for standardizing tags across 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. 86)

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 what permissions 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 multiaccount 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. 4).

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 must enable 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 come with policies that enable the specified service to perform only those tasks that are required by your configuration choices. For more information, see Enabling Trusted Access with Other AWS Services (p. 127).

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. 157).

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

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 multiaccount 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.

AWS Service	**Benefits of Using with AWS Organizations**	
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. 60) 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.*	
AWS Artifact – Download AWS security compliance reports such as ISO and PCI reports.	You can accept agreements on behalf of all accounts within your organization. To use with AWS Organizations, see AWS Artifact and AWS Organizations (p. 130).	
AWS CloudFormation StackSets – Create, update, or delete stacks across multiple accounts and Regions with a single operation.	A user in a master account can create a stack set with service-managed permissions that deploys stack instances to accounts in your organization. For information on using with AWS Organizations, see AWS CloudFormation StackSets and AWS Organizations (p. 131).	
AWS CloudTrail – Enable governance, compliance, and operational and risk auditing of your account.	A user in a master account can create an organization trail that logs all events for all accounts in that organization. For information on using with AWS Organizations, see	

AWS Service	Benefits of Using with AWS Organizations	
	AWS CloudTrail and AWS Organizations (p. 131).	
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*.	
AWS Compute Optimizer – 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*. For information on using with AWS Organizations, see AWS Compute Optimizer and AWS Organizations (p. 132).	
AWS Config – 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 across all AWS accounts in your organization. For information on using with AWS Organizations, see AWS Config and AWS Organizations (p. 132).	
AWS Control Tower – Set up and govern a secure, compliant, multiaccount AWS environment.	You can set up a landing zone, a multiaccount 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 Works and Manage Accounts Through AWS Organizations in the *AWS Control Tower User Guide*.	

AWS Service	Benefits of Using with AWS Organizations	
AWS Directory Service – 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. For information on using with AWS Organizations, see AWS Directory Service and AWS Organizations (p. 133).	
AWS Firewall Manager – Centrally configure and manage firewall rules for web applications across your accounts and applications.	You can centrally configure and manage AWS WAF rules across accounts in your organization. For information on using with AWS Organizations, see AWS Firewall Manager and AWS Organizations (p. 133).	
AWS License Manager – Streamline the process of bringing software licenses to the cloud.	You can enable cross-account discovery of computing resources throughout your organization. For information on using with AWS Organizations, see AWS License Manager and AWS Organizations (p. 134).	
AWS Private Marketplace – Create a custom digital catalog of pre-approved products from AWS Marketplace.	Any new accounts added to an AWS Organizations associated with a private marketplace automatically have access to all authorized products for their company.	

AWS Service	Benefits of Using with AWS Organizations	
AWS RAM – 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*. For information on using with AWS Organizations, see AWS RAM and AWS Organizations (p. 134).	
AWS Service Catalog – 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. For information on using with AWS Organizations, see AWS Service Catalog and AWS Organizations (p. 135).	
Service Quotas – 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. For information on using with AWS Organizations, see Service Quotas and AWS Organizations (p. 135).	
AWS Single Sign-On – Provide single sign-on services for all of your accounts and cloud applications.	Users can sign in to the AWS SSO user portal with their corporate credentials and access resources in their assigned master or member accounts. For more information about setting up trusted access to AWS Organizations, see AWS Single Sign-On and AWS Organizations (p. 136).	

AWS Service	Benefits of Using with AWS Organizations	
AWS Systems Manager – 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. For information on using with AWS Organizations, see AWS Systems Manager and AWS Organizations (p. 136).	
Tag policies (p. 86) – Help standardize tags across resources in your organization's accounts.	You can create tag policies to define tagging rules for specific resources and attach those policies to organization entities. For information on enabling trusted access for tag policies, see Tag Policies and AWS Organizations (p. 137).	

For more information about enabling trusted access to AWS Organizations, see Enabling Trusted Access with Other AWS Services (p. 127).

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. 9)

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.

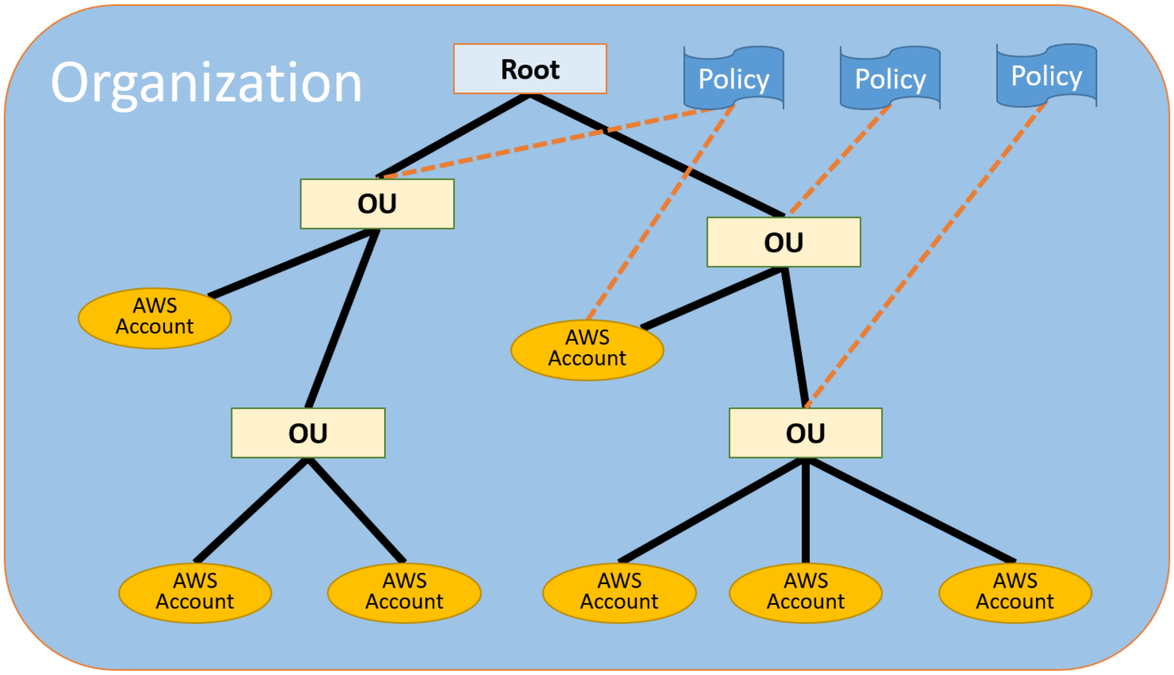
Consolidated Billing for Organizations

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 and Cost Management 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 seven 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

An entity that you create to consolidate your AWS accounts (p. 10) 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 master account along with zero or more member accounts. You can organize the accounts in a hierarchical, tree-like structure with a root (p. 10) at the top and organizational units (p. 10) 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. 11) 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. 10) and accounts (p. 10) in the organization.

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

Organization unit (OU)

A container for accounts (p. 10) within a root (p. 10). 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

A standard AWS account that contains your AWS resources. You can attach a policy to an account to apply controls to only that one account.

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

- The **master account** is the account that creates the organization. From the organization's master 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

  The master 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 master account.
- The rest of the accounts that belong to an organization are called **member accounts**. An account can be a member of only one organization at a time.

Invitation

The process of asking another account (p. 10) to join your organization (p. 10). An invitation can be issued only by the organization's master 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. 11) features to supporting all features (p. 11) in the organization. Invitations work by accounts exchanging handshakes (p. 11). 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. 10). Handshake messages are passed between and responded to by the handshake initiator and the recipient. The messages are passed in a way that ensures that both parties always know what the current status is. Handshakes also are used when changing the organization from supporting only consolidated billing (p. 11) features to supporting all features (p. 11) 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 master account of the organization has full control over what member accounts can do. The master account can apply SCPs (p. 60) to restrict the services and actions that users (including the root user) and roles in an account can access. The master account can also prevent member accounts from leaving the 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 master account starts the process.

- **Consolidated billing** – This feature set provides shared billing functionality, but does *not* include the more advanced features of AWS Organizations. For example, you can't 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. 11) 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. 60) 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.

Tag policy

A type of policy that can help you standardize tags across resources in your organization's accounts. In a tag policy (p. 86), you can specify tagging rules for specific resources.

Allow lists vs. deny lists

Allow lists and deny lists are complementary techniques for when you apply SCPs (p. 60) to filter the permissions that are available to accounts.

- **Allow list (also known as "whitelisting")** – 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 ensures 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 them back at a lower level in the hierarchy because an SCP never grants permissions; it only filters them.

- **Deny list (also known as "blacklisting")** – You explicitly specify the access that ***is not*** 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 typically 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.

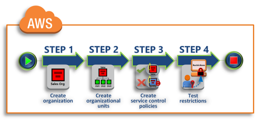
AWS Organizations Tutorials

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

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. 11) 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. 11) 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.



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 master account. The owner of this account has an email address of `masteraccount@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 master, and invite an existing account, 222222222222, to join as a member account.

1. Sign in to AWS as an administrator of account 111111111111 and open the AWS Organizations console at https://console.aws.amazon.com/organizations/.
2. On the introduction page, choose **Create organization**.
3. In the **Create organization** confirmation dialog box, choose **Create organization**.

   Note
   By default, the organization is created with all features enabled. You can also create the organization with only consolidated billing features (p. 11) enabled.

   The organization is created. You're now on the **Accounts** tab. The star next to the account email indicates that it's the master account.

   A verification email is automatically sent to the address that is associated with your master 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. 27).

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

To invite an existing account to join

1. Open the Organizations console at https://console.aws.amazon.com/organizations/.
2. Choose the **Accounts** tab. The star next to the account name indicates that it is the master account.

   Now you can invite other accounts to join as member accounts.

3. On the **Accounts** tab, choose **Add account** and then choose **Invite account**.
4. In the **Account ID or email** box, enter the email address of the owner of the account that you want to invite, similar to the following: **`member222@example.com`**.
5. Type any text that you want into the **Notes** box. This text is included in the email that is sent to the owner of the account.
6. Choose **Invite**. 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.
7. 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 master 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 (https://console.aws.amazon.com/organizations/) and sign in as an administrator of the member account. Choose **Invitations**. The number beside the link indicates how many invitations this account has.
8. On the **Invitations** page, choose **Accept** and then choose **Confirm**.
9. Sign out of your member account and sign in again as an administrator in your master 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.

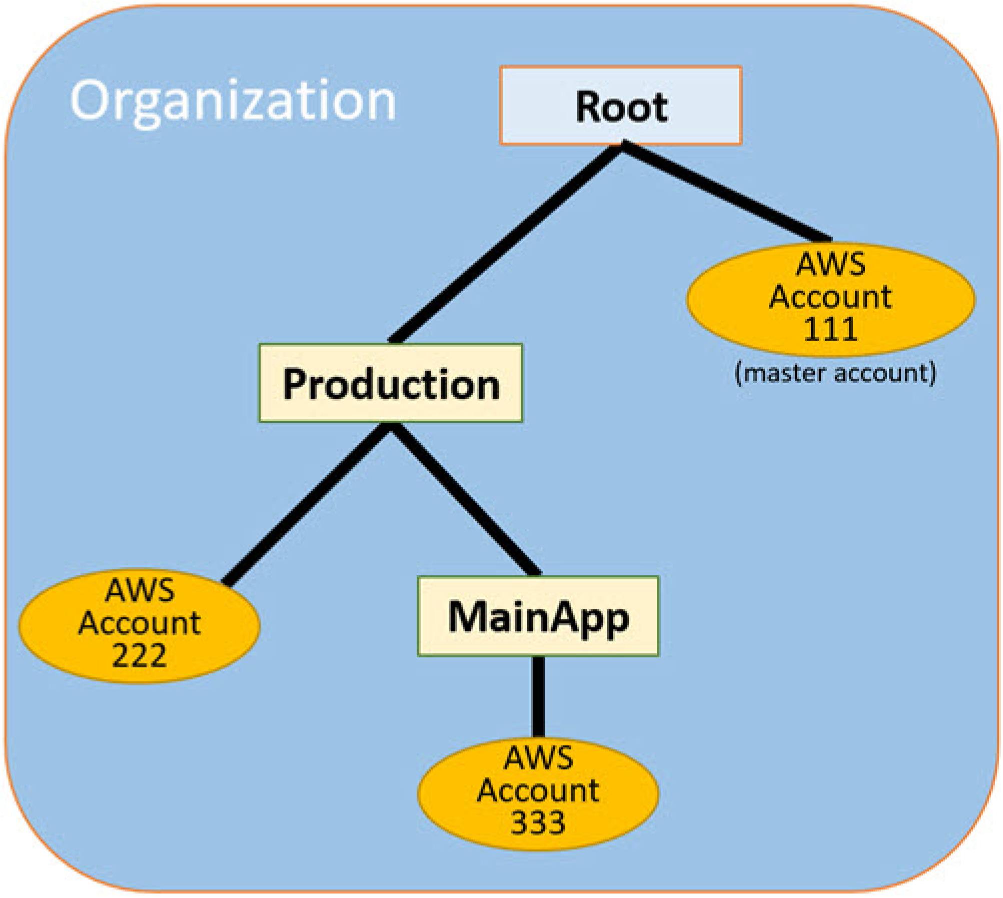
To create a member account

1. On the AWS Organizations console, on the **Accounts** tab, choose **Add account**.
2. For **Full name**, enter a name for the account, such as **`MainApp Account`**.
3. For **Email**, 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`**.
4. 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 master account. For this tutorial, leave it blank to instruct AWS Organizations to create the role with the default name.
5. Choose **Create**. You might need to wait a short while and refresh the page to see the new account appear on the **Accounts** tab.

   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. Your hierarchy looks like the following illustration when you're done. The master 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.



To create and populate the OUs

1. On the AWS Organizations console, choose the **Organize Accounts** tab and then choose **+ New organizational unit**.
2. For the name of the OU, enter **`Production`** and then choose **Create organizational unit**.
3. Choose your new **Production** OU to navigate into it and then choose **+ New organizational unit**.
4. 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.
5. In the tree view on the left, choose the **Root**.
6. Select the first member account, 222222222222, and then choose **Move**.
7. In the **Move accounts** dialog box, choose **Production** and then choose **Move**.
8. Select the second member account, 333333333333, and then choose **Move**.
9. In the **Move accounts** dialog box, choose **Production** to expose **MainApp**. Choose **MainApp** and then choose **Move**.

Step 3: Create the Service Control Policies

In the steps in this section, you create three service control policies (SCPs) (p. 60) 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 master account isn't affected by any SCP, so after you apply the CloudTrail SCP, you must create any logs from the master account.

To create the first SCP that blocks CloudTrail configuration actions

1. Choose the **Policies** tab and then choose **Create policy**.
2. For **Policy name**, enter **`Block CloudTrail Configuration Actions`**.
3. In the **Policy** section on the left, select CloudTrail for the service. Then choose the following actions: **AddTags**, **CreateTrail**, **DeleteTrail**, **RemoveTags**, **StartLogging**, **StopLogging**, and **UpdateTrail**.
4. 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": [
                "*"
            ]
        }
    ]
}
```

5. Choose **Create policy**.

The second policy defines an allow list (p. 11) 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.

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

1. From the list of policies, 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. 11) 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.

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

1. From the **Policies** tab, 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.

Enable the Service Control Policy Type in the Root

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 that root. Policy types aren't enabled in any root by default. The steps in this section show you how to enable the service control policy (SCP) type for the root in your organization.

Note
Currently, you can have only one root in your organization. It's created for you and named **Root** when you create your organization.

To enable SCPs for your root

1. On the **Organize accounts** tab, choose your root.
2. In the **Details** pane on the right, under **ENABLE/DISABLE POLICY TYPES** and next to **Service control policies**, choose **Enable**.

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.

To attach the policies to the root and the OUs

1. Still on the **Organize accounts** tab, in the **Details** pane on the right, under **POLICIES**, choose **SERVICE CONTROL POLICIES**.
2. Choose **Attach** next to the SCP named `Block CloudTrail Configuration Actions` to prevent anyone from altering the way that you configured CloudTrail. In this tutorial, you attach it to the root so that it affects all member accounts.

   The **Details** pane now shows by highlighting that two SCPs are attached to the root: the one you just created and the default `FullAWSAccess` SCP.
3. Choose the **Production** OU (not the check box) to navigate to its contents.
4. Under **POLICIES**, choose **SERVICE CONTROL POLICIES** and then choose **Attach** next to `Allow List for All Approved Services` to enable users or roles in member accounts in the Production OU to access the approved services.
5. The information pane now shows 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 detach this SCP to ensure that only your approved services are allowed.
6. To remove the default policy from the Production OU, next to **FullAWSAccess**, choose **Detach**. After you remove this default policy, all member accounts under the root immediately lose access to all actions and services that are not on the allow list SCP that you attached in the preceding step. 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.
7. 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, choose the **MainApp** OU (not the check box) to navigate to its contents.
8. In the **Details** pane, under **POLICIES**, expand the **Service control policies** section. In the list of available policies, next to **Deny List for MainApp Prohibited Services**, choose **Attach**.

Step 4: Testing Your Organization's Policies

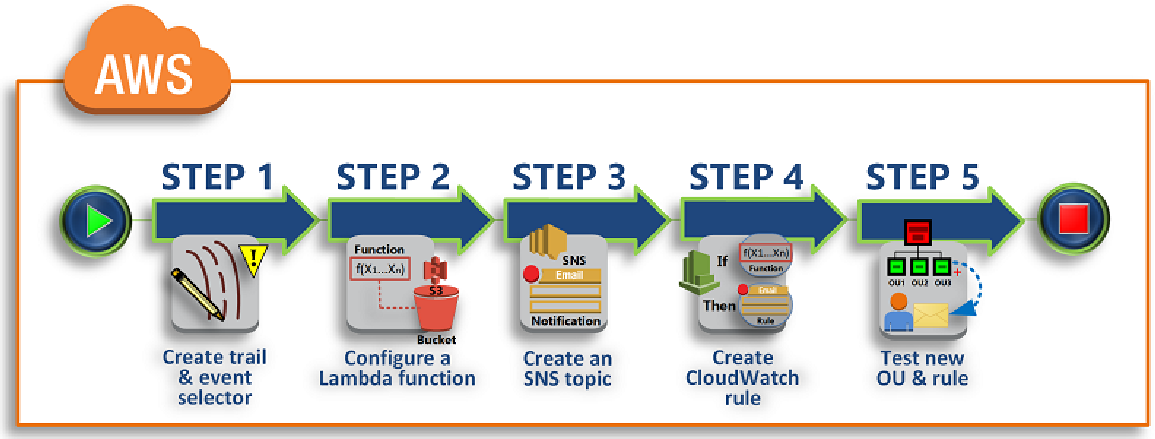
You now can sign in as a user in any of the member accounts and try to perform various AWS actions:

- If you sign in as a user in the master 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 master 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. 21)

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

Step 2: Configure a Lambda Function (p. 22)

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. 22)

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. 23)

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. 23)

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. 144).

Prerequisites

This tutorial assumes the following:

- You can sign in to the AWS Management Console as an IAM user from the master 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 master 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 master 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. In the navigation pane, choose **Rules** and then choose **Create rule**.
3. 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.
4. Under **Targets**, for **Function**, choose the function that you created in the previous procedure.
5. Under **Targets**, choose **Add target**.
6. In the new target row, choose the dropdown header and then choose **SNS topic**.
7. For **Topic**, choose the topic named **OrganizationCloudWatchTopic** that you created in the preceding procedure.
8. Choose **Configure details**.
9. 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.

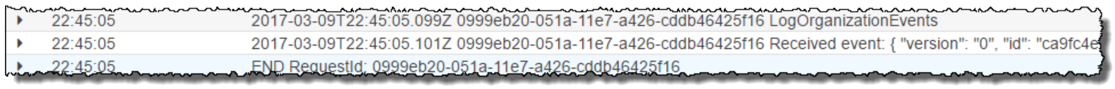
To create an OU

1. Open the AWS Organizations console at https://console.aws.amazon.com/organizations/.
2. Choose the **Organize accounts** tab and then choose **New organizational unit**.
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 at https://console.aws.amazon.com/cloudtrail/ 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 at https://console.aws.amazon.com/s3/ to delete it.
3. Use the Lambda console at https://console.aws.amazon.com/lambda/ to delete the function named **`LogOrganizationEvents`** that you created in step 2.
4. Use the Amazon SNS console at https://console.aws.amazon.com/sns/ to delete the Amazon SNS topic named **`OrganizationsCloudWatchTopic`** that you created in step 3.
5. Use the CloudWatch console at https://console.aws.amazon.com/cloudwatch/ to delete the CloudWatch rule named **`OrgsMonitorRule`** that you created in step 4.

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.

Creating and Managing an Organization

You can perform the following tasks using the AWS Organizations console:

- **Create an organization (p. 26)**. Create your organization with your current account as its master account. Create member accounts within your organization, and invite other accounts to join your organization.
- **Enable all features in your organization (p. 28)**. 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. 60). SCPs offer central control over the maximum available permissions for all accounts in your organization, enabling you to ensure your accounts stay within your organization's access control guidelines.
- **View details about your organization (p. 31)**. View details about your organization and its roots, organizational units (OUs), and accounts.
- **Delete an organization (p. 35)**. 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 select users to perform specific tasks.

Creating an Organization

Use AWS Organizations to create your own organization to consolidate and manage your AWS accounts.

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

Note
Currently, you can have only one root in your organization.

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

- Create other AWS accounts that are automatically added to your organization as member accounts
- After verifying your email address, invite existing AWS accounts to join your organization as member accounts

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

- `organizations:CreateOrganization`

To create an organization (console)

1. Sign in to the AWS Management Console and open the AWS Organizations console at https://console.aws.amazon.com/organizations/. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the account that you want to be the organization's master account.
2. On the introduction page, choose **Create organization**.
3. In the **Create organization** confirmation dialog box, choose **Create organization**.

   Note
   By default, the organization is created with all features enabled. You can also create the organization with only consolidated billing features (p. 11) enabled.

   The organization is created. You're now on the **Accounts** tab. The star next to the account email indicates that it's the master account.

   A verification email is automatically sent to the address that is associated with your master 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. 27).
5. Add accounts to your organization as follows:

   - To create an AWS account that is automatically part of your AWS organization, see Creating an AWS Account in Your Organization (p. 42).
   - To invite an existing account to your organization, see Inviting an AWS Account to Join Your Organization (p. 38).

   Note
   You can add new accounts to your organization without verifying your master account's email address. To invite existing accounts, you must first verify that email address.

To create an organization (AWS CLI, AWS API)

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

- AWS CLI: aws organizations create-organization
- AWS API: CreateOrganization

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 master account in the organization.

When you create an organization, 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 master account, see Viewing Details of an Organization from the Master Account (p. 32).
- To change the email address that is associated with your master account, see Managing an AWS Account in the *AWS Billing and Cost Management User Guide*.

To resend the verification request

1. Sign in to the AWS Management Console and open the AWS Organizations console at https://console.aws.amazon.com/organizations/. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the account that you want to be the organization's master account.
2. Choose the Settings tab and then choose **Send verification request**.
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. 38).

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

Enabling All Features in Your Organization

AWS Organizations has two available feature sets:

- All features (p. 11) – 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 service control policies (SCPs) (p. 60) and tag policies (p. 86).
- Consolidated billing features (p. 11) – 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.
- Organizations also verifies that every 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.

- While enabling all features is in progress, you can continue to *create* accounts within the organization but you can't *invite* existing accounts to join the organization. To invite accounts, you must wait until the process to enable all features is complete. Alternatively, you can cancel the process to enable all features, invite the accounts, and then restart the process to enable all features.
- Because enabling all features makes it possible to use SCPs (p. 60), 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 master account can apply SCPs that can prevent member accounts from leaving the organization.
- The master account isn't affected by any SCP. You can't limit what users and roles in the master 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.
- If your organization has only consolidated billing features enabled, member account administrators can choose to delete the service-linked role named `AWSServiceRoleForOrganizations`. However, when you 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. 129).

Beginning the Process to Enable All Features

When you sign in with permissions to your organization's master 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`

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

1. Sign in to the AWS Management Console and open the AWS Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. On the **Settings** tab, choose **Begin process to enable all features**.
3. Acknowledge your understanding that you cannot return to only consolidated billing features after you switch by choosing **Begin process to enable all features**. 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.
4. To view the status of the requests, choose **View all feature request approval status**.

   The **All feature request approval status** page shows the current request status for each account in the organization. Accounts that have agreed to the request have a green check mark and show the **Acceptance** date. Accounts that haven't yet agreed have a yellow exclamation point icon and show the date that the request was sent with a status of **Open**.

   Note

   - 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. 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.
5. If an account doesn't approve its request, you can select the account on this page and then choose **Remove**. This cancels the request for the selected account and removes that account from the organization, eliminating the blocker to enabling all features.
6. After all accounts approve the 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. Finalizing the process requires only a couple of clicks in the console. See Finalizing the Process to Enable All Features (p. 31).

To ask your invited member accounts to agree to enable all features in the organization (AWS CLI, AWS API)

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

- AWS CLI: aws organizations enable-all-features
- AWS API: EnableAllFeatures

Approving the Request to Enable All Features or to Recreate the Service-Linked Role

When signed in with permissions to one of the organization's invited member accounts, you can approve a request from the master 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.

Minimum permissions
To approve a request to enable all features for your member account, you must have the following permissions:

- `organizations:AcceptHandshake`
- `iam:CreateServiceLinkedRole` – Required only if the `AWSServiceRoleForOrganizations` role must be recreated in the member account

Important
If you perform the steps in the following procedure, the master 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. Additionally, the master account can apply a policy that might prevent your account from leaving the organization.

To agree to the request to enable all features in the organization (console)

1. Sign in to the AWS Management Console and open the AWS Organizations console at https://console.aws.amazon.com/organizations/. 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 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 master account finalizes the process.

To agree to the request to enable all features in the organization (AWS CLI, AWS API)

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

- AWS CLI: aws organizations accept-handshake
- AWS API: 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`

To finalize the process to enable all features (console)

1. Sign in to the AWS Management Console and open the AWS Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. On the **Settings** tab, under **ENABLE ALL FEATURES**, choose **View all feature request approval status**.
3. After all accounts accept the request to enable all features, in the green box at the top of the page, choose **Finalize process to enable all features**. When asked to confirm, choose **Finalize process to enable all features** again.
4. The organization now has all features enabled. The next step is to 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. 58).

To finalize the process to enable all features (AWS CLI, AWS API)

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

- AWS CLI: aws organizations accept-handshake
- AWS API: AcceptHandshake

Viewing Details About Your Organization

You can perform the following tasks using the AWS Organizations console:

- Viewing Details of an Organization from the Master Account (p. 32)
- Viewing Details of a Root (p. 32)
- Viewing Details of an OU (p. 33)

Viewing Details of an Organization from the Master Account

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

- `organizations:DescribeOrganization`

To view details of an organization (console)

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

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. Choose the **Settings** tab.

   The console displays details about the organization, including its ID and the email address of the owner of its master account.

To view details of an organization (AWS CLI, AWS API)

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

- AWS CLI: aws organizations describe-organization
- AWS API: DescribeOrganization

Viewing Details of a Root

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

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

To view details of a root (console)

When signed in to the organization's master account in the AWS Organizations console, you can view details of a root.

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. Choose the **Organize accounts** tab, and then choose **Home**.
3. Choose the **Root** entity. Root is the default name, but you can rename it using the API or command line tools.

   The **Root** pane on the right side of the page shows the details of the root.

To view details of a root (AWS CLI, AWS API)

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

- AWS CLI: aws organizations list-roots
- AWS API: ListRoots

Viewing Details of an OU

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

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

To view details of an OU (console)

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

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. On the **Organize accounts** tab, navigate to the OU that you want to examine. If the OU that you want is a child of another OU, choose each OU in the hierarchy to find the one you're looking for.
3. Select the check box for the OU.

   The Details pane on the right side of the page shows information about the OU.

To view details of an OU (AWS CLI, AWS API)

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

- AWS CLI: aws organizations describe-organizational-unit
- AWS API: DescribeOrganizationalUnit

Viewing Details of an Account

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

- `organizations:DescribeAccount`
- `organizations:DescribeOrganization` (console only)
- `organizations:ListAccounts` (console only)

To view details of an AWS account (console)

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

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. Do one of the following:

   - On the **Accounts** tab, choose the account that you want to examine.
   - On the **Organize accounts** tab, navigate to and then choose an account card.

   The **Account summary** pane on the right side of the page shows details of the selected account.

   Note
   By default, failed account creation requests are available for 90 days and hidden on the **Accounts** tab. To view a list that includes failed account creation requests, choose the switch at the top to change it to **Show**.

To view details of an account (AWS CLI, AWS API)

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

- AWS CLI: aws organizations describe-account
- AWS API: DescribeAccount

Viewing Details of a Policy

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

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

To view details of a policy (console)

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

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. On the **Policies** tab, choose the policy that you want to examine.
3. Choose **View policy details** in the pane on the right side of the page.

To view details of a policy (AWS CLI, AWS API)

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

- AWS CLI: aws organizations describe-policy
- AWS API: DescribePolicy

Remove the Master Account and Delete the Organization

When you no longer need your organization, you can delete it. This removes the master account from the organization and deletes the organization itself. The former master 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 will not be able to recover it. If you created any policies inside of the organization, they will also be deleted.
- 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 is required to operate as a standalone AWS account. For more information about how to provide that information and remove the account, see Leaving an Organization as a Member Account (p. 50).

When you remove the master account from an organization by deleting the organization, the account is affected 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 Single Sign-On requires an organization to operate, so if you remove an account from an organization that supports AWS SSO, the users in that account can no longer use that service.

The master 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.

To remove the master account from an organization and delete the organization (console)

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

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

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master 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. 48).
3. On the **Settings** tab, choose **Delete organization**.
4. In the **Delete organization** confirmation dialog box, choose **Delete organization**.
5. (Optional) If you also want to close this account, you can follow the steps at Closing an AWS Account (p. 52).

To delete an organization (AWS CLI, AWS API)

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

- AWS CLI: aws organizations delete-organization
- AWS API: DeleteOrganization

Managing the AWS Accounts in Your Organization

An organization is a collection of AWS accounts that you centrally manage. 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. 33). You can see the account's unique ID number, its Amazon Resource Name (ARN), and the policies that are attached to it.
- Invite existing AWS accounts to join your organization (p. 38). Create invitations, manage invitations that you have created, and accept or decline invitations.
- Create an AWS account as part of your organization (p. 42). Create and access an AWS account that is automatically part of your organization.
- Remove an AWS account from your organization (p. 48). As an administrator in the master 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 master account has attached a policy to your member account, you could be blocked from removing your account.
- Delete (or close) an AWS account (p. 52). When you no longer need an AWS account, you can close the account to prevent any usage or accrual of charges.

Impact on an AWS Account That You Invite to Join an Organization

You invite an AWS account to join an organization. When 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. 129). 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 service control policies (SCPs) (p. 60) or tag policies (p. 86) that are 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. 130) 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.

For invited member accounts, AWS Organizations doesn't automatically create the IAM role OrganizationAccountAccessRole (p. 47). This role grants the master account administrative control of the member account. If you want to enable that level of administrative control, you can manually add the role to the invited account. For more information, see Creating the OrganizationAccountAccessRole in an Invited Member Account (p. 45).

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. 129). 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. 47). This role grants the master account access to the new member account. This role can be deleted.
- If you have any SCPs attached to the root of the OU tree (p. 60), those SCPs 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. 130) 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 master 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 master account of your organization.

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.

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 master 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 master 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. 45).

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 master 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. 129).

You can send up to 20 invitations per day per organization. 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 returned if the invited account declines, the master 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. 42).

> **Important**
> Because of legal and billing constraints, you can invite AWS accounts only from the same AWS seller as the master account. You can't mix accounts from AWS, Amazon Internet Services Pvt. Ltd (AISPL, an AWS seller in India), or Amazon Connect Technology Services (Beijing) Co. (ACTS, an AWS seller in China) in the same organization. You can add accounts from an AWS seller only to an organization with accounts from the same AWS seller.

Sending Invitations to AWS Accounts

To invite accounts to your organization, you must first verify that you own the email address associated with the master account. After you have verified 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`

To invite another account to join your organization (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. If your email address is already verified, skip this step.

   If your email address isn't verified yet, follow the instructions in the verification email (p. 27) within 24 hours. There might be a delay before you receive the verification email. You can't invite an account until your email address is verified.
3. On the **Accounts** tab, choose **Add account**.
4. Choose **Invite account**.
5. Enter either the email address or the account ID number of the AWS account that you want to invite to your organization. If you want to invite multiple accounts, separate them with commas.
6. (Optional) For **Notes**, enter any message that you want included in the email invitation to the other account owners.
7. Choose **Invite**.

   > **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.
8. The console redirects you to the **Invitations** tab. 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 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, 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 master 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.

To invite another account to join your organization (AWS CLI, AWS API)

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

- AWS CLI: aws organizations invite-account-to-organization
- AWS API: InviteAccountToOrganization

Managing Pending Invitations for Your Organization

When signed in to your master 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` (console only)
- `organizations:ListHandshakesForOrganization`
- `organizations:CancelHandshake`

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

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. Choose the **Invitations** tab. All invitations that are sent from your organization and their current status are listed here.

   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. For any open invitations that you want to cancel, under the **Actions** column, choose **Cancel**.

   The status of the invitation changes from **Open** to **Canceled**.

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

To view or cancel invitations that are sent from your organization to other accounts (AWS CLI, AWS API)

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

- AWS CLI: aws organizations list-handshakes-for-organization, aws organizations cancel-handshake
- AWS API: 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.

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 to support integration with other AWS services. For more information, see AWS Organizations and Service-Linked Roles (p. 129).

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

- When a standalone account joins an organization, the account no longer has access to cost and usage data from the time range when the account was a standalone account.
- 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.

To accept or decline an invitation (console)

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, follow the instructions in the email invitation or go to https://console.aws.amazon.com/organizations/ in your browser, and then choose **Respond to invitations**.
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. On the **Invitations** page in the console, you can see your open invitations to join organizations. Choose **Accept** or **Decline** as appropriate.

    - If you choose **Accept** in the preceding step, in the **Confirm joining the organization** confirmation window, choose **Confirm**.

      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. 129).

AWS sends an email to the owner of the organization's master account stating that you accepted the invitation. It also sends an email 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 **Invitations** page that lists any other pending invitations.

  AWS sends an email to the organization's master 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.

To accept or decline an invitation (AWS CLI, AWS API)

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

- AWS CLI: aws organizations accept-handshake, aws organizations decline-handshake
- AWS API: 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. 43)
- Accessing a Member Account That Has a Master Account Access Role (p. 47)

> **Important**
> When you create a member account in your organization, AWS Organizations automatically creates an IAM role in the member account that enables IAM users in the master account to exercise full administrative control over the member account. This role is subject to any service control policies (SCPs) (p. 60) that apply to the member account.
> 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. 129).

Creating an AWS Account That Is Part of Your Organization

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

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

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

Important
When you create an account using the following procedure, AWS doesn't 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. 50).

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

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. On the **Accounts** tab, choose **Add account**.
3. Choose **Create account**.
4. 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. Enter the email address for the owner of the new account. This address must be unique to this account because it can be used to sign in as 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 master 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`.

   Important
   Remember this role name. You need it later to grant access to the new account for IAM users in the master account.
7. Choose **Create**.

   Important

   - If you get an error that indicates that you exceeded your account limits for the organization, contact AWS Support.
   - 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. 144).
   - If the error persists, contact AWS Support.
8. You are redirected to the **Accounts/All accounts** tab, showing your new account at the top of the list with its status set to **Pending creation**. When the account is created, this status changes to **Active**.

> **Note**
> By default, the **Accounts** tab hides account creation requests that failed. To show them, choose the switch at the top of the list and change it to **Show**.

9. Now that the account exists and has an IAM role that grants administrator access to users in the master account, you can access the account by following the steps in Accessing and Administering the Member Accounts in Your Organization (p. 44).

   When you create an 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, as it's discarded after the account is created. 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. 45).

To create an AWS account that automatically is part of your organization (AWS CLI, AWS API)

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

- AWS CLI: aws organizations create-account
- AWS API: CreateAccount

Accessing and Administering the Member Accounts in Your Organization

When you create an account in your organization, AWS Organizations automatically creates a root user and an IAM role named `OrganizationAccountAccessRole` for the account. However, AWS Organizations doesn't create any 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. 45).
- If you create an account in your organization, you can access the account by using the preconfigured role named `OrganizationAccountAccessRole` that exists in all new accounts that are created this way. See Accessing a Member Account That Has a Master Account Access Role (p. 47).
- If you invite an existing account to join your organization and the account accepts the invitation, you can then create an IAM role that allows the master account to access the invited account. This is similar 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. 45). After you create the role, you can access it using the steps in Accessing a Member Account That Has a Master Account Access Role (p. 47).
- Use AWS Single Sign-On and enable trusted access for AWS SSO with AWS Organizations. This allows users to sign in to the AWS SSO user portal with their corporate credentials and access resources in their assigned master or member accounts.

  For more information, see Manage SSO to Your AWS Accounts in the *AWS Single Sign-On User Guide.* For information about setting up trusted access for AWS SSO, see AWS Single Sign-On and AWS Organizations (p. 136).

> **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. To update the email address, see the Contact Us page, and choose the item regarding billing to contact AWS Support.

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

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 delegated IAM users in the master account. By default, that role is named `OrganizationAccountAccessRole`. For more information, see Accessing a Member Account That Has a Master Account Access Role (p. 47).

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.

To create an AWS Organizations administrator role in a member account (console)

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 that has permissions 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 master 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`, which is 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 the 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 master account who has permissions to create policies and assign the policies to users or groups.
11. Navigate to **Policies** and then choose **Create Policy**.

    Note
    This example shows how to create a policy and attach it to a group. If you already created this policy for other accounts, skip to step 18
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 Master Account Access Role (p. 47).

Accessing a Member Account That Has a Master 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 role is also configured to grant that access to the organization's master 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. 45). To use this role to access the member account, you must sign in as a user from the master 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.

To grant permissions to members of an IAM group in the master account to access the role (console)

1. Sign in to the IAM console at https://console.aws.amazon.com/iam/ as a user with administrator permissions in the master account. This is required to delegate permissions to the IAM group whose users will access the role in the member account.
2. 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.
3. Choose the **Permissions** tab and then expand the **Inline Policies** section.
4. If no inline policies exist, choose **click here** to create one. Otherwise, choose **Create Group Policy**.
5. Next to **Policy Generator**, choose **Select**.
6. On the **Edit Permissions** page, set the following options:

   - For **Effect**, choose **Allow**.
   - For **AWS Service**, choose **AWS Security Token Service**.
   - For **Actions**, choose **AssumeRole**.
   - For **Amazon Resource Name (ARN)**, enter the ARN of the role that was created in the member account. You can see the ARN in the IAM console on the role's **Summary** page. To construct this ARN, use the following template:

     ```
     arn:aws:iam::accountIdNumber:role/rolename
     ```

     Substitute the account ID number of the member account and the role name that was configured when you created the account. If you didn't specify a role name, the name defaults to `OrganizationAccountAccessRole`. The ARN should look like the following:

     ```
     arn:aws:iam::123456789012:role/OrganizationAccountAccessRole
     ```
7. Choose **Add Statement** and then choose **Next Step**.
8. On the **Review Policy** page, ensure that the ARN for the role is correct. Enter a name for the new policy and then choose **Apply Policy**.

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

To switch to the role for the member account (console)

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 the 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 have finished 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. 52).

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 *master* account, see Remove the Master Account and Delete the Organization (p. 35).

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 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.

- 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 master account's agreement with us, and (ii) the creating master 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 master account of the organization can still access the data. If the account rejoins the organization, the account can access that data again.

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 service control policies (SCPs) (p. 60) that were defined in the organization. This means that restrictions imposed by those SCPs are gone, and the users and roles in the account might have more permissions than they had before.
- Integration with other services might be disabled. For example, AWS Single Sign-On requires an organization to operate, so if you remove an account from an organization that supports AWS SSO, 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 master 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 master account, you must delete the organization.

Note
If a member account is removed from an organization, that member account will no longer be covered by organization agreements. Master 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 AWS Artifact Organization Agreements.

Minimum permissions
To remove one or more member accounts from your organization, you must sign in as an IAM user or role in the master account with the following permissions:

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

If you choose to sign in as an IAM user or role in a member account in step 5, then that user or role must have the following permissions:

- `organizations:DescribeOrganization` (console only).
- `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 and Cost Management User Guide*.

To remove a member account from your organization (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. You must sign in to the organization's master account.
2. On the **Accounts** tab, select the check box next to the member account that you want to remove from your organization. You can choose more than one.
3. Choose **Remove account**.
4. In the **Remove account** dialog box, choose **Remove**.

   A dialog box displays the success or failure status for each 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 one of the failed accounts.
   b. 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 master account and won't be able to navigate back to this dialog box.
   c. 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
      - Accept the AWS Customer Agreement
      - Provide a valid payment method
      - Verify the phone number
      - Select a support plan option
   d. 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.

To remove a member account from your organization (AWS CLI, AWS API)

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

- AWS CLI: aws organizations remove-account-from-organization
- AWS API: RemoveAccountFromOrganization

Leaving an Organization as a Member Account

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

Note
If you leave an organization, you will no longer be covered by organization agreements that were accepted on your behalf by the master account of the organization. You can view a list of

these organization agreements in AWS Artifact Organization Agreements. 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.

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

- `organizations:DescribeOrganization` (console only).
- `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 and Cost Management User Guide*.

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

- When a standalone account joins an organization, the account no longer has access to cost and usage data from the time range when the account was a standalone account.
- 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.

To leave an organization as a member account (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. You can sign in as an IAM user with the required permissions, or as the root user of the member account that you want to remove from the organization. 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. 45).
2. On the **Organization overview** 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
        - Accept the AWS Customer Agreement

- 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.

To leave an organization as a member account (AWS CLI, AWS API)

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

- AWS CLI: aws organizations leave-organization
- AWS API: LeaveOrganization

Closing an AWS Account

If you no longer need an AWS account (whether a member in an organization or not) and want to ensure that no one can accrue charges for it, you can close the account.

Before closing your account, back up any applications and data that you want to retain. All resources and data that were stored in the account are lost and cannot be recovered. For more information, see the KB article "How do I close my Amazon Web Services account?"

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.

Important

- Accounts that are closed for 90 days are subject to permanent deletion after which the account and its resources can’t be recovered.
- Closed accounts are visible in your organization with the "suspended" state. After an account is permanently deleted, it's no longer visible in your organization.

You can close an account only by using the Billing and Cost Management console, not by using the AWS Organizations console or its tools.

To close a master account, first delete the organization (p. 35), and then close it using the steps in the following procedure.

To close an AWS account (console)

Recommended: Before closing your account, back up any applications and data that you want to retain. AWS can't recover or restore your account resources and data after your account is closed.

1. Sign in as the root user of the account that you want to close, using the email address and password that are associated with the account. If you sign in as an IAM user or role, you can't close an account.

   Note
   By default, member accounts that you create with AWS Organizations don't have a password that's associated with the account's root user. To sign in, you must request a password for the root user. For more information, see Accessing a Member Account as the Root User (p. 45).

2. Open the Billing and Cost Management console at https://console.aws.amazon.com/billing/home#/.
3. On the navigation bar in the upper-right corner, choose your account name (or alias) and then choose **My Account**.
4. On the **Account Settings** page, scroll to the end of the page to the **Close Account** section. Read and ensure that you understand the text next to the check box.
5. Select the check box to confirm your understanding of the terms and then choose **Close Account**.
6. In the confirmation box, choose **Close Account**.

After you close an AWS account, you can no longer use it to access AWS services or resources. For the 90 days after you close your account (the "Post-Closure Period"), you will be able to log in to view past bills and access AWS Support. 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.

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
Currently, you can have only a single root, which AWS Organizations creates for you when you first set up your organization. The name of the default root is "root."

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

- Viewing details of an OU (p. 33)
- Creating an OU (p. 55)
- Renaming an OU (p. 56)
- Moving an account to an OU or between the root and OUs (p. 56)

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 tree view.

To enable and use the tree view of the organization

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/.
2. Choose the **Organize accounts** tab.
3. If the tree view pane isn't visible on the left side of the page, choose the **TREE VIEW** switch icon .
4. The tree initially appears showing the root, with only the first level of child OUs displayed. To expand the tree to show deeper levels, choose the + icon next to any parent entity. To reduce clutter and collapse a branch of the tree, choose the — icon next to an expanded parent entity.
5. Choose the OU or root that you want to navigate to. The node in the tree view that is displayed in bold text is the one that you are currently viewing in the center pane.

Notes

- **Rename, Delete, and Move operations in the center pane**: When you view the contents of a root or OU in the console, you can interact with the child entities of that root or OU. For example, if you select the check box for a child OU or account, you can choose the **Rename**, **Delete**, or **Move** links above that section to perform those operations on the selected entity. The operations apply *only* to the child entities that you select. They don't apply to the containing root or OU. To perform the same operations for the containing OU, you must navigate to the OU's parent OU or root, and then select the check box for the child OU that you want to manage.

- **Details pane**: The details pane on the right side of the console shows information about the root or OU that you are viewing. If you select a check box for a child entity, the details pane switches to show information about the selected entity. To see the details of the containing root or OU again, you must clear the check box. Alternatively, you can navigate to the parent root or OU, and then select the check box for the OU whose information you want to see.

To navigate without using the tree view

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/.
2. Choose the **Organize accounts** tab.
3. Navigate down a branch by choosing the name of the OU (not the check box) that you want to view in the center pane.
4. Navigate up the branch by choosing the back button (<) on the title bar of the center pane.

Creating an OU

When signed in to your organization's master 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.

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

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

To create an OU (Console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.

   The console displays the contents of the root. The first time you visit a 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.
2. (Optional) If you want to create an OU inside an existing OU, navigate to the child OU (p. 54) by choosing the name (not the check box) of the child OU, or by choosing the OU in the tree view.
3. When you're in the correct location in the hierarchy, choose **Create organizational unit (OU)**.
4. In the **Create organizational unit** dialog box, type the name of the OU that you want to create, and then choose **Create organizational unit**.

   Your new OU appears inside the parent. You now can move accounts to this OU (p. 56) or attach policies to it.

To create an OU (AWS CLI, AWS API)

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

- AWS CLI: aws organizations create-organizational-unit
- AWS API: CreateOrganizationalUnit

Renaming an OU

When signed in to your organization's master 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` (console only)
- `organizations:UpdateOrganizationalUnit`

To rename an OU (Console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. On the **Organize accounts** tab, navigate to the parent of the OU (p. 54) that you want to rename. Select the check box for the child OU that you want to rename.
3. Choose **Rename** above the list of OUs.
4. In the **Rename organizational unit** dialog box, type a new name, and then choose **Rename organizational unit**.

To rename an OU (AWS CLI, AWS API)

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

- AWS CLI: aws organizations update-organizational-unit
- AWS API: UpdateOrganizationalUnit

Moving an Account to an OU or Between the Root and OUs

When signed in to your organization's master 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 other 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 to the root and any that are attached directly to the account. To move an account, complete the following steps.

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

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

To move an account to an OU (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.

2. Choose the **Organize accounts** tab and then navigate to the OU (p. 54) that contains the account that you want to move. When you find the account, select its check box. Select multiple check boxes if you want to move multiple accounts.
3. Choose **Move** above the list of accounts.
4. In the **Move accounts** dialog box, choose the OU or the root that you want to move the accounts to and then choose **Select**.

To move an account to an OU (AWS CLI, AWS API)

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

- AWS CLI: aws organizations move-account
- AWS API: MoveAccount

Deleting an OU That You No Longer Need

When signed in to your organization's master account, you can delete OUs that you no longer need. You first must move all accounts out of the OU and any child OUs, and then delete the child OUs.

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

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

To delete an OU (Console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. On the **Organize accounts** tab, navigate to the parent container (p. 54) of the OU that you want to delete. Select the OU's check box. You can select check boxes for multiple OUs if you want to delete more than one.
3. Choose **Delete** above the list of OUs.

   AWS Organizations deletes the OU and removes it from the list.

To delete an OU (AWS CLI, AWS API)

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

- AWS CLI: aws organizations delete-organizational-unit
- AWS API: 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. 28) in your organization.

Policy Types

Organizations offers the following policy types:

- **Service control policies (SCPs)** (p. 60) offer central control over the maximum available permissions for all accounts your organization.
- **Tag policies** (p. 86) help you standardize tags across resources in your organization's accounts.

The AWS Organizations console displays the enabled and disabled status of 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.

Listing Policy Information

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`

To list all policies in your organization (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. Choose the **Policies** tab.
3. Choose the policy type: **Service control policies** or **Tag policies**.

   The displayed list includes the all policies of that type that are currently defined in the organization.

To list all policies in your organization (AWS CLI, AWS API)

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

- AWS CLI: aws organizations list-policies

- AWS API: ListPolicies

Listing All Policies Attached to a Root, OU, or Account

Minimum permissions
To list the policies that are attached to a root, 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 ARN of the specified target (or "*")

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

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. On the **Organize accounts** tab, navigate (p. 54) to the root, OU, or account whose policy attachments you want to see.

   a. For a root or OU, don't select any check boxes. This way, the details pane on the right shows the information about the root or OU that you are viewing. Alternatively, you can navigate to the parent of the OU, and then select the check box for the OU whose information you want to see.
   b. For an account, check the box for the account.
3. In the details pane on the right, expand the **Service control policies** or **Tag policies** section.

   The displayed list shows all policies of that type that are attached directly to this entity. It also shows policies that affect this entity because of inheritance from the root or a parent OU.

To list all policies that are attached directly to a specified root, OU, or account (AWS CLI, AWS API)

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

- AWS CLI: aws organizations list-policies-for-target
- AWS API: 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 "*")

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

1. Choose the **Policies** tab.
2. Choose the policy type: **Service control polices** or **Tag policies**.
3. Select the check box next to the policy that you're interested in.
4. In the details pane on the right, choose one of the following:

   - **Accounts** to see the list of accounts that the policy is directly attached to

- **Organizational units** to see the list of OUs that the policy is directly attached to
- **Roots** to see the list of roots that the policy is directly attached to

To list all roots, OUs, and accounts that have a specified policy attached (AWS CLI, AWS API)

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

- AWS CLI: aws organizations list-targets-for-policy
- AWS API: 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 "*")

To get details about a policy (console)

1. Choose the **Policies** tab.
2. Choose the policy type: **Service control policies** or **Tag policies**.
3. Select the check box next to the policy that you're interested in.

   The details pane on the right displays the available information about the policy, including its ARN, description, and attachments.
4. To view the content of the policy, choose **Policy editor**.

   The center pane shows the following information:

   - The details about the policy: its name, description, unique ID, and ARN.
   - The list of roots, OUs, and accounts that the policy is attached to. Choose each item to see the individual entities of each type.
   - The policy's content (specific to the type of policy):
     - For SCPs, the JSON text that defines the permissions that are allowed in attached accounts.
     - For tag policies, the JSON text that defines compliant tags for specified resource types.

     To update the contents of the policy document, choose **Edit**. Choose **Save** when you are done. For more details, see the next section.

To get details about a policy (AWS CLI, AWS API)

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

- AWS CLI: aws organizations describe-policy
- AWS API: DescribePolicy

Service Control Policies

Service control policies (SCPs) are one type of policy that you can use to manage your organization. SCPs offer central control over the maximum available permissions for all accounts in your organization, allowing 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. 28). SCPs aren't available if your organization has enabled only the consolidated billing features. For instructions on enabling SCPs, see Enabling and Disabling SCPs (p. 63).

SCPs alone are not sufficient for allowing access in the accounts in your organization. Attaching an SCP to an AWS Organizations entity (root, OU, or account) defines a guardrail for what actions the principals can perform. You still need to attach identity-based or resource-based policies to principals or resources in your organization's accounts to actually grant permissions to them. When a principal belongs to an account that is a member of an organization, the SCPs contribute to the principal's effective permissions (p. 61).

Topics

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. 152). 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.

Effects on Permissions

SCPs are similar to 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 an organization or organizational unit (OU). For more information, see Policy Evaluation Logic in the *IAM User Guide*.

- SCPs ***affect only principals*** 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 accounts outside the organization. Account A has an SCP attached. That SCP doesn't apply to those outside users. It applies only to users that are managed by account A in the organization.

- An SCP restricts permissions for principals in member accounts, including each AWS account 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.
- Users and roles must still be granted permissions with appropriate IAM permission policies. A user without any IAM permission policies has no access at all, 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. 62).
- 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 master 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 CLI or AWS API in IAM to retrieve service last accessed data. This data includes information about which allowed services that principals in an AWS Organizations account last attempted to access and when. You can use this information to identify unnecessary 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. 70) 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 pages 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

The following tasks and AWS Organizations entities are not restricted by SCPs:

- Actions performed by the master account.
- Any action performed using permissions that are attached to a service-linked role.
- Managing root credentials. No matter what SCPs are attached, the root user in an account can always do the following:
  - Changing the root user's password
  - Creating, updating, or deleting root access keys

    Note
    For some accounts created before 9/15/2017, this restriction doesn't apply. AWS recommends that you don't rely on this restriction for these accounts.
  - Enabling or disabling multi-factor authentication on the root user
  - Creating, updating, or deleting x.509 keys for the root user
- Registering for the Enterprise support plan as the root user
- Changing the AWS support level as the root user
- Managing Amazon CloudFront keys
- Trusted signer functionality for CloudFront private content
- Modifying AWS account email allowance/reverse DNS
- Performing tasks on some AWS-related services:
  - Alexa Top Sites
  - Alexa Web Information Service
  - Amazon Mechanical Turk
  - Amazon Product Marketing API

Enabling and Disabling SCPs

Before you can create and attach service control policies (SCPs), you must enable this feature. Enabling the use of SCPs is a one-time task on the organization root. You must be signed in to the organization's master account to enable SCPs.

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

- `organizations:EnablePolicyType`
- `organizations:DescribeOrganization`

To enable SCPs (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. On the **Organize accounts** tab, choose **Root** in the left navigation pane.
3. In the details pane on the right side, next to **Service control policies**, choose **Enable**.
4. Under **What is a service control policy?**, choose **Enable service control polices**.

To enable SCPs (AWS CLI, AWS API)

You can use one of the following to enable SCPs:

- AWS CLI: aws organizations enable-policy-type
- AWS API: EnablePolicyType

Disabling SCPs

When you disable SCPs, all SCPs are automatically detached from all entities in the organization root. If you reenable SCPs, all entities in the organization root are initially attached to only the default `FullAWSAccess` SCP. Any attachments of SCPs to entities from before SCPs were disabled are lost.

You can only disable SCPs from the organization's master account.

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

- `organizations:DescribeOrganization`
- `organizations:DisablePolicyType`

To disable SCPs (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. On the **Organize accounts** tab, choose **Root** in the left navigation pane.
3. In the details pane on the right, next to **Service control policies**, choose **Disable**.

To disable SCPs (AWS CLI, AWS API)

You can use one of the following to disable SCPs:

- AWS CLI: aws organizations disable-policy-type
- AWS API: DisablePolicyType

Creating and Updating SCPs

When signed in with permissions to your organization's master account, you can create and update service control policies (SCPs) (p. 60). 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 create statements that deny access. With deny statements, you can also 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. 70).

> **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.*

Creating an SCP

To create SCPs, you need permission to run the following action:

- `organizations:CreatePolicy`

To create a service control policy (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. On the **Policies** tab, choose **Service control policies**.
3. On the **Service control policies** page, choose **Create policy**.
4. On the **Create policy** page, enter a name and description for the policy.

   To build the policy, your next steps vary depending on whether you want to add a statement that denies or allows access. With 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 more information, see SCP Syntax (p. 72).
5. To add a statement that *denies* access:

   a. In the left pane of the **Policy** section, choose the service to add actions for.

      As you choose options on the left, the right pane updates to show the JSON policy. You can also type or paste policies in the editor in the **Policy** section's right pane. However, this procedure describes how to use the visual editor on the left to build your SCP.

   b. From the list that opens of available actions for that service, choose the action or actions to deny.

   c. Specify resources to include in the statement.

      - Choose **Add resource**.
      - On the **Add resource** screen, choose the service from the list and then choose the **Resource type**. Enter the **Resource ARN**.
      - Choose **Add resource**.

      Tip
      The resource element is required. If you want to specify all resources for the selected service, edit the resource statement in the right pane to read `"Resource":"*"`.

   d. Optional: To specify conditions for when a policy is in effect, choose **Add condition**. For the selected service, specify the following:

      - **Condition key** – You can specify a condition key that is available for all AWS services (for example, `aws:SourceIp`) or a service-specific key(for example, `ec2:InstanceType`).
      - **Qualifier** – (Optional) If you enter multiple values for the condition, you can specify a qualifier for testing requests against the values.
      - **Operator** – You can use operators to restrict access based on comparing a key to a value.

        For any condition operator except the `Null` condition, you can choose the IfExists option.
      - **Value** – (Optional) Specify one or more values for the condition.

      Choose **Add condition**.

      For more information on condition keys, see IAM JSON Policy Elements: Condition in the *IAM User Guide*.

   e. Optional: To use the `NotAction` element to deny access to all of the listed resources except for specified actions, 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 right pane of the **Policy** section, change `"Effect": "Deny"` to `"Effect": "Allow"`.
   b. In the left pane of the **Policy** section, choose the service to add actions for.

      As you choose options on the left, the right pane updates to show the JSON policy. You can also type or paste policies in the editor in the **Policy** section's right pane. However, this procedure describes how to use the visual editor on the left to build your SCP.
   c. From the list that opens of available actions for that service, choose the action or actions to allow.
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. 67).

> **Note**
> SCPs don't take effect on the master account and in a few other situations. For more information, see Tasks and Entities Not Restricted by SCPs (p. 62).

To create a service control policy (AWS CLI, AWS API)

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

- AWS CLI: aws organizations create-policy
- AWS API: CreatePolicy

Updating an SCP

When signed in to your organization's master 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.

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 "*")

To update a policy (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. Choose the **Policies** tab.
3. On the **Policies** tab, choose **Service control policies**.
4. Choose the policy that you want to update.
5. In the details pane on the right, choose **Policy editor**.
6. Choose **Edit** to enable making changes to the policy.
7. Make your changes by editing the policy in the right pane. For deny statements, you can also use the visual editor in the left pane to make changes. When you're finished, choose **Save changes**.

To update a policy (AWS CLI, AWS API)

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

- AWS CLI: aws organizations update-policy
- AWS API: UpdatePolicy

For More Information

For more information on creating SCPs, see the following pages:

- Example Service Control Policies (p. 78)
- SCP Syntax (p. 72)

Deleting a Policy

When signed in to your organization's master 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 one named `FullAWSAccess`.

To delete an SCP, you permission to run the following action:

- `organizations:DeletePolicy`

To delete a policy (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. The policy that you want to delete must first be detached from all roots, OUs, and accounts. Follow the steps in Detaching an SCP from the Organization Root, OUs, or Accounts (p. 68) to detach the policy from all entities in the organization.
3. On the **Policies** tab, choose **Service control policies**.
4. On the **Service control policies** page, choose the SCP to delete.
5. Choose **Delete policy**.

To delete a policy (AWS CLI, AWS API)

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

- AWS CLI: aws organizations delete-policy
- AWS API: DeletePolicy

Attaching SCPs

When signed in to your organization's master account, you can attach a service control policy (SCP) that you previously created. You can attach an SCP to the organization root, to an OU, or directly to an account. To attach an SCP, complete the following steps.

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 ARN of the specified policy and the ARN of the root, OU, or account that you want to attach the policy to

To attach an SCP to a root, OU, or account (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. On the **Organize accounts** tab, navigate to (p. 54) and select the check box for the root, OU, or account you want to attach the SCP to.
3. In the **Details** pane on the right, expand the **Service control policies** section to see the list of the currently attached SCPs.
4. On the list of available SCPs, find the one that you want and choose **Attach**. The list of attached SCPs is updated with the new addition. The SCP goes into effect immediately. For example, an SCP immediately affects the permissions of IAM users and roles in the attached account or all accounts under the attached root or OU.

To attach an SCP to a root, OU, or account (AWS CLI, AWS API)

You can use one of the following commands to attach an SCP:

- AWS CLI: aws organizations attach-policy
- AWS API: AttachPolicy

Detaching an SCP from the Organization Root, OUs, or Accounts

When signed in to your organization's master 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 an entity. There must be at least one SCP attached to all entities at all times.

To detach an SCP from the organization root, OU, or account, you need permission to run the following action:

- `organizations:DetachPolicy`

To detach an SCP from the organization root, OU, or account (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. On the **Organize accounts** tab, navigate to (p. 54) and select the check box for the root, OU, or account from which you want to detach the SCP.
3. In the **Details** pane on the right, expand the **Service control policies** section to see the list of the currently attached SCPs. The **Source** field tells you where the SCP comes from. It can be attached directly to the account or OU, or it could be attached to a parent OU or organization root.

4. Find the SCP that you want to detach and choose **Detach**. The list of attached SCPs is updated. The SCP change caused by detaching the SCP goes into 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.

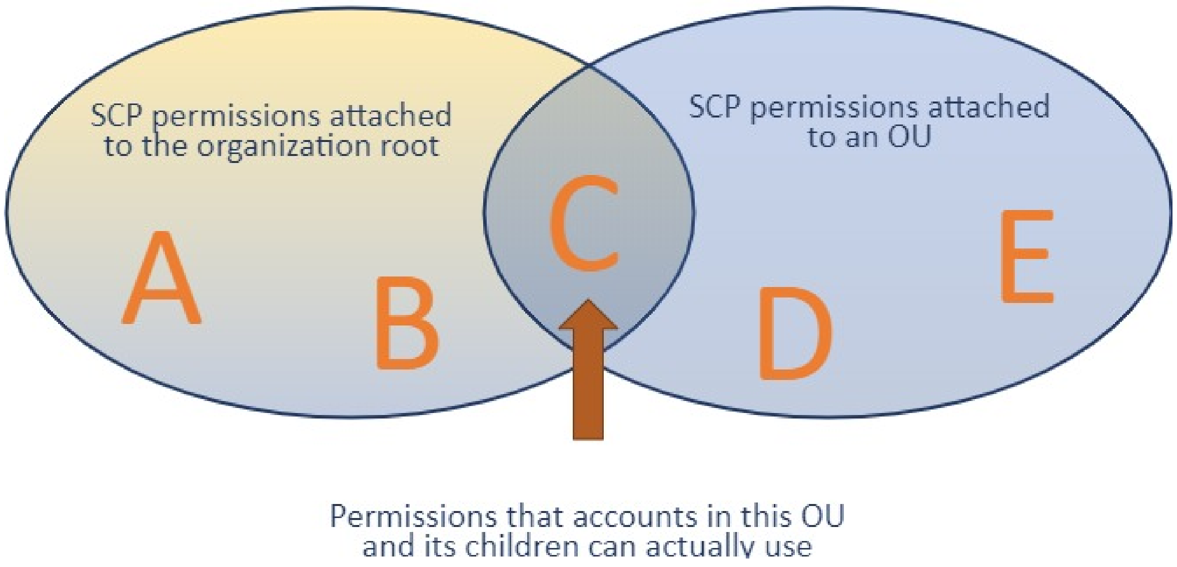
To detach an SCP from a root, OU, or account (AWS CLI, AWS API)

You can use one of the following commands to detach an SCP:

- AWS CLI: aws organizations detach-policy
- AWS API: DetachPolicy

How SCPs Work

The following illustration shows how SCPs (p. 60) work.


SCP permissions attached to the organization root
SCP permissions attached to an OU
A
B
C
D
E
Permissions that accounts in this OU and its children can actually use

In this illustration, an SCP attached to the organization root allows permissions A, B, and C. The organization root contains an organizational unit (OU), and an SCP that allows C, D, and E is attached to that OU. Because the SCP attached to the organization 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 end up blocked because they're blocked by the organization root. Also, 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 under the organization root that are peers to the parent OU could allow A and B.

Users and roles must still be granted permissions using IAM permission policies attached to them or to groups. The SCPs filter the permissions granted by such policies, and 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. 72).

Strategies for Using SCPs

You can configure the SCPs in your organization to work as either of the following:

- A deny list (p. 70) – actions are allowed by default, and you specify what services and actions are prohibited
- An allow list (p. 71) – 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. 152). 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 action 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.

An allow list policy might look like the following example, which enables account users to perform operations for 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 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 on this page.

> **Note**
> All characters in your SCP count against its maximum size (p. 152). 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 (p. 60).

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	Specifies the language syntax rules to use for processing the policy.	`Allow`, `Deny`
Statement	Serves as the container for policy elements. You can have	`Allow`, `Deny`

Element	Purpose	Supported Effects
	multiple statements in SCPs.	
Statement ID (Sid)	(Optional) Provides a friendly name for the statement.	`Allow`, `Deny`
Effect	Defines whether the SCP statement allows (p. 11) or denies (p. 11) principal and root access in an account.	`Allow`, `Deny`
Action	Specifies AWS service/ actions that the SCP allows or denies.	`Allow`, `Deny`
NotAction	Specifies AWS service/ actions that are exempt from the SCP. Used instead of the `Action` element.	`Deny`
Resource	Specifies the AWS resources that the SCP applies to.	`Deny`
Condition	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",
```

`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 EC2:

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "*",
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": "ec2:*",
            "Resource": "*"
        }
    ]
```

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": "*"
    }
}
```

`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.

`"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 where the default `FullAWSAccess` policies are all detached so that permissions are implicitly denied by default. The result is that it allows (p. 11) the Amazon S3 permissions for any attached accounts:

```
{
    "Statement": {
        "Effect": "Allow",
        "Action": "s3:*"
    }
}
```

Even though it uses the same `Allow` value keyword as an IAM permission policy, in an SCP it doesn't actually grant a user permissions to do anything. Instead, SCPs specify the maximum permissions for an organization, organizational unit (OU), or account. In the preceding example, even if a user in the account had the `AdministratorAccess` managed policy attached, the SCP limits ***all*** users in the account 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 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".

> **Note**
> In an SCP, the wildcard (*) character 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*.

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. 11), 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. 11) 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 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).

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 principals in accounts from making changes to a common administrative IAM role created in all accounts in your organization.

`Condition` Element

You can specify a `Condition` element in deny statements in an SCP. For example:

```
{
    "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.

Unsupported Elements

The following elements aren't supported in SCPs:

- `Principal`
- `NotPrincipal`
- `NotResource`

Example Service Control Policies

The example service control policies (SCPs) (p. 60) displayed in this topic are for information purposes only.

Before Using These Examples
Before you attempt to use these example SCPs in your organization, do the following:

- Carefully review and customize them for your unique requirements.
- Test your policies before using them in a production capacity. Remember that an SCP affects every user and role and even 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. 70) 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 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.

Topics

Example 1: Prevent Users from Disabling AWS CloudTrail

This SCP prevents users or roles in any affected account from disabling a CloudTrail log, either directly as a command or through the console.

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

Example 2: Prevent Users from Disabling Amazon CloudWatch or Altering Its Configuration

A lower-level CloudWatch operator needs to monitor dashboards and alarms, but 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 3: Prevent Users from Deleting Amazon VPC Flow Logs

This SCP prevents users or roles in any affected account from deleting 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": "*"
    }
  ]
 }
```

Example 4: 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 5: 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": "*"
    }
  ]
}
```

Example 6: Denies Access to AWS Based on the Requested Region

This SCP denies access to any operations outside of the `eu-central-1` and `eu-west-1` Regions, except for actions in the listed services. To use this SCP, replace the red italicized text in the example policy with your own information.

This policy uses the NotAction element with the `Deny` effect to deny access to all of the actions *not* listed in the statement. The listed services are examples of AWS global services with a single endpoint that is physically located in the `us-east-1` Region. Requests made to services in the `us-east-1` Region aren't denied if they're included in the `NotAction` element. Any other requests to services in the `us-east-1` Region are denied.

Notes

- Not all AWS global services are shown in this example policy. Replace the list of services in red italicized text with the global services used by accounts in your organization.
- This example policy blocks access to the AWS Security Token Service global endpoint (`sts.amazonaws.com`). To use AWS STS with this policy, use regional endpoints or add `"sts:*"` to the `NotAction` element. For more information on AWS STS endpoints, see Activating and Deactivating AWS STS in an AWS Region in the *IAM User Guide*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyAllOutsideEU",
            "Effect": "Deny",
            "NotAction": [
               "iam:*",
               "organizations:*",
               "route53:*",
               "budgets:*",
               "waf:*",
               "cloudfront:*",
               "globalaccelerator:*",
               "importexport:*",
               "support:*"
            ],
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestedRegion": [
                        "eu-central-1",
                        "eu-west-1"
                    ]
                }
            }
        }
    ]
}
```

Example 7: Prevent IAM Principals from Making Certain Changes

This SCP restricts IAM principals in accounts from making changes to a common administrative IAM role 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/role-to-deny"
      ]
    }
  ]
}
```

Example 8: Prevent IAM Principals from Making Certain Changes, with Exceptions for Admins

This SCP builds on the previous example to make an exception for administrators. It prevents IAM principals in 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/role-to-deny"
      ],
      "Condition": {
        "StringNotLike": {
          "aws:PrincipalARN":"arn:aws:iam::*:role/role-to-allow"
        }
      }
    }
  ]
}
```

Example 9: Require Encryption on Amazon S3 Buckets

This SCP requires that principals use AES256 encryption when writing to Amazon S3 buckets.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyIncorrectEncryptionHeader",
      "Effect": "Deny",
      "Action": "s3:PutObject",
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "s3:x-amz-server-side-encryption": "AES256"
        }
      }
    },
    {
      "Sid": "DenyUnEncryptedObjectUploads",
      "Effect": "Deny",
      "Action": "s3:PutObject",
      "Resource": "*",
      "Condition": {
        "Null": {
          "s3:x-amz-server-side-encryption": true
        }
      }
    }
  ]
}
```

Example 10: 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 11: Require MFA to Stop an Amazon EC2 Instance

Use an SCP like the following to require that multi-factor authentication (MFA) is enabled before a principal or root user can 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}}
    }
  ]
}
```

Example 12: Restrict Access to Amazon EC2 for Root User

The following policy restricts all access to Amazon EC2 actions for the root user in an 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"
          ]
        }
      }
    }
  ]
}
```

Example 13: Require a Tag Upon Resource Creation

The following SCP prevents account principals from creating certain resource types without the specified tags.

```
{
  "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*.

Tag Policies

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. 123).
- 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 signed in to the organization's master 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 master 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 signed 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 master 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. 28). For more information on what's required to use tag policies, see Prerequisites and Permissions for Managing Tag Policies (p. 87).

Important
To get started with tag policies, AWS strongly recommends that you follow the example workflow described in Getting Started with Tag Policies (p. 89) 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 this 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. 28).
- You must be signed in to your organization's master account.
- You need the permissions that are listed in Permissions for Managing Tag Policies (p. 87).

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. 89).
- **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. 98). Also see the workflows described in Getting Started with Tag Policies (p. 89).
- **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 Example 13: Require a Tag Upon Resource Creation (p. 85). 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. 87)
- Best Practices for Using Tag Policies (p. 88)

Using Tag Policies for the First Time

Follow these steps to get started using tag policies for the first time.

Task	How to Perform
Step 1: Enable tag policies for your organization. (p. 92)	• **Account to sign in to:** The organization's master account.¹ • **AWS service console to use:** AWS Organizations.
Step 2: Create a tag policy (p. 93). 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.	• **Account to sign in to:** The organization's master account.¹ • **AWS service console to use:** Organizations.
Step 3: Attach a tag policy to a single member account that you can use for testing. (p. 95) You'll need to sign in to this account in the next step.	• **Account to sign in to:** The organization's master account.¹ • **AWS service console to use:** Organizations.
Step 4: Create some resources with compliant tags and some with noncompliant tags.	• **Account to sign in to:** The member account that you're using for testing purposes. • **AWS service console to use:** 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 noncompliant secrets.
Step 5: View the effective tag policy and evaluate the compliance status of the account.	• **Account to sign in to:** The member account that you're using for testing purposes. • **AWS service console to use:** Resource Groups and the AWS service where the resource was created. If you created resources with compliant and noncompliant tags, you should see the noncompliant 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.	• **Account to sign in to:** The member account that you're using for testing purposes.

Task	How to Perform
	• **AWS service console to use:** Resource Groups and the AWS service where the resource was created.
At any time, you can evaluate organization-wide compliance.	• **Account to sign in to:** The organization's master account.[1] • **AWS service console to use:** 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 master account.

If you're using the AWS Command Line Interface, the complete process and examples are described in Using Tag Policies in the AWS CLI (p. 117).

Expanding Use of Tag Policies

You can perform the following tasks in any order to expand your use of tag policies.

Advanced Task	How to Perform
Create more advanced tag policies (p. 93). 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 How Policy Inheritance Works (p. 108) and Tag Policy Syntax (p. 104) to create more detailed tag policies.	• **Account to sign in to:** The organization's master account.[1] • **AWS service console to use:** Organizations.
Attach tag policies to additional accounts or OUs. (p. 95) Check the effective tag policy for an account (p. 97) after you attach more policies to it or to any OU in which the account is a member.	• **Account to sign in to:** The organization's master account.[1] • **AWS service console to use:** Organizations.
Create an SCP to require tags when anyone creates new resources. For an example, see Example 13: Require a Tag Upon Resource Creation (p. 85).	• **Account to sign in to:** The organization's master account.[1] • **AWS service console to use:** Organizations.
Continue to evaluate the compliance status of the account against the effective tag policy as it changes. Correct noncompliant tags.	• **Account to sign in to:** A member account with an effective tag policy. • **AWS service console to use:** Resource Groups and the AWS service where the resource was created.
Evaluate organization-wide compliance.	• **Account to sign in to:** The organization's master account.[1] • **AWS service console to use:** 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 master 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. 98).

Enforcement Tasks	How to Perform
Step 1: Create a tag policy (p. 93). 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 on. Continuing with our earlier example, you can choose to enforce it on Secrets Manager secrets.	• **Account to sign in to:** The organization's master account.¹ • **AWS service console to use:** Organizations.
Step 2: Attach a tag policy to a single, test account. (p. 95)	• **Account to sign in to:** The organization's master account.¹ • **AWS service console to use:** 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.	• **Account to sign in to:** The member account that you're using for testing purposes. • **AWS service console to use:** Any AWS service 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 noncompliant secrets.
Step 4: Evaluate the compliance status of the account against the effective tag policy and correct noncompliant tags.	• **Account to sign in to:** The member account that you're using for testing purposes. • **AWS service console to use:** 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.	• **Account to sign in to:** The member account that you're using for testing purposes. • **AWS service console to use:** Resource Groups and the AWS service where the resource was created.
At any time, you can evaluate organization-wide compliance.	• **Account to sign in to:** The organization's master account.¹ • **AWS service console to use:** 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 master account.

Enabling Tag Policies

Before you can create and attach tag policies, you must enable this feature. Enabling the use of tag policies is a one-time task. You enable tag policies on the organization root, even if you plan to attach tag policies to individual accounts only. You must be signed in to the organization's master account to enable tag policies.

To enable tag policies, you need permissions to run the following actions:

- `organizations:EnablePolicyType`
- `organizations:DescribeOrganization`
- `organizations:ListRoots`

To enable tag policies (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. In the upper-right corner, choose **Settings**.
3. In the **Trusted access for AWS services** section, find **Tag policies** and then choose **Enable access**. When prompted, confirm the action.
4. Choose the **Policies** tab, and then choose **Tag policies**.
5. Under **What is a tag policy?**, choose **Enable tag policies**.

To enable tag policies (AWS CLI, AWS API)

You can use one of the following to enable tag policies:

- AWS CLI: aws organizations enable-policy-type

  For the complete procedure for using tag policies in the AWS CLI, see Using Tag Policies in the AWS CLI (p. 117).
- AWS API: EnablePolicyType

What to Do Next

After you enable tag policies, you can create tag policies (p. 93).

Disabling Tag Policies

When you disable tag policies, all tag policies are automatically detached, but not deleted, from all entities in the organization root.

You can disable tag policies from the organization's master account only.

To disable tag policies, you should have permissions to run the following actions:

- `organizations:DescribeOrganization`
- `organizations:DisablePolicyType`
- `organizations:ListRoots`

To disable tag policies (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. On the **Organize accounts** tab, choose **Root** in the left navigation pane.
3. In the details pane on the right side, next to **Tag policies**, choose **Disable**.

To disable tag policies (AWS CLI, AWS API)

You can use one of the following to disable tag policies:

- AWS CLI: aws organizations disable-policy-type
- AWS API: DisablePolicyType

Creating and Updating Tag Policies

After you enable tag policies for your organization, you can create a tag policy.

> **Important**
> Untagged resources don't appear as noncompliant in results.

Creating a Tag Policy

To create tag policies, you need permission to run the following action:

- `organizations:CreatePolicy`

To create a tag policy (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. On the **Policies** tab, choose **Tag policies**.
3. On the **Tag policies** page, choose **Create policy**.
4. On the **Create policy** page, enter a name and description for the policy.

   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 on tag policy syntax, see Tag Policy Syntax (p. 104).
5. For **Tag Key**, 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 parent tag policy should define the case treatment for the tag key.

   Select this option only if you want to specify different capitalization for the tag key. If you select this option, the capitalization you specified for **Tag Key** overrides the case treatment specified in a parent policy.

   If a parent policy doesn't exist and you don't select this option, tag keys in all lowercase characters are considered compliant. For more information about parent policies, see How Policy Inheritance Works (p. 108).

   > **Tip**
   > Consider using the example tag policy shown in Example 1: Define Organization-wide Tag Key Case (p. 106) 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**, select the check box if you want to add allowed values for this tag key.

   By default, this option is cleared, which means that only any values inherited from a parent policy are considered compliant. If a parent policy doesn't exist or specify tag values, 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 **Specify values**. When prompted, enter the new values and choose **Save changes**.
8. For **Prevent noncompliant operations for this tag**, 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. 98). 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 allowed values**. When prompted, enter the resource types to add. Then choose **Save changes**.
9. (Optional) To add another tag key to this tag policy, choose **Add tag key**. Then perform steps 5–8 to define the tag key.
10. When you're finished building your tag policy, choose **Save changes**.

To create a tag policy (AWS CLI, AWS API)

You can use one of the following to create a tag policy:

- AWS CLI: aws organizations create-policy

  For the complete procedure for using tag policies in the AWS CLI, see Using Tag Policies in the AWS CLI (p. 117).
- AWS API: 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. 95) to the organization root, organizational units (OUs), AWS accounts within your organization, or a combination of organization entities.

Updating a Tag Policy

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 "*")

To update a tag policy (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. On the **Policies** tab, choose **Tag policies**.
3. On the **Tag policies** page, choose the tag policy that you want to update.
4. In the details pane on the right, choose **View details**.

5. On the page that shows the tag policy, choose **Edit policy**.
6. Make your changes either by using the visual editor or by editing the JSON.
7. When you're finished updating the tag policy, choose **Save changes**.

To update a policy (AWS CLI, AWS API)

You can use one of the following to update a policy:

- AWS CLI: aws organizations update-policy
- AWS API: UpdatePolicy

Deleting a Tag Policy

When signed in to your organization's master 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.

To delete a tag policy, you must have permission to run the following action:

- `organizations:DeletePolicy`

To delete a tag policy (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. The tag policy that you want to delete must first be detached from all roots, OUs, and accounts. Follow the steps in Detaching a Tag Policy (p. 96) to detach the tag policy from all entities in the organization.
3. On the **Policies** tab, choose **Tag policies**.
4. From the **Tag policies** page, choose the tag policy that you want to delete.
5. Choose **Delete policy**.

To delete a tag policy (AWS CLI, AWS API)

You can use one of the following to delete a policy:

- AWS CLI: aws organizations delete-policy
- AWS API: DeletePolicy

Attaching 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. 97). For more information, see How Policy Inheritance Works (p. 108).

Important
Untagged resources don't appear as noncompliant in results.

To attach tag policies, you must have permission to run the following action:

- `organizations:AttachPolicy`

To attach a tag policy to the organization root, OU, or account (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. On the **Organize accounts** tab, navigate to (p. 54) and select the check box for the root, OU, or account you want to attach the tag policy to.
3. In the details pane on the right, expand the **Tag policies** section to see the list of the currently attached tag policies.
4. On the list of available tag policies, find the one that you want and choose **Attach**.

To attach a tag policy to the organization root, OU, or account (AWS CLI, AWS API)

You can use one of the following to attach a tag policy:

- AWS CLI: aws organizations attach-policy
- AWS API: AttachPolicy

What to Do Next

After you attach a tag policy, you can find out how compliant that account's 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 signed in to your organization's master 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.

To detach a tag policy from the organization root, OU, or account, you must have permission to run the following action:

- `organizations:DetachPolicy`

To detach a tag policy from the organization root, OU, or account (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. On the **Organize accounts** tab, navigate to (p. 54) and select the check box for the organization root, OU, or account from which you want to detach the policy.
3. In the **Details** pane on the right, expand the **Tag policies** section to see the list of the currently attached tag policies.

4. Find the tag policy that you want to detach and choose **Detach**. The list of attached tag policies is updated with the chosen policy removed.

To detach a tag policy from the organization root, OU, or account (AWS CLI, AWS API)

You can use one of the following to detach a tag policy:

- AWS CLI: aws organizations detach-policy
- AWS API: DetachPolicy

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 five 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 How Policy Inheritance Works (p. 108).

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.

To view the effective tag policy for an account, you must have permission to run the following actions:

- `organizations:DescribeEffectivePolicy`
- `organizations:DescribeOrganization`

To view the effective policy for an account (console)

1. Sign in to the organization's master account.

   Note
   When you are signed in to a member account, the procedure for viewing the effective policy is different. When signed in to an account, you can view the effective tag policy in the context of evaluating compliance for the acount. For more information, see Evaluating Compliance for an Account in the *AWS Resource Groups User Guide.*
2. On the **Accounts** tab, choose the account.
3. In the details pane on the right, expand the **Tag policies** section.
4. Choose **View effective policy**.

To view the effective policy for an account (AWS CLI, AWS API)

You can use one of the following to view the effective tag policy:

- AWS CLI: aws organizations describe-effective-policy

  For the complete procedure for using tag policies in the AWS CLI, see Using Tag Policies in the AWS CLI (p. 117).
- AWS API: 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. 93):

- From the **Visual editor** tab, select **Prevent noncompliant operations for this tag** (p. 94).
- From the **JSON** tab, use the `enforced_for` field. For information on tag policy syntax, see Tag Policy Syntax and Examples (p. 104).

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. 89). 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. 99). 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.

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 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 Certificate Manager	• All • Certificates	• `"acm:*"` • `"acm:certificate"`
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"`
AWS CodeBuild	• All	• `"codebuild:*"`

Service Name	Resource Type	JSON Syntax
	• Project	• `"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"`
AWS Database Migration Service	• All • Endpoint • ES • Rep • Subgrp • Task	• `"dms:*"` • `"dms:endpoint"` • `"dms:es"` • `"dms:rep"` • `"dms:subgrp"` • `"dms:task"`
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"`
AWS Elastic Beanstalk	• Application • Application version • Configuration template • Platform	• `"elasticbeanstalk:application"` • `"elasticbeanstalk:applicationversi` • `"elasticbeanstalk:configurationtem` • `"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 ElastiCache	• Cluster	• `"elasticache:cluster"`
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"`
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 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 RDS	• Cluster PG • ES • OG • PG • RI • Secgrp • Subgrp	• `"rds:cluster-pg"` • `"rds:es"` • `"rds:og"` • `"rds:pg"` • `"rds:ri"` • `"rds:secgrp"` • `"rds:subgrp"`

Service Name	Resource Type	JSON Syntax
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"`
Amazon Route 53 Resolver	• All • Resolver endpoint • Resolver rule	• `"route53resolver:*"` • `"route53resolver:resolver-endpoint"` • `"route53resolver:resolver-rule"`
Amazon S3	• Bucket	• `"s3:bucket"`
AWS Secrets Manager	• All • Secret	• `"secretsmanager:*"` • `"secretsmanager:secret"`
Amazon Simple Queue Service (SQS)	• Queue	• `"sqs:queue"`
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	• `"ssm:automation-execution"` • `"ssm:document"` • `"ssm:maintenancewindowtask"` • `"ssm:managed-instance"` • `"ssm:opsitem"` • `"ssm:patchbaseline"` • `"ssm:session"`
Amazon WorkSpaces	• All • Directory • WorkSpace • WorkSpaces bundle • WorkSpaces image • WorkSpaces IP group	• `"workspaces:*"` • `"workspaces:directory"` • `"workspaces:workspace"` • `"workspaces:workspacebundle"` • `"workspaces:workspaceimage"` • `"workspaces:workspaceipgroup"`

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 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 components:

- 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 key *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. 97). 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. 109).
- You can use the `*` wildcard in tag values and `enforced_for` fields:
  - You can use 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. 99).

    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. 86) 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. 89) 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. 152). 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 changes 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 How Policy Inheritance Works (p. 108).

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. 109) are allowed in child tag policies. Therefore, any `Color` tags on resources in affected accounts are noncompliant. In addition, the `enforced_for` option actually prevents affected accounts from tagging 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"
                ],
                "@@append": []
            },
            "enforced_for": {
                "@@assign": [
                    "dynamodb:table"
                ]
            }
        }
    }
}
```

How Policy Inheritance Works

You can attach tag policies to any organization entity (organization root, organizational unit, or account):

- When you attach a tag policy to the organization root, all accounts in the organization inherit that policy.
- When you attach a tag policy to a specific OU, accounts that are directly under that OU or any child OU inherit the policy.
- When you attach tag policies to an account, they affect only that account.

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

The *effective tag policy* is the set of tagging rules that are inherited from the organization root and OUs, plus the attached account tag policies. The effective tag policy specifies the tagging rules that apply to the account. To learn how to view the effective tag policy for an account, see Viewing Effective Tag Policies (p. 97).

Important
Untagged resources don't appear as noncompliant in results.

This page describes how parent policies and child policies are aggregated into the effective policy for an account.

Terminology

The following table describes common terms used in defining how policy inheritance works.

Term	Definition
Policy inheritance	The interaction of tag policies at differing levels of an organization. You can attach tag policies to the organization root, organizational units (OUs), individual accounts, and to a 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

Term	Definition
	members of the organization root or OU where a tag policy is attached *inherit* that tag 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 policies are attached to a specific OU, accounts that are directly under that OU or any child OU inherit the 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, 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 policy (p. 97)	A single policy document that specifies the tagging rules that apply to an account. The effective policy is the aggregation of any tag policies the account inherits, plus any tag policy that is directly attached to the account.
Inheritance operators (p. 109)	Operators that control how inherited policies merge into a single effective policy. These operators are considered an advanced feature. Experienced tag 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 tag policies and account tag policies merge into the account's effective tag 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 tag policy authors can use them to control what tag values are applied to the effective tag policy and limit what changes child policies can make.

Value-Setting Operators

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

- `@@assign` – Overwrites inherited tag keys and values with the specified tag keys and values. If the specified tag isn't inherited, this operator adds the key and value to the effective policy.
- `@@append` – Adds the specified tag keys and values to the inherited tag keys and values. If the specified tag isn't inherited, this operator adds the keys and values to the effective policy.

- `@@remove` – Removes specific inherited tag keys and values from the effective policy, if they exist. If this operator removes all values from a tag policy key, accounts that are directly under that OU or any child OU can use any value for that tag.

Child Control Operators

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

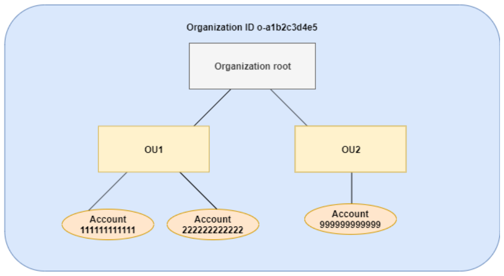
- `"@@operators_allowed_for_child_policies":["@@all"]` – Child OUs and accounts can use any operator in tag policies. By default, all operators are allowed in child policies.
- `"@@operators_allowed_for_child_policies":["@@assign", "@@append", "@@remove"]` – Child OUs and accounts can use only the specified operators in tag 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 tag policies. To effectively lock down values that are defined in a parent policy so that child policies can't add, append, or remove values, use this operator.

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

Tag Policy Inheritance Examples

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

The examples assume that you have the following organization structure.



Example 1: Allow Child Policies to Overwrite Tag Values

The following tag policy defines the `CostCenter` tag key and acceptable values. 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 different tag value for a key, and you want to enforce it 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. `Sandbox` is only compliant value for the `CostCenter` tag key.
- The addition of `enforced_for` specifies that the `CostCenter` tag must be used 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.

Effective tag policy for accounts 111111111111 and 222222222222

```
{
    "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 `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 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

```
{
    "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 do that 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 to 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

```
{
    "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, 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 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

```
{
    "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 `Maintenance`.

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 can include conflicting information. These 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.

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

Using Tag Policies in the AWS CLI

To use tag policies in the AWS Command Line Interface, AWS recommends that you first follow the basic workflow described in this topic.

Recommended Workflow

The recommended workflow for using tag policies is as follows:

1. Enable tag policies for your organization. (p. 117)
2. Create a tag policy. (p. 118)
3. Attach a tag policy to a single, test account. (p. 118)
4. View the effective tag policy for that account. (p. 119)
5. Find and correct noncompliant tags on resources. (p. 119)
6. Find and correct more noncompliant tags on resources. (p. 120)
7. At any time, you can find out the compliance status for all tagged resources across all accounts across your organization. To do this, generate the organization-wide report. (p. 120)

Enabling Tag Policies for Your Organization

Enabling the use of tag policies is a one-time task. You enable tag policies on the organization root, even if you plan to attach tag policies to individual accounts only.

To enable tag policies

1. Find the root ID of your organization so you can specify where to enable and attach a tag policy. To find the root ID, run the following from a command prompt:

```
aws --region us-east-1 organizations list-roots
```

The following is an example result:

```
{
    "Roots": [
        {
            "Id": "r-examplerootid111",
            "Arn": "arn:aws:organizations::111111111111:root/o-exampleorgid/r-
examplerootid111",
            "Name": "Root",
            "PolicyTypes": []
        }
    ]
}
```

In this example, `r-examplerootid111` is the root ID of the organization. You use this root ID in the next step.

2. Run the following to enable tag policies for the organization:

```
aws --region us-east-1 resourcegroupstaggingapi enable-tag-policies --root-id r-
examplerootid111
```

This command enables tag policies for the organization with the root ID `r-examplerootid111.`

Creating a Tag Policy

After you enable tag policies, you're ready to create your first tag 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.

To create a tag policy

- Create a tag policy like the following:

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

This tag policy defines the `CostCenter` tag key. The tag can accept any value or no value. That's because when you don't specify at least one value, any value or no value is compliant.

Attach a Tag Policy

After you create a tag policy, you're ready to attach it to your organization root, an OU, or to an individual account. Attaching a tag policy to the organization root affects all of your organization's member accounts. When you attach a tag policy to an individual account, only that account is subject to that tag policy. It is also subject to any tag policy that is attached to the organization root.

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 --region us-east-1 organizations attach-policy --target-id <account-id> --policy
 file://<path-and-filename>
```

Determining the Effective Policy for an Account

To start checking compliance status for tagged resources in an account, it's helpful to first determine the effective tag policy for the account.

> **Note**
> If you did not specify a default Region when you configured the AWS CLI, you must specify the Region for account-level commands. You must also specify a Region if you want the command to apply to a Region other than your default. For more information, see Configuring the AWS CLI in the *AWS Command Line Interface User Guide*.

To determine what tagging rules are attached to an account, run the following from the account and save the results to a file:

```
aws --region region-name organizations describe-effective-policy
```

If a tag policy is attached to the account as well as to the organization root, the combination of both policies defines the account's effective tag policy. In these cases, running `describe-effective-policy` from the account returns the contents of both tag policies.

Finding Noncompliant Resources for an Account

For each account, you can get information about noncompliant resources. You should run this command from every Region in which the account has resources.

> **Note**
> If you did not specify a default Region when you configured the AWS CLI, you must specify the Region for account-level commands. You must also specify a Region if you want the command to apply to a Region other than your default. For more information, see Configuring the AWS CLI in the *AWS Command Line Interface User Guide*.

To find noncompliant resources for an account that uses the effective policy, run the following when signed in to the account and save the results to a file:

```
aws --region region-name resourcegroupstaggingapi get-resources --include-compliance-
details --exclude-compliant-resources
```

Here is an example:

```
aws --region us-east-1 resourcegroupstaggingapi get-resources --include-compliance-details
 --exclude-compliant-resources
```

Correcting Noncompliant Tags in Resources

After finding noncompliant tags, make corrections using any of the following methods. You must be signed in to the account that has the resource with noncompliant tags:

- Use the console or API operations of the service that has the noncompliant resources.

  Use TagResources to add tags that are compliant with the effective policy.
- Use UntagResources to remove tags that are noncompliant with the effective tag policy.

Finding and Correcting Additional Noncompliance Issues

Finding and correcting compliance issues is an iterative process.

To continue finding and correcting compliance issues

1. After finding and correcting noncompliant tags in resources, rerun the `get-resources` command with the `--include-compliance-details` and `--exclude-compliant-resources` parameters to ensure that you corrected all the previously returned issues.

   Run this command in all Regions where you have resources.
2. Make additional corrections.
3. Repeat the process of finding and correcting compliance issues 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 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 master 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
```

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
```

When the above command returns `SUCCEEDED`, you can open the report from the Amazon S3 bucket.

Supported Regions

Tag policy features are available in the following Regions:

Region Name	Region Parameter
US East (Ohio) Region	`us-east-2`

Region Name	Region Parameter
US East (N. Virginia) Region[1]	**`us-east-1`**
US West (N. California) Region	`us-west-1`
US West (Oregon) Region	`us-west-2`
Asia Pacific (Hong Kong) Region[2]	`ap-east-1`
Asia Pacific (Mumbai) Region	`ap-south-1`
Asia Pacific (Osaka-Local) Region[3]	`ap-northeast-3`
Asia Pacific (Seoul) Region	`ap-northeast-2`
Asia Pacific (Singapore) Region	`ap-southeast-1`
Asia Pacific (Sydney) Region	`ap-southeast-2`
Asia Pacific (Tokyo) Region	`ap-northeast-1`
Canada (Central) Region	`ca-central-1`
Europe (Frankfurt) Region	`eu-central-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 is not available in these Regions.

[3]This Region doesn't support reporting on resources.

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 are case sensitive.
- A *tag value* (for example, `111122223333` or `Production`). 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. Like tag keys, tag values are case sensitive.

You can use tags to categorize resources by purpose, owner, environment, or other criteria. For more information, see AWS Tagging Strategies.

Tip
You can use tag policies (p. 86) to help standardize tags across resources in your organization's accounts.

Supported Resources in AWS Organizations

Currently, AWS Organizations supports the following tagging operations when you are logged in to the master account:

- You can tag and untag accounts in AWS Organizations.
- You can view tags on an account in AWS Organizations.

AWS Organizations doesn't currently support tagging resources within an account, cost allocation tags, or the tag-based access control feature of AWS Identity and Access Management (IAM).

Adding Tags

When signed in with permissions to your organization's master account, you can add tags to accounts in your organization.

To add tags to accounts in your organization, you need permission to run the following actions:

- `organizations:ListTagsForResource` (console only)
- `organizations:TagResource`

To add a tag to an account in your organization (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. On the **Accounts** tab, choose an account.
3. In the **TAGS** section of the details pane on the right, choose **EDIT TAGS**.

4. Enter a key and, optionally, a value for the tag.

   Tag keys and values are case sensitive. Use the capitalization that you want to standardize on.
5. Choose **Save changes**.

Any tags that you added to the account appear in the **TAGS** section of the details pane on the right.

To add a tag to an account in your organization (AWS CLI, AWS API)

You can use one of the following commands to add tags to accounts:

- AWS CLI: aws organizations tag-resource
- AWS API: TagResource

Viewing Tags on an Account

When signed in with permissions to your organization's master account, you can view tags on an account in your organization.

To view tags on an account in your organization, you need permission to run the following action:

- `organizations:ListTagsForResource`

To view tags on an account in your organization (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. On the **Accounts** tab, choose an account.
3. In the details pane on the right, find the **TAGS** section.

All tags that are attached to the selected account are displayed.

To view tags on an account in your organization (AWS CLI, AWS API)

You can use one of the following commands to view tags on an account:

- AWS CLI: aws organizations list-tags-for-resource
- AWS API: ListTagsForResource

Editing Tag Values

When signed in with permissions to your organization's master account, you can edit *tag values* on tags that are attached to accounts.

To edit *tag keys*, you need to delete the tag key and then add a new tag key. For more information, see Deleting Tags (p. 125) and Adding Tags (p. 123).

To edit tag values on tags that are attached to accounts, you need permission to run the following actions:

- `organizations:ListTagsForResource`

- `organizations:TagResource`

To edit a tag value for a tag on an account in your organization (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. On the **Accounts** tab, choose an account.
3. In the **TAG** section of the details pane on the right, choose **EDIT TAGS**.
4. Modify the value of the tag that you want to change.
5. Choose **Save changes**.

The **TAGS** section in the details pane on the right updates with any changes that you made to tag values for tags on the account.

To edit a tag value on an account in an organization (AWS CLI, AWS API)

1. Delete the existing tag value by using one of the following commands:
   - AWS CLI: aws organizations untag-resource
   - AWS API: UntagResource
2. Add a new tag value by using one of the following commands:
   - AWS CLI: aws organizations tag-resource
   - AWS API: TagResource

Deleting Tags

When signed in with permissions to your organization's master account, you can delete tags that are attached to accounts in your organization.

To delete tags, you need permission to run the following actions:

- `organizations:ListTagsForResource`
- `organizations:UntagResource`

To delete a tag from an account in your organization (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/. 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 master account.
2. On the **Accounts** tab, choose an account.
3. In the **TAGS** section of the details pane on the right, choose **EDIT TAGS**.
4. Choose **Remove** next to the tag to delete it.
5. Choose **Save changes**.

The **TAGS** section in the details pane no longer displays the tags that you deleted.

To delete a tag from an account in your organization (AWS CLI, AWS API)

You can use one of the following commands to delete tags:

- AWS CLI: aws organizations untag-resource
- AWS API: UntagResource

Enabling Trusted Access with Other AWS Services

You can use *trusted access* to enable an 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 serviced 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 recommend that you enable trusted access by using the trusted service's console, the AWS CLI, or the API operations in one of the AWS SDKs. This enables the trusted service to perform any required initialization or create any required resources. To learn what configuration the trusted service performs, see the documentation for that 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) that are available in the trusted service. This allows the trusted 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. 141).
  - `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. 129).

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 Services That Support Trusted Access with Your Organization (p. 130).

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. 60) in AWS Organizations.

- 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. 141).
  - `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 Services That Support Trusted Access with Your Organization (p. 130).
  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. 129).

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.

To enable or disable trusted service access (console)

1. Sign in to the Organizations console at https://console.aws.amazon.com/organizations/.
2. In the upper-right corner, choose **Settings**.
3. If you are *enabling* access, proceed to the next step. If you are *disabling* access, wait until the administrator tells you that the service is disabled and the resources have been cleaned up.
4. In the **Trusted access for AWS services** section, find the service that you want and then choose either **Enable access** or **Disable access** as appropriate.
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 enable or disable trusted service access (AWS CLI, AWS API)

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 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 account with a service-linked role named `AWSServiceRoleForOrganizations`. Only the AWS Organizations service itself can assume this role. The role has permissions that allow only 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. 11) enabled, the service-linked role named `AWSServiceRoleForOrganizations` is never used, and you can delete it. If you later want to enable all features (p. 11) 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 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.

Services That Support Trusted Access with Your Organization

The following sections describe the AWS services for which you can enable trusted access with your organization. Each section includes the following:

- A summary of the trusted service and how it works when you enable trusted access
- Links to instructions for enabling and disabling trusted access with your organization
- The principal name of the trusted service that you can specify in policies to grant the trusted access to the accounts in your organization
- If applicable, the name of the IAM service-linked role created in all accounts when you enable trusted access

For more information about the benefits of using other AWS services with Organizations, see AWS Services That You Can Use with AWS Organizations (p. 4).

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 a master 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 about AWS Artifact, see the AWS Artifact User Guide.

The following list provides information that is useful to know when you want to integrate AWS Artifact and Organizations:

- **To enable trusted access with AWS Organizations:** You must sign in with your AWS Organizations master account to configure an account within the organization as the AWS Artifact administrator account. For information, see Step 1: Create an Admin Group and Add an IAM User in the *AWS Artifact User Guide*.
- **To disable trusted access with AWS Organizations:** 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.

- **Service principal name for AWS Artifact:** `aws-artifact-account-sync.amazonaws.com`.
- **Role name created to synchronize with AWS Artifact:** `AWSArtifactAccountSync`.

AWS CloudFormation StackSets and AWS Organizations

AWS CloudFormation StackSets enables you to create, update, or delete stacks across multiple accounts and Regions with a single operation. For more information about StackSets, see Working with AWS CloudFormation StackSets in the *AWS CloudFormation User Guide*.

The following list provides information that you need when integrating AWS CloudFormation StackSets with AWS Organizations:

- **To enable trusted access with AWS Organizations:** Only an administrator in an AWS Organizations master account has permissions to enable trusted access. You can enable trusted access using either the AWS CloudFormation console or the AWS Organizations console. To enable trusted access using the AWS CloudFormation console, see Enable Trusted Access with AWS Organizations in the AWS CloudFormation User Guide. To enable trusted access using the AWS Organizations console:

  1. Sign in to AWS as an administrator of the master account and open the AWS Organizations console at https://console.aws.amazon.com/organizations/.
  2. On the **Settings** page, next to **AWS CloudFormation**, choose **Enable access**.
- **To disable trusted access with AWS Organizations:** Only an administrator in an AWS Organizations master account has permissions to disable trusted access. You can only disable trusted access using the AWS Organizations console. If you disable trusted access with AWS Organizations while you are using AWS CloudFormation StackSets, all previously created stack instances are retained. However, stack sets with service-managed permissions will no longer perform deployments to accounts managed by AWS Organizations.

  1. Sign in to AWS as an administrator of the master account and open the AWS Organizations console at https://console.aws.amazon.com/organizations/.
  2. On the **Settings** page, next to **AWS CloudFormation**, choose **Disable access**.
- **Name suffixes of the IAM service-linked roles that are automatically created in administrator and target accounts** when trusted access is enabled:
  - Administrator (master) account: `CloudFormationStackSetsOrgAdmin`. You can modify or delete this role only if trusted access with AWS Organizations is disabled.
  - Target accounts: `CloudFormationStackSetsOrgMember`. You can modify or delete this role only if trusted access with AWS Organizations is disabled, or if the account is removed from the target organization or organizational unit (OU).

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 master 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*.

The following list provides information that you need when integrating AWS CloudTrail with AWS Organizations:

- **To enable trusted access with AWS Organizations:** You must sign in with your AWS Organizations master 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.*
- **To disable trusted access with AWS Organizations:** 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.
- **Service principal name for AWS CloudTrail:** `cloudtrail.amazonaws.com`.
- **Role name created to synchronize with AWS CloudTrail:** `AWSServiceRoleForCloudTrail`.

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.

The following list provides information that is useful to know when you want to integrate AWS Compute Optimizer and AWS Organizations:

- **To enable trusted access with AWS Organizations** – You must sign in with your AWS Organizations master account and opt in your accounts. For information, see Opting in your Account in the *AWS Compute Optimizer User Guide*.
- **Service principal name for AWS Compute Optimizer** – `compute-optimizer.amazonaws.com`.
- **Name of the IAM service-linked role that can be created in accounts when trusted access is enabled** – `AWSServiceRoleForComputeOptimizer`.

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*.

The following list provides information that is useful to know when you want to integrate AWS Config and AWS Organizations:

- **To enable trusted access with AWS Organizations:** To enable trusted access to AWS Organizations from AWS Config, you 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*.
- **Service principal name**
  - For AWS Config: `config.amazonaws.com`
  - For AWS Config rules: `config-multiaccountsetup.amazonaws.com`
- **Name of the IAM service-linked role that can be created in accounts** when trusted access is enabled
  - For AWS Config: `AWSConfigRoleForOrganizations`
  - For AWS Config rules: `AWSServiceRoleForConfigMultiAccountSetup`

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.

The following list provides information that is useful to know when you want to integrate AWS Directory Service for Microsoft Active Directory and AWS Organizations:

- **To enable trusted access with AWS Organizations:** AWS Directory Service requires trusted access to AWS Organizations before you can share a Microsoft AD directory with an account inside your organization. For more information, see Share Your Directory in the *AWS Directory Service Administration Guide*.
- **To disable trusted access with AWS Organizations:** 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 will no longer be able to share new directories within the organization until you have reenabled trusted access.
- **Service principal name for AWS Directory Service:** `ds.amazonaws.com`.

AWS Firewall Manager and AWS Organizations

AWS Firewall Manager is a security management service that centrally configures and manages firewall rules for web applications across your accounts and applications. Using AWS Firewall Manager, you can roll out AWS WAF rules all at once for your Application Load Balancers and Amazon CloudFront distributions across all of the accounts in your AWS organization. Use AWS Firewall Manager to set up your firewall rules 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 Developer Guide.

The following list provides information that is useful to know when you want to integrate AWS Firewall Manager and AWS Organizations:

- **To enable trusted access with AWS Organizations:** You must sign in with your AWS Organizations master account to configure an account within the organization as the AWS Firewall Manager administrator account. For information, see Step 2: Set the AWS Firewall Manager Administrator Account in the *AWS Firewall Manager Developer Guide*.
- **To disable trusted access with AWS Organizations:** 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 master account and set a new administrator account for AWS Firewall Manager.
- **Service principal name for AWS Firewall Manager:** `fms.amazonaws.com`.
- **Name of the IAM service-linked role that can be created in accounts** when trusted access is enabled: `AWSServiceRoleForFMS`.

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 AWS License Manager with AWS Organizations, you can enable cross-account discovery of computing resources throughout your organization. For more information about AWS License Manager, see the AWS License Manager Guide.

The following list provides information that is useful to know when you want to integrate AWS License Manager and AWS Organizations:

- **To enable trusted access with AWS Organizations:** You must sign in with your AWS Organizations master account to associate it with your AWS License Manager account and then configure your License Manager settings. For information, see Configuring AWS License Manager Guide Settings.
- **Service principal names for AWS License Manager:** `license-manager.amazonaws.com` and `license-manager.member-account.amazonaws.com`.
- **Names of the IAM service-linked roles that can be created in accounts** when trusted access is enabled: `AWSLicenseManagerMasterAccountRole`, `AWSLicenseManagerMemberAccountRole`, and `AWSServiceRoleForAWSLicenseManagerRole`.

  For more information, see Using the License Manager–Master Account Role and Using the License Manager–Member Account Role.

AWS RAM 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*.

The following list provides information that you need when integrating AWS RAM with AWS Organizations:

- **To enable trusted access with AWS Organizations:** From the AWS RAM CLI, use the `enable-sharing-with-aws-organizations` command. For more information, see Sharing Your Resources in the *AWS RAM User Guide*.
- **Service principal name for AWS RAM:** `ram.amazonaws.com`.

- **Name of the IAM service-linked role that can be created in accounts** when trusted access is enabled: `AWSResourceAccessManagerServiceRolePolicy`.

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*.

The following list provides information that is useful to know when you want to integrate AWS Service Catalog and AWS Organizations:

- **To enable trusted access with AWS Organizations:** Call the AWSServiceCatalog::EnableAWSOrganizationsAccess action or perform the action from the AWS Service Catalog console's **Portfolio Sharing** page. For more information, see Portfolio Sharing in the *AWS Service Catalog Administrator Guide*.
- **To disable trusted access with AWS Organizations:** Call the AWSServiceCatalog::DisableAWSOrganizationsAccess action or perform the action from the AWS Service Catalog console's **Portfolio Sharing** page. 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.
- **Service principal name for AWS Service Catalog:** `servicecatalog.amazonaws.com`.

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*.

The following list provides information that is useful to know when you want to associate Service Quotas and AWS Organizations:

- **To enable trusted access with AWS Organizations:** Sign in with your AWS Organizations master 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*.

  You can also call the AssociateServiceQuotaTemplate operation. For more information, see the *Service Quotas API Reference*.
- **To disable trusted access with AWS Organizations:** Call the DisableAWSServiceAccess operation.
- **Service principal name for Service Quotas:** `servicequotas.amazonaws.com`.
- **Name of the IAM service-linked role that can be created in accounts** when trusted access is enabled: `AWSServiceRoleForServiceQuotas`.

AWS Single Sign-On and AWS Organizations

AWS Single Sign-On (AWS SSO) provides single sign-on services 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 user portal using their existing Active Directory user names and passwords. From the portal, users have access to all the AWS accounts and cloud applications that you provide in the portal. For more information about AWS SSO, see the AWS Single Sign-On User Guide.

The following list provides information that is useful to know when you want to integrate AWS SSO and AWS Organizations:

- **To enable trusted access with AWS Organizations:** AWS SSO requires trusted access with AWS Organizations to function. Trusted access is enabled when you set up AWS SSO. For more information, see Getting Started - Step 1: Enable AWS Single Sign-On in the *AWS Single Sign-On User Guide*.
- **To disable trusted access with AWS Organizations:** AWS SSO requires trusted access with AWS Organizations to operate. If you disable trusted access using AWS Organizations while you are using AWS SSO, it stops functioning because it can't access the organization. Users can't use AWS SSO to access accounts. Any roles that AWS SSO creates remain, but the AWS SSO service can't access them. The AWS SSO service-linked roles remain. If you reenable trusted access, AWS SSO continues to operate as before, without the need for you to reconfigure the service.

  If you remove an account from your organization, AWS SSO 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 AWS SSO.
- **Service principal name for AWS SSO:** `sso.amazonaws.com`.
- **Name of the IAM service-linked role that can be created in accounts** when trusted access is enabled: `AWSServiceRoleForSSO`.

  For more information, see Using Service-Linked Roles for AWS SSO in the *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. One of these capabilities, 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.

To integrate Systems Manager and Organizations, all features (p. 28) must be enabled for your organization.

The following list provides information that is useful to know when you want to integrate Systems Manager and Organizations:

- **To enable trusted access with AWS Organizations** – You must sign in with your AWS Organizations master 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*.
- **To disable trusted access with AWS Organizations** – 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. To reenable trusted access, you must create a new Resource Data Sync for Systems Manager Explorer.
- **Service principal name for Systems Manager** – `ssm.amazonaws.com`.
- **Role name created to synchronize with Systems Manager Explorer** – `AWSServiceRoleForAmazonSSM_AccountDiscovery`.

Tag Policies and AWS Organizations

Tag policies are a type of policy that can help you standardize tags across resources in your organization's accounts. For more information about tag policies, see Tag Policies (p. 86).

The following list provides information that is useful to know when you want to enable access for tag policies:

- **To enable trusted access:** Call the Organizations::EnableAWSServiceAccess action or perform the action from the Organizations console's **Settings** page.

  To disable trusted access with AWS Organizations: Call the AWSOrganizations::DisableAWSServiceAccess action or perform the action from the Organizations console's **Settings** page.

  Service principal name for tag policies: `tagpolicies.tag.amazonaws.com`.

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 in 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 master 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 master 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 master 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 master 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 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. 60) to an account.

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 master 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 master account of an organization, you can perform administrative tasks or delegate administrator permissions to other IAM users or roles in the master 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. 31) in the `Condition` element.

  Note
  This global condition also applies to the master 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:ServicePrincipal` – Available as a condition if you use the EnableAWSServiceAccess or DisableAWSServiceAccess operations to enable or disable trusted access (p. 127) 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": {
                "ForAllValues:StringLike": {
                    "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 master 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 master account to create an OU, your master account is the owner of the resource. (In AWS Organizations, the resource is the OU.)
- If you create an IAM user in your master account and grant permissions to create an OU to that user, the user can create an OU. However, the master account, to which the user belongs, owns the OU resource.
- If you create an IAM role in your master account with permissions to create an OU, anyone who can assume the role can create an OU. The master 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 Policies (IAM Policies)

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

- **Attach a permissions policy to a user or a group in your account** – To grant a user permissions to create an AWS Organizations resource, such as a service control policy (SCP) (p. 60) 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 master 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 master account can create a role to grant cross-account permissions to a user in a member account as follows:
  1. The master account administrator creates an IAM role and attaches a permissions policy to the role that grants permissions to the organization's resources.
  2. The master 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 master 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 is an example policy 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": "*"
      }
   ]
}
```

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*.

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: CreateAccount

The following example shows a CloudTrail log entry for a sample `CreateAccount` call that is generated when the API is called.

```
{
    "eventVersion": "1.06",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAMVNPBQA3EXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/diego",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "diego"
    },
    "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 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/55.0.2883.95 Safari/537.36",
    "requestParameters": {
        "email": "anaya@amazon.com",
        "accountName": "****"
    },
    "responseElements": {
        "createAccountStatus": {
            "accountName": "****",
            "state": "IN_PROGRESS",
            "id": "car-examplecreateaccountrequestid111",
            "requestedTimestamp": "Jun 21, 2018 10:06:27 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 it successfully completed.

```
{
  "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": "192.0.2.0",
  "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": "SUCCEEDED",
      "accountName": "****",
      "accountId": "444455556666",
      "requestedTimestamp": Jun 21, 2018 10:06:27 PM,
      "completedTimestamp": Jun 21, 2018 10:07:15 PM
    }
  }
}
```

The following example shows a CloudTrail log entry that is generated after a `CreateAccount` call failed.

```

  {
  "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": "1.06",
    "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-examplerootid111"
    },
    "responseElements": {
        "organizationalUnit": {
            "arn": "arn:aws:organizations::111111111111:ou/o-exampleorgid/ou-
exampleroottid111-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": "1.06",
    "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-exampleorgid/invite/h-
examplehandshakeid111",
            "id": "h-examplehandshakeid111",
            "parties": [
                {
                    "type": "ORGANIZATION",
                    "id": "o-exampleorgid"
                },
                {
                    "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": "Master account for organization"
                        },
                        {
                            "type": "ORGANIZATION_FEATURE_SET",
                            "value": "ALL"
                        }
                    ],
                    "type": "ORGANIZATION",
                    "value": "o-exampleorgid"
                },
                {
                    "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. 20).

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 Organizations as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others.

For a list of AWS services in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using Organizations is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- Architecting for HIPAA Security and Compliance Whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Config – This AWS 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.

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) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

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
- They must not exceed 250 characters in length

Maximum and Minimum Values

The following are the default maximums for entities in AWS Organizations.

Number of AWS accounts in an organization	4 is the default maximum number of accounts allowed in an organization. *If you need to increase your quota, contact AWS Support. In the upper-right corner of the console, choose **Support** and then **Support Center**. On the **Support Center** page, choose **Create case**.* An invitation sent to an account counts against this quota. The count is returned if the invited account declines, the master account cancels the invitation, or the invitation expires.
Number of roots in an organization	1.
Number of OUs in an organization	1,000.
Number of policies in an organization	1,000.
Maximum size of a service control policy (SCP) (p. 60) document	5,120 bytes. This includes all characters, including white space (such as spaces and line breaks). To reduce the size of your SCP if you approach the quota, you can remove all white space characters that are outside quotation marks.
Maximum size of a tag policy (p. 86) document	2,500 characters. This includes white space (such as spaces and line breaks). To reduce the size of your tag policy if you approach the quota, you can remove all white space characters that are outside quotation marks.

OU maximum nesting in a root	Five levels of OUs deep under a root.
Number of open invitations you can add in a 24-hour period	20 — Accepted invitations don't count against this quota. As soon as one invitation is accepted, you can send another invitation that same 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 entities that you can attach a policy to	Unlimited.
Number of tags that you can attach to an 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 maximum depends 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.

Policy Type	Policies per Root	Policies per OU	Policies per Account
Service control policy	5	5	5

Note
Currently, you can have only one root in an organization.

The minimum depends on the policy type. The following table shows each policy type and the minimum number of entities that you can attach each type to.

Policy Type	Minimum Allowed Attached to an Entity
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.

AWS Managed Policies Available for Use with AWS Organizations

This section identifies the AWS-managed policies provided for your use to manage your administration. 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 Service Control Policies

Service control policies (SCPs) (p. 60) 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 master account access to member accounts.	arn:aws:iam::aws:policy/AWSFullAccess

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 master 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 and Cost Management 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. 50).

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 master account cancels the invitation, or the invitation expires.

- Before you close or delete an AWS account, remove it from your organization (p. 48) so that it doesn't continue to count against your quota.
- Contact AWS Support 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 later. If you experience issues with account quotas while adding and removing accounts, contact AWS Support 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 used an incorrect email address when I created a member account

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. In this case, try accessing or creating a master account access role for the account. For more information, see Accessing a Member Account That Has a Master Account Access Role (p. 47). If you can't access or create the role, see the Contact Us page, and choose the item regarding billing to contact AWS Support.

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 Developer 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. 72).

Common policy errors

- More than one policy object (p. 158)
- More than one Statement element (p. 158)
- Policy document exceeds maximum size (p. 159)

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**

update-history-change	update-history-description	update-history-date
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 master account before you can invite existing accounts to your organization.	September 20, 2018
CreateAccount notifications	`CreateAccount` notifications are published to the master 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. AWS SSO 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 Single Sign-On	AWS Organizations now supports integration with AWS Single Sign-On (AWS SSO).	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. 162)	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*.