User Guide

AWS Organizations

AWS Organizations: User Guide

Table of Contents

What is AWS Organizations? 1
Features 2
Use cases 3
Terminology and concepts 4
Available feature sets 5
Organization structure 6
Invitations and handshakes 9
Organization policies 10
Quotas and service limits 12
Naming guidelines 12
Considerations 12
Maximum and minimum values 12
Expiration times for handshakes 16
Number of policies that you can attach to an entity 17
Throttling limits 18
Region support 22
List of available Regions 23
Billing and pricing 27
Payment responsibility 27
Payment structure 28
Support and feedback 28
Other AWS resources 28
Best practices 29
Account and credentials 29
Enable root access management to simplify managing root user credentials for member accounts 29
Keep the contact phone number updated 30
Use a group email address for root accounts 30
Organization structure and workloads 31
Manage your accounts within a single organization 31
Group workloads based on business purpose and not reporting structure 31
Use multiple accounts to organize your workloads 31
Service and cost management 31

Enable AWS services at the organizational level using the service console or API/CLI operations ... 31
Use billing tools to track costs and optimize resource usage ... 32
Plan the tagging strategy and enforcement of tags across your organization resources ... 32
Getting started ... 33
Signing up for AWS ... 33
Sign up for an AWS account ... 33
Create a user with administrative access ... 34
Accessing AWS Organizations ... 35
Tutorial: Creating and configuring an organization ... 37
Prerequisites ... 38
Step 1: Create your organization ... 38
Step 2: Create the organizational units ... 41
Step 3: Create the service control policies ... 44
Step 4: Testing your organization's policies ... 48
Tutorial: Monitor an organization with Amazon EventBridge ... 49
Prerequisites ... 50
Step 1: Configure a trail and event selector ... 51
Step 2: Configure a Lambda function ... 52
Step 3: Create an Amazon SNS topic that sends emails to subscribers ... 53
Step 4: Create an Amazon EventBridge rule ... 53
Step 5: Test your Amazon EventBridge rule ... 54
Clean up: Remove the resources you no longer need ... 56
Working with AWS SDKs ... 56
Managing an entire organization ... 58
Creating an organization ... 58
Create an organization ... 58
Verifying your email address ... 63
Verify your email address ... 63
Resending the verification email ... 63
Changing your email address ... 64
Enabling all features ... 64
Considerations ... 65
Standard migration process ... 66
Assisted migration process ... 75
Viewing details of an organization ... 77

Deleting an organization ... 78
Considerations ... 79
Delete an organization ... 80
Managing accounts in an organization ... 83
Management account ... 83
Best practices for the management account ... 84
Closing a management account ... 85
Member accounts ... 87
Best practices for member accounts ... 87
Creating a member account ... 90
Accessing member accounts ... 95
Closing a member account ... 102
Protecting member accounts from closure ... 104
Removing a member account ... 106
Leaving an organization from a member account ... 111
Updating the account name for a member account ... 115
Updating the root user email for a member account ... 115
Account invitations ... 116
Considerations ... 117
Sending invitations ... 119
Managing pending invitations ... 122
Accepting or declining invitations ... 127
Migrate an account ... 131
Pre-migration ... 132
Migration ... 135
Post-migration ... 136
View details of an account ... 136
Export account details ... 138
Export a list of all AWS accounts in your organization ... 138
Update alternate contacts for an account ... 140
Update primary contact for an account ... 140
Update AWS Regions for an account ... 140
Organizational units (OUs) ... 141
Best practices for OUs ... 142
Understanding AWS Organizations ... 143
Recommended foundational OUs ... 143

Recommended additional OUs ... 144
Conclusion ... 146
Navigating the root and tree ... 147
Viewing details of an OU ... 148
Creating an OU ... 150
Renaming an OU ... 154
Tagging an OU ... 155
Moving accounts between OUs ... 157
Viewing details of the root ... 158
Deleting an OU ... 160
Organization policies ... 163
Policy types ... 163
Authorization policies ... 164
Management policies ... 164
Authorization policies ... 166
Differences between SCPs and RCPs ... 166
Using SCPs and RCPs ... 167
Service control policies ... 169
Resource control policies ... 221
Management policies ... 237
Prerequisites and permissions ... 237
Understanding policy inheritance ... 239
Viewing effective policies ... 255
Declarative policies ... 258
Backup policies ... 277
Tag policies ... 317
Chat applications policies ... 358
AI services opt-out policies ... 372
Delegated administrator for AWS Organizations ... 381
Create a resource-based delegation policy ... 382
Update a resource-based delegation policy ... 387
View a resource-based delegation policy ... 391
Delete a resource-based delegation policy ... 392
Enabling a policy type ... 394
Disabling a policy type ... 395
Considerations ... 395

Disable a policy type ... 396
Creating policies ... 397
Create a service control policy (SCP) ... 397
Create a resource control policy (RCP) ... 403
Create a declarative policy ... 407
Create a backup policy ... 409
Create a tag policy ... 414
Create a chat applications policy ... 418
Create an AI services opt-out policy ... 422
Updating policies ... 424
Update a service control policy (SCP) ... 425
Update a resource control policy (RCP) ... 428
Update a declarative policy ... 430
Update a backup policy ... 432
Update a tag policy ... 436
Update a chat applications policy ... 439
Update an AI services opt-out policy ... 440
Editing tags attached to policies ... 443
Edit tags attached to a service control policy (SCP) ... 444
Edit tags attached to a resource control policy (RCP) ... 445
Edit tags attached to an declarative policy ... 446
Edit tags attached to a backup policy ... 448
Edit tags attached to a tag policy ... 449
Edit tags attached to a chat applications policy ... 450
Edit tags attached to an AI services opt-out policy ... 452
Attaching policies ... 453
Attach policies ... 453
Detaching policies ... 464
Detach policies ... 464
Getting policy details ... 476
Listing all policies ... 476
Listing attached policies ... 481
Listing all attachments ... 482
Getting details about a policy ... 484
Deleting policies ... 486
Delete policies ... 487

Tagging resources 494
Considerations 494
Using tags 495
Adding, updating, and removing tags 495
Adding tags to a resource when you create it 496
Adding or updating tags for an existing resource 496
Using other AWS services 499
Permissions required to enable trusted access 500
Permissions required to disable trusted access 501
How to enable or disable trusted access 502
AWS Organizations and service-linked roles 504
Using the AWSServiceRoleForDeclarativePoliciesEC2Report service-linked role 506
Services that work with Organizations 506
AWS Account Management 562
AWS Application Migration Service 566
AWS Artifact 571
AWS Audit Manager 574
AWS Backup 578
AWS Billing and Cost Management 581
AWS CloudFormation StackSets 583
AWS CloudTrail 587
Amazon CloudWatch 591
AWS Compute Optimizer 596
AWS Config 601
AWS Cost Optimization Hub 604
AWS Control Tower 607
Amazon Detective 610
Amazon DevOps Guru 614
AWS Directory Service 618
Amazon Elastic Compute Cloud 620
AWS Firewall Manager 623
Amazon GuardDuty 628
AWS Health 630
AWS Identity and Access Management 634
Amazon Inspector 637
AWS License Manager 641

AWS Managed Services (AMS) Self-Service Reporting (SSR) ... 644
Amazon Macie ... 646
AWS Marketplace ... 649
AWS Marketplace Private Marketplace ... 652
AWS Marketplace procurement insights dashboard ... 656
AWS Network Manager ... 660
Amazon Q Developer ... 663
AWS Resource Access Manager ... 664
AWS Resource Explorer ... 668
AWS Security Hub ... 672
Amazon S3 Storage Lens ... 675
AWS Security Incident Response ... 679
Amazon Security Lake ... 684
AWS Service Catalog ... 688
Service Quotas ... 692
AWS IAM Identity Center ... 694
AWS Systems Manager ... 698
AWS User Notifications ... 703
Tag policies ... 705
AWS Trusted Advisor ... 707
AWS Well-Architected Tool ... 710
Amazon VPC IP Address Manager (IPAM) ... 714
Amazon VPC Reachability Analyzer ... 717
Delegated administrator for integrated AWS services ... 721
Permissions granted to delegated administrator accounts ... 722
Security ... 724
AWS PrivateLink ... 724
Limitations and restrictions of AWS PrivateLink for AWS Organizations ... 725
Creating a VPC endpoint ... 725
Creating a VPC endpoint policy ... 726
Identity and Access Management ... 726
Audience ... 727
Authenticating with identities ... 728
Managing access using policies ... 731
How AWS Organizations works with IAM ... 734
Managing access permissions for an organization ... 740

Identity-based policy examples ... 749
Resource-based policy examples ... 755
AWS managed policies ... 764
Attribute-based access control with tags ... 769
Troubleshooting ... 773
Logging and monitoring ... 775
AWS CloudTrail ... 776
Amazon EventBridge ... 786
Compliance validation ... 786
Resilience ... 787
Infrastructure security ... 788
Troubleshooting ... 789
Troubleshooting general issues ... 789
I get an "access denied" message when I make a request to AWS Organizations ... 789
I get an "access denied" message when I make a request with temporary security credentials ... 790
I get an "access denied" message when I try to leave an organization as a member account or remove a member account as the management account ... 790
I get a "quota exceeded" message when I try to add an account to my organization ... 791
I get a "this operation requires a wait period" message while adding or removing accounts ... 791
I get an "organization is still initializing" message when I try to add an account to my organization ... 791
I get an "Invitations are disabled" message when I try to invite an account to my organization. ... 791
Changes that I make aren't always immediately visible ... 792
Making HTTP Query requests ... 793
Endpoints ... 793
HTTPS required ... 794
Signing AWS Organizations API requests ... 794
Code examples ... 795
Basics ... 795
Actions ... 796
Document history ... 833

What is AWS Organizations?

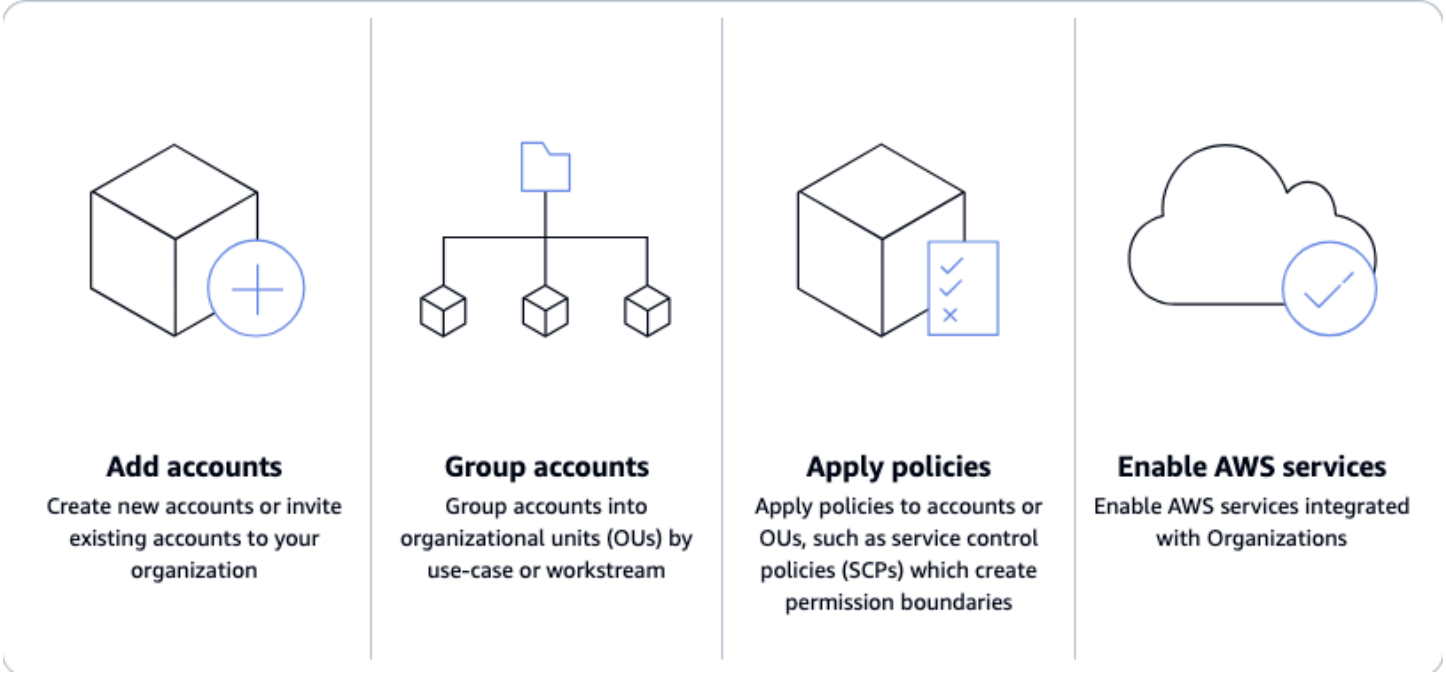
Centrally manage your environment as you scale your AWS resources

AWS Organizations helps you centrally manage and govern your environment as you grow and scale your AWS resources. Using Organizations, you can create accounts and allocate resources, group accounts to organize your workflows, apply policies for governance, and simplify billing by using a single payment method for all of your accounts.

Organizations is integrated with other AWS services so you can define central configurations, security mechanisms, audit requirements, and resource sharing across accounts in your organization. For more information, see Using AWS Organizations with other AWS services.

The following diagram shows a high-level explanation of how you can use AWS Organizations:

- Add accounts
- Group accounts
- Apply policies
- Enable AWS services.



Topics

- Features for AWS Organizations
- Use cases for AWS Organizations
- Terminology and concepts for AWS Organizations
- Quotas and service limits for AWS Organizations
- Region support for AWS Organizations
- Billing and pricing for AWS Organizations
- Support and feedback for AWS Organizations

Features for AWS Organizations

AWS Organizations offers the following features:

Manage your AWS accounts

AWS accounts are natural boundaries for permission, security, costs, and workloads. Using a multi-account environment is a recommended best-practice when scaling your cloud environment. You can simplify account creation by programmatically creating new accounts using the AWS Command Line Interface (AWS CLI), SDKs, or APIs, and centrally provision recommended resources and permissions to those accounts with AWS CloudFormation StackSets.

Define and manage your organization

As you create new accounts, you can group them into organizational units (OUs), or groups of accounts that serve a single application or service. Apply tag polices to classify or track resources in your organization, and provide attribute-based access control for users or applications. In addition, you can delegate responsibility for supported AWS services to accounts so users can manage them on behalf of your organization.

Secure and monitor your accounts

You can centrally provide tools and access for your security team to manage security needs on behalf of the organization. For example, you can provide read-only security access across accounts, detect and mitigate threats with Amazon GuardDuty, review unintended access to resources with IAM Access Analyzer, and secure sensitive data with Amazon Macie.

Control access and permissions

Set up AWS IAM Identity Center to provide access to AWS accounts and resources using your active directory, and customize permissions based on separate job roles. You can also

apply organization policies to users, accounts, or OUs. For example, service control policies (SCPs) enable you to to control access to AWS resources, services, and Regions within your organization. Resource control policies (RCPs) enable you to centrally prevent the unintended use of your AWS resources. Chat applications policies enable you to control access to your organization's accounts from chat applications such as Slack and Microsoft Teams.

Share resources across accounts

You can share AWS resources within your organization using AWS Resource Access Manager (AWS RAM). For example, you can create your Amazon Virtual Private Cloud (Amazon VPC) subnets once and share them across your organization. You can also centrally agree to software licenses with AWS License Manager, and share a catalog of IT services and custom products across accounts with AWS Service Catalog.

Audit your environment for compliance

You can activate AWS CloudTrail across accounts, which creates a log of all activity in your cloud environment that cannot be turned off or modified by member accounts. In addition, you can set policies to enforce backups on your specified cadence with AWS Backup, or define recommended configuration settings for resources across accounts and AWS Regions with AWS Config.

Centrally manage billing and costs

Organizations provides you with a single consolidated bill. In addition, you can view usage from resources across accounts and track costs using AWS Cost Explorer, and optimize your usage of compute resources using AWS Compute Optimizer.

Use cases for AWS Organizations

The following are some use cases for AWS Organizations:

Automate the creation of AWS accounts and categorize workloads

You can automate the creation of AWS accounts to quickly launch new workloads. Add the accounts to user-defined groups for instant security policy application, touchless infrastructure deployments, and auditing. Create separate groups to categorize development and production accounts and use AWS CloudFormation StackSets to provision services and permissions to each group.

Define and enforce audit and compliance policies

You can apply service control policies (SCPs) to ensure that your users perform only the actions that meet your security and compliance requirements. Create a central log of all actions performed across your organization using AWS CloudTrail. View and enforce standard resource configurations across accounts and AWS Regions using AWS Config. Automatically apply regular backups using AWS Backup. Use AWS Control Tower to apply pre-packaged governance rules for security, operations, and compliance for your AWS workloads.

Provide tools and access for your Security teams while encouraging development

Create a Security group and provide it with read-only access to all of your resources to identify and mitigate security concerns. You can allow that group to manage Amazon GuardDuty so they can actively monitor and mitigate threats to your workloads, and IAM Access Analyzer to quickly identify unintended access to your resources.

Share common resources across accounts

Organizations makes it easy for you to share critical central resources across your accounts. For example, you can share your central AWS Directory Service for Microsoft Active Directory so that applications can access your central identity store.

Share critical central resources across your accounts

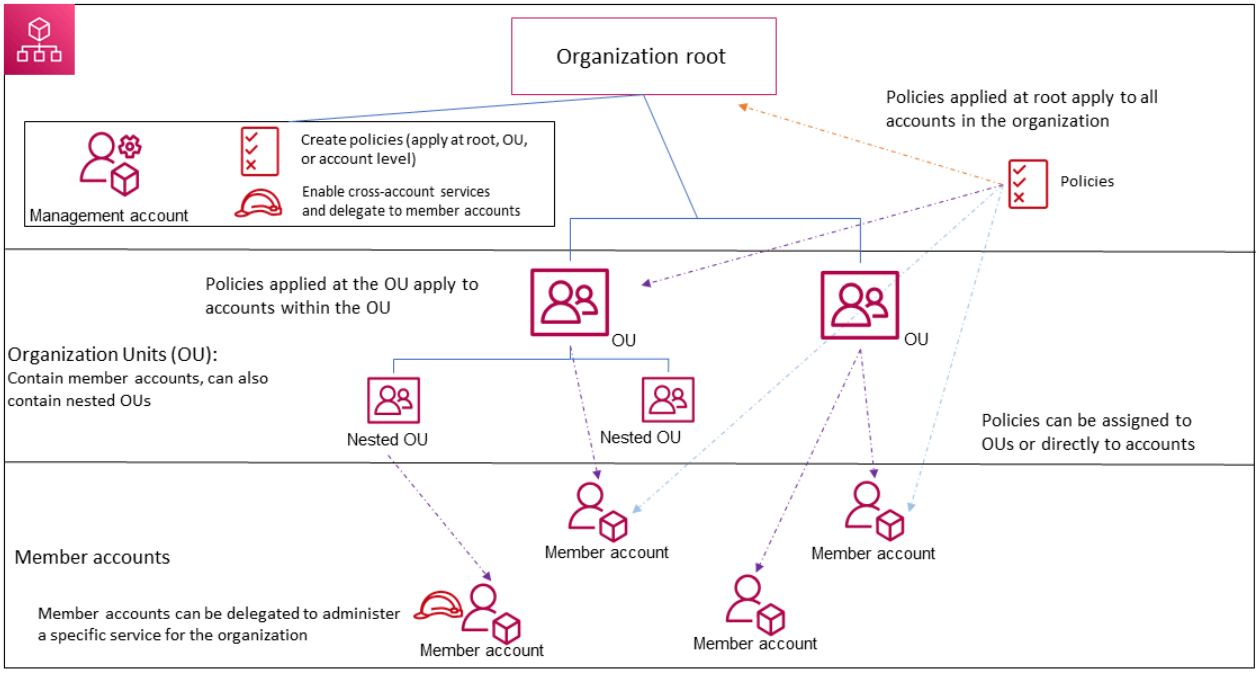
Share your AWS Directory Service for Microsoft Active Directory as a central identity store for your applications. Use AWS Service Catalog to share IT services in designated accounts so users can quickly discover and deploy approved services. Ensure that application resources are created on your Amazon Virtual Private Cloud (Amazon VPC) subnets by centrally defining them once and sharing them across your organization using AWS Resource Access Manager (AWS RAM).

Terminology and concepts for AWS Organizations

This topic explains some of the key concepts for AWS Organizations.

The following diagram shows an organization that consists of five accounts that are organized into four organizational units (OUs) under the root. The organization also has several policies that are attached to some of the OUs or directly to accounts.

For a description of each of these items, refer to the definitions in this topic.



Topics

- Available feature sets
- Organization structure
- Invitations and handshakes
- Organization policies

Available feature sets

All features (Recommended)

All features is the default feature set that is available to AWS Organizations. You can set central policies and configuration requirements for an entire organization, create custom permissions or capabilities within the organization, manage and organize your accounts under a single bill, and delegate responsibilities to other accounts on behalf of the organization. You can also use integrations with other AWS services to define central configurations, security mechanisms, audit requirements, and resource sharing across all member accounts in your organization. For more information, see Using AWS Organizations with other AWS services.

All features mode provides all the capabilities of consolidated billing along with the administrative capabilities.

Consolidated billing

Consolidated billing is the feature set that provide shared billing functionality, but doesn't include the more advanced features of AWS Organizations. For example, you can't enable other AWS services to integrate with your organization to work across all of its accounts, or use policies to restrict what users and roles in different accounts can do.

You can enable all features for an organization that originally supported only the consolidated billing features. To enable all features, all invited member accounts must approve the change by accepting the invitation that is sent when the management account starts the process. For more information, see Enabling all features for an organization with AWS Organizations.

Organization structure

Organization

An *organization* is a collection of AWS accounts that you can manage centrally and organize into a hierarchical, tree-like structure with a root at the top and organizational units nested under the root. Each account can be directly in the root, or placed in one of the OUs in the hierarchy.

Each organization consists of:

- A management account
- Zero or more member accounts
- Zero or more organizational units (OUs)
- Zero or more policies.

An organization has the functionality that is determined by the feature set that you enable.

Root

An *administrative root (root)* is contained in the management account and is the starting point for organizing your AWS accounts. The root is the top-most container in your organization's hierarchy. Under this root, you can create organizational units (OUs) to logically group your accounts and organize these OUs into a hierarchy that best matches your needs.

If you apply a management policy to the root, it applies to all organizational units (OUs) and accounts, including the management account for the organization.

If you apply an authorization policy (for example, a service control policy (SCP)), to the root, it applies to all organizational units (OUs) and member accounts in the organization. It does not apply to the management account in the organization.

> **Note**
>
> You can have only one root. AWS Organizations automatically creates the root for you when you create an organization.

Organizational unit (OU)

An *organizational unit (OU)* is a group of AWS accounts within an organization. An OU can also contain other OUs enabling you to create a hierarchy. For example, you can group all accounts that belong to the same department into a departmental OU. Similarly, you can group all accounts running security services into a security OU.

OUs are useful when you need to apply the same controls to a subset of accounts in your organization. Nesting OUs enables smaller units of management. For example, you can create OUs for each workload, then create two nested OUs in each workload OU to divide production workloads from pre-production. These OUs inherit the policies from the parent OU in addition to any controls assigned directly to the team-level OU. Including the root and AWS accounts created in the lowest OUs, your hierarchy can be five levels deep.

AWS account

An *AWS account* is a container for your AWS resources. You create and manage your AWS resources in an AWS account, and the AWS account provides administrative capabilities for access and billing.

Using multiple AWS accounts is a best practice for scaling your environment, as it provides a billing boundary for costs, isolates resources for security, gives flexibility or individuals and teams, in addition to being adaptable for new processes.

> **Note**
>
> An AWS account is different from a user. A user is an identity that you create using AWS Identity and Access Management (IAM) and takes the form of either an IAM user with long-term credentials or an IAM role with short-term credentials. A single AWS account can, and typically does, contain many users and roles.

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

Management account

A management account is the AWS account you use to create your organization. From the management account, you can do the following:

- Create other accounts in your organization
- Invite and manage invitations for other accounts to join your organization
- Designate delegated administrator accounts
- Remove accounts from your organization
- Attach policies to entities such as roots, organizational units (OUs), or accounts within your organization
- Enable integration with supported AWS services to provide service functionality across all of the accounts in the organization.

The management account is the ultimate owner of the organization, having final control over security, infrastructure, and finance policies. This account has the role of a payer account and is responsible for paying all charges accrued by the accounts in its organization.

> **Note**
>
> You cannot change which account in your organization is the management account.

Member account

A *member account* is an AWS account, other than the management account, that is part of an organization. If you are an administrator of an organization, you can create member accounts in the organization and invite existing accounts to join the organization. You also can apply policies to member accounts.

> **Note**
>
> A member account can belong to only one organization at a time. You can designate member accounts to be delegated administrator accounts.

Delegated administrator

We recommend that you use the management account and its users and roles only for tasks that must be performed by that account. We recommend that you store your AWS resources in other member accounts in the organization and keep them out of the management account. This is because security features like Organizations service control policies (SCPs) do not restrict any users or roles in the management account. Separating your resources from your management account can also help you understand the charges on your invoices. From the organization's management account, you can designate one or more member accounts as a delegated administrator account to help you implement this recommendation. There are two types of delegated administrators:

- Delegated administrator for Organizations: From these accounts, you can manage organization policies and attach policies to entities (roots, OUs, or accounts) within the organization. The management account can control delegation permissions at granular levels. For more information, see Delegated administrator for AWS Organizations.
- Delegated administrator for an AWS service: From these accounts, you can manage AWS services that integrate with Organizations. The management account can register different member accounts as delegated administrators for different services as needed. These accounts have administrative permissions for a specific service, as well as permissions for Organizations read-only actions. For more information, see Delegated administrator for AWS services that work with Organizations

Invitations and handshakes

Invitation

An *invitation* is a request made by the management account of an organization to another account. For example, the process of asking a standalone account to join an organization is an invitation.

Invitations are implemented as handshakes. 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 *handshake* is the secure exchange of information between two AWS accounts: a sender and a recipient.

The following handshakes are supported:

- **INVITE**: Handshake sent to a standalone account for it to join the sender's organization.
- **ENABLE_ALL_FEATURES**: Handshake sent to invited member accounts to enable all features for the organization.
- **APPROVE_ALL_FEATURES**: Handshake sent to the management account when all invited member accounts have approved to enable all features.

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.

Organization policies

A *policy* is a "document" with one or more statements that define the controls that you want to apply to a group of AWS accounts. AWS Organizations supports authorization policies and management policies.

Authorization policies

Authorization policies help you to centrally manage the security of AWS accounts across an organization.

Service control policy (SCP)

A *service control policy* is a type of policy that offers central control over the maximum available permissions for IAM users and IAM roles in an organization.

This means that SCPs specify principal-centric controls. SCPs create a permissions guardrail, or set limits on the maximum permissions available to principals in your member accounts. You use an SCP when you want to centrally enforce consistent access controls on principals in your organization.

This can include specifying which services your IAM users and IAM roles can access, which resources they can access, or the conditions under which they can make requests (for example, from specific regions or networks). For more information, see SCPs.

Resource control policy (RCP)

A *resource control policy* is a type of policy that offers central control over the maximum available permissions for resources in an organization.

This means that RCPs specify resource-centric controls. RCPs create a permissions guardrail, or set limits, on the maximum permissions available for resources in your member accounts. Use an RCP when you want to centrally enforce consistent access controls across resources in your organization.

This can include restricting access to your resources so that they can only be accessed by identities that belong to your organization, or specifying the conditions under which identities external to your organization can access your resources. For more information, see RCPs.

Management policies

Management policies help you centrally configure and manage AWS services and their features across an organization.

Declarative policy

A *declarative policy* is a type of policy that allows you to centrally declare and enforce desired configurations for a given AWS service at scale across an organization. Once attached, the configuration is always maintained when the service adds new features or APIs. for more information, see declarative policy.

Backup policy

A *backup policy* is type of policy that allows you to centrally manage and apply backup plans to the AWS resources across an organization's accounts. For more information, see backup policy.

Tag policy

A *tag policy* is type of policy that allows you to standardize the tags attached to the AWS resources in an organization's accounts. For more information, see tag policy.

Chat applications policy

A *chat applications policy* is a type of policy that allows you to control access to an organization's accounts from chat applications such as Slack and Microsoft Teams. For more information, see Chat applications policy.

AI services opt-out policy

An *AI services opt-out policy* is a type of policy that allows you to control data collection for AWS AI services for all the accounts in an organization. For more information, see AI service opt-out policy.

Quotas and service limits for AWS Organizations

This topic describes quotas and service limits for 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:

- Names must be composed of Unicode characters.
- Maximum string length for names vary by the object. For information about the actual limit for each object, see the AWS Organizations API Reference and find the API operation that creates the object, and look at the details for that operation's `Name` parameter. For example: Account name, or OU name.

Considerations

Service quota codes might change over time due to updates. This does not impact the quota values or names. To find the quota code for a specific quota, use the ListServiceQuotas operation, and look for the `QuotaCode` response in the output for the quota you want.

Maximum and minimum values

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

> **Note**
>
> Consider the following information about AWS Organizations quotas:
>
> - You can request increases for some of these values by using the Service Quotas console.
> - AWS Organizations limits apply at the organization level, unless otherwise specified. Many quotas apply only to actions performed from the AWS Organizations management account.
> - AWS Organizations is a global service that is physically hosted in the US East (N. Virginia) Region (`us-east-1`). Therefore, you must use `us-east-1` to access these quotas when using the Service Quotas console, the AWS CLI, or an AWS SDK.

Description	Limit
Default maximum number of accounts	10 — The default maximum number of accounts allowed in an organization. This quota is adjustable, and can be increased by using the Service Quotas console. **Note:** Only the Management account of an organization can submit this quota increase request. Limit increases can be granted up to 10,000 accounts based on customer qualifications and requirements. Newly created accounts and organizations might experience a quota below the default of 10 accounts. An invitation sent to an account counts against this quota. The count is returned if the invited account declines, the management account cancels the invitation, or the invitation expires. When an account is closed it does not stop counting against this quota until it is permanently closed. For more information on when an account is permanently closed, see Post-closure period in the *AWS Account Management Reference Guide*. Some services have account limits separate from the maximum number of accounts allowed in an organization. For more information, see Limits by AWS service.
Number of roots in an organization	1
Number of OUs in an organization	1000
Number of policies of each type in an organization	Service control policies: 2000 Resource control policies: 1000 Declarative policies: 1000 Backup policies: 1000 Tag policies: 1000

Description	Limit
	Chat applications policies: 1000 AI services opt-out policies: 1000
Maximum size of a policy document	Service control policies: 5120 characters Resource control policies: 5120 characters Declarative policies: 10,000 characters Backup policies: 10,000 characters Chat applications policies: 10,000 characters AI services opt-out policies: 2500 characters Tag policies: 10,000 characters **Note:** If you save the policy by using the AWS Management Console, extra white space (such as spaces and line breaks) between JSON elements and outside of quotation marks, is removed and not counted. If you save the policy using an SDK operation or the AWS CLI, then the policy is saved exactly as you provided and no automatic removal of characters occurs.
OU maximum nesting in a root	Five levels of OUs deep under a root.
Maximum number of invitation attempts you can perform in a 24-hour period	Either 20 or the maximum number of accounts allowed in your organization, whichever is greater. Accepted invitations don't count against this quota. As soon as one invitation is accepted, you can send another invitation that same day. If the maximum number of accounts allowed in your organization is less than 20, then you get an "account limit exceeded" exception if you attempt to invite more accounts than your organization can contain. However, you can cancel invitations and send new ones up to the maximum of 20 attempts in one day.

Description	Limit
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 accounts you can close within a 30-day period	10% of member accounts in an organization, with a maximum of 1000. This quota is not adjustable. • **< 100 accounts** – You can close up to 10 member accounts • **100 - 10,000 accounts** – You can close up to 10% of your member accounts • **> 10,000 accounts** – You can close up to 1000 member accounts After you reach this quota, you can close additional accounts or wait until your quota resets. For more information, see Close an AWS account in the *AWS Account Management Guide*.
Number of member accounts you can close concurrently	3 — Only three account closures can be in progress at the same time. As soon as one finishes, you can close another account.
Number of entities to which you can attach a policy	Unlimited
Number of tags that you can attach to a root, OU, or account	50
Maximum size of the resource-based delegation policy	40,000 characters

Limits by AWS service

Most AWS services support the stated maximum number of accounts that you can have in an organization. However, some services have account limits separate from the maximum number of accounts allowed in an organization.

The following tables shows services with separate account limits.

AWS service	Limit	Can be increased
AWS IAM Identity Center	3000	Yes
AWS Application Migration Service	5000	No
AWS Directory Service	250	Yes

For more information, see AWS IAM Identity Center quotas in the *IAM Identity Center User Guide* and AWS MGN service quota limits in the *Application Migration Service User Guide*.

Expiration times for handshakes

The following are the timeouts for handshakes in AWS Organizations.

Description	Limit
Invitation to join an organization	15 days
Request to enable all features in an organization	90 days
Handshake is deleted and no longer appears in lists	30 days after the handshake is completed

Number of policies that you can attach to an entity

The minimum and maximum depend on the policy type and the entity that you're attaching the policy to. The following table shows each policy type and the number of entities that you can attach each type to.

> **Note**
>
> These numbers apply to only those policies that are directly attached to an OU or an account. Policies that affect an OU or account by inheritance do ***not*** count against these limits. All policy limits are hard limits.

Policy type	Minimum attached to an entity	Maximum attached to root	Maximum attached per OU	Maximum attached per account
Service control policy	1 — Every entity must have *at least* one SCP attached at all times when you enable SCPs. You can't remove the last SCP from an entity.	5	5	5
Resource control policy	1 — The `RCPFullAWSAccess` policy is automatically attached to the root, every OU, and every account in your organization when you enable RCPs. You cannot detach this policy and it counts towards the 5 policies quota.	5	5	5

Policy type	Minimum attached to an entity	Maximum attached to root	Maximum attached per OU	Maximum attached per account
Declarative policy	0	10	10	10
Backup policy	0	10	10	10
Tag policy	0	10	10	10
Chat applications policy	0	5	5	5
AI services opt-out policy	0	5	5	5

Note

You can have only one root in an organization.

Throttling limits

The following tables lists the AWS Organizations APIs by management category, and shows their respective throttle rates at the account and organizational level.

AWS Organizations uses the token bucket algorithm to implement API throttling. With this algorithm, your account has a *bucket* that holds a specific number of *tokens*. The number of tokens in the bucket represents your throttling quota at any given second.

Rate is the fixed pace that tokens are added to the token bucket per second.

Burst is the maximum number of token that can be added and the maximum number of token that can be used per second.

For example, the `DescribeAccount` API is limited for a single AWS account to 20 requests per second as the baseline rate and to 30 requests per second as the burst rate. The burst rate of 30 requests per second allows you to temporarily exceed the baseline rate of 20 requests per second.

You can makes 20 requests in the first second, which is the baseline rate. In the next second, you can make 30 requests, exceeding the baseline but staying within the burst rate of 30. However, in the third second, if your try to make more than 20 requests, you will be throttled since you have exceeded the baseline rate and the burst capacity has been used.

The burst rate allows you to handle temporary spikes in traffic without getting throttled, as long as the average requests per second stay within the baseline limit over time.

Account management limits

The following table lists the AWS Organizations APIs for account management.

AWS Organizations API	Per account limit (rate, burst)	Per organization limit (rate, burst)
CloseAccount	.05, 1	
CreateAccount, CreateGovCloudAccount	0.1, 3	
DescribeAccount	20, 30	24, 36
DescribeCreateAccountStatus	2, 2	2, 3
LeaveOrganization	1, 1	
ListCreateAccountStatus	5, 8	6, 10

Handshake management limits

The following table lists the AWS Organizations APIs for account handshake.

AWS Organizations API	Per account limit (rate, burst)	Per organization limit (rate, burst)
AcceptHandshake	1, 2	5, 5
DescribeHandshake	1, 2	6, 10

AWS Organizations API	Per account limit (rate, burst)	Per organization limit (rate, burst)
CancelHandshake	2, 3	
DeclineHandshake	1, 1	5, 5
InviteAccountToOrganization	3, 5	
ListHandshakesForAccount, ListHandshakesForOrganization	5, 8	6, 10

Organization management limits

The following table lists the AWS Organizations APIs for organization management.

AWS Organizations API	Per account limit (rate, burst)	Per organization limit (rate, burst)
CreateOrganization , DeleteOrganization, EnableFullControl	1, 1	
CreateOrganizationalUnit, DescribeOrganization	1, 2	
MoveAccount, UpdateOrganizationalUnit, DeleteOrganizationalUnit	2, 3	
DescribeOrganizationalUnit	2, 2	2, 3
ListAccounts	8, 12	9, 15
ListChildren	6, 10	7, 12

AWS Organizations API	Per account limit (rate, burst)	Per organization limit (rate, burst)
ListParents, ListAccountsForParent, ListOrganizationalUnitsForParent	5, 8	6, 10
ListRoots	1, 2	1, 3
ListTagsForResource	10, 15	12, 18
RemoveAccountFromOrganization	2, 2	
TagResource, UntagResource	4, 6	

Policy management limits

The following table lists the AWS Organizations APIs for policy management.

AWS Organizations API	Per account limit (rate, burst)	Per organization limit (rate, burst)
CreatePolicy, DeletePolicy, AttachPolicy, DetachPolicy	2, 3	
DescribePolicy	2, 2	2, 3
DisablePolicyType, EnablePolicyType	1, 1	
ListPolicies, ListPoliciesForTarget, ListTargetsForPolicy	5, 8	6, 10
UpdatePolicy	2, 3	

Service management limits

The following table lists the AWS Organizations APIs for service management.

AWS Organizations API	Per account limit (rate, burst)	Per organization limit (rate, burst)
EnableAWSServiceAccess, DisableAWSServiceAccess	1, 2	
ListAWSServiceAccessForOrganization, ListDelegatedServicesForAccount	1, 3	1, 4
ListDelegatedAdministrators	5, 8	6, 10
RegisterDelegatedAdministrator, DeregisterDelegatedAdministrator	1, 2	

Region support for AWS Organizations

AWS Organizations is available in all AWS commercial Regions, AWS GovCloud (US) Regions, and China Regions.

For a list of functionality differences in AWS GovCloud (US) Regions, see AWS Organizations in AWS GovCloud (US).

For a list of functionality differences in China Regions, see AWS Organizations in China.

The service endpoints for Organizations are located:

- In US East (N. Virginia) for commercial organizations
- In AWS GovCloud (US-West) for AWS GovCloud (US) organizations
- In China (Ningxia) for China organizations, operated by Ningxia Western Cloud Data Technology Co. Ltd (NWCD).

All organization entities are globally accessible, except for organizations managed in China, similar to how AWS Identity and Access Management (IAM) works today. You do not need to specify an AWS Region when you create and manage your organization, but you will need to create a separate organization for accounts used in China. Users in your AWS accounts can use AWS services in any geographic Region where that service is available.

> **Note**
>
> **Tag policies are only supported in a subset of Regions**
> Tag policies are a type of policy that can help you standardize tags across resources in your organization's accounts. Tag policies are only supported in a subet of Regions where Organizations is supported. For a list of Regions where tag policies are supported, see Tag policies | Support Regions.

List of available AWS Regions

The following table lists all the available AWS Regions.

Region Name	Region	Endpoint	Protocol
US East (Ohio)	us-east-2	organizations.us-east-1.amazonaws.com	HTTPS
		organizations-fips.us-east-1.amazonaws.com	HTTPS
US East (N. Virginia)	us-east-1	organizations.us-east-1.amazonaws.com	HTTPS
		organizations-fips.us-east-1.amazonaws.com	HTTPS
US West (N. California)	us-west-1	organizations.us-east-1.amazonaws.com	HTTPS
		organizations-fips.us-east-1.amazonaws.com	HTTPS
US West (Oregon)	us-west-2	organizations.us-east-1.amazonaws.com	HTTPS
		organizations-fips.us-east-1.amazonaws.com	HTTPS

Region Name	Region	Endpoint	Protocol
Africa (Cape Town)	af-south-1	organizations.us-east-1.amazonaws.com organizations-fips.us-east-1.amazonaws.com	HTTPS HTTPS
Asia Pacific (Hong Kong)	ap-east-1	organizations.us-east-1.amazonaws.com organizations-fips.us-east-1.amazonaws.com	HTTPS HTTPS
Asia Pacific (Hyderabad)	ap-south-2	organizations.us-east-1.amazonaws.com organizations-fips.us-east-1.amazonaws.com	HTTPS HTTPS
Asia Pacific (Jakarta)	ap-southeast-3	organizations.us-east-1.amazonaws.com organizations-fips.us-east-1.amazonaws.com	HTTPS HTTPS
Asia Pacific (Malaysia)	ap-southeast-5	organizations.us-east-1.amazonaws.com organizations-fips.us-east-1.amazonaws.com	HTTPS HTTPS
Asia Pacific (Melbourne)	ap-southeast-4	organizations.us-east-1.amazonaws.com organizations-fips.us-east-1.amazonaws.com	HTTPS HTTPS
Asia Pacific (Mumbai)	ap-south-1	organizations.us-east-1.amazonaws.com organizations-fips.us-east-1.amazonaws.com	HTTPS HTTPS
Asia Pacific (Osaka)	ap-northeast-3	organizations.us-east-1.amazonaws.com organizations-fips.us-east-1.amazonaws.com	HTTPS HTTPS

Region Name	Region	Endpoint	Protocol
Asia Pacific (Seoul)	ap-northeast-2	organizations.us-east-1.amazonaws.com organizations-fips.us-east-1.amazonaws.com	HTTPS HTTPS
Asia Pacific (Singapore)	ap-southeast-1	organizations.us-east-1.amazonaws.com organizations-fips.us-east-1.amazonaws.com	HTTPS HTTPS
Asia Pacific (Sydney)	ap-southeast-2	organizations.us-east-1.amazonaws.com organizations-fips.us-east-1.amazonaws.com	HTTPS HTTPS
Asia Pacific (Thailand)	ap-southeast-7	organizations.us-east-1.amazonaws.com organizations-fips.us-east-1.amazonaws.com	HTTPS HTTPS
Asia Pacific (Tokyo)	ap-northeast-1	organizations.us-east-1.amazonaws.com organizations-fips.us-east-1.amazonaws.com	HTTPS HTTPS
Canada (Central)	ca-central-1	organizations.us-east-1.amazonaws.com organizations-fips.us-east-1.amazonaws.com	HTTPS HTTPS
Canada West (Calgary)	ca-west-1	organizations.us-east-1.amazonaws.com organizations-fips.us-east-1.amazonaws.com	HTTPS HTTPS
Europe (Frankfurt)	eu-central-1	organizations.us-east-1.amazonaws.com organizations-fips.us-east-1.amazonaws.com	HTTPS HTTPS

Region Name	Region	Endpoint	Protocol
Europe (Ireland)	eu-west-1	organizations.us-east-1.amazonaws.com	HTTPS
		organizations-fips.us-east-1.amazonaws.com	HTTPS
Europe (London)	eu-west-2	organizations.us-east-1.amazonaws.com	HTTPS
		organizations-fips.us-east-1.amazonaws.com	HTTPS
Europe (Milan)	eu-south-1	organizations.us-east-1.amazonaws.com	HTTPS
		organizations-fips.us-east-1.amazonaws.com	HTTPS
Europe (Paris)	eu-west-3	organizations.us-east-1.amazonaws.com	HTTPS
		organizations-fips.us-east-1.amazonaws.com	HTTPS
Europe (Spain)	eu-south-2	organizations.us-east-1.amazonaws.com	HTTPS
		organizations-fips.us-east-1.amazonaws.com	HTTPS
Europe (Stockholm)	eu-north-1	organizations.us-east-1.amazonaws.com	HTTPS
		organizations-fips.us-east-1.amazonaws.com	HTTPS
Europe (Zurich)	eu-central-2	organizations.us-east-1.amazonaws.com	HTTPS
		organizations-fips.us-east-1.amazonaws.com	HTTPS
Israel (Tel Aviv)	il-central-1	organizations.us-east-1.amazonaws.com	HTTPS
		organizations-fips.us-east-1.amazonaws.com	HTTPS
Mexico (Central)	mx-central-1	organizations.us-east-1.amazonaws.com	HTTPS
		organizations-fips.us-east-1.amazonaws.com	HTTPS

Region Name	Region	Endpoint	Protocol
Middle East (Bahrain)	me-south-1	organizations.us-east-1.amazonaws.com organizations-fips.us-east-1.amazonaws.com	HTTPS HTTPS
Middle East (UAE)	me-central-1	organizations.us-east-1.amazonaws.com organizations-fips.us-east-1.amazonaws.com	HTTPS HTTPS
South America (São Paulo)	sa-east-1	organizations.us-east-1.amazonaws.com organizations-fips.us-east-1.amazonaws.com	HTTPS HTTPS
AWS GovCloud (US-East)	us-gov-east-1	organizations.us-gov-west-1.amazonaws.com	HTTPS
AWS GovCloud (US-West)	us-gov-west-1	organizations.us-gov-west-1.amazonaws.com	HTTPS

Billing and pricing for AWS Organizations

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.

Who pays for usage incurred by users under an AWS member account in my organization?

The owner of the management account is responsible for paying for all usage, data, and resources used by the accounts in the organization.

Will my bill reflect the organizational unit structure that I created in my organization?

Your bill will not reflect the structure that you have defined in your organization. You can use cost allocation tags in individual AWS accounts to categorize and track your AWS costs, and this allocation will be visible in the consolidated bill for your organization.

Support and feedback for AWS Organizations

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

Other AWS resources

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

Best practices for a multi-account environment

Follow these recommendations to help walk you through setting up and managing a multi-account environment in AWS Organizations.

Topics

- Account and credentials
- Organization structure and workloads
- Service and cost management

Account and credentials

Enable root access management to simplify managing root user credentials for member accounts

We recommend you enable root access management to help you monitor and remove root user credentials for member accounts. Root access management prevents recovery of root user credentials, improving account security in your organization.

- Remove root user credentials for member accounts to prevent sign in to the root user. This also prevents member accounts from recovery of the root user.
- Assume a privileged session to perform the following tasks on member accounts:
  - Remove a misconfigured bucket policy that denies all principals from accessing an Amazon S3 bucket.
  - Delete an Amazon Simple Queue Service resource-based policy that denies all principals from accessing an Amazon SQS queue.
  - Allow a member account to recover their root user credentials. The person with access to the root user email inbox for the member account can reset the root user password and sign in as the member account root user.

After root access management is enabled, newly created member accounts are secure-by-default, having no root user credentials, which eliminates the need for additional security, such as MFA after provisioning.

For more information, see Centralize root user credentials for member accounts in the *AWS Identity and Access Management User Guide*.

Keep the contact phone number updated

To recover access to your AWS account, it is crucial to have a valid and active contact phone number that allows you to receive text messages or calls. We recommend using a dedicated phone number to make sure that AWS can contact you for account support and recovery purposes. You can easily view and manage your account phone numbers via the AWS Management Console or Account Management APIs.

There are various ways to obtain a dedicated phone number that ensures AWS can contact you. We strongly recommend that you obtain a dedicated SIM card and physical phone. Safely store the phone and the SIM long-term to guarantee the phone number remains available for account recovery. Also make sure the team responsible for the mobile bill understands the importance of this number, even if it remains inactive for extended periods. It is essential to keep this phone number confidential within your organization for additional protection.

Document the phone number in the AWS Contact Information console page, and share its details with the specific teams that must know about it in your organization. This approach helps minimize the risk associated with transferring the phone number to a different SIM. Store the phone according to your existing information security policy. However, do not store the phone in the same location as the other related credential information. Any access to the phone or its storage location should be logged and monitored. If the phone number associated with an account changes, implement processes to update the phone number in your existing documentation.

Use a group email address for root accounts

Use an email address that is managed by your business. Use an email address that forwards received messages directly to a group of users. In the event that AWS must contact the owner of the account, for example, to confirm access, the email message is distributed to multiple parties. This approach helps to reduce the risk of delays in responding, even if individuals are on vacation, out sick, or leave the business.

Organization structure and workloads

Manage your accounts within a single organization

We recommend creating a single organization and managing all your accounts within this organization. An organization is a security boundary that lets you maintain consistency across accounts in your environment. You can centrally apply policies or service-level configurations across accounts within an organization. If you want to enable consistent policies, central visibility, and programmatic controls across your multi-account environment, this is best achieved within a single organization.

Group workloads based on business purpose and not reporting structure

We recommend that you isolate production workload environments and data under your top-level workload-oriented OUs. Your OUs should be based on a common set of controls rather than mirroring your company's reporting structure. Apart from production OUs, we recommend that you define one or more non-production OUs that contain accounts and workload environments that are used to develop and test workloads. For additional guidance, see Organizing workload-oriented OUs.

Use multiple accounts to organize your workloads

An AWS account provides natural security, access, and billing boundaries for your AWS resources. There are benefits of using multiple accounts because it lets you distribute account level quotas and API request-rate limits, and additional benefits listed here. We recommend that you use a number of organization-wide foundational accounts, such as accounts for security, logging, and infrastructure. For workload accounts, you should separate production workloads from test/development workloads in separate accounts.

Service and cost management

Enable AWS services at the organizational level using the service console or API/CLI operations

As a best practice, we recommend enabling or disabling any services you'd like to integrate with across AWS Organizations using that service's console, or API operations/CLI command

equivalents. Using this method, the AWS service can perform all required initialization steps for your organization, such as creating any required resources and cleaning up resources when disabling the service. AWS Account Management is the only service that requires use of the AWS Organizations Console or APIs to enable. To review the list of services that are integrated with AWS Organizations, see AWS services that you can use with AWS Organizations.

Use billing tools to track costs and optimize resource usage

When managing an organization, you get a consolidated bill that covers all charges from accounts in your organization. For business users who need access to cost visibility, you can provide a role in the management account with restricted read-only permissions to review billing and cost tools. For example, you can create a permission set that provides access to billing reports, or use the AWS Cost Explorer Service (a tool for viewing cost trends over time), and cost-efficiency services such as Amazon S3 Storage Lens and AWS Compute Optimizer.

Plan the tagging strategy and enforcement of tags across your organization resources

As your accounts and workloads scale, tags can be a useful feature for cost tracking, access control, and resource organization. For tagging naming strategies, follow the guidance in Tagging your AWS resources. In addition to resources, you can create tags on the organization root, accounts, OUs, and policies. Refer to the Building your tagging strategy for additional information.

Getting started with AWS Organizations

The following topics provide information to help you start using AWS Organizations. You can also use the following tutorials to begin performing tasks using AWS Organizations.

Tutorial: Creating and configuring an organization

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

Tutorial: Monitor important changes to your organization with Amazon EventBridge

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

Topics

- Signing up for AWS
- Accessing AWS Organizations
- Tutorial: Creating and configuring an organization
- Tutorial: Monitor important changes to your organization with Amazon EventBridge
- Using AWS Organizations with an AWS SDK

Signing up for AWS

Topics

- Sign up for an AWS account
- Create a user with administrative access

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

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

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

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform tasks that require root user access.

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

Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

   For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.
2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create a user with administrative access

1. Enable IAM Identity Center.

For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to a user.

   For a tutorial about using the IAM Identity Center directory as your identity source, see Configure user access with the default IAM Identity Center directory in the *AWS IAM Identity Center User Guide*.

Sign in as the user with administrative access

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

  For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the *AWS Sign-In User Guide*.

Assign access to additional users

1. In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

   For instructions, see Create a permission set in the *AWS IAM Identity Center User Guide*.

2. Assign users to a group, and then assign single sign-on access to the group.

   For instructions, see Add groups in the *AWS IAM Identity Center User Guide*.

Accessing AWS Organizations

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

AWS Management Console

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

AWS Command Line Tools

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

AWS provides two sets of command line tools:

- AWS Command Line Interface

  The AWS Command Line Interface (AWS CLI) is a unified tool to manage your AWS services. With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.

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

  The Tools for Windows PowerShell let developers and administrators manage their AWS services and resources in the PowerShell scripting environment. You can manage your AWS resources with the same PowerShell tools you use to manage your Windows, Linux, and MacOS environments.

  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.

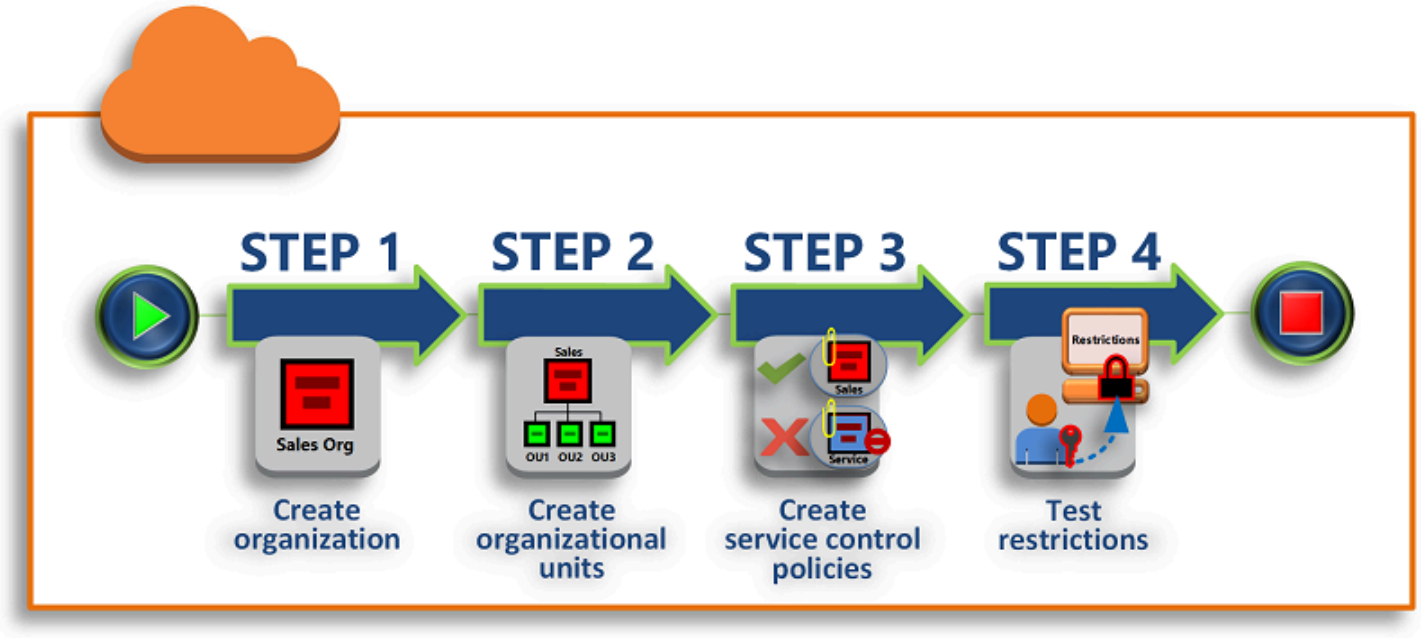
AWS Organizations HTTPS Query API

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.

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 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 to ensure that no users in a member account can change the configuration for the auditing logs that AWS CloudTrail creates.

The following illustration shows the main steps of the tutorial.



Step 1: Create your organization

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

Step 2: Create the organizational units

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

Step 3: Create the service control policies

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

Step 4: Testing your organization's policies

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

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

Prerequisites

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

The tutorial refers to the accounts as the following:

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

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

Step 1: Create your organization

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

AWS Management Console

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

   > **Note**
   >
   > By default, the organization is created with all features enabled. You can also create the organization with only consolidated billing features enabled.

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

   If the account you use has never had its email address verified by AWS, a verification email is automatically sent to the address that is associated with your management account. There might be a delay before you receive the verification email.
4. Verify your email address within 24 hours. For more information, see Email address verification with AWS Organizations.

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

Invite an existing account to join your organization

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

AWS Management Console

To invite an existing account to join

1. Navigate to the **AWS accounts** page, and choose **Add an AWS account**.
2. On the **Add an AWS account** page, choose **Invite an existing AWS account**.
3. In the box **Email address or account ID of an AWS account to invite** box, enter the email address of the owner of the account that you want to invite, similar to the following:

member222@example.com. Alternatively, if you know the AWS account ID number, then you can enter it instead.

4. Type any text that you want into the **Message to include in the invitation email message** box. This text is included in the email that is sent to the owner of the account.
5. Choose **Send invitation**. AWS Organizations sends the invitation to the account owner.

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

6. For the purposes of this tutorial, you now need to accept your own invitation. Do one of the following to get to the **Invitations** page in the console:
   - Open the email that AWS sent from the management account and choose the link to accept the invitation. When prompted to sign in, do so as an administrator in the invited member account.
   - Open the AWS Organizations console and navigate to the **Invitations** page.
7. On the **AWS accounts** page, choose **Accept** and then choose **Confirm**.

> **Tip**
>
> The invitation receipt could be delayed and you might need to wait before you can accept the invitation.

8. Sign out of your member account and sign in again as an administrator in your management account.

Create a member account

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

AWS Management Console

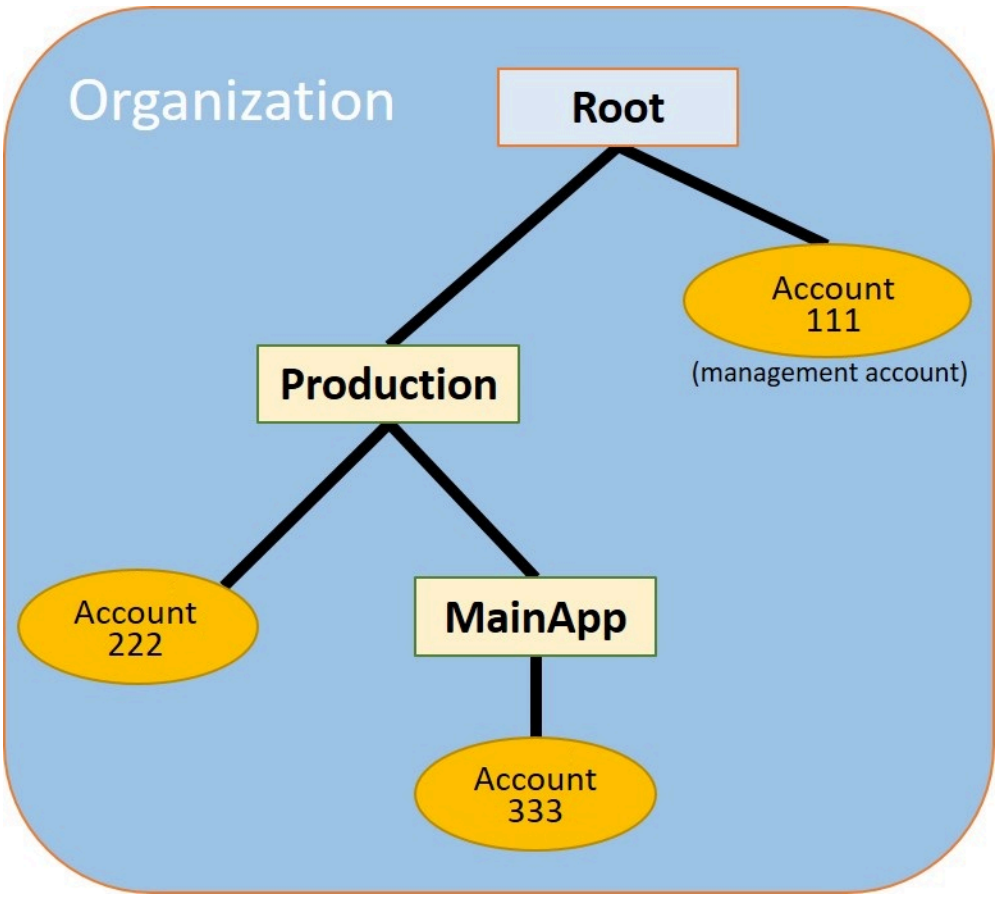
To create a member account

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

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

Step 2: Create the organizational units

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



AWS Management Console

To create and populate the OUs

> **Note**
>
> In the steps that follow, you interact with objects for which you can choose either the name of the object itself, or the radio button next to the object.
>
> - If you choose the name of the object, you open a new page that displays the objects details.
> - If you choose the radio button next to the object, you are identifying that object to be acted upon by another action, such as choosing a menu option.
>
> The steps that follow have you choose the radio button so that you can then act on the associated object by making menu choices.

1. On the AWS Organizations console navigate to the **AWS accounts** page.
2. Choose the check box

   next to the **Root** container.
3. Choose the **Actions** dropdown, and then under **Organizational unit**, choose **Create new**.
4. On the **Create organizational unit in Root** page, for the **Organizational unit name**, enter **`Production`** and then choose **Create organizational unit**.
5. Choose the check box

   next to your new **Production** OU.
6. Choose **Actions**, and then under **Organizational unit**, choose **Create new**.
7. On the **Create organizational unit in Production** page, for the name of the second OU, enter **`MainApp`** and then choose **Create organizational unit**.

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

   next to it. This displays the **MainApp** OU as a child of **Production**.
9. Next to **333333333333**, choose the check box

   (not its name), choose **Actions**, and then under **AWS account**, choose **Move**.
10. On the **Move AWS account '333333333333'** page, choose the triangle next to **Production** to expand it. Next to **MainApp**, choose the radio button

    (not its name), and then choose **Move AWS account**.
11. Next to **222222222222**, choose the check box

    (not its name), choose **Actions**, and then under **AWS account**, choose **Move**.
12. On the **Move AWS account '222222222222'** page, next to **Production**, choose the radio button (not its name), and then choose **Move AWS account**.

Step 3: Create the service control policies

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

Enable the service control policy type for the organization

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

AWS Management Console

To enable SCPs for your organization

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

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

Create your SCPs

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

AWS Management Console

To create the first SCP that blocks CloudTrail configuration actions

1. Navigate to the **Policies** page, and then choose **Service control policies**.
2. On the **Service control policies** page, choose **Create policy**.
3. For **Policy name**, enter **`Block CloudTrail Configuration Actions`**.
4. In the **Policy** section, in the list of services on the right, select CloudTrail for the service. Then choose the following actions: **AddTags**, **CreateTrail**, **DeleteTrail**, **RemoveTags**, **StartLogging**, **StopLogging**, and **UpdateTrail**.

5. Still in the right pane, choose **Add resource** and specify **CloudTrail** and **All Resources**. Then choose **Add resource**.

   The policy statement on the left should look similar to the following.

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

6. Choose **Create policy**.

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

AWS Management Console

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

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

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

4. Choose **Create policy**.

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

AWS Management Console

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

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

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

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

```
    {
      "Effect": "Deny",
      "Action": [ "dynamodb:*" ],
      "Resource": [ "*" ]
    }
  ]
}
```

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

Attach the SCPs to your OUs

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

AWS Management Console

To attach the policies to the root and the OUs

1. Navigate to the **AWS accounts** page.
2. On the **AWS accounts** page, choose **Root** (its name, not the radio button) to navigate to its details page.
3. On the **Root** details page, choose the **Policies** tab, and then under **Service Control Policies**, choose **Attach**.
4. On the **Attach a service control policy** page, choose the radio button next to the SCP named `Block CloudTrail Configuration Actions`, and then choose **Attach**. In this tutorial, you attach it to the root so that it affects all member accounts to prevent anyone from altering the way that you configured CloudTrail.

   The **Root** details page, **Policies** tab now shows that two SCPs are attached to the root: the one you just attached and the default `FullAWSAccess` SCP.
5. Navigate back to the **AWS accounts** page, and choose the **Production** OU (it's name, not the radio button) to navigate to its details page.
6. On the **Production** OU's details page, choose the **Policies** tab.
7. Under **Service Control Policies**, choose **Attach**.
8. On the **Attach a service control policy** page, choose the radio button next to `Allow List for All Approved Services`, and then choose **Attach**. This enables users or roles in member accounts in the **Production** OU to access the approved services.

9. Choose the **Policies** tab again to see that two SCPs are attached to the OU: the one that you just attached and the default `FullAWSAccess` SCP. However, because the `FullAWSAccess` SCP is also an allow list that allows all services and actions, you must now detach this SCP to ensure that only your approved services are allowed.
10. To remove the default policy from the **Production** OU, choose the radio button to **FullAWSAccess**, choose **Detach**, and then on the confirmation dialog box, choose **Detach policy**.

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

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

Step 4: Testing your organization's policies

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

- If you sign in as a user in the management account, you can perform any operation that is allowed by your IAM permissions policies. The SCPs don't affect any user or role in the management account, no matter which root or OU the account is located in.
- If you sign in as a 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

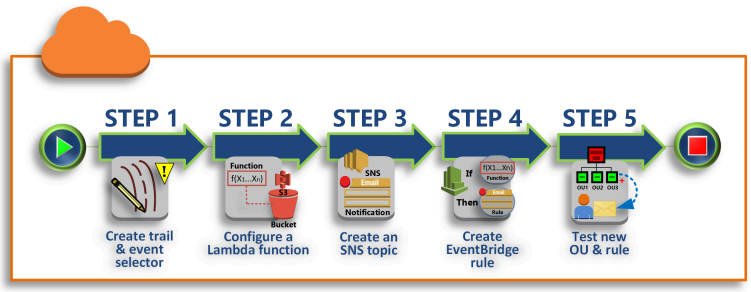
that isn't in the allow list. Also, AWS Organizations denies any attempt to perform one of the CloudTrail configuration actions.

- 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 Amazon EventBridge

This tutorial shows how to configure Amazon EventBridge, formerly Amazon 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 Amazon EventBridge 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

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

Step 2: Configure a Lambda function

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

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

Step 4: Create an Amazon EventBridge rule

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

Step 5: Test your Amazon EventBridge rule

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 Amazon EventBridge with AWS Organizations, see Logging and monitoring in AWS Organizations.

Prerequisites

This tutorial assumes the following:

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

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

Step 1: Configure a trail and event selector

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

To create a trail

1. Sign in to AWS as an administrator of the organization's management account and then open the CloudTrail console at https://console.aws.amazon.com/cloudtrail/.
2. On the navigation bar in the upper-right corner of the console, choose the **US East (N. Virginia)** Region. If you choose a different region, AWS Organizations doesn't appear as an option in the Amazon EventBridge 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 need to create a bucket, choose **Create new S3 bucket** and then, for **Trail log bucket and folder**, enter a name for the new bucket.

     > **Note**
     >
     > S3 bucket names must be ***globally*** unique.

   - If you already have a bucket, choose **Use existing S3 bucket** and then choose the bucket name from the **S3 bucket** list.
7. Choose **Next**.
8. On the **Choose log events** page, in the **Management events** section, choose **Read** and **Write**.
9. Choose **Next**.
10. Review your selections and choose **Create trail**.

Amazon EventBridge 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 Amazon EventBridge rule that you configure later.

To create a Lambda function that logs Amazon EventBridge 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 function**.
3. On the **Create function** page, choose **Use a blueprint**.
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 new role with basic Lambda permissions**. This role grants your Lambda function permissions to access the data it requires and to write its output log.
7. Edit the Lambda function code, 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 **Create function**.

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 Amazon EventBridge 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 an Amazon EventBridge rule

Now that the required Lambda function exists in your account, you create an Amazon EventBridge rule that invokes it when the criteria in the rule are met.

To create an EventBridge rule

1. Open the Amazon EventBridge console at https://console.aws.amazon.com/events/.

2. Set the console to the **US East (N. Virginia)** Region or information about Organizations is not available. On the navigation bar in the upper-right corner of the console, choose the **US East (N. Virginia)** Region.
3. For instructions on creating rules, see Rules in Amazon EventBridge in the Amazon EventBridge user guide.

Step 5: Test your Amazon EventBridge rule

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

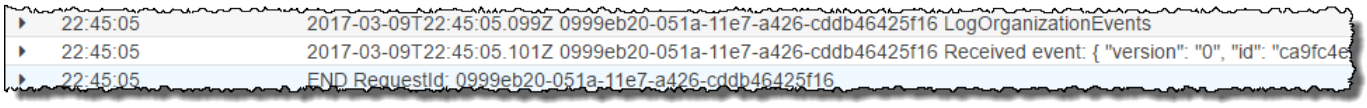
AWS Management Console

To create an OU

1. Open the AWS Organizations console to the **AWS accounts** page.
2. Choose the check box
   ☑
   Root OU, choose **Actions**, and then under **Organizational unit** choose **Create new**.
3. For the name of the OU, enter **`TestCWEOU`** and then choose **Create organizational unit**.

To see the EventBridge 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.



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

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

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

Clean up: Remove the resources you no longer need

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

To clean up your AWS environment

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

That's it. In this tutorial, you configured EventBridge 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.

Using AWS Organizations with an AWS SDK

AWS software development kits (SDKs) are available for many popular programming languages. Each SDK provides an API, code examples, and documentation that make it easier for developers to build applications in their preferred language.

SDK documentation	Code examples
AWS SDK for C++	AWS SDK for C++ code examples
AWS CLI	AWS CLI code examples

SDK documentation	Code examples
AWS SDK for Go	AWS SDK for Go code examples
AWS SDK for Java	AWS SDK for Java code examples
AWS SDK for JavaScript	AWS SDK for JavaScript code examples
AWS SDK for Kotlin	AWS SDK for Kotlin code examples
AWS SDK for .NET	AWS SDK for .NET code examples
AWS SDK for PHP	AWS SDK for PHP code examples
AWS Tools for PowerShell	Tools for PowerShell code examples
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) code examples
AWS SDK for Ruby	AWS SDK for Ruby code examples
AWS SDK for Rust	AWS SDK for Rust code examples
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP code examples
AWS SDK for Swift	AWS SDK for Swift code examples

Example availability

Can't find what you need? Request a code example by using the **Provide feedback** link at the bottom of this page.

Managing an organization with AWS Organizations

An *organization* is a collection of AWS accounts that you can manage centrally and organize into a hierarchical, tree-like structure with a root at the top and organizational units nested under the root. Each account can be directly in the root, or placed in one of the OUs in the hierarchy.

Each organization consists of:

- A management account
- Zero or more member accounts
- Zero or more organizational units (OUs)
- Zero or more policies.

An organization has the functionality that is determined by the feature set that you enable.

Topics

- Creating an organization with AWS Organizations
- Email address verification with AWS Organizations
- Resend the verification email with AWS Organizations
- Changing your email address for an organization with AWS Organizations
- Enabling all features for an organization with AWS Organizations
- Viewing details of an organization from the management account
- Deleting an organization with AWS Organizations

Creating an organization with AWS Organizations

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

Create an organization

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

Minimum permissions

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

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

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

AWS Management Console

To create an organization

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

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

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

   If required, Organizations automatically sends a verification email to the address that is associated with your management account. There might be a delay before you receive the verification email. Verify your email address within 24 hours. For more information, see Email address verification with AWS Organizations. You can create accounts to grow your

organization without verifying your management account's email address. However, to invite existing accounts, you must first complete email verification.

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

AWS CLI & AWS SDKs

The following code examples show how to use `CreateOrganization`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Creates an organization in AWS Organizations.
/// </summary>
public class CreateOrganization
{
    /// <summary>
    /// Creates an Organizations client object and then uses it to create
    /// a new organization with the default user as the administrator, and
    /// then displays information about the new organization.
    /// </summary>
    public static async Task Main()
    {
        IAmazonOrganizations client = new AmazonOrganizationsClient();
```

```
            var response = await client.CreateOrganizationAsync(new
CreateOrganizationRequest
            {
                FeatureSet = "ALL",
            });

            Organization newOrg = response.Organization;

            Console.WriteLine($"Organization: {newOrg.Id} Main Accoount:
{newOrg.MasterAccountId}");
        }
    }
```

- For API details, see CreateOrganization in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

Example 1: To create a new organization

Bill wants to create an organization using credentials from account 111111111111. The following example shows that the account becomes the master account in the new organization. Because he does not specify a features set, the new organization defaults to all features enabled and service control policies are enabled on the root.

```
aws organizations create-organization
```

The output includes an organization object with details about the new organization:

```
{
        "Organization": {
                "AvailablePolicyTypes": [
                        {
                                "Status": "ENABLED",
                                "Type": "SERVICE_CONTROL_POLICY"
                        }
                ],
                "MasterAccountId": "111111111111",
```

```
                "MasterAccountArn": "arn:aws:organizations::111111111111:account/
o-exampleorgid/111111111111",
                "MasterAccountEmail": "bill@example.com",
                "FeatureSet": "ALL",
                "Id": "o-exampleorgid",
                "Arn": "arn:aws:organizations::111111111111:organization/o-
exampleorgid"
        }
}
```

Example 2: To create a new organization with only consolidated billing features enabled

The following example creates an organization that supports only the consolidated billing features:

```
aws organizations create-organization --feature-set CONSOLIDATED_BILLING
```

The output includes an organization object with details about the new organization:

```
{
        "Organization": {
                "Arn": "arn:aws:organizations::111111111111:organization/o-
exampleorgid",
                "AvailablePolicyTypes": [],
                "Id": "o-exampleorgid",
                "MasterAccountArn": "arn:aws:organizations::111111111111:account/
o-exampleorgid/111111111111",
                "MasterAccountEmail": "bill@example.com",
                "MasterAccountId": "111111111111",
                "FeatureSet": "CONSOLIDATED_BILLING"
        }
}
```

For more information, see Creating an Organization in the *AWS Organizations Users Guide*.

- For API details, see CreateOrganization in *AWS CLI Command Reference*.

After you have created an organization, you can add accounts to your organization in these ways from the management 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.

Email address verification with AWS Organizations

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

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

Verify your email address

Within 24 hours, follow the instructions in the email to verify your email address. If more than 24 hours have passed, see Resending the verification email.

Resend the verification email with AWS Organizations

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

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

AWS Management Console

To resend the verification request

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

3. Verify your email address within 24 hours.

After verifying your email address, you can invite other AWS accounts to your organization. For more information, see Managing account invitations with AWS Organizations.

Changing your email address for an organization with AWS Organizations

To change the email address that is associated with your management account, see Update the AWS account name, email address, or password for the root user in the *AWS Account Management Reference Guide*.

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

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

Enabling all features for an organization with AWS Organizations

AWS Organizations has two available feature sets:

- All features – This feature set is the preferred and default way to work with AWS Organizations, and it includes all the features of consolidating billing. 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 Organizations such as integration with supported AWS services and organization policies.
- Consolidated billing features – This feature set is limited to generating a single bill across an organization. No other management capabilities are available with consolidated billing.

If you create an organization with the consolidated billing feature set, you can later enable all features. However, you cannot migrate from all features to consolidated billing after all features is enabled.

Standard migration and assisted migration

The two approaches for migrating to all features are *standard migration* and *assisted migration*.

Standard migration is the self-service process available to all AWS Organizations customers to enable the all features mode.

Assisted migration is process available to Enterprise Support plan customers to request that AWS migrate their organization to the all features mode of your behalf.

> **Note**
>
> **One-way processes and rollback processes**
>
> - 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.
> - After you have begun the assisted migration process, it cannot be rolled back. You will need to wait 90 days until the process expires if you want to go through the standard process instead.

Topics

- Considerations
- Standard migration process to enable all features with Organizations
- Assisted migration process to enable all features with Organizations

Considerations

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

Invited accounts must approve the migration

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.

Invited accounts are notified which feature set is currently enabled

The owner of an invited account is informed by the invitation whether they are joining an organization with consolidated billing only, or with all features enabled. You can continue inviting accounts to your organization while enabling all features.

If you invite an account *during* the process to enable all features, the invitation states that the organization they are joining has all features enabled. If you cancel the process to enable all features before the account accepts the invitation, that invitation is canceled. You must invite the account again to be a member of an organization with consolidated billing features only.

If you invite an account and the invitation is not yet accepted *before* you begin the process to enable all features, that invitation is canceled because the invitation states that the organization has consolidated billing features only. You must invite the account again to be a member of an organization with all features enabled.

The process of creating accounts in an organization is unaffected by the migration

You can continue creating accounts in the organization. That process isn't affected by this change.

The service-linked role `AWSServiceRoleForOrganizations` is required

AWS Organizations verifies that every member account has a service-linked role named `AWSServiceRoleForOrganizations`. This role is mandatory in all accounts to enable all features. If you deleted the role in an invited account, accepting the invitation to enable all features recreates the role. If you deleted the role in an account that was created using AWS Organizations, that account receives an invitation specifically to recreate that role. All of these invitations must be accepted for the organization to complete the process of enabling all features.

Standard migration process to enable all features with Organizations

This topic describes how to enable all features with the standard migration process.

Step 1: Request invited accounts to approve the migration (Management account)

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

Minimum permissions

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

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

AWS Management Console

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

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

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

> **Tip**
>
> To get back to this page later, open the **Settings** page and in the **Request sent *date*** section, choose **View status**.

4. The **Enable all features** page shows the current request status for each account in the organization. Accounts that have agreed to the request show a status of **ACCEPTED**. Accounts that haven't yet agreed show a status of **OPEN**.

AWS CLI & AWS SDKs

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

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

- AWS CLI: enable-all-features

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

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

```
    }
}
```

The output shows the details of the handshake that invited member accounts must agree to.

- AWS SDKs: EnableAllFeatures

Notes

- A countdown of 90 days begins when the request is sent to the member accounts. All accounts must approve the request within that time period or the request expires. If the request expires, all requests related to this attempt are canceled, and you have to start over with step 2.
- Once you make the request to enable all features, any existing unaccepted account invitations will be cancelled.
- During the all features migration process, you can still initiate new account invitations and create new accounts.

After all invited accounts in the organization approve their requests, you can finalize the process and enable all features. You can also immediately finalize the process if your organization doesn't have any *invited* member accounts. To finalizing the process, continue with Step 3: Finalize the migration process to enable all features (Management account).

Step 2: Approve the request to enable all features or to recreate the service-linked role (Invited account)

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

Important

If you enable all features, the management account in the organization can apply policy-based controls on your member account. These controls can restrict what users and even what you as the administrator can do in your account. Such restrictions might prevent your account from leaving the organization.

Minimum permissions

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

- `organizations:AcceptHandshake`
- `organizations:DescribeOrganization` – required only when using the Organizations console
- `organizations:ListHandshakesForAccount`– required only when using the Organizations console
- `iam:CreateServiceLinkedRole` – required only if the `AWSServiceRoleForOrganizations` role must be recreated in the member account

AWS Management Console

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

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

AWS CLI & AWS SDKs

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

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

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

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

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

```
                    "Type": "ACCOUNT"
                }
            ]
        }
    ]
}
```

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

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

```
            }
        ]
    }
}
```

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

Step 3: Finalize the migration process to enable all features (Management account)

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

Minimum permissions

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

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

AWS Management Console

To finalize the process to enable all features

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

3. On the **Enable all features** page, choose **Finalize**, and then in the confirmation dialog box, choose **Finalize** again.
4. The organization now has all features enabled.

AWS CLI & AWS SDKs

To finalize the process to enable all features

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

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

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

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

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

- AWS SDKs:
  - ListHandshakesForOrganization
  - AcceptHandshake

Assisted migration process to enable all features with Organizations

If you are an Enterprise customer, it can be difficult to complete the standard migration process due to the large number of accounts you might manage. For example, you might have difficulty obtaining approval to migrate all invited accounts in large organizations.

Assisted migration help with this process by enabling customers with an Enterprise Support plan to request that AWS migrate their organization to all features on your behalf. This process requires that you sign an agreement contract affirming that you own all accounts, followed by a 14-day waiting period. This waiting period provides accounts time to leave the organization if the accounts want to before the migration to all features takes effect.

AWS Management Console

To migrate to all features with assisted migration

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Settings** page choose **Enable all feature** and then select **Assisted migration**.
3. Read the terms and conditions of the agreement, choose **Accept** and choose **Begin process to enable all features** to start the migration.

> **Note**
>
> **Beginning the assisted migration process overrides the standard migration process**
> If you are currently enabling all features using the standard migration process, it will be canceled, and the assisted migration process will kick-off.
> **The assisted migration process is one-way and cannot be rolled back**
> After you have begun the assisted migration process, it cannot be rolled back. You will need to wait 90 days until the process expires if you want to go through the standard process instead.

If you use assisted migration, you do not need to worry about accessing your invited account as the root user to accept the migration to all features.

You can reach out to your Technical Account Manager (TAM) for exact details, progress, and timelines for the assisted migration.

Viewing details of an organization from the management account

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

Minimum permissions

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

- `organizations:DescribeOrganization`

AWS Management Console

To view the details for your organization

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

AWS CLI & AWS SDKs

To view the details for your organization

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

- AWS CLI: describe-organization

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

```
$ aws organizations describe-organization
{
    "Organization": {
        "Id": "o-aa111bb222",
        "Arn": "arn:aws:organizations::123456789012:organization/o-aa111bb222",
```

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

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

- AWS SDKs: DescribeOrganization

Deleting an organization with AWS Organizations

When you no longer need your organization, you can delete it. Deleting an organization does not close the management account, instead it removes the management account from the organization and deletes the organization itself.

The former management account becomes a standalone AWS account that is no longer managed by AWS Organizations. You then have three options:

- You can continue to use it as a standalone account
- You can use it to create a different organization
- You can accept an invitation from another organization to add the account to that organization as a member account.

Topics

- Considerations
- Delete an organization

Considerations

Deleted organizations cannot be recovered

If you delete an organization, you can't recover it. If you created any policies inside of the organization, they're also deleted and you can't recover them.

Organizations can only be deleted after all member account have been removed

You can delete an organization only after you remove all member accounts from the organization. If you created some of your member accounts using AWS Organizations, you might be blocked from removing those accounts. You can remove a member account only if it has all the information that's required to operate as a standalone AWS account. For more information about how to provide that information and then remove the account, see Leaving an organization from a member account with AWS Organizations.

Member accounts in a 'suspended' state cannot be removed from an organization

If you closed a member account before you remove it from the organization, it enters a 'suspended' state for a period of time and you can't remove the account from the organization until it is finally closed. This can take up to 90 days and can prevent you from deleting the organization until all member accounts are completely closed.

Removing the management account from an organization by deleting the organization can affect the account in the following ways:

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

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

Back up all reports

Make sure to export or back up reports from the management account, especially billing reports. Organizational level reports and history are not stored when you delete an organization. All cost

data (such as the Cost Explorer data set) is deleted. It is recommended that you do a full export of all billing history.

For more information, see Cost and Usage Reports, Cost Explorer Reports, Savings Plans Reports, and Reserved Instance (RI) utilization and coverage.

Delete an organization

Use the following procedure to delete an organization which reverts the former management account to a standalone AWS account that is no longer managed by AWS Organizations.

Minimum permissions

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

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

AWS Management Console

To delete an organization

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

> **Important**
>
> This operation does **not** close the management account but does return it to a standalone AWS account. To close the account, follow the steps at Closing a member account in an organization with AWS Organizations.

AWS CLI & AWS SDKs

The following code examples show how to use `DeleteOrganization`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to delete an existing organization using the AWS
/// Organizations Service.
/// </summary>
public class DeleteOrganization
{
    /// <summary>
    /// Initializes the Organizations client and then calls
    /// DeleteOrganizationAsync to delete the organization.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();
```

```
            var response = await client.DeleteOrganizationAsync(new
DeleteOrganizationRequest());

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine("Successfully deleted organization.");
            }
            else
            {
                Console.WriteLine("Could not delete organization.");
            }
        }
    }
```

- For API details, see DeleteOrganization in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

To delete an organization

The following example shows how to delete an organization. To perform this operation, you must be an admin of the master account in the organization. The example assumes that you previously removed all the member accounts, OUs, and policies from the organization:

```
aws organizations delete-organization
```

- For API details, see DeleteOrganization in *AWS CLI Command Reference*.

Managing accounts in an organization with AWS Organizations

An *AWS account* is a container for your AWS resources. You create and manage your AWS resources in an AWS account.

This topic describes how to manage accounts for AWS Organizations.

Topics

- Managing the management account with AWS Organizations
- Managing member accounts with AWS Organizations
- Managing account invitations with AWS Organizations
- Migrate an account to another organization with AWS Organizations
- View details of an account in AWS Organizations
- Export details for all accounts in AWS Organizations
- Update the alternate contacts for an account in AWS Organizations
- Update the primary contact information for an account in AWS Organizations
- Update AWS Regions for an account in AWS Organizations

Managing the management account with AWS Organizations

A management account is the AWS account you use to create your organization.

The management account is the ultimate owner of the organization, having final control over security, infrastructure, and finance policies. This account has the role of a payer account and is responsible for paying all charges accrued by the accounts in its organization.

This topic describes how to manage the management account with AWS Organizations.

Topics

- Best practices for the management account
- Closing a management account in your organization

Best practices for the management account

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

Topics

- Limit who has access to the management account
- Review and track who has access
- Use the management account only for tasks that require the management account
- Avoid deploying workloads to the organization's management account
- Delegate responsibilities outside the management account for decentralization

Limit who has access to the management account

The management account is key to all the mentioned administrative tasks such as account management, policies, integration with other AWS services, consolidated billing, and so on. Therefore, you should restrict and limit access to the management account only to those admin users who need rights to make changes to the organization.

Review and track who has access

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

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

We recommend that you use the management account and its users and roles for tasks that must be performed only by that account. Store all of your AWS resources in other AWS accounts in the organization and keep them out of the management account. One important reason to keep your resources in other accounts is because Organizations service control policies (SCPs) do not work to restrict any users or roles in the management account. Separating your resources from your management account also helps you to understand the charges on your invoices.

For a list of tasks that must be called from the management account, see Operations you can call from only the organization's management account .

Avoid deploying workloads to the organization's management account

Privileged operations can be performed within an organization's management account, and SCPs do not apply to the management account. That's why you should limit the cloud resources and data contained in the management account to only those that must be managed in the management account.

Delegate responsibilities outside the management account for decentralization

Where possible, we recommend delegating responsibilities and services outside the management account. Provide your teams with permissions in their own accounts to manage the needs of the organization, without requiring access to the management account. In addition, you can register multiple delegated administrators for services that support this functionality such as AWS Service Catalog for sharing software across the organization, or AWS CloudFormation StackSets for authoring and deploying stacks.

For more information, see Security Reference Architecture, Organizing Your AWS Environment Using Multiple Accounts, and AWS services that you can use with AWS Organizations for suggestions on registering member accounts as delegated administrator for various AWS services.

For more information about setting up delegated admins, see Enabling a delegated admin account for AWS Account Management and Delegated administrator for AWS Organizations.

Closing a management account in your organization

To close the management account in your organization, you must first either close or remove all member accounts in the organization. The act of closing the management account also deletes the instance of AWS Organizations and any policies that you created inside of that organization after the post-closure period has expired.

Close the management account

Use the following procedure to close a management account.

> **Important**
>
> Before you close your management account, we highly recommend that you review considerations and understand the impact for closing an account. For more information,

see What you need to know before closing your account and What to expect after you close your account in the *AWS Account Management Guide*.

AWS Management Console

To close a management account from the Accounts page

> **Note**
>
> You cannot close a management account directly from the AWS Organizations console.

1. Sign in to the AWS Management Console as the root user for the management account that you want to close. You can't close an account while signed in as an IAM user or role.
2. Verify that there are no active member accounts remaining in your organization. To do this, go to the AWS Organizations console. If you have a member account that is still active, you will need to follow the guidance provided in Closing a member account in an organization with AWS Organizations or Remove a member account from an organization before you can move to the next step.
3. On the navigation bar in the upper-right corner, choose your account name or number, and then choose **Account**.
4. On the **Account** page, choose the **Close account** button. Read and ensure that you understand the account closure guidance.
5. Choose the **Close account** button to initiate the account closure process.
6. Within a few minutes, you should receive an email confirmation that your account has been closed.

AWS CLI & AWS SDKs

This task isn't supported in the AWS CLI or by an API operation from one of the AWS SDKs. You can perform this task only by using the AWS Management Console.

Managing member accounts with AWS Organizations

A *member account* is an AWS account, other than the management account, that is part of an organization.

This topic describes how to manage member accounts with AWS Organizations.

Topics

- Best practices for member accounts
- Creating a member account in an organization with AWS Organizations
- Accessing member accounts in an organization with AWS Organizations
- Closing a member account in an organization with AWS Organizations
- Protecting member accounts from closure with AWS Organizations
- Removing a member account from an organization with AWS Organizations
- Leaving an organization from a member account with AWS Organizations
- Updating the account name for a member account with AWS Organizations
- Updating the root user email address for a member account with AWS Organizations

Best practices for member accounts

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

Topics

- Define account name and attributes
- Efficiently scale your environment and account usage
- Enable root access management to simplify managing root user credentials for member accounts

Define account name and attributes

For your member accounts, use a naming structure and email address that reflects the account usage. For example, `Workloads+fooA+dev@domain.com` for `WorkloadsFooADev`, `Workloads+fooB+dev@domain.com` for `WorkloadsFooBDev`. If you have custom tags defined for your organization, we recommend that you assign those tags on accounts that reflect account usage,

cost center, environment, and project. This makes it easier to identify, organize, and search for accounts.

Efficiently scale your environment and account usage

As you scale, before creating new accounts, make sure accounts for similar needs do not already exist, to avoid unnecessary duplication. AWS accounts should be based on common access requirements. If you are planning to reuse the accounts, such as a sandbox account or equivalent, we recommend that you clean up unneeded resources or workloads from the accounts, but save the accounts for a future use.

Before closing accounts, note that they are subject to close account quota limits. For more information, see Quotas and service limits for AWS Organizations. Consider implementing a cleanup process to reuse accounts instead of closing them and creating new ones when possible. This way, you will avoid running into incurring costs from running resources, and reaching CloseAccount API limits.

Enable root access management to simplify managing root user credentials for member accounts

We recommend you enable root access management to help you monitor and remove root user credentials for member accounts. Root access management prevents recovery of root user credentials, improving account security in your organization.

- Remove root user credentials for member accounts to prevent sign in to the root user. This also prevents member accounts from recovery of the root user.
- Assume a privileged session to perform the following tasks on member accounts:
  - Remove a misconfigured bucket policy that denies all principals from accessing an Amazon S3 bucket.
  - Delete an Amazon Simple Queue Service resource-based policy that denies all principals from accessing an Amazon SQS queue.
  - Allow a member account to recover their root user credentials. The person with access to the root user email inbox for the member account can reset the root user password and sign in as the member account root user.

After root access management is enabled, newly created member accounts are secure-by-default, having no root user credentials, which eliminates the need for additional security, such as MFA after provisioning.

For more information, see Centralize root user credentials for member accounts in the *AWS Identity and Access Management User Guide*.

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

We recommend that you create a service control policy (SCP) in the organization and attach it to the organization's root so that it applies to all member accounts. For more information, see Secure your Organizations account root user credentials.

You can deny all root actions except a specific root only action that you must perform in your member account. For example, the following SCP prevents the root user in any member account from making any AWS service API calls except “Updating a S3 bucket policy that was misconfigured and denies access to all principals” (one of the actions that requires root credentials). For more information, see Tasks that require root user credentials in the *IAM User Guide*.

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

    "Statement": [

        {

            "Effect": "Deny",

            "NotAction":[

            "s3:GetBucketPolicy",

            "s3:PutBucketPolicy",

            "s3:DeleteBucketPolicy"

                 ],

            "Resource": "*",

            "Condition": {
 "StringLike": { "aws:PrincipalArn": "arn:aws:iam::*:root" }

            }

        }
```

```
    ]
}
```

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

Creating a member account in an organization with AWS Organizations

This topic describes how to create AWS accounts within your organization in AWS Organizations. For information about creating a single AWS account, see the Getting Started Resource Center.

Considerations before creating a member account

Organizations automatically creates the IAM role `OrganizationAccountAccessRole` for the member account

When you create a member account in your organization, Organizations automatically creates the IAM role `OrganizationAccountAccessRole` in the member account that enables users and roles in the management account to exercise full administrative control over the member account. Any additional accounts attached to the same managed policy will be updated automatically whenever the policy gets updated. This role is subject to any service control policies (SCPs) that apply to the member account.

Organizations automatically creates the service-linked role `AWSServiceRoleForOrganizations` for the member account

When you create a member account in your organization, Organizations automatically creates service-linked role `AWSServiceRoleForOrganizations` in the member account 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.

Member accounts can only be created in the root of an organization

Member accounts in an organization can only be created in the root of an organization. After you create a member account root of an organization, you can move it between OUs. For more information, see Moving accounts to an organizational unit (OU) or between the root and OUs with AWS Organizations.

Policies attached to the root immediately apply

If you have any policies attached to the root, those policies immediately apply to all users and roles in the created account.

If you have enabled service trust for another AWS service 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.

Member accounts must opt in to receive marketing emails

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

Member accounts for organizations managed by AWS Control Tower should be created in AWS Control Tower

If your organization is managed by AWS Control Tower, we recommend that you create your member accounts using the AWS Control Tower account factory in the AWS Control Tower console or using the AWS Control Tower APIs.

If you create an member account in Organizations when the organization is managed by AWS Control Tower, the account won't be enrolled with AWS Control Tower. For more information, see Referring to Resources Outside of AWS Control Tower in the *AWS Control Tower User Guide*.

Create a member account

After you sign in to the organization's management account, you can create member accounts that are part of your organization.

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

- Phone number
- Company name
- Website URL
- Address

Organizations also copies the communication language and Marketplace information (vendor of the account in some AWS Regions) from the management account.

Minimum permissions

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

- `organizations:DescribeOrganization` – required only when using the Organizations console
- `organizations:CreateAccount`
- `iam:CreateServiceLinkedRole`

AWS Management Console

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

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page, choose **Add an AWS account**.
3. On the **Add an AWS account** page, choose **Create an AWS account** (it is chosen by default).
4. On the **Create an AWS account** page, for **AWS account name** enter the name that you want to assign to the account. This name helps you distinguish the account from all other accounts in the organization and is separate from the IAM alias or the email name of the owner.
5. For **Email address of the account's owner**, enter the email address of the account's owner. This email address cannot already be associated with another AWS account because it becomes the user name credential for the root user of the account.
6. (Optional) Specify the name to assign to the IAM role that is automatically created in the new account. This role grants the organization's management account permission to access the newly created member account. If you don't specify a name, AWS Organizations gives the role a default name of `OrganizationAccountAccessRole`. We recommend that you use the default name across all of your accounts for consistency.

> **Important**
>
> Remember this role name. You need it later to grant access to the new account for users and roles in the management account.

7. (Optional) In the **Tags** section, add one or more tags to the new account by choosing **Add tag** and then entering a key and an optional value. Leaving the value blank sets it to an empty string; it isn't `null`. You can attach up to 50 tags to an account.
8. Choose **Create AWS account**.

   - If you get an error that indicates that you exceeded your account quota for the organization, see I get a "quota exceeded" message when I try to add an account to my organization.
   - 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.
   - If the error persists, contact AWS Support.

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

AWS CLI & AWS SDKs

The following code examples show how to use `CreateAccount`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Organizations;
    using Amazon.Organizations.Model;

    /// <summary>
    /// Creates a new AWS Organizations account.
    /// </summary>
    public class CreateAccount
    {
        /// <summary>
        /// Initializes an Organizations client object and uses it to create
        /// the new account with the name specified in accountName.
        /// </summary>
        public static async Task Main()
        {
            IAmazonOrganizations client = new AmazonOrganizationsClient();
            var accountName = "ExampleAccount";
            var email = "someone@example.com";

            var request = new CreateAccountRequest
            {
                AccountName = accountName,
                Email = email,
            };

            var response = await client.CreateAccountAsync(request);
            var status = response.CreateAccountStatus;

            Console.WriteLine($"The staus of {status.AccountName} is
{status.State}.");
        }
    }
```

- For API details, see CreateAccount in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

To create a member account that is automatically part of the organization

The following example shows how to create a member account in an organization. The member account is configured with the name Production Account and the email address of susan@example.com. Organizations automatically creates an IAM role using the default name of OrganizationAccountAccessRole because the roleName parameter is not specified. Also, the setting that allows IAM users or roles with sufficient permissions to access account billing data is set to the default value of ALLOW because the IamUserAccessToBilling parameter is not specified. Organizations automatically sends Susan a "Welcome to AWS" email:

```
aws organizations create-account --email susan@example.com --account-
name "Production Account"
```

The output includes a request object that shows that the status is now `IN_PROGRESS`:

```
{
        "CreateAccountStatus": {
                "State": "IN_PROGRESS",
                "Id": "car-examplecreateaccountrequestid111"
        }
}
```

You can later query the current status of the request by providing the Id response value to the describe-create-account-status command as the value for the create-account-request-id parameter.

For more information, see Creating an AWS Account in Your Organization in the *AWS Organizations Users Guide*.

- For API details, see CreateAccount in *AWS CLI Command Reference*.

Accessing member accounts in an organization with AWS Organizations

When you create an account in your organization, in addition to the root user, AWS Organizations automatically creates an IAM role that is by default named `OrganizationAccountAccessRole`.

You can specify a different name when you create it, however we recommend that you name it consistently across all of your accounts. AWS Organizations doesn't create any other users or roles.

To access the accounts in your organization, you must use one of the following methods:

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

Using the root user (Not recommended for everyday tasks)

When you create new member account in your organization, the account has no root user credentials by default. Member accounts can't sign in to their root user or perform password recovery for their root user unless account recovery is enabled.

You can centralize root access for member accounts to remove root user credentials for existing member accounts in your organization. Deleting root user credentials removes the root user password, access keys, signing certificates, and deactivates multi-factor authentication (MFA). These member accounts do not have root user credentials, can't sign in as a root user, and are prevented from recovering the root user password. New accounts you create in Organizations have no root user credentials by default.

Contact your administrator if you need to perform a task that requires root user credentials on a member account where root user credentials are not present.

To access your member account as the root user, you must go through the process for password recovery. For more information, see I forgot my root user password for my AWS account in the *AWS Sign-In User Guide*.

If you must access a member account using the root user, follow these best practices:

- 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.

- Enable multi-factor authentication (MFA) on the root user. Reset the password, and assign an MFA device to the root user.

For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*. For additional root user security recommendations, see Root user best practices for your AWS account in the *IAM User Guide*.

Using trusted access for IAM Identity Center

Use AWS IAM Identity Center and enable trusted access for IAM Identity Center with AWS Organizations. This allows users to sign in to the AWS access portal with their corporate credentials and access resources in their assigned management account or member accounts.

For more information, see Multi-account permissions in the *AWS IAM Identity Center User Guide.* For information about setting up trusted access for IAM Identity Center, see AWS IAM Identity Center and AWS Organizations.

Using the IAM role OrganizationAccountAccessRole

If you create an account by using the tools provided as part of AWS Organizations, you can access the account by using the preconfigured role named `OrganizationAccountAccessRole` that exists in all new accounts that you create this way. For more information, see Accessing a member account that has OrganizationAccountAccessRole with AWS Organizations.

If you invite an existing account to join your organization and the account accepts the invitation, you can then choose to create an IAM role that allows the management account to access the invited member account. This role is intended to be identical to the role automatically added to an account that is created with AWS Organizations.

To create this role, see Creating OrganizationAccountAccessRole for an invited account with AWS Organizations.

After you create the role, you can access it using the steps in Accessing a member account that has OrganizationAccountAccessRole with AWS Organizations.

Topics

- Creating OrganizationAccountAccessRole for an invited account with AWS Organizations
- Accessing a member account that has OrganizationAccountAccessRole with AWS Organizations

Creating OrganizationAccountAccessRole for an invited account with AWS Organizations

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

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

AWS Management Console

To create an AWS Organizations administrator role in a member account

1. Sign in to the IAM console at https://console.aws.amazon.com/iam/. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the member account. The user or role must have permission to create IAM roles and policies.
2. In the IAM console, navigate to **Roles** and then choose **Create role**.
3. Choose **AWS account**, and then select **Another AWS account**.
4. Enter the 12-digit account ID number of the management account that you want to grant administrator access to. Under **Options**, please note the following:
   - 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 an external ID? in the *IAM User Guide*.
   - 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*.
5. Choose **Next**.
6. On the **Add permissions** page, choose the AWS managed policy named `AdministratorAccess` and then choose **Next**.

7. On the **Name, review, and create** page, specify a role name and an optional description. We recommend that you use `OrganizationAccountAccessRole`, for consistency with the default name assigned to the role in new accounts. To commit your changes, choose **Create role**.
8. Your new role appears on the list of available roles. Choose the new role's name to view its details, paying special note to the link URL that is provided. Give this URL to users in the member account who need to access the role. Also, note the **Role ARN** because you need it in step 15.
9. Sign in to the IAM console at https://console.aws.amazon.com/iam/. This time, sign in as a user in the management account who has permissions to create policies and assign the policies to users or groups.
10. Navigate to **Policies** and then choose **Create policy**.
11. For **Service**, choose **STS**.
12. For **Actions**, start typing **`AssumeRole`** in the **Filter** box and then select the check box next to it when it appears.
13. Under **Resources**, ensure that **Specific** is selected and then choose **Add ARNs**.
14. 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 ARNs**.
15. If you're granting permission to assume the role in multiple member accounts, repeats steps 14 and 15 for each account.
16. Choose **Next**.
17. On the **Review and create** page, enter a name for the new policy and then choose **Create policy** to save your changes.
18. Choose **User 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.
19. Choose the **Permissions** tab.
20. Choose **Add permissions**, choose **Attach policies**, and then select the policy that you created in steps 11–18.

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 OrganizationAccountAccessRole with AWS Organizations.

Accessing a member account that has OrganizationAccountAccessRole with AWS Organizations

When you create a member account using the AWS Organizations console, AWS Organizations *automatically* creates an IAM role named `OrganizationAccountAccessRole` in the account. This role has full administrative permissions in the member account. The scope of access for this role includes all principals in the management account, such that the role is configured to grant that access to the organization's management account.

You can create an identical role for an invited member account by following the steps in Creating OrganizationAccountAccessRole for an invited account with AWS Organizations.

To use this role to access the member account, you must sign in as a user from the management account that has permissions to assume the role. To configure these permissions, perform the following procedure. We recommend that you grant permissions to groups instead of users for ease of maintenance.

AWS Management Console

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

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

   In the navigation pane, choose **Policies** and then choose **Create policy**.
3. On the Visual editor tab, choose **Choose a service**, enter **STS** in the search box to filter the list, and then choose the **STS** option.
4. In the **Actions** section, enter **`assume`** in the search box to filter the list, and then choose the **AssumeRole** option.
5. In the **Resources** section, choose **Specific**, choose **Add ARNs**
6. In the **Specify ARN(s)** section, choose **Other account** for Resource in.
7. Enter the ID of the member account you just created
8. For **Resource role name with path**, enter the name of the role that you created in the previous section (we recommended naming it `OrganizationAccountAccessRole`).

9. Choose **Add ARNs** when the dialog box displays the correct ARN.
10. (Optional) If you want to require multi-factor authentication (MFA), or restrict access to the role from a specified IP address range, then expand the Request conditions section, and select the options you want to enforce.
11. Choose **Next**.
12. On the **Review and create** page, enter a name for the new policy. For example : **`GrantAccessToOrganizationAccountAccessRole`**. You can also add an optional description.
13. Choose **Create policy** to save your new managed policy.
14. Now that you have the policy available, you can attach it to a group.

    In the navigation pane, choose **User 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.
15. Choose the **Permissions** tab, choose **Add permissions**, and then choose **Attach policies**.
16. (Optional) In the **Search** box, you can start typing the name of your policy to filter the list until you can see the name of the policy you just created in Step 2 through Step 13. You can also filter out all of the AWS managed policies by choosing **All types** and then choosing **Customer managed**.
17. Check the box next to your policy, and then choose **Attach policies**.

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

AWS Management Console

To switch to the role for the member account

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

1. From the upper-right corner of the AWS Organizations console, choose the link that contains your current sign-in name and then choose **Switch Role**.
2. Enter the administrator-provided account ID number and role name.

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

Closing a member account in an organization with AWS Organizations

If you no longer need a member account in your organization, you can close it from the AWS Organizations console following the instructions in this topic. You can only close a member account using the AWS Organizations console if your organization is in All features mode.

You can also close an AWS account directly from the **Account** page in the AWS Management Console after signing in as the root user. For step-by-step instructions, see Close an AWS account in the *AWS Account Management Guide*.

To close a management account, see Closing a management account in your organization.

Close a member account

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

> **Important**
>
> Before you close your member account, we highly recommend that you review considerations and understand the impact for closing an account. For more information, see What you need to know before closing your account and What to expect after you close your account in the *AWS Account Management Guide*.

AWS Management Console

To close a member account from the AWS Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page, find and choose the name of the member account you want to close. You can navigate the OU hierarchy, or look at a flat list of accounts without the OU structure.
3. Choose **Close** next to the account name at the top of the page. This option is only available when an AWS organization is in All features mode.

> **Note**
>
> If your organization is using Consolidated billing mode, you won't be able to see the **Close** button in the console. To close an account in consolidated billing mode, sign in to the account you want to close as the root user. On the **Accounts** page, choose the **Close account** button, enter your account ID, and then choose the **Close account** button.

4. Read and ensure that you understand the account closure guidance.
5. Enter the member account ID, and then choose **Close account**.

> **Note**
>
> Any member account that you close will display a SUSPENDED label next to its account name in the AWS Organizations console for up to 90 days after the original closure date. After 90 days, the member account will no longer be displayed in the AWS Organizations.

To close a member account from the Accounts page

Optionally, you can close an AWS member account directly from the **Accounts** page in the AWS Management Console. For step-by-step guidance, follow the instructions in Close an AWS account in the *AWS Account Management Guide*.

AWS CLI & AWS SDKs

To close an AWS account

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

- AWS CLI: close-account

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

This command produces no output when successful.

- AWS SDKs: CloseAccount

Protecting member accounts from closure with AWS Organizations

To protect member accounts from accidental closure, create an IAM policy that specifies which accounts are exempt. This policy prevents closure of protected member accounts.

Create an IAM policy to deny account closure using one of these methods:

- Explicitly list protected accounts in the policy's `Resource` element using their ARNs.
- Tag individual accounts and use the `aws:ResourceTag` global condition key to prevent closure of tagged accounts.

> **Service control policies cannot protect member accounts**
>
> Service Control Policies (SCPs) can't prtected member accounts because SCPs don't affect IAM principals in the management account.

Example IAM policies that prevent member account closures

The following code examples show two different methods you can use to restrict member accounts from closing their account.

Prevent member accounts with tags from getting closed

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

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

Prevent member accounts listed in this policy from getting closed

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

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

```
    ]
}
```

Removing a member account from an organization with AWS Organizations

Removing a member account does not close the account, instead it removes the member account from the organization. The former member account becomes a standalone AWS account that is no longer managed by AWS Organizations.

Afterwards, the account is no longer subject to any policies and is responsible for its own bill payments. The organization's management account is no longer charged for any expenses accrued by the account after it's been removed from the organization.

Considerations

IAM access roles created by the management account are not automatically deleted

When you remove a member account from the organization, any IAM role that was created to enable access by the organization's management account isn't automatically deleted. If you want to terminate this access from the former organization's management account, then you must manually delete the IAM role. For information about how to delete a role, see Deleting roles or instance profiles in the *IAM User Guide*.

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

You can remove an account from your organization only if the account has the information that is required for it to operate as a standalone account. When you create an account in an organization using the AWS Organizations console, API, or AWS CLI commands, all the information that is required of standalone accounts is *not* automatically collected.

For each account that you want to make standalone, you must choose a support plan, provide and verify the required contact information, and provide a current payment method. AWS uses the payment method to charge for any billable (not AWS Free Tier) AWS activity that occurs while the account isn't attached to an organization. To remove an account that doesn't yet have this information, follow the steps in Leaving an organization from a member account with AWS Organizations.

You must wait until at least seven days after the account was created

To remove an account that you created in the organization, you must wait until at least seven days after the account was created. Invited accounts aren't subject to this waiting period.

The owner of the account that leaves becomes responsible for all new costs accrued

At the moment the account successfully leaves the organization, the owner of the AWS account becomes responsible for all new AWS costs accrued, and the account's payment method is used. The management account of the organization is no longer responsible.

The account cannot be a delegated administrator account for any AWS service enabled for the organization

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

The account no longer has access to cost and usage data

When a member account leaves an organization, that account no longer has access to cost and usage data from the time range when the account was a member of the organization. However, the management account of the organization can still access the data. If the account rejoins the organization, the account can access that data again.

Tags attached to the account are deleted

When a member account leaves an organization, all tags attached to the account are deleted.

Principals in the account are no longer affected by any organization policies

The principals in the account are no longer affected by any policies that applied in the organization. This means that restrictions imposed by SCPs are gone, and the users and roles in the account might have more permissions than they had before. Other organization policy types can no longer be enforced or processed.

The account is no longer be covered by organization agreements

If a member account is removed from an organization, that member account will no longer be covered by organization agreements. Management account administrators should communicate this to member accounts before removing member accounts from the organization, so that

member accounts can put new agreements in place if necessary. A list of active organization agreements can be viewed in the AWS Artifact console on the AWS Artifact Organization Agreements page.

Integration with other services might be disabled

Integration with other services might be disabled. If you remove an account from an organization that has integration with an AWS service enabled, the users in that account can no longer use that service.

Remove a member account from an organization

When you sign in to the organization's management account, you can remove member accounts from the organization that you no longer need. To do this, complete the following procedure. This procedure applies only to member accounts. To remove the management account, you must delete the organization.

Minimum permissions

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

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

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

- `organizations:DescribeOrganization` – required only when using the Organizations console.
- `organizations:LeaveOrganization` – Note that the organization administrator can apply a policy to your account that removes this permission, preventing you from removing your account from the organization.
- If you sign in as an IAM user and the account is missing payment information, the user must have either `aws-portal:ModifyBilling` and `aws-portal:ModifyPaymentMethods` permissions (if the account has not yet migrated to fine-grained permissions) OR `payments:CreatePaymentInstrument` and

`payments:UpdatePaymentPreferences` permissions (if the account has migrated to fine-grained permissions). Also, the member account must have IAM user access to billing enabled. If this isn't already enabled, see Activating Access to the Billing and Cost Management Console in the *AWS Billing User Guide*.

AWS Management Console

To remove a member account from your organization

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page, find and choose the check box

next to each member account that you want to remove from your organization. You can navigate the OU hierarchy or enable **View AWS accounts only** to see a flat list of accounts without the OU structure. If you have a lot of accounts, you might have to choose **Load more accounts in '*ou-name*'** at the bottom of the list to find all of those you want to move.

   On the **AWS accounts** page, find and choose the name of the member account that you want to remove from your organization. You might have to expand OUs (choose the

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

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

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

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

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

d. Remove the IAM roles that grant access to your account from the organization.

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

AWS CLI & AWS SDKs

To remove a member account from your organization

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

- AWS CLI: remove-account-from-organization

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

This command produces no output when successful.

- AWS SDKs: RemoveAccountFromOrganization

After the member account has been removed from the organization, make sure to remove the IAM roles that grant access to your account from the organization.

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

Member accounts can remove themselves with leave-organization instead. For more information, see Leaving an organization from a member account with AWS Organizations.

Leaving an organization from a member account with AWS Organizations

When you sign in to a member account, you can leave an organization. The management account can't leave the organization using this technique. To remove the management account, you must delete the organization.

Considerations

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

If a member account leaves an organization and becomes a standalone account, the account no longer has access to cost and usage data from the time range when the account was a member of the organization. The account has access only to the data that is generated as a standalone account.

If a member account leaves organization A to join organization B, the account no longer has access to cost and usage data from the time range when the account was a member of organization A. The account has access only to the data that is generated as a member of organization B.

If an account rejoins an organization that it previously belonged to, the account regains access to its historical cost and usage data.

The account is no longer covered by organization agreements that were accepted on its behalf

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

Leave an organization from a member account

To leave an organization, complete the following procedure.

Minimum permissions

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

- `organizations:DescribeOrganization` – required only when using the Organizations console.
- `organizations:LeaveOrganization` – Note that the organization administrator can apply a policy to your account that removes this permission, preventing you from removing your account from the organization.
- If you sign in as an IAM user and the account is missing payment information, the user must have either `aws-portal:ModifyBilling` and `aws-portal:ModifyPaymentMethods` permissions (if the account has not yet migrated to fine-grained permissions) OR `payments:CreatePaymentInstrument` and `payments:UpdatePaymentPreferences` permissions (if the account has migrated

to fine-grained permissions). Also, the member account must have IAM user access to billing enabled. If this isn't already enabled, see Activating Access to the Billing and Cost Management Console in the *AWS Billing User Guide*.

AWS Management Console

To leave an organization from your member account

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

   By default, you don't have access to the root user password in a member account that was created using AWS Organizations. If required, recover the root user password by following the steps in **Using the root user (Not recommended for everyday tasks)** in Accessing member accounts in an organization with AWS Organizations.
2. On the **Organizations Dashboard** page, choose **Leave this organization**.
3. In the **Confirm leaving the organization?** dialog box, choose **Leave organization**. When prompted, confirm your choice to remove the account. After you have confirmed, 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 you see a **You can't leave the organization yet** message, your account doesn't have all the required information to operate as a standalone account. If this is the case, proceed to the next step.
4. If the **Confirm leaving the organization?** dialog box displays the message **You can't leave the organization yet**, choose the **Complete the account sign-up steps** link.

   If you do not see the **Complete the account sign-up steps** link, use this link to go the **Sign up for AWS** page complete the missing registration steps.
5. On the **Sign up for AWS** page, enter all of the required information necessary for this to become a standalone account. This might include the following types of information:

   - Contact name and address
   - Valid payment method
   - Phone number verification

- Support plan options

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

   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.

7. Remove the IAM roles that grant access to your account from the organization.

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

AWS CLI & AWS SDKs

To leave an organization as a member account

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

- AWS CLI: leave-organization

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

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

  This command produces no output when successful.

- AWS SDKs: LeaveOrganization

After the member account has left the organization, make sure to remove the IAM roles that grant access to your account from the organization.

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

Member accounts can also be removed by a user in the management account with remove-account-from-organization instead. For more information, see Remove a member account from an organization.

Updating the account name for a member account with AWS Organizations

When you sign in to your organization's management account, you can update the account name for a member account. To learn how to update a member account name, follow the steps in Update the account name for any AWS account in your organization in the *AWS Account Management Reference Guide*.

Updating the root user email address for a member account with AWS Organizations

For increased security and administrative resilience, IAM principals in the management account (that have the necessary IAM permissions) can centrally update a root user email address (also referred to as the primary email address) for any of their member accounts without having to sign into each account individually. This gives administrators in the management account (or in a delegated administrator account) more control over their member accounts. It also ensures that root user email addresses from any member accounts across your AWS Organizations can be

kept up to date, even when you may have lost access to the original root user email address or administrative credentials.

When the root user email address is changed centrally by a management account administrator, both the password and MFA configuration will remain the same as they were before the change. Note that MFA can be bypassed by a user with control of an account's root user email address and primary contact phone number.

To update the root user email address of a member account in your organization, your organization must have previously enabled all features mode. AWS Organizations in consolidated billing mode or accounts that are not part of an organization, cannot update their root user email address centrally. Users that want to change the root user email address for accounts that are unsupported by the API should continue to use the Billing Console to manage their root user email address.

For step-by-step instructions on how to update your member account's root user email address, see Update the root user email for any AWS account in your organization in the *AWS Account Management Reference Guide*.

Managing account invitations with AWS Organizations

After you create an organization and verify that you own the email address associated with the management account, you can invite existing AWS accounts to join your organization. Use the AWS Organizations console to initiate and manage invitations that you send to other accounts. You can send an invitation to other accounts only from the management account of your organization.

When you invite an account, AWS Organizations sends an invitation to the account owner, who can decide to accept or decline the invitation.

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.

To create an account that automatically is part of an organization, see Creating a member account in an organization with AWS Organizations.

> **Important**
>
> All accounts in an organization must come from the same AWS partition as the management account. Accounts in the commercial AWS Regions partition can't be in

an organization with accounts from the China Regions partition or accounts in the AWS GovCloud (US) Regions partition.

Topics

- Considerations
- Sending account invitations with AWS Organizations
- Managing pending account invitations with AWS Organizations
- Accepting or declining account invitations with AWS Organizations

Considerations

Limitations on the number of invite you can send per day

For limitations on the number of invitations you can send per day, see Maximum and minimum values. 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 reset if the invited account declines, the management account cancels the invitation, or the invitation expires.

An account can only join one organization

An account can only join one organization. If you receive multiple invitations, you can accept only one.

Billing history and reports stay with the management account

Billing history and reports for all accounts stay with the management account in an Organization. Before you move the account to a new Organization, export or back up any billing and report histories for any member accounts that you want to keep. This might include Cost and Usage Reports, Cost Explorer Reports, Savings Plans Reports, and Reserved Instance (RI) utilization and coverage.

The management account is responsible for all charges accrued by member accounts

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

Organizations automatically creates the service-linked role `AWSServiceRoleForOrganizations`

AWS Organizations creates a service-linked role called `AWSServiceRoleForOrganizations` to support integrations between AWS Organizations and other AWS services. For more information, see AWS Organizations and service-linked roles. The invited account must have this role if your organization supports all features. You can delete this role if the organization supports only the consolidated billing feature set. If you delete this role and later you enable all features in your organization, AWS Organizations recreates this role for the account.

Organizations does not automatically create the IAM role `OrganizationAccountAccessRole`

For invited member accounts, AWS Organizations doesn't automatically create the IAM role `OrganizationAccountAccessRole`. This role grants users in the management account administrative access to the member account. If you want to enable that level of administrative control to an invited account, you can manually add the role. For more information, see Creating OrganizationAccountAccessRole for an invited account with AWS Organizations.

> **Note**
>
> When you create an account in your organization instead of inviting an existing account to join, AWS Organizations automatically creates the IAM role `OrganizationAccountAccessRole`by default.

Policies attached to the root or OU that contain the account immediately apply

If you have any policies attached to the root or the organizational unit (OU) that contains the invited account, those policies immediately apply to all users and roles in the invited account.

You can enable service trust for another AWS service 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.

Organizations with only the consolidated billing feature set can still invite accounts

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.

Sending account invitations with AWS Organizations

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

Minimum permissions

To invite an AWS account to join your organization, you must have the following permissions:

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

AWS Management Console

To invite another account to join your organization

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

   If you haven't yet verified your email address, follow the instructions in the verification email within 24 hours after you create the organization. There might be a delay before you receive the verification email message. You can't invite an account to join your organization until you verify your email address.
3. Navigate to the **AWS accounts** page, and choose **Add an AWS account**.
4. On the **Add an AWS account** page, choose **Invite an existing AWS account**.

5. On the **Invite an existing AWS** page, for **Email address or account ID of the AWS account to invite** enter either the email address associated with the account to be invited, or its account ID number.
6. (Optional) For **Message to include in the invitation email message**, enter any text that you want to include in the email invitation to the invited account owner.
7. (Optional) In the **Add tags** section, specify one or more tags that are automatically applied to the account after its administrator accepts the invitation. To do this, choose **Add tag** and then enter a key and an optional value. Leaving the value blank sets it to an empty string; it isn't `null`. You can attach up to 50 tags to an AWS account.
8. Choose **Send invitation**.

   > **Important**
   >
   > If you get a message that you exceeded your account quotas for the organization or that you can't add an account because your organization is still initializing, contact AWS Support.

9. The console redirects you to the **Invitations** page page where you can view all open and accepted invitations here. The invitation that you just created appears at the top of the list with its status set to **OPEN**.

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

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

AWS CLI & AWS SDKs

To invite another account to join your organization

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

- AWS CLI: invite-account-to-organization

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

```
                ],
                "Type": "ORGANIZATION",
                "Value": "o-exampleorgid"
            },
            {
                "Type": "EMAIL",
                "Value": "juan@example.com"
            }
        ],
        "State": "OPEN"
    }
}
```

- AWS SDKs: InviteAccountToOrganization

Managing pending account invitations with AWS Organizations

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

Minimum permissions

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

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

AWS Management Console

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

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

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

If you can't see an invitation, check if the invited account is the management account of another organization. Only member accounts and standalone accounts are able to receive invitations. Management accounts cannot receive invitations.

If you want to invite an account that is a management account in another organization, it is recommended that you make that account a standalone account.

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

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

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

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

AWS CLI & AWS SDKs

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

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

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

```
$ aws organizations list-handshakes-for-organization
{
    "Handshakes": [
        {
            "Action": "INVITE",
```

```
            "Arn": "arn:aws:organizations::111111111111:handshake/o-exampleorgid/
invite/h-examplehandshakeid111",
            "ExpirationTimestamp": 1482952459.257,
            "Id": "h-examplehandshakeid111",
            "Parties": [
                {
                    "Id": "o-exampleorgid",
                    "Type": "ORGANIZATION"
                },
                {
                    "Id": "juan@example.com",
                    "Type": "EMAIL"
                }
            ],
            "RequestedTimestamp": 1481656459.257,
            "Resources": [
                {
                    "Resources": [
                        {
                            "Type": "MASTER_EMAIL",
                            "Value": "bill@amazon.com"
                        },
                        {
                            "Type": "MASTER_NAME",
                            "Value": "Management Account"
                        },
                        {
                            "Type": "ORGANIZATION_FEATURE_SET",
                            "Value": "FULL"
                        }
                    ],
                    "Type": "ORGANIZATION",
                    "Value": "o-exampleorgid"
                },
                {
                    "Type": "EMAIL",
                    "Value": "juan@example.com"
                },
                {
                    "Type":"NOTES",
                    "Value":"This is an invitation to Juan's account to join
 Bill's organization."
                }
            ],
```

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

```
            ]
        }
    ]
}
```

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

```
$ aws organizations cancel-handshake --handshake-id h-examplehandshakeid111
{
    "Handshake": {
        "Id": "h-examplehandshakeid111",
        "State":"CANCELED",
        "Action": "INVITE",
        "Arn": "arn:aws:organizations::111111111111:handshake/o-exampleorgid/
invite/h-examplehandshakeid111",
        "Parties": [
            {
                "Id": "o-exampleorgid",
                "Type": "ORGANIZATION"
            },
            {
                "Id": "susan@example.com",
                "Type": "EMAIL"
            }
        ],
        "Resources": [
            {
                "Type": "ORGANIZATION",
                "Value": "o-exampleorgid",
                "Resources": [
                    {
                        "Type": "MASTER_EMAIL",
                        "Value": "bill@example.com"
                    },
                    {
                        "Type": "MASTER_NAME",
                        "Value": "Management Account"
                    },
                    {
                        "Type": "ORGANIZATION_FEATURE_SET",
                        "Value": "CONSOLIDATED_BILLING"
                    }
                ]
```

```
            },
            {
                "Type": "EMAIL",
                "Value": "anika@example.com"
            },
            {
                "Type": "NOTES",
                "Value": "This is a request for Susan's account to join Bob's
 organization."
            }
        ],
        "RequestedTimestamp": 1.47008383521E9,
        "ExpirationTimestamp": 1.47137983521E9
    }
}
```

- AWS SDKs: ListHandshakesForOrganization, CancelHandshake

Accepting or declining account invitations with AWS Organizations

If you receive an invitation to join an organization, you can accept or decline the invitation.

Considerations

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

If a member account leaves an organization and becomes a standalone account, the account no longer has access to cost and usage data from the time range when the account was a member of the organization. The account has access only to the data that is generated as a standalone account.

If a member account leaves organization A to join organization B, the account no longer has access to cost and usage data from the time range when the account was a member of organization A. The account has access only to the data that is generated as a member of organization B.

If an account rejoins an organization that it previously belonged to, the account regains access to its historical cost and usage data.

Only member accounts and standalone accounts can accept or decline an invitation

Only member accounts and standalone accounts can accept or decline an invitation to join an organization. If an invitation is sent to a member account, that account should leave the current

organization before accepting the invitation. If an invitation is sent to a management account that is already part of an organization, that account won't be able to view the invitation until they remove all member accounts from their organization and delete the organization.

Accept or decline to an account invitation

To accept or decline the invitation, complete the following steps.

Minimum permissions

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

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

AWS Management Console

To accept or decline an invitation

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

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

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

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

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

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

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

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

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

AWS CLI & AWS SDKs

To accept or decline an invitation

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

- AWS CLI: accept-handshake, decline-handshake

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

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

```
        "State": "ACCEPTED"
    }
}
```

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

- AWS SDKs: AcceptHandshake, DeclineHandshake

Migrate an account to another organization with AWS Organizations

You can migrate an AWS account from one organization to another at any time. For example, migrating an account can be helpful in the case of a merger and acquisition when you need to consolidate one or more AWS accounts from multiple organizations into one organization.

Whatever your use case, migrating an account between organizations requires for you to remove the account from the old organization, for you to make the account a standalone account, and for the account to accept the invitation from the new organization to join the new organization. Your workloads and services will continue to operate according to your specifications during the migration. However, it is important to be aware of any dependencies you might have in your organization.

Note

Closed or suspended accounts cannot be migrated
You cannot migrate a closed or suspended account. To reactive an account, contact Support.
Seven day age requirement
To migrate an account that you created in an organization, you must wait until at least seven days after the account was created. Invited accounts aren't subject to this waiting period.
Replicating data between accounts
The following AWS Prescriptive Guidance provides information about strategies for replicating data between AWS accounts: Resource replication or migration between AWS accounts.

What you need to do before migrating an account

Before migrating your AWS account from one organization to another, make sure you have completed the following steps.

Step 1: Check that you have the necessary IAM permissions to migrate an account

Step 1

Make sure you have applied the necessary permissions for migrating an account to the respective organizations.

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

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

For more information, see Leave an organization from your member account.

To invite an AWS account to join an organization, you must have the following permissions:

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

For more information, see Inviting an AWS account to join your organization.

To migrate an account, you cannot have IAM policies or service control policies that prevent migration

If you are the management account or a delegated administrator, you can control access to AWS resources by attaching permissions policies to IAM identities (users, groups, and roles) within an organization. For more information, see IAM policies for AWS Organizations.

Before migrating an account:

- Check that there are no IAM policies or service control policies (SCPs) that prevent you from migrating the account.
- Identify existing IAM policies and service control policies (SCPs) that you need to replicate in the organization where you are migrating the account.

- Identify existing IAM policies which specify your organization ID. For example, `aws:PrincipalOrgID`.

For more information, see Managing IAM policies in the *IAM User Guide* and Service control policies (SCPs).

Step 2: Check that you have removed IAM permissions that enable access to the old management account

Step 2

Make sure you have removed IAM permissions that enable access to the old management account such as `OrganizationAccountAccessRole`.

When you remove a member account from an organization, any IAM role that was created to enable access by the organization's management account isn't automatically deleted. If you want to terminate this access from the former organization's management account, then you must manually delete the IAM role.

For information about how to delete a role, see Deleting roles or instance profiles in the *IAM User Guide*.

Step 3: Check your phone verification and payment method

Step 3

The migrating account must operate as a standalone account for a period of time before migrating to the new organization.

To allow an account operate as a standalone account, check the following:

- Make sure your phone verification is up-to-date.
- Make sure have have added a valid payment method for the account to address any charges that are incurred while the account is migrating.
- If you use invoicing for your payment method, make sure your invoice is up-to-date.

Step 4: Back up all reports

Step 4

Make sure to export or back up reports from the management account, especially billing reports. Organizational level reports and history are not stored when you migrate an account. It is recommended that you do a full export of all billing history. You can still access reports for member account such as AWS CloudTrail Event history and account billing history.

> **Important**
>
> All organizational level reporting and history, such as organizational billing information in the management account, will be deleted after an account is removed from an organization.

For more information, see Cost and Usage Reports, Cost Explorer Reports, Savings Plans Reports, and Reserved Instance (RI) utilization and coverage.

Step 5: Check for organization dependencies

Step 5

Make sure the migrating account does not have any organization-related dependencies.

Dependencies to check:

- If the account is a delegated administrator, you must deregister the delegated administrator permissions before migrating the account. For more information, see Services you can use with AWS Organizations.
- If the account is the management account, you must remove all member accounts from the organization and delete the organization before migrating. After you have deleted the organization, your management account will operate as a standalone account. After migration, the management account will be a member account of the new organization. For more information, see Deleting an organization.
- If any IAM permissions depend on the account, you will need to adjust the permissions for the old organization after you have migrated the account to the new organization in order for the old organization to function as before. For more information, see Managing access permissions for your organization.

- If you are using any account or organizational unit (OU) tags, you will need to recreate the tags in the new organization.

(Optional) Step 6: Review guidance if you use AWS Control Tower

(Optional) Step 6

If you are migrating an account to or from an organization managed by AWS Control Tower, review the following AWS Prescriptive Guidance: Migrate an AWS member account from AWS Organizations to AWS Control Tower.

What you need to do to migrate an account

The migration process requires for the new organization to send an invitation to the migrating account, for the old organization to remove the migration account, and for the migrating account to accept the invitation from the new organization to join the new organization.

To migrate an account

1. Send an invitation from the management account of the new organization to the migrating account. You should send the invitation to the account before it leaves the old organization. This helps to minimize the costs incurred when the migrating account temporarily operates as a standalone account. For information about inviting accounts, see Inviting an AWS account to join your organization.
2. Remove the migrating account from the old organization. You can remove a member account from your organization using the management account or leave an organization from as a member account.
3. Accept the invitation to join the new organization. For more information, see Accepting an invitation from an organization. Accounts that are migrated from one another organization to another will be automatically added to the root of the new organization. Before moving an account to an organizational unit (OU) in the new organization, it is recommended that you check that migrating account has the appropriate organization policies and OU permissions.
4. If you want to migrate the management account, you must remove all member accounts from the organization and delete the organization before migrating the management account to the new organization. After you have deleted the old organization, your management account will operate as a standalone account and can accept the invitation from the new organization to join the new organization. If you accept the invitation, the management account will be a member account of the new organization.

What you need to do after migrating an account

After migration your account from one organization to another, make sure you have completed the following steps.

Post-migration review

1. Evaluate all of the billing tool configurations for the migrated account, such as cost categories, budgets, and billing alarms.
2. Review and update the following monetary information for any accounts that you migrated from one organization to another:
   a. If necessary, update the tax settings on the account.
   b. Make sure the Support plan for migrating account matches payer account for the new organization.
   c. Review any possible tax exemptions that you might want to apply to the account you migrated.
3. Validate and confirm existing IAM policies and service control policies (SCPs) for the migrated account. For example, you might need to update the organization ID for some IAM policies to reflect the new organization.
4. Update cost allocation tags for new organization where you migrated the account. You will need to update all the previous cost allocation tags collected by account you migrated.
5. Any Reserved Instances and Saving Plans will migrate along with the account. These are not retained in the old organization. Contact Support if these need to be transferred to the old organization.

View details of an account in AWS Organizations

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

Minimum permissions

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

- `organizations:DescribeAccount`

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

AWS Management Console

To view details of an AWS account

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

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

AWS CLI & AWS SDKs

To view details of an AWS account

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

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

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

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

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

{
    "Account": {
        "Id": "123456789012",
```

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

- AWS SDKs:
  - ListAccounts
  - DescribeAccount

Export details for all accounts in AWS Organizations

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

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

Export a list of all AWS accounts in your organization

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

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

- **Account ID** - Numeric account identifier. For example: 123456789012
- **ARN** - Amazon Resource Name for the account. For example: `arn:aws:organizations::123456789012account/o-o1gb0d1234/123456789012`

- **Email** - Email address associated with the account. For example: marymajor@example.com
- **Name** - Account name provided by account creator. For example: stage testing account
- **Status** - Account status within the organization. Value can be `PENDING`, `ACTIVE` or `SUSPENDED`.
- **Joined method** - Specifies how the account was created. Value can be `INVITED` or `CREATED`.
- **Joined timestamp** - Date and time the account joined the organization.

Minimum permissions

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

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

AWS Management Console

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

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

AWS CLI & AWS SDKs

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

Update the alternate contacts for an account in AWS Organizations

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

Update the primary contact information for an account in AWS Organizations

You can update primary contact information for accounts within your organization using the AWS Organizations console, or programmatically using the AWS CLI or AWS SDKs. To learn how to update primary contact information, see Update the primary contact for any AWS account in your organization in the *AWS Account Management Reference*.

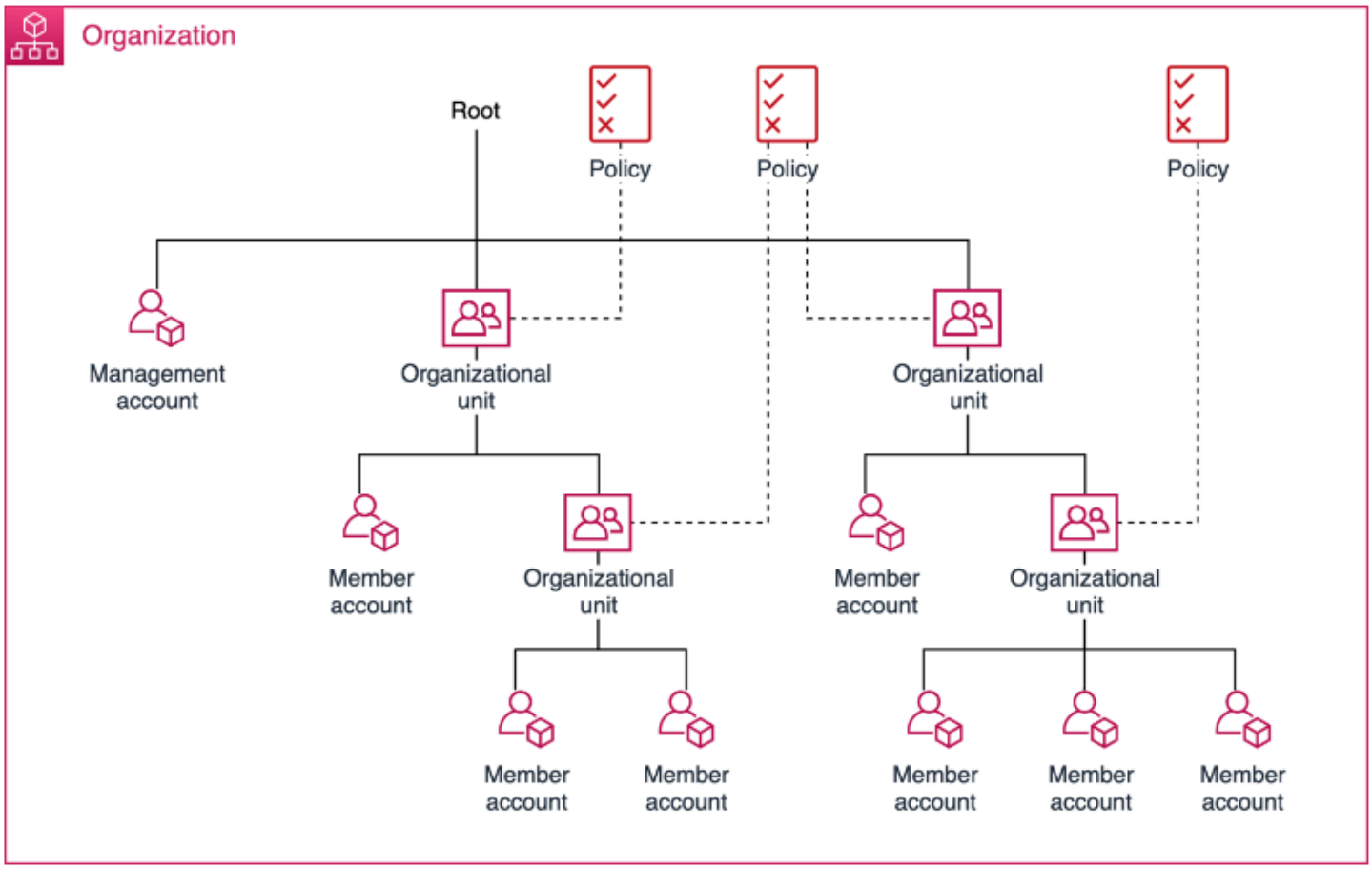
Update AWS Regions for an account in AWS Organizations

You can update enabled AWS Regions for accounts within your organization using the AWS Organizations console. To learn how to update enabled AWS Regions, see Enable or disable AWS Regions in your account in the *AWS Account Management Reference*.

Managing organizational units (OUs) with AWS Organizations

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.

The following diagram shows an organization that consists of seven accounts that are organized into four OUs under the root. The organization also has a few policies that are applied to OUs.



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

Topics

- Best practices for managing organizational units (OUs) with AWS Organizations
- Navigating the root and organizational unit (OU) hierarchy with AWS Organizations
- Viewing details of an organizational unit (OU) with AWS Organizations
- Creating an organizational unit (OU) with AWS Organizations
- Renaming an organizational unit (OU) with AWS Organizations
- Tagging an organizational unit (OU) with AWS Organizations
- Moving accounts to an organizational unit (OU) or between the root and OUs with AWS Organizations
- Viewing details of the root with AWS Organizations
- Deleting an organizational unit (OU) with AWS Organizations

Best practices for managing organizational units (OUs) with AWS Organizations

Follow these recommendations to help walk you through managing a multi-account environment in AWS Organizations using organization units (OUs).

Topics

- Understanding AWS Organizations
- Recommended foundational organizational unit (OUs)
- Recommended additional organizational unit (OUs)
- Conclusion

Understanding AWS Organizations

The basis of a well-architected multi-account AWS environment is AWS Organizations, which enables you to centrally manage and govern multiple accounts. An organizational unit (OU) is a logical grouping of accounts in an organization. OUs enable you to organize your accounts into a hierarchy, and help you apply management controls. Organizations policies define the controls that you can apply to a group of AWS accounts. For example, a service control policy (SCP) is a policy that defines the AWS service actions, such as Amazon EC2 Run Instance, that accounts in your organization can perform.

While you might begin your AWS journey with a single account, AWS recommends that you set up multiple accounts as your workloads grow in size and complexity. Using a multi-account environment is an AWS best practice that can offer several benefits:

- **Rapid innovation with various requirements**: You can allocate AWS accounts to different teams, projects, or products within your company to help ensure that each of them can rapidly innovate while allowing for their own security requirements.
- **Simplified billing**: Using multiple AWS accounts can simplify how you allocate your AWS cost by helping identify which product or service line is responsible for an AWS charge.
- **Flexible security controls**: You can use multiple AWS accounts to isolate workloads or applications that have specific security requirements, or need to meet strict guidelines for compliance such as HIPAA or PCI.
- **Adapt to business processes**: You can organize multiple AWS accounts in a manner that best reflects the diverse needs of your company's business processes that have different operational, regulatory, and budgetary requirements.

Recommended foundational organizational unit (OUs)

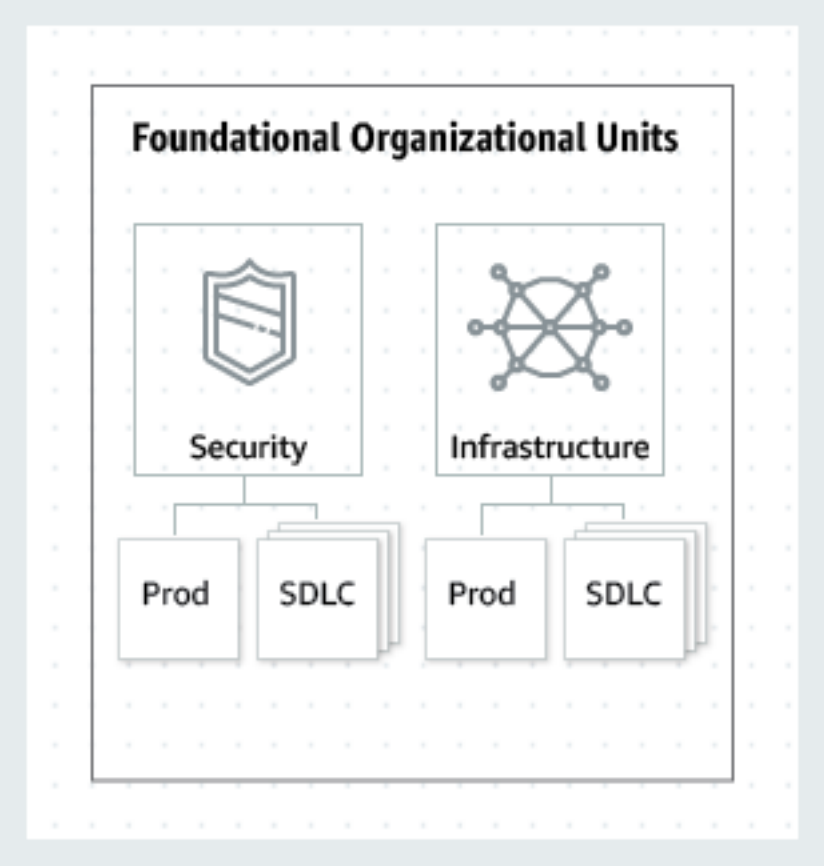
Your organizational unit (OUs) should be based on function or common set of controls instead of mirroring your company’s reporting structure. AWS recommends that you start with security and infrastructure in mind. Most businesses have centralized teams that serve the entire organization for those needs. We recommend creating a set of foundational OUs for these specific functions:

- **Security**: Used for security services. Create accounts for log archives, security read-only access, security tooling, and break-glass.
- **Infrastructure**: Used for shared infrastructure services such as networking and IT services. Create accounts for each type of infrastructure service you require.

Given that most companies have different policy requirements for production workloads, infrastructure and security can have nested OUs for *non-production* (SDLC) and *production* (Prod). Accounts in the SDLC OU host non-production workloads and should not have production dependencies from other accounts. If there are variations in OU policies between life cycle stages, SDLC can be split into multiple OUs (for example, development and pre-prod). Accounts in the Prod OU host the production workloads.

Apply policies at the OU-level to govern the Prod and SDLC environment according to your requirements. In general, applying policies at the OU-level is a better practice than at the individual account-level as it simplifies policy management and any potential troubleshooting.

The following diagram shows the foundational OUs (Prod and SDLC) for security and infrastructure:



Recommended additional organizational unit (OUs)

After the central services are in place, we recommend creating OUs that directly relate to building or running your products or services. Many AWS customers build the following OUs after establishing a foundation:

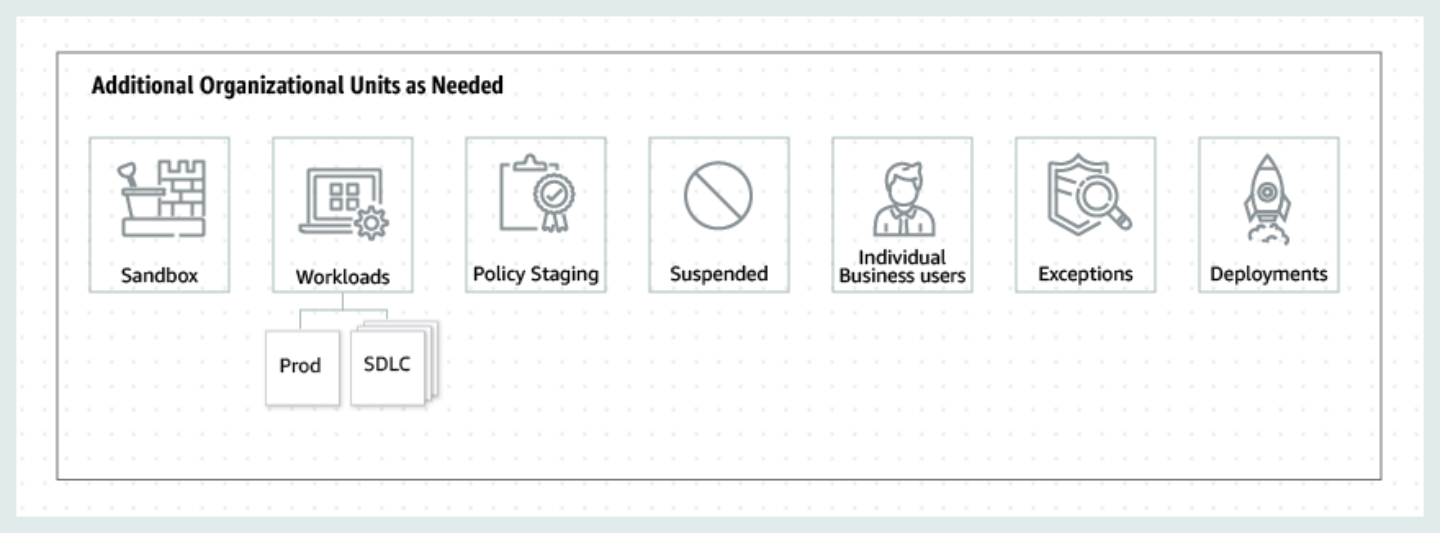
- **Sandbox**: Holds AWS accounts that individual developers can use to experiment with AWS services. Ensure that these accounts can be detached from internal networks.
- **Workloads**: Contains AWS accounts that host your external-facing application services. You should structure OUs under SDLC and Prod environments (similar to the foundational OUs) in order to isolate and tightly control production workloads.

We also recommend adding additional OUs for maintenance and continued expansion depending on your specific needs. The following are some common themes based on practices from existing AWS customers:

- **Policy Staging**: Holds AWS accounts where you can test proposed policy changes before applying them broadly to the organization. Start by implementing changes at the account level in the intended OU, and slowly work out into other accounts, OUs, and across the rest of the organization.
- **Suspended**: Contains AWS accounts that have been closed and are waiting to be deleted from the organization. Attach an SCP to this OU that denies all actions. Ensure that the accounts are tagged with details for traceability if they need to be restored.
- **Individual Business Users**: A limited access OU that contains AWS accounts for business users (not developers) who might need to create business productivity-related applications, for example set up an S3 bucket to share reports or files with a partner.
- **Exceptions**: Holds AWS accounts used for business use-cases that have highly customized security or auditing requirements, different from those defined in the Workloads OU. For example, setting up an AWS account specifically for a confidential new application or feature. Use SCPs at the account level to meet customized needs. Consider setting up a Detect and React system using Amazon EventBridge and AWS Config rules.
- **Deployments**: Contains AWS accounts meant for continuous integration and continuous delivery/deployment (CI/CD deployments). You can create this OU if you have a different governance and operational model for CI/CD deployments as compared to accounts in the Workloads OUs (Prod and SDLC). Distribution of CI/CD helps reduce the organizational dependency on a shared CI/CD environment operated by a central team. For each set of SDLC/Prod AWS accounts for an application in the Workloads OU, create an account for CI/CD under Deployments OU.
- **Transitional**: This is used as a temporary holding area for existing accounts and workloads before moving them to standard areas of your organization. This might be because accounts

are part of an acquisition, previously managed by a third party, or legacy accounts from an old organization structure.

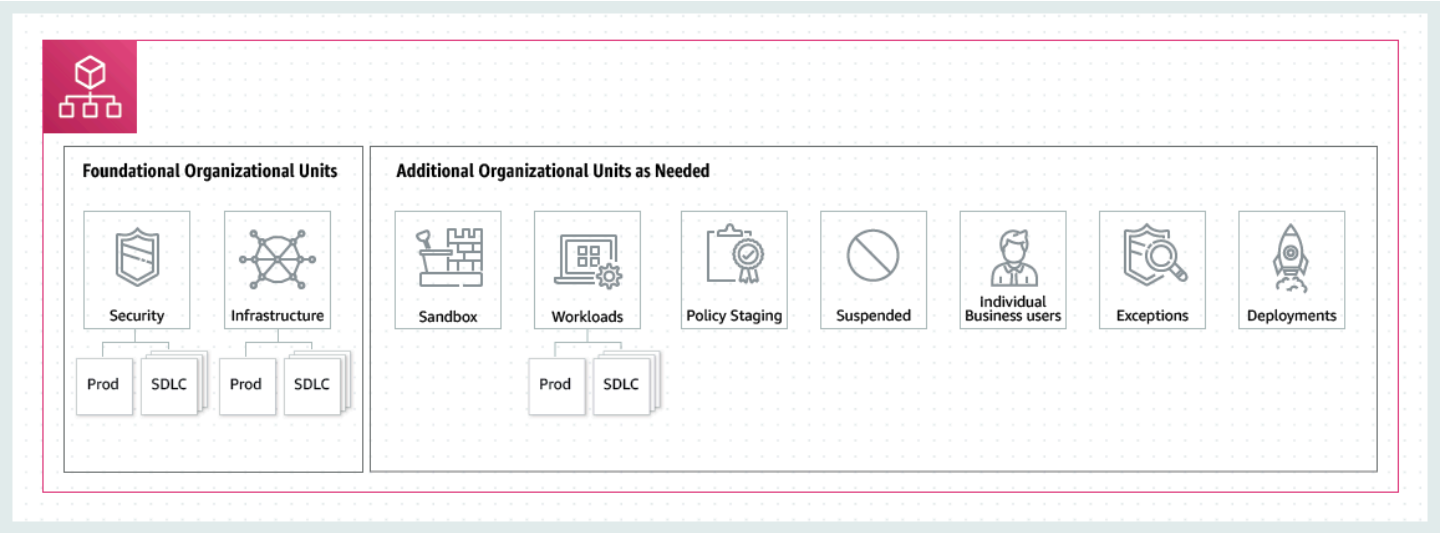
The following diagram shows additional OUs for sandbox, workloads, policy staging, suspended, individual business users, exceptions, deployments, and transitional accounts:



Conclusion

A well-architected multi-account strategy can help you innovate in AWS, while helping to ensure that you meet your security and scalability needs. The framework described in this topic represents AWS best practices that you should use as a starting point for your AWS journey.

The following diagram shows recommended foundational OUs and additional OUs:



Navigating the root and organizational unit (OU) hierarchy with AWS Organizations

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

AWS Management Console

To navigate the organization as a 'tree'

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

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

AWS Management Console

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

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

2. On the **AWS accounts** page, at the top of the **Organization** section, choose the **View AWS accounts only** switch icon to turn it on.

.
3. The list of accounts is displayed without any hierarchy.

Viewing details of an organizational unit (OU) with AWS Organizations

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

Minimum permissions

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

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

AWS Management Console

To view details of an OU

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

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

AWS CLI & AWS SDKs

To view details of an OU

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

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

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

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

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

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

- AWS SDKs:
  - ListRoots
  - ListChildren
  - DescribeOrganizationalUnit

Creating an organizational unit (OU) with AWS Organizations

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

Important

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

Minimum permissions

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

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

AWS Management Console

To create an OU

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

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

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

AWS CLI & AWS SDKs

To create an OU

The following code examples show how to use `CreateOrganizationalUnit`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Creates a new organizational unit in AWS Organizations.
/// </summary>
public class CreateOrganizationalUnit
{
    /// <summary>
    /// Initializes an Organizations client object and then uses it to call
    /// the CreateOrganizationalUnit method. If the call succeeds, it
    /// displays information about the new organizational unit.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var orgUnitName = "ProductDevelopmentUnit";

        var request = new CreateOrganizationalUnitRequest
        {
            Name = orgUnitName,
            ParentId = "r-0000",
        };

        var response = await client.CreateOrganizationalUnitAsync(request);
```

```
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully created organizational unit:
 {orgUnitName}.");
                Console.WriteLine($"Organizational unit {orgUnitName} Details");
                Console.WriteLine($"ARN: {response.OrganizationalUnit.Arn} Id:
 {response.OrganizationalUnit.Id}");
            }
            else
            {
                Console.WriteLine("Could not create new organizational unit.");
            }
        }
    }
```

- For API details, see CreateOrganizationalUnit in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

To create an OU in a root or parent OU

The following example shows how to create an OU that is named AccountingOU:

```
aws organizations create-organizational-unit --parent-id r-examplerootid111 --
name AccountingOU
```

The output includes an organizationalUnit object with details about the new OU:

```
{
        "OrganizationalUnit": {
                "Id": "ou-examplerootid111-exampleouid111",
                "Arn": "arn:aws:organizations::111111111111:ou/o-exampleorgid/ou-
examplerootid111-exampleouid111",
                "Name": "AccountingOU"
        }
}
```

- For API details, see CreateOrganizationalUnit in *AWS CLI Command Reference*.

Renaming an organizational unit (OU) with AWS Organizations

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

Minimum permissions

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

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

AWS Management Console

To rename an OU

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

AWS CLI & AWS SDKs

To rename an OU

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

- AWS CLI: update-organizational-unit

  The following example shows how to rename an OU.

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

- AWS SDKs: UpdateOrganizationalUnit

Tagging an organizational unit (OU) with AWS Organizations

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

Minimum permissions

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

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

AWS Management Console

To edit the tags attached to an OU

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

AWS CLI & AWS SDKs

To edit the tags attached to an OU

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

- AWS CLI: tag-resource and untag-resource

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

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

  This command produces no output when successful.

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

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

This command produces no output when successful.

- AWS SDKs: TagResource and UntagResource

Moving accounts to an organizational unit (OU) or between the root and OUs with AWS Organizations

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

Minimum permissions

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

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

AWS Management Console

To move accounts to an OU

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

2. On the **AWS accounts** page, find the account or accounts that you want to move. You can navigate the OU hierarchy or enable **View AWS accounts only** to see a flat list of accounts without the OU structure. If you have a lot of accounts, you might have to choose **Load more accounts in '*ou-name*'** at the bottom of the list to find all of those you want to move.
3. Choose the check box
   ☑
   next to the name of each account that you want to move.
4. On the **Actions** menu, under **AWS account**, choose **Move** .
5. In the **Move AWS account** dialog box, navigate to and then choose the OU or root that you want to move the account to, and then choose **Move AWS account**.

AWS CLI & AWS SDKs

To move accounts to an OU

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

- AWS CLI: move-account

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

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

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

Viewing details of the root with AWS Organizations

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

Minimum permissions

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

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

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

AWS Management Console

To view the details of the root

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

AWS CLI & AWS SDKs

To view the details of the root

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

- AWS CLI: list-roots

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

```
$ aws organizations list-roots
{
    "Roots": [
        {
            "Id": "r-a1b2",
            "Arn": "arn:aws:organizations::123456789012:root/o-aa111bb222/r-a1b2",
            "Name": "Root",
            "PolicyTypes": [
                {
```

```
                    "Type": "BACKUP_POLICY",
                    "Status": "ENABLED"
                }
            ]
        }
    ]
}
```

- AWS SDKs: ListRoots

Deleting an organizational unit (OU) with AWS Organizations

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

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

Minimum permissions

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

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

AWS Management Console

To delete an OU

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

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

AWS CLI & AWS SDKs

To delete an OU

The following code examples show how to use `DeleteOrganizationalUnit`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to delete an existing AWS Organizations organizational unit.
/// </summary>
public class DeleteOrganizationalUnit
{
    /// <summary>
    /// Initializes the Organizations client object and calls
    /// DeleteOrganizationalUnitAsync to delete the organizational unit
    /// with the selected ID.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var orgUnitId = "ou-0000-00000000";

        var request = new DeleteOrganizationalUnitRequest
```

```
            {
                OrganizationalUnitId = orgUnitId,
            };

            var response = await client.DeleteOrganizationalUnitAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully deleted the organizational unit
with ID: {orgUnitId}.");
            }
            else
            {
                Console.WriteLine($"Could not delete the organizational unit with
ID: {orgUnitId}.");
            }
        }
    }
```

- For API details, see DeleteOrganizationalUnit in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

To delete an OU

The following example shows how to delete an OU. The example assumes that you previously removed all accounts and other OUs from the OU:

```
aws organizations delete-organizational-unit --organizational-unit-id ou-
examplerootid111-exampleouid111
```

- For API details, see DeleteOrganizationalUnit in *AWS CLI Command Reference*.

Managing organization policies with AWS Organizations

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 in your organization.

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

Topics

- Policy types
- Authorization policies in AWS Organizations
- Management policies in AWS Organizations
- Delegated administrator for AWS Organizations
- Enabling a policy type
- Disabling a policy type
- Creating organization policies with AWS Organizations
- Updating organization policies with AWS Organizations
- Editing tags attached to organization policies with AWS Organizations
- Attaching organization policies with AWS Organizations
- Detaching organization policies with AWS Organizations
- Getting information about your organization's policies
- Deleting organization policies with AWS Organizations

Policy types

Organizations offers policy types in the following two broad categories:

Authorization policies

Authorization policies help you to centrally manage the security of AWS accounts across an organization.

- **Service control policies (SCPs)** offer central control over the maximum available permissions for IAM users and IAM roles in an organization.
- **Resource control policies (RCPs)** offer central control over the maximum available permissions for resources in an organization.

Management policies

Management policies help you centrally configure and manage AWS services and their features across an organization.

- **Declarative policies** allow you to centrally declare and enforce desired configurations for a given AWS service at scale across an organization. Once attached, the configuration is always maintained when the service adds new features or APIs.
- **Backup policies** allow you to centrally manage and apply backup plans to the AWS resources across an organization's accounts.
- **Tag policies** allow you to standardize the tags attached to the AWS resources in an organization's accounts.
- **Chat applications policies** allow you to control access to an organization's accounts from chat applications such as Slack and Microsoft Teams.
- **AI services opt-out policies** allow you to control data collection for AWS AI services for all the accounts in an organization.

The following table summarizes some of the characteristics of each policy type. For additional characteristics about these policy types, see Quotas and service limits for AWS Organizations.

Policy type	Policy category	Affects management account	Maximum number you can attach to a root, OU, or account	Maximum size	Supports viewing effective policy for OU or account
SCP	Authorization	 No	5	5120 characters	 No
RCP	Authorization	 No	5	5120 characters	 No
Declarative policy	Management	 Yes	10	10,000 characters	 Yes
Backup policy	Management	 Yes	10	10,000 characters	 Yes
Tag policy	Management	 Yes	10	10,000 characters	 Yes
Chat applications policy	Management	 Yes	5	10,000 characters	 Yes

Policy type	Policy category	Affects management account	Maximum number you can attach to a root, OU, or account	Maximum size	Supports viewing effective policy for OU or account
AI services opt-out policy	Management	 Yes	5	2500 characters	 Yes

Authorization policies in AWS Organizations

Authorization policies in AWS Organizations enable you to centrally configure and manage access for principals and resources in your member accounts. How those policies affect the organizational units (OUs) and accounts that you apply them to depends on the type of authorization policy that you apply.

There are two different types of authorization policies in AWS Organizations: service control policies (SCPs) and resource control policies (RCPs).

Topics

- Differences between SCPs and RCPs
- Using SCPs and RCPs
- Service control policies (SCPs)
- Resource control policies (RCPs)

Differences between SCPs and RCPs

SCPs are principal-centric controls. SCPs create a permissions guardrail, or set limits, on the maximum permissions available to principals in your member accounts. You can use an SCP when you want to centrally enforce consistent access controls on principals in your organization. This can include specifying which services your IAM users and IAM roles can access, which resources they can

access, or the conditions under which they can make requests (for example, from specific regions or networks).

RCPs are resource-centric controls. RCPs create a permissions guardrail, or set limits, on the maximum permissions available for resources in your member accounts. You can use an RCP when you want to centrally enforce consistent access controls across resources in your organization. This can restrict access to your resources so that they can only be accessed by identities that belong to your organization, or specifying the conditions under which identities external to your organization can access your resources.

Some controls can be applied in a similar way through SCPs and RCPs. For example, you might want to prevent your users from uploading unencrypted objects to S3 which can be written as an SCP to enforce a control on the actions that your principals can take on your S3 buckets. This control can also be written as an RCP to require encryption whenever any principal uploads objects to your S3 bucket. The second option might be preferred if your bucket allows principals outside of your organization, such as third-party vendors, to upload objects to your S3 bucket. However, some controls can only be implemented in an RCP, and some controls can only be implemented in an SCP. For more information, see General use cases for SCPs and RCPs.

Using SCPs and RCPs

SCPs and RCPs are independent controls. You can choose to enable only SCPs or RCPs, or use both policy types together. By using both SCPs and RCPs, you can create a data perimeter around your identities and your resources.

SCPs provide an ability to control which resources your identities can access. For example, you may want to allow your identities to access resources in your AWS organization. However, you may want to prevent your identities from accessing resources outside of your organization. You can enforce this control using SCPs.

RCPs provide an ability to control which identities can access your resources. For example, you may want to allow identities in your organization to be able to access resources in your organization. However, you may want to prevent identities external to your organization from accessing your resources. You can enforce this control using RCPs. RCPs provide an ability to impact the effective permissions for principals external to your organization that are accessing your resources. SCPs can only impact the effective permissions for principals within your AWS organization.

General use cases for SCPs and RCPs

The following table details general use cases for using an SCP and RCPs

		Impacts			
Use case	**Policy type**	**Your identities**	**External identities**	**Your Resources**	**External resources (target of the request)**
Restrict which services or actions your identities can use	SCP	X		X	X
Restrict which resources your identities can access	SCP	X		X	X
Enforce requirements on how your identities can access resources	SCP	X		X	X
Restrict which identities can access your resources	RCP	X	X	X	
Protect sensitive resources in your organization	RCP	X	X	X	

		Impacts		
Enforce requirements on how your resources can be accessed	RCP	X	X	X

Service control policies (SCPs)

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

SCPs do not grant permissions to the IAM users and IAM roles in your organization. No permissions are granted by an SCP. An SCP defines a permission guardrail, or sets limits, on the actions that the IAM users and IAM roles in your organization can perform. To grant permissions, the administrator must attach policies to control access, such as identity-based policies that are attached to IAM users and IAM roles, and resource-based policies that are attached to the resources in your accounts. For more information, see Identity-based policies and resource-based policies in the *IAM User Guide*.

The effective permissions are the logical intersection between what is allowed by the SCP and resource control policies (RCPs) and what is allowed by the identity-based and resource-based policies.

SCPs don't affect users or roles in the management account

SCPs don't affect users or roles in the management account. They affect only the member accounts in your organization. This also means that SCPs apply to member accounts that are designated as delegated administrators.

Topics on this page

- Testing effects of SCPs
- Maximum size of SCPs
- Attaching SCPs to different levels in the organization
- SCP effects on permissions
- Using access data to improve SCPs
- Tasks and entities not restricted by SCPs
- SCP evaluation
- SCP syntax
- Service control policy examples
- Troubleshooting service control policies (SCPs) with AWS Organizations

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.

> **Note**
>
> You should not remove the **FullAWSAccess** policy unless you modify or replace it with a separate policy with allowed actions, otherwise all AWS actions from member accounts will fail.

Maximum size of SCPs

All characters in your SCP count against its maximum size. 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.

Attaching SCPs to different levels in the organization

For a detailed explanation of how SCPs work, see SCP evaluation.

SCP effects on permissions

SCPs are similar to AWS Identity and Access Management permission policies and use almost the same syntax. However, an SCP never grants permissions. Instead, SCPs are access controls that specify the maximum available permissions for the IAM users and IAM roles in your organization. For more information, see Policy Evaluation Logic in the *IAM User Guide*.

- SCPs ***affect only IAM users and roles*** that are managed by accounts that are part of the organization. SCPs don't affect resource-based policies directly. They also don't affect users or roles from accounts outside the organization. For example, consider an Amazon S3 bucket that's owned by account A in an organization. The bucket policy (a resource-based policy) grants access to users from account B outside the organization. Account A has an SCP attached. That SCP doesn't apply to those outside users in account B. The SCP applies only to users that are managed by account A in the organization.
- An SCP restricts permissions for IAM users and roles in member accounts, including the member account's root user. Any account has only those permissions permitted by ***every*** parent above it. If a permission is blocked at any level above the account, either implicitly (by not being included in an `Allow` policy statement) or explicitly (by being included in a `Deny` policy statement), a user or role in the affected account can't use that permission, even if the account administrator attaches the `AdministratorAccess` IAM policy with */* permissions to the user.
- SCPs affect only ***member*** accounts in the organization. They have no effect on users or roles in the management account. This also means that SCPs apply to member accounts that are designated as delegated administrators. For more information, see Best practices for the management account.
- Users and roles must still be granted permissions with appropriate IAM permission policies. A user without any IAM permission policies has no access, even if the applicable SCPs allow all services and all actions.

- If a user or role has an IAM permission policy that grants access to an action that is also allowed by the applicable SCPs, the user or role can perform that action.
- If a user or role has an IAM permission policy that grants access to an action that is either not allowed or explicitly denied by the applicable SCPs, the user or role can't perform that action.
- SCPs affect all users and roles in attached accounts, ***including the root user***. The only exceptions are those described in Tasks and entities not restricted by SCPs.
- SCPs ***do not*** affect any service-linked role. Service-linked roles enable other AWS services to integrate with AWS Organizations and can't be restricted by SCPs.
- When you disable the SCP policy type in a root, all SCPs are automatically detached from all AWS Organizations entities in that root. AWS Organizations entities include organizational units, organizations, and accounts. If you reenable SCPs in a root, that root reverts to only the default `FullAWSAccess` policy automatically attached to all entities in the root. Any attachments of SCPs to AWS Organizations entities from before SCPs were disabled are lost and aren't automatically recoverable, although you can manually reattach them.
- If both a permissions boundary (an advanced IAM feature) and an SCP are present, then the boundary, the SCP, and the identity-based policy must all allow the action.

Using access data to improve SCPs

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

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

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

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

Tasks and entities not restricted by SCPs

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

- Any action performed by the management account
- Any action performed using permissions that are attached to a service-linked role
- Register for the Enterprise support plan as the root user
- Provide trusted signer functionality for CloudFront private content
- Configure reverse DNS for an Amazon Lightsail email server and Amazon EC2 instance as the root user
- Tasks on some AWS-related services:
  - Alexa Top Sites
  - Alexa Web Information Service
  - Amazon Mechanical Turk
  - Amazon Product Marketing API

SCP evaluation

> **Note**
>
> The information in this section does ***not*** apply to management policy types, including backup policies, tag policies, chat applications policies, or AI services opt-out policies. For more information, see Understanding management policy inheritance.

As you can attach multiple service control policies (SCPs) at different levels in AWS Organizations, understanding how SCPs are evaluated can help you write SCPs that yield the right outcome.

Topics

- How SCPs work with Allow
- How SCPs work with Deny
- Strategy for using SCPs

How SCPs work with Allow

For a permission to be **allowed** for a specific account, there must be an **explicit `Allow` statement** at every level from the root through each OU in the direct path to the account (including the target account itself). This is why when you enable SCPs, AWS Organizations attaches an AWS managed SCP policy named FullAWSAccess which allows all services and actions. If this policy is removed and not replaced at any level of the organization, all OUs and accounts under that level would be blocked from taking any actions.

For example, let's walk through the scenario shown in figures 1 and 2. For a permission or a service to be allowed at Account B, a SCP that allows the permission or service should be attached to Root, the Production OU, and to Account B itself.

SCP evaluation follows a deny-by-default model, meaning that any permissions not explicitly allowed in the SCPs are denied. If an allow statement is not present in the SCPs at any of the levels such as Root, Production OU or Account B, the access is denied.

Notes

- An `Allow` statement in an SCP permits the `Resource` element to only have a `"*"` entry.
- An `Allow` statement in an SCP can't have a `Condition` element at all.

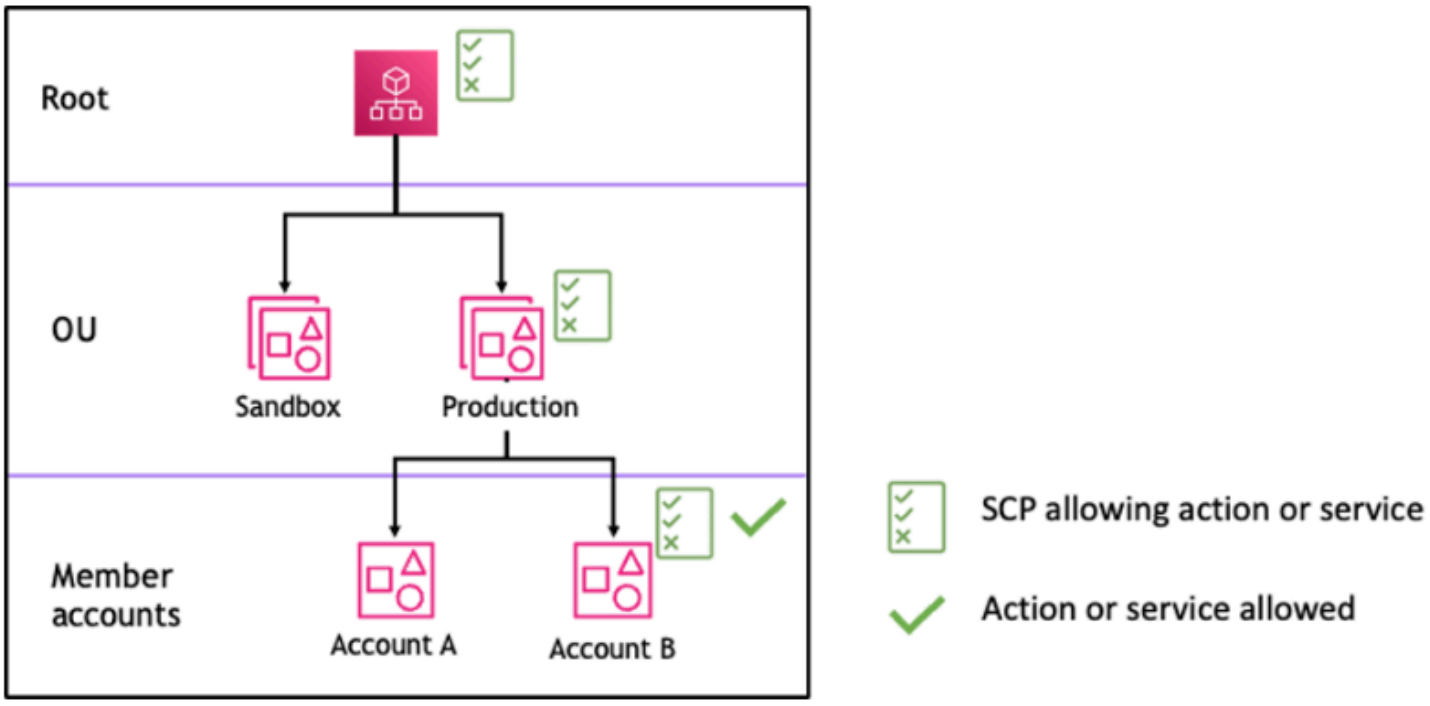


Figure 1: Example organization structure with an `Allow` statement attached at Root, Production OU and Account B

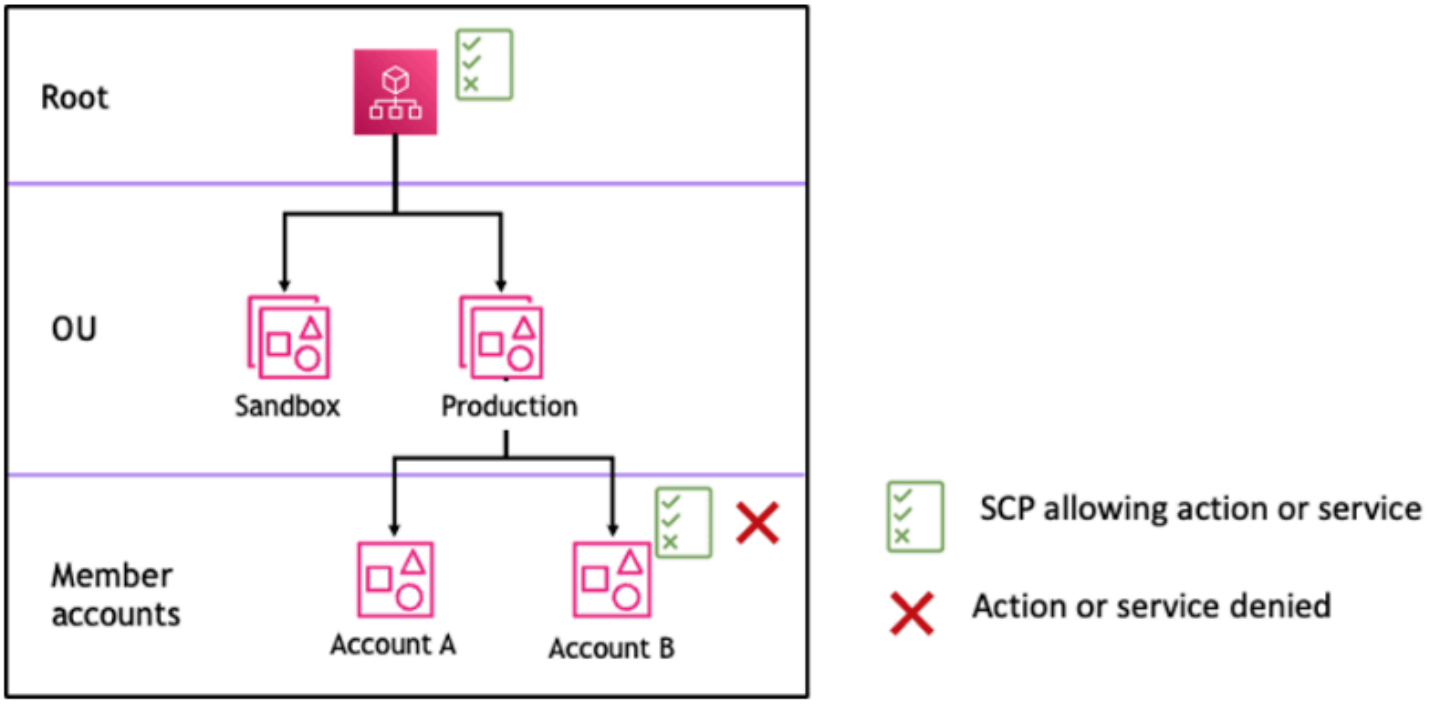


Figure 2: Example organization structure with an `Allow` statement missing at Production OU and its impact on Account B

How SCPs work with Deny

For a permission to be **denied** for a specific account, **any SCP** from the root through each OU in the direct path to the account (including the target account itself) can deny that permission.

For example, let's say there is an SCP attached to the Production OU that has an explicit `Deny` statement specified for a given service. There also happens to be another SCP attached to Root and to Account B that explicitly allows access to that same service, as shown in Figure 3. As a result, both Account A and Account B will be denied access to the service as a deny policy attached to any level in the organization is evaluated for all the OUs and member accounts underneath it.

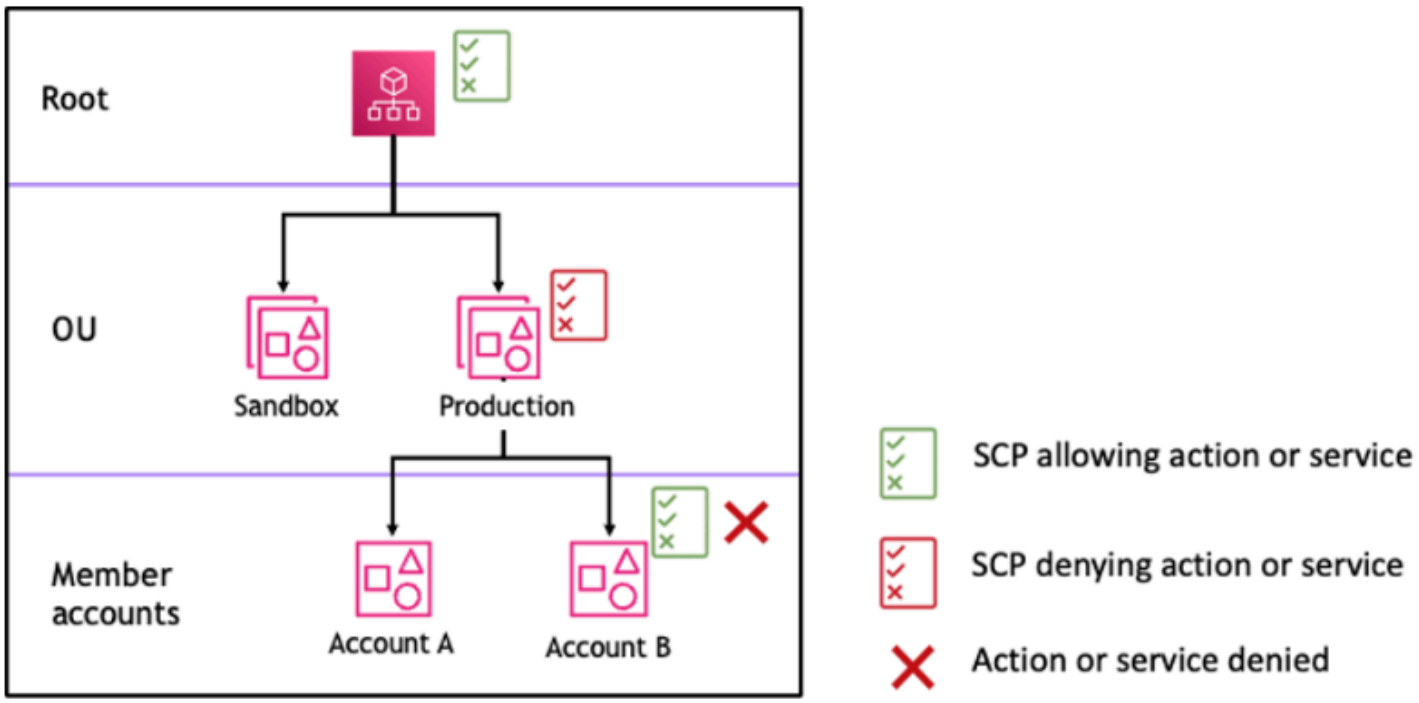


Figure 3: Example organization structure with an `Deny` statement attached at Production OU and its impact on Account B

Strategy for using SCPs

While writing SCPs you can make use of a combination of `Allow` and `Deny` statements to allow intended actions and services in your organization. `Deny` statements are a powerful way to implement restrictions that should be true for a broader part of your organization or OUs because when they are applied at the root or the OU level they affect all the accounts under it.

For example, you can implement a policy using `Deny` statements to Prevent member accounts from leaving the organization at the root level, which will be effective for all the accounts in the

organization. Deny statements also support condition element which can be helpful to create exceptions.

> **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*.

AWS Organizations attaches an AWS managed SCP named **FullAWSAccess** to every root, OU and account when it's created. This policy allows all services and actions. You can replace **FullAWSAccess** with a policy allowing only a set of services so that new AWS services are not allowed unless they are explicitly allowed by updating SCPs. For example, if your organization wants to only allow the use of a subset of services in your environment, you can use an `Allow` statement to only allow specific services.

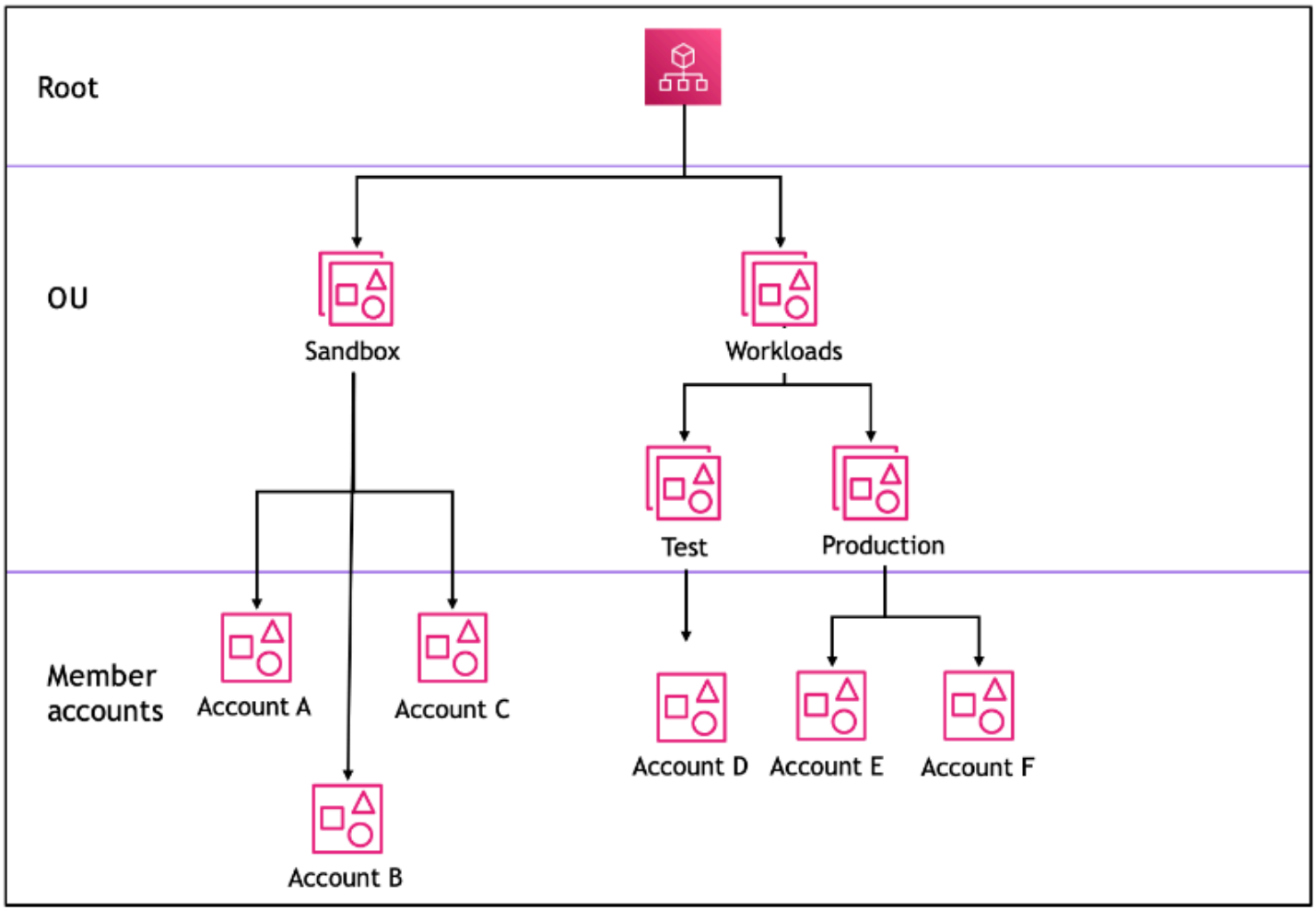
```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:*",
                "cloudwatch:*",
                "organizations:*"
            ],
            "Resource": "*"
        }
    ]
}
```

A policy combining the two statements might look like the following example, which prevents member accounts from leaving the organization and allows use of desired AWS services. The organization administrator can detach the **FullAWSAccess** policy and attach this one instead.

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

```
            "Effect": "Allow",
            "Action": [
                "ec2:*",
                "cloudwatch:*",
                "organizations:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action":"organizations:LeaveOrganization",
            "Resource": "*"
        }
    ]
}
```

Now, consider the following sample organization structure to understand how you can apply multiple SCPs at different levels in an organization.



The following table shows the effective policies in the Sandbox OU.

Scenario	SCP at Root	SCP at Sandbox OU	SCP at Account A	Resultant policy at Account A	Resultant policy at Account B and Account C
1	Full AWS access	Full AWS access + deny S3 access	Full AWS access + deny EC2 access	No S3, no EC2 access	No S3 access
2	Full AWS access	Allow EC2 access	Allow EC2 access	Allow EC2 access	Allow EC2 access
3	Deny S3 access	Allow S3 access	Full AWS access	No service access	No service access

The following table shows the effective policies in the Workloads OU.

Scenario	SCP at Root	SCP at Workloads OU	SCP at Test OU	Resultant policy at Account D	Resultant policies at Production OU, Account E and Account F
1	Full AWS access	Full AWS access	Full AWS access + deny EC2 access	No EC2 access	Full AWS access
2	Full AWS access	Full AWS access	Allow EC2 access	Allow EC2 access	Full AWS access
3	Deny S3 access	Full AWS access	Allow S3 access	No service access	No S3 access

SCP syntax

Service control policies (SCPs) use a similar syntax to that used by AWS Identity and Access Management (IAM) permission policies and resource-based policies (like Amazon S3 bucket policies). For more information about IAM policies and their syntax, see Overview of IAM Policies in the *IAM User Guide*.

An SCP is a plaintext file that is structured according to the rules of JSON. It uses the elements that are described in this topic.

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

For general information about SCPs, see Service control policies (SCPs).

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
Action	Specifies AWS service and actions that the SCP allows or denies.	`Allow`, `Deny`

Element	Purpose	Supported effects
Effect	Defines whether the SCP statement allows or denies access to the IAM users and roles in an account.	`Allow`, `Deny`
Statement	Serves as the container for policy elements. You can have multiple statements in SCPs.	`Allow`, `Deny`
Statement ID (Sid)	(Optional) Provides a friendly name for the statement.	`Allow`, `Deny`

Element	Purpose	Supported effects
Version	Specifies the language syntax rules to use for processing the policy.	`Allow`, `Deny`
Condition	Specifies conditions for when the statement is in effect.	`Deny`
NotAction	Specifies AWS service and actions that are exempt from the SCP. Used instead of the `Action` element.	`Deny`

Element	Purpose	Supported effects
Resource	Specifies the AWS resources that the SCP applies to.	Deny

The following sections provide more information and examples of how policy elements are used in SCPs.

Topics

- Action and NotAction elements
- Condition element
- Effect element
- Resource element
- Statement element
- Statement ID (Sid) element
- Version element
- Unsupported elements

`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-insensitive. Generally, they are all entered with each word starting with an uppercase letter and the rest lowercase. For example: `"s3:ListAllMyBuckets"`.

You also can use wildcard characters such as asterisk (*) or question mark (?) in an SCP:

- Use an asterisk (*) as a wildcard to match multiple actions that share part of a name. The value `"s3:*"` means all actions in the Amazon S3 service. The value `"ec2:Describe*"` matches only the EC2 actions that begin with "Describe".
- Use the question mark (?) wildcard to match a single character.

> **Note**
>
> In an SCP, the wildcard characters (*) and (?) in an `Action` or `NotAction` element can be used only by itself or at the end of the string. It can't appear at the beginning or middle of the string. Therefore, `"servicename:action*"` is valid, but `"servicename:*action"` and `"servicename:some*action"` are both invalid in SCPs.

For a list of all the services and the actions that they support in both AWS Organizations SCPs and IAM permission policies, see Actions, Resources, and Condition Keys for AWS Services in the *IAM User Guide*.

For more information, see IAM JSON Policy Elements: Action and IAM JSON Policy Elements: NotAction in the *IAM User Guide*.

Example of `Action` element

The following example shows an SCP with a statement that permits account administrators to delegate describe, start, stop, and terminate permissions for EC2 instances in the account. This is an example of an allow list, 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 to services that you don't want used in attached accounts. It assumes that the default `"Allow *"` SCPs are still attached to all OUs and the root. This example policy prevents the account administrators in attached accounts from delegating any permissions for the IAM, Amazon EC2, and Amazon RDS services. Any action from other services can be delegated as long as there isn't another attached policy that denies them.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Deny",
        "Action": [ "iam:*", "ec2:*", "rds:*" ],
        "Resource": "*"
    }
}
```

Example of `NotAction` element

The following example shows how you can use a `NotAction` element to exclude AWS services from the effect of the policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "LimitActionsInRegion",
      "Effect": "Deny",
      "NotAction": "iam:*",
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:RequestedRegion": "us-west-1"
         }
       }
```

```
      }
   ]
}
```

With this statement, affected accounts are limited to taking actions in the specified AWS Region, except when using IAM actions.

`Condition` element

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

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

This SCP denies access to any operations outside the `eu-central-1` and `eu-west-1` Regions, except for actions in the listed services.

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

`Effect` element

Each statement must contain one `Effect` element. The value can be either `Allow` or `Deny`. It affects any actions listed in the same statement.

For more information, see IAM JSON Policy Elements: Effect in the *IAM User Guide*.

`"Effect": "Allow"`

The following example shows an SCP with a statement that contains an `Effect` element with a value of `Allow` that permits account users to perform actions for the Amazon S3 service. This example is useful in an organization that uses the allow list strategy (where the default `FullAWSAccess` policies are all detached so that permissions are implicitly denied by default). The result is that the statement allows the Amazon S3 permissions for any attached accounts:

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

Even though this statement uses the same `Allow` value keyword as an IAM permission policy, in an SCP it doesn't actually grant a user permission to do anything. Instead, SCPs act as *filters* that specify the maximum permissions for the accounts in an organization, organizational unit (OU), or account. In the preceding example, even if a user in the account had the `AdministratorAccess` managed policy attached, this SCP limits ***all*** users in affected accounts to only Amazon S3 actions.

`"Effect": "Deny"`

In a statement where the `Effect` element has a value of `Deny`, you can also restrict access to specific resources or define conditions for when SCPs are in effect.

The following shows an example of how to use a condition key in a deny statement.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Deny",
        "Action": "ec2:RunInstances",
```

```
        "Resource": "arn:aws:ec2:*:*:instance/*",
        "Condition": {
            "StringNotEquals": {
                "ec2:InstanceType": "t2.micro"
            }
        }
    }
}
```

This statement in an SCP sets a guardrail to prevent affected accounts (where the SCP is attached to the account itself or to the organization root or OU that contains the account), from launching Amazon EC2 instances if the Amazon EC2 instance isn't set to `t2.micro`. Even if an IAM policy that allows this action is attached to the account, the guardrail created by the SCP prevents it.

`Resource` element

In statements where the `Effect` element has a value of `Allow`, you can specify only "*" in the `Resource` element of an SCP. You can't specify individual resource Amazon Resource Names (ARNs).

You also can use wildcard characters such as asterisk (*) or question mark (?) in the resource element:

- Use an asterisk (*) as a wildcard to match multiple actions that share part of a name.
- Use the question mark (?) wildcard to match a single character.

In statements where the `Effect` element has a value of `Deny`, you *can* specify individual ARNs, as shown in the following example.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAccessToAdminRole",
      "Effect": "Deny",
      "Action": [
        "iam:AttachRolePolicy",
        "iam:DeleteRole",
        "iam:DeleteRolePermissionsBoundary",
        "iam:DeleteRolePolicy",
        "iam:DetachRolePolicy",
```

```
        "iam:PutRolePermissionsBoundary",
        "iam:PutRolePolicy",
        "iam:UpdateAssumeRolePolicy",
        "iam:UpdateRole",
        "iam:UpdateRoleDescription"
      ],
      "Resource": [
        "arn:aws:iam::*:role/role-to-deny"
      ]
    }
  ]
}
```

This SCP restricts IAM users and roles in affected accounts from making changes to a common administrative IAM role created in all accounts in your organization.

For more information, see IAM JSON Policy Elements: Resource in the *IAM User Guide*.

`Statement` element

An SCP consists of one or more `Statement` elements. You can have only one `Statement` keyword in a policy, but the value can be a JSON array of statements (surrounded by [ ] characters).

The following example shows a single statement that consists of single `Effect`, `Action`, and `Resource` elements.

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

The following example includes two statements as an array list inside one `Statement` element. The first statement allows all actions, while the second denies any EC2 actions. The result is that an administrator in the account can delegate any permission *except* those from Amazon Elastic Compute Cloud (Amazon EC2).

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "*",
            "Resource": "*"
```

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

For more information, see IAM JSON Policy Elements: Statement in the *IAM User Guide*.

Statement ID (`Sid`) element

The `Sid` is an optional identifier that you provide for the policy statement. You can assign a `Sid` value to each statement in a statement array. The following example SCP shows a sample `Sid` statement.

```
{
    "Statement": {
        "Sid": "AllowsAllActions",
        "Effect": "Allow",
        "Action": "*",
        "Resource": "*"
    }
}
```

For more information, see IAM JSON Policy Elements: Id in the *IAM User Guide*.

`Version` element

Every SCP must include a `Version` element with the value `"2012-10-17"`. This is the same version value as the most recent version of IAM permission policies.

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

For more information, see IAM JSON Policy Elements: Version in the *IAM User Guide*.

Unsupported elements

The following elements aren't supported in SCPs:

- `Principal`
- `NotPrincipal`

- `NotResource`

Service control policy examples

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

Before using these examples

Before you use these example SCPs in your organization, do the following:

- Carefully review and customize the SCPs for your unique requirements.
- Thoroughly test the SCPs in your environment with the AWS services that you use.

  The example policies in this section demonstrate the implementation and use of SCPs. They're ***not*** intended to be interpreted as official AWS recommendations or best practices to be implemented exactly as shown. It is your responsibility to carefully test any deny-based policies for its suitability to solve the business requirements of your environment. Deny-based service control policies can unintentionally limit or block your use of AWS services unless you add the necessary exceptions to the policy. For an example of such an exception, see the first example that exempts global services from the rules that block access to unwanted AWS Regions.
- Remember that an SCP affects every user and role, including the root user, in every account that it's attached to.
- Remember that an SCP does not affect service-linked roles. Service-linked roles enable other AWS services to integrate with AWS Organizations and can't be restricted by SCPs.

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 strategy. Deny list policies must be attached along with other policies that allow the approved actions in the affected accounts.

For example, the default `FullAWSAccess` policy permits the use of all services in an account. This policy is attached by default to the root, all organizational units (OUs), and all accounts. It doesn't actually grant the permissions; no SCP does. Instead, it enables administrators in that account to delegate access to those actions by attaching standard AWS Identity and Access Management (IAM) permissions policies to users, roles, or groups in the account. Each of these deny list policies then overrides any policy by blocking access to the specified services or actions.

Examples

- General examples
  - Deny access to AWS based on the requested AWS Region
  - Prevent IAM users and roles from making certain changes
  - Prevent IAM users and roles from making specified changes, with an exception for a specified admin role
  - Require MFA to perform an API operation
  - Block service access for the root user
  - Prevent member accounts from leaving the organization
- Example SCPs for Amazon Q Developer in chat applications
  - Deny all IAM operation
  - Deny S3 bucket put requests from a specified Slack channel
- Example SCPs for Amazon CloudWatch
  - Prevent users from disabling CloudWatch or altering its configuration
- Example SCPs for AWS Config
  - Prevent users from disabling AWS Config or changing its rules
- Example SCPs for Amazon Elastic Compute Cloud (Amazon EC2)
  - Require Amazon EC2 instances to use a specific type
  - Prevent launching EC2 instances without IMDSv2
  - Prevent disabling of default Amazon EBS encryption
  - Prevent creating and attaching non-gp3 volumes
- Example SCPs for Amazon GuardDuty
  - Prevent users from disabling GuardDuty or modifying its configuration
- Example SCPs for AWS Resource Access Manager
  - Preventing external sharing

- Allowing specific accounts to share only specified resource types
- Prevent sharing with organizations or organizational units (OUs)
- Allow sharing with only specified IAM users and roles
- Example SCPs for Amazon Application Recovery Controller (ARC)
  - Prevent users from updating ARC routing control states
- Example SCPs for Amazon S3
  - Prevent Amazon S3 unencrypted object uploads
- Example SCPs for tagging resources
  - Require a tag on specified created resources
  - Prevent tags from being modified except by authorized principals
- Example SCPs for Amazon Virtual Private Cloud (Amazon VPC)
  - Prevent users from deleting Amazon VPC flow logs
  - Prevent any VPC that doesn't already have internet access from getting it

General examples

Deny access to AWS based on the requested AWS Region

This SCP denies access to any operations outside of the specified Regions. Replace `eu-central-1` and `eu-west-1` with the AWS Regions you want to use. It provides exemptions for operations in approved global services. This example also shows how to exempt requests made by either of two specified administrator roles.

> **Note**
>
> To use the Region deny SCP with AWS Control Tower, see Deny access to AWS based on the requested AWS Region in the *AWS Control Tower Controls Reference Guide*.

This policy uses the `Deny` effect to deny access to all requests for operations that don't target one of the two approved regions (`eu-central-1` and `eu-west-1`). The NotAction element enables you to list services whose operations (or individual operations) are exempted from this restriction. Because global services have endpoints that are physically hosted by the `us-east-1` Region , they must be exempted in this way. With an SCP structured this way, requests made to global services in

the `us-east-1` Region are allowed if the requested service is included in the `NotAction` element. Any other requests to services in the `us-east-1` Region are denied by this example policy.

Note

This example might not include all of the latest global AWS services or operations. Replace the list of services and operations with the global services used by accounts in your organization.

Tip

You can view the service last accessed data in the IAM console to determine what global services your organization uses. The **Access Advisor** tab on the details page for an IAM user, group, or role displays the AWS services that have been used by that entity, sorted by most recent access.

Considerations

- AWS KMS and AWS Certificate Manager support Regional endpoints. However, if you want to use them with a global service such as Amazon CloudFront you must include them in the global service exclusion list in the following example SCP. A global service like Amazon CloudFront typically requires access to AWS KMS and ACM in the same region, which for a global service is the US East (N. Virginia) Region (`us-east-1`).
- By default, AWS STS is a global service and must be included in the global service exclusion list. However, you can enable AWS STS to use Region endpoints instead of a single global endpoint. If you do this, you can remove STS from the global service exemption list in the following example SCP. For more information see Managing AWS STS in an AWS Region.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyAllOutsideEU",
            "Effect": "Deny",
```

```
            "NotAction": [
                "a4b:*",
                "acm:*",
                "aws-marketplace-management:*",
                "aws-marketplace:*",
                "aws-portal:*",
                "budgets:*",
                "ce:*",
                "chime:*",
                "cloudfront:*",
                "config:*",
                "cur:*",
                "directconnect:*",
                "ec2:DescribeRegions",
                "ec2:DescribeTransitGateways",
                "ec2:DescribeVpnGateways",
                "fms:*",
                "globalaccelerator:*",
                "health:*",
                "iam:*",
                "importexport:*",
                "kms:*",
                "mobileanalytics:*",
                "networkmanager:*",
                "organizations:*",
                "pricing:*",
                "route53:*",
                "route53domains:*",
                "route53-recovery-cluster:*",
                "route53-recovery-control-config:*",
                "route53-recovery-readiness:*",
                "s3:GetAccountPublic*",
                "s3:ListAllMyBuckets",
                "s3:ListMultiRegionAccessPoints",
                "s3:PutAccountPublic*",
                "shield:*",
                "sts:*",
                "support:*",
                "trustedadvisor:*",
                "waf-regional:*",
                "waf:*",
                "wafv2:*",
                "wellarchitected:*"
            ],
```

```
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestedRegion": [
                        "eu-central-1",
                        "eu-west-1"
                    ]
                },
                "ArnNotLike": {
                    "aws:PrincipalARN": [
                        "arn:aws:iam::*:role/Role1AllowedToBypassThisSCP",
                        "arn:aws:iam::*:role/Role2AllowedToBypassThisSCP"
                    ]
                }
            }
        }
    ]
}
```

Prevent IAM users and roles from making certain changes

This SCP restricts IAM users and roles from making changes to the specified IAM role that you created in all accounts in your organization.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAccessToASpecificRole",
      "Effect": "Deny",
      "Action": [
        "iam:AttachRolePolicy",
        "iam:DeleteRole",
        "iam:DeleteRolePermissionsBoundary",
        "iam:DeleteRolePolicy",
        "iam:DetachRolePolicy",
        "iam:PutRolePermissionsBoundary",
        "iam:PutRolePolicy",
        "iam:UpdateAssumeRolePolicy",
        "iam:UpdateRole",
        "iam:UpdateRoleDescription"
      ],
      "Resource": [
```

```
        "arn:aws:iam::*:role/name-of-role-to-deny"
      ]
    }
  ]
}
```

Prevent IAM users and roles from making specified changes, with an exception for a specified admin role

This SCP builds on the previous example to make an exception for administrators. It prevents IAM users and roles in affected accounts from making changes to a common administrative IAM role created in all accounts in your organization *except* for administrators using a specified role.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAccessWithException",
      "Effect": "Deny",
      "Action": [
        "iam:AttachRolePolicy",
        "iam:DeleteRole",
        "iam:DeleteRolePermissionsBoundary",
        "iam:DeleteRolePolicy",
        "iam:DetachRolePolicy",
        "iam:PutRolePermissionsBoundary",
        "iam:PutRolePolicy",
        "iam:UpdateAssumeRolePolicy",
        "iam:UpdateRole",
        "iam:UpdateRoleDescription"
      ],
      "Resource": [
        "arn:aws:iam::*:role/name-of-role-to-deny"
      ],
      "Condition": {
        "ArnNotLike": {
          "aws:PrincipalARN":"arn:aws:iam::*:role/name-of-admin-role-to-allow"
        }
      }
    }
  ]
}
```

Require MFA to perform an API operation

Use an SCP like the following to require that multi-factor authentication (MFA) is enabled before an IAM user or role can perform an action. In this example, the action is to stop an Amazon EC2 instance.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyStopAndTerminateWhenMFAIsNotPresent",
      "Effect": "Deny",
      "Action": [
        "ec2:StopInstances",
        "ec2:TerminateInstances"
      ],
      "Resource": "*",
      "Condition": {"BoolIfExists": {"aws:MultiFactorAuthPresent": false}}
    }
  ]
}
```

Block service access for the root user

The following policy restricts all access to the specified actions for the root user in a member account. If you want to prevent your accounts from using root credentials in specific ways, add your own actions to this policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RestrictEC2ForRoot",
      "Effect": "Deny",
      "Action": [
        "ec2:*"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringLike": {
```

```
          "aws:PrincipalArn": [
            "arn:aws:iam::*:root"
          ]
        }
      }
    }
  ]
}
```

Prevent member accounts from leaving the organization

The following policy blocks use of the `LeaveOrganization` API operation so that administrators of member accounts can't remove their accounts from the organization.

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

Example SCPs for Amazon Q Developer in chat applications

Examples in this category

- Deny all IAM operation
- Deny S3 bucket put requests from a specified Slack channel

Deny all IAM operation

The following SCP denies all IAM operations invoked through all Amazon Q Developer in chat applications configurations.

```
{
    "Effect": "Deny",
```

```
    "Action": "iam:*",
    "Resource": "*",
    "Condition": {
        "ArnLike": {
            "aws:ChatbotSourceArn": "arn:aws:chatbot::*:*"
        }
    }
}
```

Deny S3 bucket put requests from a specified Slack channel

The following policy denies S3 put requests on the specified bucket for all requests originating from a Slack channel.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ExampleS3Deny",
            "Effect": "Deny",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringLike": {
                      "aws:ChatbotSourceArn": "arn:aws:chatbot::*:chat-configuration/
slack-channel/*"
                }
            }
        }
    ]
}
```

Example SCPs for Amazon CloudWatch

Examples in this category

- Prevent users from disabling CloudWatch or altering its configuration

Prevent users from disabling CloudWatch or altering its configuration

A lower-level CloudWatch operator needs to monitor dashboards and alarms. However, the operator must not be able to delete or change any dashboard or alarm that senior people might

put into place. This SCP prevents users or roles in any affected account from running any of the CloudWatch commands that could delete or change your dashboards or alarms.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "cloudwatch:DeleteAlarms",
        "cloudwatch:DeleteDashboards",
        "cloudwatch:DisableAlarmActions",
        "cloudwatch:PutDashboard",
        "cloudwatch:PutMetricAlarm",
        "cloudwatch:SetAlarmState"
      ],
      "Resource": "*"
    }
  ]
}
```

Example SCPs for AWS Config

Examples in this category

- Prevent users from disabling AWS Config or changing its rules

Prevent users from disabling AWS Config or changing its rules

This SCP prevents users or roles in any affected account from running AWS Config operations that could disable AWS Config or alter its rules or triggers.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "config:DeleteConfigRule",
        "config:DeleteConfigurationRecorder",
        "config:DeleteDeliveryChannel",
        "config:StopConfigurationRecorder"
      ],
```

```
      "Resource": "*"
    }
  ]
}
```

Example SCPs for Amazon Elastic Compute Cloud (Amazon EC2)

Examples in this category

- Require Amazon EC2 instances to use a specific type
- Prevent launching EC2 instances without IMDSv2
- Prevent disabling of default Amazon EBS encryption
- Prevent creating and attaching non-gp3 volumes

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"
        }
      }
    }
  ]
}
```

Prevent launching EC2 instances without IMDSv2

The following policy restricts all users from launching EC2 instances without IMDSv2.

```
[
```

```
   {
      "Effect":"Deny",
      "Action":"ec2:RunInstances",
      "Resource":"arn:aws:ec2:*:*:instance/*",
      "Condition":{
         "StringNotEquals":{
            "ec2:MetadataHttpTokens":"required"
         }
      }
   },
   {
      "Effect":"Deny",
      "Action":"ec2:RunInstances",
      "Resource":"arn:aws:ec2:*:*:instance/*",
      "Condition":{
         "NumericGreaterThan":{
            "ec2:MetadataHttpPutResponseHopLimit":"2"
         }
      }
   },
   {
      "Effect":"Deny",
      "Action":"*",
      "Resource":"*",
      "Condition":{
         "NumericLessThan":{
            "ec2:RoleDelivery":"2.0"
         }
      }
   },
   {
      "Effect":"Deny",
      "Action":"ec2:ModifyInstanceMetadataOptions",
      "Resource":"*"
   }
]
```

The following policy restricts all users from launching EC2 instances without IMDSv2 but allows specific IAM identities to modify instance metadata options.

```
[
  {
    "Effect": "Deny",
```

```
    "Action": "ec2:RunInstances",
    "Resource": "arn:aws:ec2:*:*:instance/*",
    "Condition": {
      "StringNotEquals": {
        "ec2:MetadataHttpTokens": "required"
      }
    }
  },
  {
    "Effect": "Deny",
    "Action": "ec2:RunInstances",
    "Resource": "arn:aws:ec2:*:*:instance/*",
    "Condition": {
      "NumericGreaterThan": {
        "ec2:MetadataHttpPutResponseHopLimit": "2"
      }
    }
  },
  {
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
      "NumericLessThan": {
        "ec2:RoleDelivery": "2.0"
      }
    }
  },
  {
    "Effect": "Deny",
    "Action": "ec2:ModifyInstanceMetadataOptions",
    "Resource": "*",
    "Condition": {
      "ArnNotLike": {
        "aws:PrincipalARN": [
          "arn:aws:iam::{ACCOUNT_ID}:{RESOURCE_TYPE}/{RESOURCE_NAME}"
        ]
      }
    }
  }
]
```

Prevent disabling of default Amazon EBS encryption

The following policy restricts all users from disabling the default Amazon EBS Encryption.

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

Prevent creating and attaching non-gp3 volumes

The following policy restricts all users from creating or attaching any Amazon EBS volumes that are not of the gp3 volume type. For more information, see Amazon EBS volume types.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyCreationAndAttachmentOfNonGP3Volumes",
      "Effect": "Deny",
      "Action": [
        "ec2:AttachVolume",
        "ec2:CreateVolume",
        "ec2:RunInstances"
      ],
      "Resource": "arn:aws:ec2:*:*:volume/*",
      "Condition": {
        "StringNotEquals": {
          "ec2:VolumeType": "gp3"
        }
      }
    }
  ]
}
```

This can help enforce a standardized volume configuration across an organization.

Volume type modifications are not prevented

You cannot restrict the action of modifying an existing gp3 volume to an Amazon EBS volume of another type using SCPs. For example, this SCP would not prevent you from modifying an existing gp3 volume to a gp2 volume. This is because the condition key `ec2:VolumeType` checks the volume type before it is modified.

Example SCPs for Amazon GuardDuty

Examples in this category

- Prevent users from disabling GuardDuty or modifying its configuration

Prevent users from disabling GuardDuty or modifying its configuration

This SCP prevents users or roles in any affected account from disabling GuardDuty or altering its configuration, either directly as a command or through the console. It effectively enables read-only access to the GuardDuty information and resources.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "guardduty:AcceptInvitation",
                "guardduty:ArchiveFindings",
                "guardduty:CreateDetector",
                "guardduty:CreateFilter",
                "guardduty:CreateIPSet",
                "guardduty:CreateMembers",
                "guardduty:CreatePublishingDestination",
                "guardduty:CreateSampleFindings",
                "guardduty:CreateThreatIntelSet",
                "guardduty:DeclineInvitations",
                "guardduty:DeleteDetector",
                "guardduty:DeleteFilter",
                "guardduty:DeleteInvitations",
                "guardduty:DeleteIPSet",
                "guardduty:DeleteMembers",
                "guardduty:DeletePublishingDestination",
```

```
                "guardduty:DeleteThreatIntelSet",
                "guardduty:DisassociateFromMasterAccount",
                "guardduty:DisassociateMembers",
                "guardduty:InviteMembers",
                "guardduty:StartMonitoringMembers",
                "guardduty:StopMonitoringMembers",
                "guardduty:TagResource",
                "guardduty:UnarchiveFindings",
                "guardduty:UntagResource",
                "guardduty:UpdateDetector",
                "guardduty:UpdateFilter",
                "guardduty:UpdateFindingsFeedback",
                "guardduty:UpdateIPSet",
                "guardduty:UpdatePublishingDestination",
                "guardduty:UpdateThreatIntelSet"
            ],
            "Resource": "*"
        }
    ]
}
```

Example SCPs for AWS Resource Access Manager

Examples in this category

- Preventing external sharing
- Allowing specific accounts to share only specified resource types
- Prevent sharing with organizations or organizational units (OUs)
- Allow sharing with only specified IAM users and roles

Preventing external sharing

The following example SCP prevents users from creating resource shares that allow sharing with IAM users and roles that aren't part of the organization.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
```

```
                "ram:CreateResourceShare",
                "ram:UpdateResourceShare"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "ram:RequestedAllowsExternalPrincipals": "true"
                }
            }
        }
    ]
}
```

Allowing specific accounts to share only specified resource types

The following SCP allows accounts 111111111111 and 222222222222 to create resource shares that share prefix lists, and to associate prefix lists with existing resource shares.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "OnlyNamedAccountsCanSharePrefixLists",
            "Effect": "Allow",
            "Action": [
                "ram:AssociateResourceShare",
                "ram:CreateResourceShare"
            ],
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:PrincipalAccount": [
                        "111111111111",
                        "222222222222"
                    ]
                },
                "StringEquals": {
                    "ram:RequestedResourceType": "ec2:PrefixList"
                }
            }
        }
    ]
}
```

Prevent sharing with organizations or organizational units (OUs)

The following SCP prevents users from creating resource shares that share resources with an organization or OUs.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "ram:CreateResourceShare",
                "ram:AssociateResourceShare"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringLike": {
                    "ram:Principal": [
                        "arn:aws:organizations::*:organization/*",
                        "arn:aws:organizations::*:ou/*"
                    ]
                }
            }
        }
    ]
}
```

Allow sharing with only specified IAM users and roles

The following example SCP allows users to share resources with *only* organization `o-12345abcdef`, organizational unit `ou-98765fedcba`, and account 111111111111.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ram:AssociateResourceShare",
                "ram:CreateResourceShare"
            ],
            "Resource": "*",
            "Condition": {
```

```
                "ForAnyValue:StringEquals": {
                    "ram:Principal": [
                        "arn:aws:organizations::123456789012:organization/
o-12345abcdef",
                        "arn:aws:organizations::123456789012:ou/o-12345abcdef/
ou-98765fedcba",
                        "111111111111"
                    ]
                }
            }
        }
    ]
}
```

Example SCPs for Amazon Application Recovery Controller (ARC)

Examples in this category

- Prevent users from updating ARC routing control states

Prevent users from updating ARC routing control states

A lower-level ARC operator needs to monitor dashboards and view ARC information. However, the operator must not be able to update routing controls to fail over the application from one AWS Region to another, as a senior operator might be allowed to. This SCP prevents users or roles in any affected account from running ARC operations that update ARC routing controls.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyAll",
            "Effect": "Deny",
            "Action": [
                "route53-recovery-cluster:UpdateRoutingControlState",
                "route53-recovery-cluster:UpdateRoutingControlStates"
            ],
            "Resource": "*",
            "Condition": {
                "ArnNotLike": {
                    "aws:PrincipalARN": [
                        "arn:aws:iam::*:role/Role1AllowedToBypassThisSCP",
                        "arn:aws:iam::*:role/Role2AllowedToBypassThisSCP"
```

```
                    ]
                }
            }
        }
    ]
}
```

Example SCPs for Amazon S3

> **Note**
>
> Amazon Simple Storage Service (Amazon S3) automatically applies server-side encryption (SSE-S3) for each new object, unless you specify a different encryption option. For more information, see Amazon S3 now automatically encrypts all new objects in the *Amazon S3 User Guide*.

Examples in this category

- Prevent Amazon S3 unencrypted object uploads

Prevent Amazon S3 unencrypted object uploads

The following policy restricts all users from uploading unencrypted objects to S3 buckets.

```
{
  "Effect": "Deny",
  "Action": "s3:PutObject",
  "Resource": "*",
  "Condition": {
    "Null": {
      "s3:x-amz-server-side-encryption": "true"
    }
  }
}
```

The following policy restricts all users from uploading unencrypted objects to S3 buckets and also enforces a specified encryption type (either AES256 or aws:kms) for object upload in their buckets.

```
[
  {
```

```
    "Effect": "Deny",
    "Action": "s3:PutObject",
    "Resource": "*",
    "Condition": {
      "Null": {
        "s3:x-amz-server-side-encryption": "true"
      }
    }
  },
  {
    "Effect": "Deny",
    "Action": "s3:PutObject",
    "Resource": "*",
    "Condition": {
      "StringNotEquals": {
        "s3:x-amz-server-side-encryption": "AES256"
      }
    }
  }
]
```

Example SCPs for tagging resources

Examples in this category

- Require a tag on specified created resources
- Prevent tags from being modified except by authorized principals

Require a tag on specified created resources

The following SCP prevents IAM users and roles in the affected accounts from creating certain resource types if the request doesn't include the specified tags.

> **Important**
>
> Remember to test Deny-based policies with the services you use in your environment. The following example is a simple block of creating untagged secrets or running untagged Amazon EC2 instances, and doesn't include any exceptions.
> The following example policy is not compatible with AWS CloudFormation as written, because that service creates a secret and then tags it as two separate steps. This example policy effectively blocks AWS CloudFormation from creating a secret as part of a stack,

because such an action would result, however briefly, in a secret that is not tagged as required.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyCreateSecretWithNoProjectTag",
      "Effect": "Deny",
      "Action": "secretsmanager:CreateSecret",
      "Resource": "*",
      "Condition": {
        "Null": {
          "aws:RequestTag/Project": "true"
        }
      }
    },
    {
      "Sid": "DenyRunInstanceWithNoProjectTag",
      "Effect": "Deny",
      "Action": "ec2:RunInstances",
      "Resource": [
        "arn:aws:ec2:*:*:instance/*"
      ],
      "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/*"
      ],
      "Condition": {
        "Null": {
          "aws:RequestTag/CostCenter": "true"
        }
      }
    }
  ]
}
```

For a list of all the services and the actions that they support in both AWS Organizations SCPs and IAM permission policies, see Actions, Resources, and Condition Keys for AWS services in the *IAM User Guide*.

Prevent tags from being modified except by authorized principals

The following SCP shows how a policy can allow only authorized principals to modify the tags attached to your resources. This is an important part of using attribute-based access control (ABAC) as part of your AWS cloud security strategy. The policy allows a caller to modify the tags on only those resources where the authorization tag (in this example, `access-project`) exactly matches the same authorization tag attached to the user or role making the request. The policy also prevents the authorized user from changing the *value* of the tag that is used for authorization. The calling principal must have the authorization tag to make any changes at all.

This policy only blocks unauthorized users from changing tags. An authorized user who isn't blocked by this policy must still have a separate IAM policy that explicitly grants the `Allow` permission on the relevant tagging APIs. As an example, if your user has an administrator policy with `Allow */*` (allow all services and all operations), then the combination results in the administrator user being allowed to change *only* those tags that have an authorization tag value that matches the authorization tag value attached to the user's principal. This is because the explicit `Deny` in the this policy overrides the explicit `Allow` in the administrator policy.

Important

This is not a complete policy solution and should not be used as shown here. This example is intended only to illustrate part of an ABAC strategy and needs to be customized and tested for production environments.
For the complete policy with a detailed analysis of how it works, see Securing resource tags used for authorization using a service control policy in AWS Organizations
Remember to test Deny-based policies with the services you use in your environment.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyModifyTagsIfResAuthzTagAndPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ec2:CreateTags",
                "ec2:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "ec2:ResourceTag/access-project": "${aws:PrincipalTag/access-
project}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-admins/iam-
admin"
                },
                "Null": {
                    "ec2:ResourceTag/access-project": false
                }
            }
        },
        {
            "Sid": "DenyModifyResAuthzTagIfPrinTagDontMatch",
            "Effect": "Deny",
            "Action": [
                "ec2:CreateTags",
                "ec2:DeleteTags"
```

```
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestTag/access-project": "${aws:PrincipalTag/access-
project}",
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-admins/iam-
admin"
                },
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": [
                        "access-project"
                    ]
                }
            }
        },
        {
            "Sid": "DenyModifyTagsIfPrinTagNotExists",
            "Effect": "Deny",
            "Action": [
                "ec2:CreateTags",
                "ec2:DeleteTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:PrincipalArn": "arn:aws:iam::123456789012:role/org-admins/iam-
admin"
                },
                "Null": {
                    "aws:PrincipalTag/access-project": true
                }
            }
        }
    ]
}
```

Example SCPs for Amazon Virtual Private Cloud (Amazon VPC)

Examples in this category

- Prevent users from deleting Amazon VPC flow logs
- Prevent any VPC that doesn't already have internet access from getting it

Prevent users from deleting Amazon VPC flow logs

This SCP prevents users or roles in any affected account from deleting Amazon Elastic Compute Cloud (Amazon EC2) flow logs or CloudWatch log groups or log streams.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "ec2:DeleteFlowLogs",
        "logs:DeleteLogGroup",
        "logs:DeleteLogStream"
      ],
      "Resource": "*"
    }
  ]
 }
```

Prevent any VPC that doesn't already have internet access from getting it

This SCP prevents users or roles in any affected account from changing the configuration of your Amazon EC2 virtual private clouds (VPCs) to grant them direct access to the internet. It doesn't block existing direct access or any access that routes through your on-premises network environment.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "ec2:AttachInternetGateway",
        "ec2:CreateInternetGateway",
```

```
        "ec2:CreateEgressOnlyInternetGateway",
        "ec2:CreateVpcPeeringConnection",
        "ec2:AcceptVpcPeeringConnection",
        "globalaccelerator:Create*",
        "globalaccelerator:Update*"
      ],
      "Resource": "*"
    }
  ]
}
```

Troubleshooting service control policies (SCPs) with AWS Organizations

Use the information here to help you diagnose and fix common errors found in service control policies (SCPs).

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.

Common policy errors

- More than one policy object
- More than one statement element
- Policy document exceeds maximum size

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 characters. 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.

> **Note**
>
> If you save the policy by using the AWS Management Console, extra white space between JSON elements and outside of quotation marks is removed and not counted. If you save the policy using an SDK operation or the AWS CLI, then the policy is saved exactly as you provided and no automatic removal of characters occurs.

Resource control policies (RCPs)

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

RCPs alone are not sufficient in granting permissions to the resources in your organization. No permissions are granted by an RCP. An RCP defines a permissions guardrail, or sets limits, on the actions that identities can take on resources in your organizations. The administrator must still attach identity-based policies to IAM users or roles, or resource-based policies to resources in your accounts to actually grant permissions. For more information, see Identity-based policies and resource-based policies in the *IAM User Guide*.

The effective permissions are the logical intersection between what is allowed by the RCPs and service control policies (SCPs) and what is allowed by the identity-based and resource-based policies.

RCPs don't affect resources in the management account

RCPs don't affect resources in the management account. They only affect resources in the member accounts within your organization. This also means that RCPs apply to member accounts that are designated as delegated administrators.

Topics on this page

- List of AWS services that support RCPs
- Testing effects of RCPs
- Maximum size of RCPs
- Attaching RCPs to different levels in the organization
- RCP effects on permissions
- Resources and entities not restricted by RCPs
- RCP evaluation
- RCP syntax
- Resource control policy examples

List of AWS services that support RCPs

RCPs apply to actions for the following AWS services:

- Amazon S3
- AWS Security Token Service
- AWS Key Management Service
- Amazon SQS
- AWS Secrets Manager

Testing effects of RCPs

AWS strongly recommends that you don't attach RCPs to the root of your organization without thoroughly testing the impact that the policy has on resources in your accounts. You can begin by attaching RCPs to individual test accounts, moving them up to OUs lower in the hierarchy, and then

working your way up through the organization structure as needed. One way to determine impact is to review AWS CloudTrail logs for Access Denied errors.

Maximum size of RCPs

All characters in your RCP count against its maximum size. The examples in this guide show the RCPs 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 RCP. It automatically removes extra white space.

Attaching RCPs to different levels in the organization

You can attach RCPs directly to individual accounts, OUs, or the organization root. For a detailed explanation of how RCPs work, see RCP evaluation.

RCP effects on permissions

RCPs are a type of AWS Identity and Access Management (IAM) policy. They are most closely related to resource-based policies. However, an RCP never grants permissions. Instead, RCPs are access controls that specify the maximum available permissions for resources in your organization. For more information, see Policy evaluation logic in the *IAM User Guide*.

- RCPs apply to resources for a subset of AWS services. For more information, see List of AWS services that support RCPs.
- RCPs ***affect only resources*** that are managed by accounts that are part of the organization which has attached the RCPs. They don't affect resources from accounts outside the organization. For example, consider an Amazon S3 bucket that's owned by Account A in an organization. The bucket policy (a resource-based policy) grants access to users from Account B outside the organization. Account A has an RCP attached. That RCP applies to the S3 bucket in Account A even when accessed by users from Account B. However, that RCP does not apply to resources in Account B when accessed by users in Account A.
- An RCP restricts permissions for resources in member accounts. Any resource in an account has only those permissions permitted by ***every*** parent above it. If a permission is blocked at any level

above the account, a resource in the affected account does not have that permission, even if the resource owner attaches a resource-based policy that allows full access to any user.

- RCPs apply to the resources that are authorized as part of an operation request. These resources can be found in the “Resource type” column of the Action table in the Service Authorization Reference. If a resource is specified in the "Resource type" column, then the RCPs of the calling principal account are applied. For example, `s3:GetObject` authorizes the object resource. Whenever a `GetObject` request is made, an applicable RCP will apply to determine whether the requesting principal can invoke the `GetObject` operation. An *applicable RCP* is an RCP that has been attached to an account, to an organizational unit (OU), or to the root of the organization that owns the resource being accessed.
- RCPs affect only resources in ***member*** accounts in the organization. They have no effect on resources in the management account. This also means that RCPs apply to member accounts that are designated as delegated administrators. For more information, see Best practices for the management account.
- When a principal makes a request to access a resource within an account that has an attached RCP (a resource with an applicable RCP), the RCP is included in the policy evaluation logic to determine whether the principal is allowed or denied access.
- RCPs impact the effective permissions of principals trying to access resources in a member account with an applicable RCP, regardless of whether the principals belong to the same organizations or not. This includes root users. The exception is when principals are service-linked roles because RCPs do not apply to calls made by service-linked roles. Service-linked roles enable AWS services to perform necessary actions on your behalf and can't be restricted by RCPs.
- Users and roles must still be granted permissions with appropriate IAM permission policies, including identity-based and resource-based policies. A user or role without any IAM permission policies has no access, even if an applicable RCP allows all services, all actions, and all resources.

Resources and entities not restricted by RCPs

You ***can't*** use RCPs to restrict the following:

- Any action on resources in the management account.
- RCPs do not impact the effective permissions of any service-linked role. Service-linked roles are a unique type of IAM role that is linked directly to an AWS service and include all the permissions that the service requires to call other AWS services on your behalf. The permissions of service-linked roles can't be restricted by RCPs. RCPs also do not impact AWS services' ability to assume a service-linked role; that is, the service-linked role's trust policy is also not impacted by RCPs.

- RCPs do not apply to AWS managed keys for AWS Key Management Service. AWS managed keys are created, managed, and used on your behalf by an AWS service. You cannot change or manage their permissions.
- RCPs do not impact following permissions:

Service	API	Resources not authorized by RCPs
AWS Key Management Service	`kms:RetireGrant`	RCPs do not impact the `kms:RetireGrant` permission. For more information on how permission to `kms:RetireGrant` is determined, see Retiring and revoking grants in the *AWS KMS Developer Guide*.

RCP evaluation

> **Note**
>
> The information in this section does ***not*** apply to management policy types, including backup policies, tag policies, chat applications policies, or AI services opt-out policies. For more information, see Understanding management policy inheritance.

As you can attach multiple resource control policies (RCPs) at different levels in AWS Organizations, understanding how RCPs are evaluated can help you write RCPs that yield the right outcome.

Strategy for using RCPs

The `RCPFullAWSAccess` policy is an AWS managed policy. It is automatically attached to the organization root, every OU, and every account in your organization, when you enable resource control policies (RCPs). You cannot detach this policy. This default RCP allows all principals and actions access to pass through RCP evaluation, meaning until you start creating and attaching

RCPs, all your existing IAM permissions continue to operate as they did. This AWS managed policy does not grant access.

You can make use of Deny statements to block access to resources in your organization. For a permission to be **denied** for a resource in a specific account, **any RCP** from the root through each OU in the direct path to the account (including the target account itself) can deny that permission.

Deny statements are a powerful way to implement restrictions that should be true for a broader part of your organization. For example, you can attach a policy to help prevent identities external to your organization from accessing your resources root level, and that will be effective for all accounts in the organization. AWS strongly recommends that you don't attach RCPs to the root of your organization without thoroughly testing the impact that the policy has on resources in your accounts. For more information, see Testing effects of RCPs.

In Figure 1, there is an RCP attached to the Production OU that has an explicit Deny statement specified for a given service. As a result, both Account A and Account B will be denied access to the service as a deny policy attached to any level in the organization is evaluated for all the OUs and member accounts underneath it.

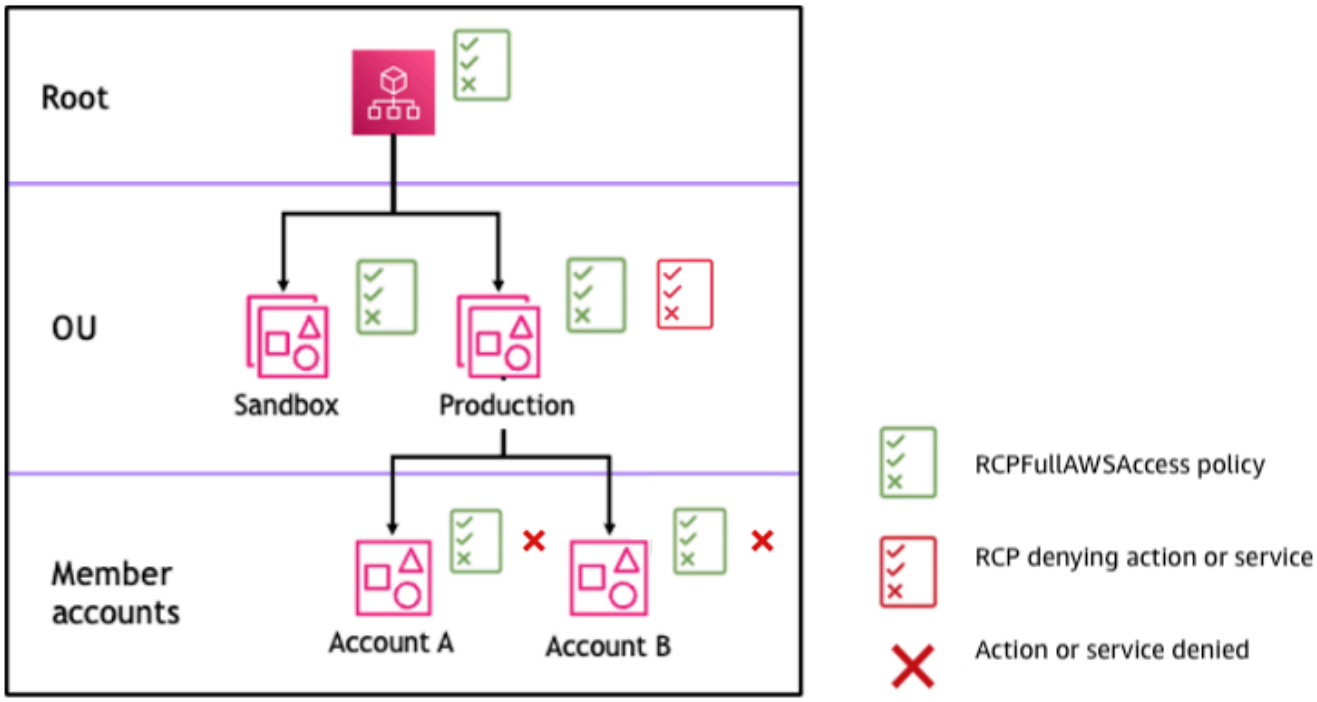


Figure 1: Example organization structure with an Deny statement attached at Production OU and its impact on Account A and Account B

RCP syntax

Resource control policies (RCPs) use a similar syntax to that used by resource-based policies. For more information about IAM policies and their syntax, see Overview of IAM Policies in the *IAM User Guide*.

An RCP is structured according to the rules of JSON. It uses the elements that are described in this topic.

> **Note**
>
> All characters in your RCP count against its maximum size. The examples in this guide show the RCPs 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 RCPs, see Resource control policies (RCPs).

Elements summary

The following table summarizes the policy elements that you can use in RCPs.

> **Note**
>
> **The effect of `Allow` is only supported for the `RCPFullAWSAccess` policy**
> The effect of `Allow` is only supported for the `RCPFullAWSAccess` policy. This policy is automatically attached to the organization root, every OU, and every account in your organization, when you enable resource control policies (RCPs). You cannot detach this policy. This default RCP allows all principals and actions access to pass through RCP evaluation, meaning until you start creating and attaching RCPs, all your existing IAM permissions continue to operate as they did. This does not grant access.

Element	Purpose	
Version	Specifies the language syntax rules to use for	

Element	Purpose
	processing the policy.
Statement	Serves as the container for policy elements. You can have multiple statements in RCPs.
Statement ID (Sid)	(Optional) Provides a friendly name for the statement.
Effect	Defines whether the RCP statement denies access to the resources in an account.
Principal	Specifies the principal that is allowed or denied access to resources in an account.
Action	Specifies AWS service and actions that the RCP allows or denies.
Resource	Specifies the AWS resources that the RCP applies to.

Element	Purpose
NotResource	Specifies the AWS resources that are exempt from the RCP. Used instead of the `Resource` element.
Condition	Specifies condition s for when the statement is in effect.

Topics

- Version element
- Statement element
- Statement ID (Sid) element
- Effect element
- Principal element
- Action element
- Resource and NotResource elements
- Condition element
- Unsupported elements

`Version` element

Every RCP must include a `Version` element with the value `"2012-10-17"`. This is the same version value as the most recent version of IAM permission policies.

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

For more information, see IAM JSON Policy Elements: Version in the *IAM User Guide*.

`Statement` element

An RCP 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`, `Principal`, `Action`, and `Resource` elements.

```
 {
    "Statement": {
        "Effect": "Deny",
        "Principal": "*",
        "Action": "*",
        "Resource": "*"
    }
}
```

For more information, see IAM JSON Policy Elements: Statement in the *IAM User Guide*.

Statement ID (`Sid`) element

The `Sid` is an optional identifier that you provide for the policy statement. You can assign a `Sid` value to each statement in a statement array. The following example RCP shows a sample `Sid` statement.

```
{
    "Statement": {
        "Sid": "DenyAllActions",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "*",
        "Resource": "*"
    }
}
```

For more information, see IAM JSON Policy Elements: Sid in the *IAM User Guide*.

`Effect` element

Each statement must contain one `Effect` element. Using the value of `Deny` in the `Effect` element, you can restrict access to specific resources or define conditions for when RCPs are in

effect. For RCPs that you create, the value must be `Deny`. For more information, see RCP evaluation and IAM JSON Policy Elements: Effect in the *IAM User Guide*.

`Principal` element

Each statement must contain the `Principal` element. You can only specify "`*`" in the `Principal` element of an RCP. Use the `Conditions` element to restrict specific principals.

For more information, see IAM JSON Policy Elements: Principal in the *IAM User Guide*.

`Action` element

Each statement must contain the `Action` element.

The value for the `Action` element is a string or 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", "sqs", or "sts"), in all lowercase, followed by a colon and then an action from that service. Generally, they are all entered with each word starting with an uppercase letter and the rest lowercase. For example: `"s3:ListAllMyBuckets"`.

You also can use wildcard characters such as asterisk (*) or question mark (?) in an RCP:

- 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 `"sts:Get*"` matches only the AWS STS actions that begin with "Get".
- Use the question mark (?) wildcard to match a single character.

> **Note**
>
> **Wildcards (*) and question marks (?) can be used anywhere in the action name**
> Unlike with SCPs, you can use wildcard characters such as asterisk (*) or question mark (?) anywhere in the action name.

For a list of the services that support RCPs, see List of AWS services that support RCPs. For a list of the actions an AWS service supports, see Actions, Resources, and Condition Keys for AWS Services in the *Service Authorization Reference*.

For more information, see IAM JSON Policy Elements: Action in the *IAM User Guide*.

`Resource` and `NotResource` elements

Each statement must contain the `Resource` or `NotResource` element.

You can use wildcard characters such as asterisk (*) or question mark (?) in the resource element:

- Use an asterisk (*) as a wildcard to match multiple resources that share part of a name.
- Use the question mark (?) wildcard to match a single character.

For more information, see IAM JSON Policy Elements: Resource and see IAM JSON Policy Elements: NotResource in the *IAM User Guide*.

`Condition` element

You can specify a `Condition` element in deny statements in an RCP.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "*",
            "Condition:": {
                "BoolIfExists": {
                    "aws:SecureTransport": "false"
                }
            }
        }
    ]
}
```

This RCP denies access to Amazon S3 operations and resources unless the request occurs over secure transport (the request was sent over TLS).

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

Unsupported elements

The following elements are not supported in RCPs:

- `NotPrincipal`
- `NotAction`

Resource control policy examples

The example resource control policies (RCPs) displayed in this topic are for information purposes only. For data perimeter examples, see Data Perimeter Policy Examples in GitHub.

Before using these examples

Before you use these example RCPs in your organization, do the following:

- Carefully review and customize the RCPs for your unique requirements.
- Thoroughly test the RCPs in your environment with the AWS services that you use.

The example policies in this section demonstrate the implementation and use of RCPs. They're ***not*** intended to be interpreted as official AWS recommendations or best practices to be implemented exactly as shown. It is your responsibility to carefully test any policies for its suitability to solve the business requirements of your environment. Deny-based resource control policies can unintentionally limit or block your use of AWS services unless you add the necessary exceptions to the policy.

General examples

Topics

- RCPFullAWSAccess
- Cross-service confused deputy protection
- Restrict access to only HTTPS connections to your resources
- Consistent Amazon S3 bucket policy controls

RCPFullAWSAccess

The following policy is an AWS managed policy and is automatically attached to the organization root, every OU, and every account in your organization, when you enable resource control policies (RCPs). You cannot detach this policy. This default RCP allows all principals and actions access

to your resources, meaning until you start creating and attaching RCPs, all your existing IAM permissions continue to operate as they did. You do not need to test the effect of this policy as it will allow existing authorization behavior to continue for your resources.

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

Cross-service confused deputy protection

Some AWS services (calling services) use their AWS service principal to access AWS resources from other AWS services (called services). When an actor not intended to have access to an AWS resource attempts to use the trust of an AWS service principal to interact with resources that they are not intended to have access to it is known as the cross-service confused deputy problem. For more information, see The confused deputy problem in the *IAM User Guide*

The following policy requires that AWS service principals accessing your resources only do so on behalf of requests from your organization. This policy applies the control only on requests that have `aws:SourceAccount` present so that service integrations that do not require the use of `aws:SourceAccount` aren't impacted. If the `aws:SourceAccount` is present in the request context, the `Null` condition will evaluate to `true`, causing the `aws:SourceOrgID` key to be enforced.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnforceConfusedDeputyProtection",
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "s3:*",
                "sqs:*",
```

```
                "kms:*",
                "secretsmanager:*",
                "sts:*"
            ],
            "Resource": "*",
            "Condition": {
                "StringNotEqualsIfExists": {
                    "aws:SourceOrgID": "my-org-id",
                    "aws:SourceAccount": [
                        "third-party-account-a",
                        "third-party-account-b"
                    ]
                },
                "Bool": {
                    "aws:PrincipalIsAWSService": "true"
                },
                 "Null": {
                    "aws:SourceArn": "false"
                }
            }
        }
    ]
}
```

Restrict access to only HTTPS connections to your resources

The following policy requires that access to your resources only occurs on encrypted connections over HTTPS (TLS). This can help you prevent potential attackers from manipulating network traffic.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnforceSecureTransport",
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "sts:*",
                "s3:*",
                "sqs:*",
                "secretsmanager:*",
                "kms:*"
            ],
```

```
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "aws:SecureTransport": "false"
                }
            }
        }
    ]
}
```

Consistent Amazon S3 bucket policy controls

The following RCP contains multiple statements to enforce consistent access controls on Amazon S3 buckets in your organization.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnforceS3TlsVersion",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "*",
            "Condition": {
                "NumericLessThan": {
                    "s3:TlsVersion": [
                        "1.2"
                    ]
                }
            }
        },
        {
            "Sid": "EnforceKMSEncryption",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "*",
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption-aws-kms-key-id": "true"
                }
            }
```

```
        }
    ]
}
```

- The statement ID `EnforceS3TlsVersion` – Require a minimum TLS version of 1.2 for access to S3 buckets.
- The statement ID `EnforceKMSEncryption` – Require objects to be server-side encrypted with KMS keys.

Management policies in AWS Organizations

Management policies enable you to centrally configure and manage AWS services and their features. How those policies affect the OUs and accounts that inherit them depends on the type of management policy you apply in AWS Organizations. Review the topics in this section to understand relevant terms and concepts about management policies.

Topics

- Prerequisites and permissions for management policies for AWS Organizations
- Understanding management policy inheritance
- Viewing effective management policies
- Declarative policies
- Backup policies
- Tag policies
- Chat applications policies
- AI services opt-out policies

Prerequisites and permissions for management policies for AWS Organizations

This page describes the prerequisites and required permissions for management policies for AWS Organizations.

Topics

- Prerequisites for management policies

- Permissions for management policies

Prerequisites for management policies

Using management policies for an organization requires the following:

- Your organization must have all features enabled.
- You must be signed in to your organization's management account or be a delegated administrator.
- Your AWS Identity and Access Management (IAM) user or role must have the permissions that are listed in the following section.

Permissions for management policies

The following example IAM policy provides permissions to use all aspects of management policies in an organization.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "OrganizationPolicies",
            "Effect": "Allow",
            "Action": [
                "organizations:AttachPolicy",
                "organizations:CreatePolicy",
                "organizations:DeletePolicy",
                "organizations:DescribeAccount",
                "organizations:DescribeCreateAccountStatus",
                "organizations:DescribeEffectivePolicy",
                "organizations:DescribeOrganization",
                "organizations:DescribeOrganizationalUnit",
                "organizations:DescribePolicy",
                "organizations:DetachPolicy",
                "organizations:DisableAWSServiceAccess",
                "organizations:DisablePolicyType",
                "organizations:EnableAWSServiceAccess",
                "organizations:EnablePolicyType",
                "organizations:ListAccounts",
                "organizations:ListAccountsForParent",
```

```
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:ListCreateAccountStatus",
                "organizations:ListOrganizationalUnitsForParent",
                "organizations:ListParents",
                "organizations:ListPolicies",
                "organizations:ListPoliciesForTarget",
                "organizations:ListRoots",
                "organizations:ListTargetsForPolicy",
                "organizations:UpdatePolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

For more information about IAM policies and permissions, see the IAM User Guide.

Understanding management policy inheritance

> **Important**
>
> The information in this section does ***not*** apply to authorization policies: service control policies (SCPs) and resource control policies (RCPs). For more information about how SCPs and RCPs work in an AWS Organizations hierarchy, see SCP evaluation and RCP evaluation.

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

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

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

This following topics explain how parent policies and child policies are processed into the effective policy for an account.

Topics

- Inheritance terminology
- Policy syntax and inheritance for management policy types
- Inheritance operators
- Inheritance examples

Inheritance terminology

This topic uses the following terms when discussing management policy inheritance.

Policy inheritance

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

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

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

Parent policies

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

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

Child policies

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

Effective policies

The final, single policy document that specifies the rules that apply to an account. The effective policy is the aggregation of any policies the account inherits, plus any policy that is directly attached to the account. For more information, see Viewing effective management policies.

Inheritance operators

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

Policy syntax and inheritance for management policy types

Exactly how policies affect the OUs and accounts that inherit them depends on the type of management policy you choose. Management policy types include:

- Declarative policies
- Backup policies
- Tag policies
- Chat applications policies
- AI services opt-out policies

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

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

Inheritance operators

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

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

For information about how policy inheritance works in an organization, see Inheritance examples.

Value-setting operators

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

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

Child control operators

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

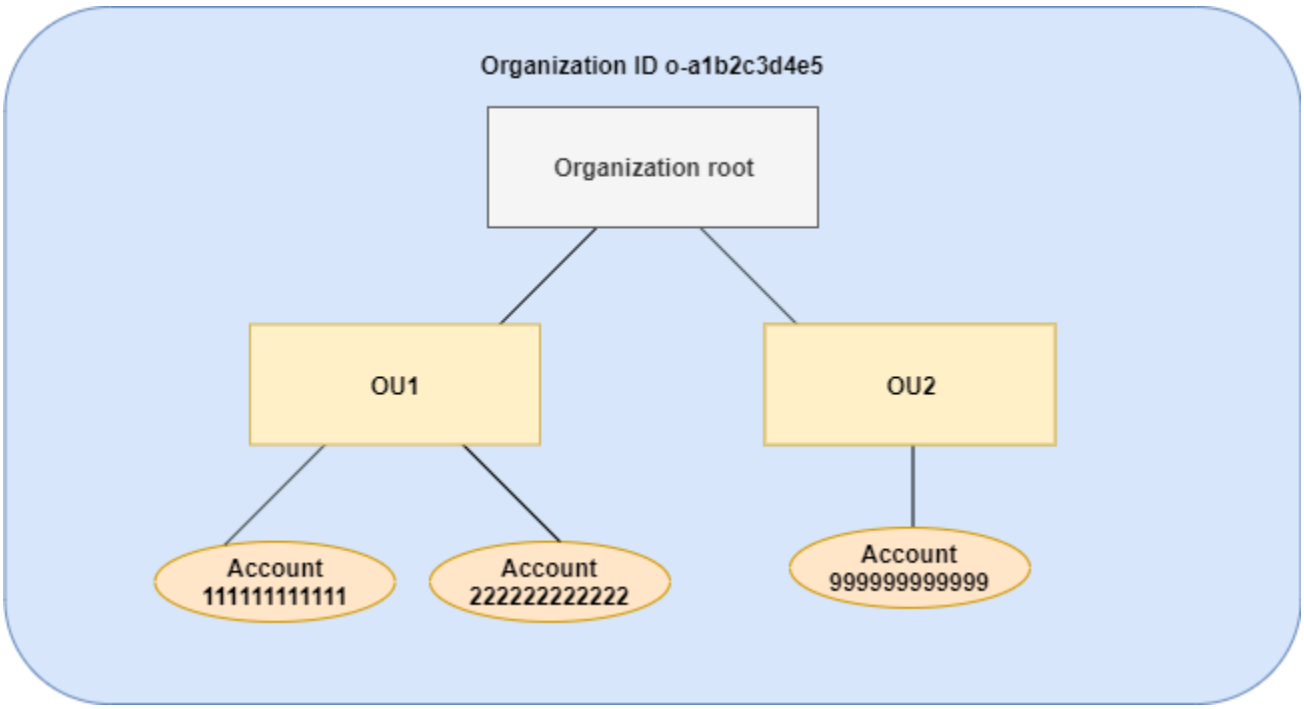
- `"@@operators_allowed_for_child_policies":["@@all"]` – Child OUs and accounts can use any operator in policies. By default, all operators are allowed in child policies.
- `"@@operators_allowed_for_child_policies":["@@assign", "@@append", "@@remove"]` – Child OUs and accounts can use only the specified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

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

Inheritance examples

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

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



Examples

- Example 1: Allow child policies to overwrite only tag values
- Example 2: Append new values to inherited tags
- Example 3: Remove values from inherited tags
- Example 4: Restrict changes to child policies
- Example 5: Conflicts with child control operators
- Example 6: Conflicts with appending values at same hierarchy level

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

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

Policy A – Organization root tag policy

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

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

Policy B – OU1 tag policy

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

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

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

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

Resultant effective tag policy for accounts 111111111111 and 222222222222

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

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

Example 2: Append new values to inherited tags

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

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

Policy A – Organization root tag policy

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

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

Policy C – OU2 tag policy for appending values

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

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

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

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

Effective tag policy for account 999999999999

Note

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

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

Example 3: Remove values from inherited tags

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

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

Policy A – Organization root tag policy

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

```
            "tag_value": {
                "@@assign": [
                    "Development",
                    "Support"
                ]
            }
        }
    }
}
```

For this example, attach policy D to account 999999999999.

Policy D – Account 999999999999 tag policy for removing values

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

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

- Assuming you performed all of the previous examples, policies B, C, and C are child policies of A. Policy B is only attached to OU1, so it has no effect on account 9999999999999.

- For account 9999999999999, the only acceptable value for the `CostCenter` tag key is `Support`.
- Compliance is not enforced for the `CostCenter` tag key.

New effective tag policy for account 999999999999

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

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

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

Example 4: Restrict changes to child policies

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

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

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

```
{
```

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

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

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

Policy F – OU tag policy

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

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

- Policy F is a child policy to Policy E.

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

Effective policy for accounts in the OU

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

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

Example 5: Conflicts with child control operators

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

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

Policy G – Organization root tag policy 1

```
{
    "tags": {
```

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

Policy H – Organization root tag policy 2

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

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

Example 6: Conflicts with appending values at same hierarchy level

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

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

Policy J – First tag policy attached to the organization root

```
{
```

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

Policy K – Second tag policy attached to the organization root

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

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

Policy JK – Effective tag policy for account

The effective policy for the account is as follows.

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

```
{
    "tags": {
```

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

Viewing effective management policies

Determine the effective management policy for an account in your organization.

What is an effective management policy?

The *effective policy* specifies the final rules that apply to an AWS account for a management policy type. It is the aggregation for a management policy that the account inherits, plus any policies for that management policy type that are directly attached to the account. When you attach a management policy to the organization's root, it applies to all accounts in your organization. When you attach a management policy to an organiztional unit (OU), it applies to all accounts and OUs that belong to the OU. When you attach a management policy directly to an account, it applies only to that one AWS account.

For information about how policies are combined into the final effective policy, see Understanding management policy inheritance.

Backup policy example

The backup policy attached to the organization root might specify that all accounts in the organization back up all Amazon DynamoDB tables with a default backup frequency of once per week. A separate backup policy attached directly to one member account with critical information in a table can override the frequency with a value of once per day. The combination of these backup policies comprises the effective backup policy. This effective backup policy is determined for each account in the organization individually. In this example, the result is that all accounts in the organization back up their DynamoDB tables once per week, with the exception of one account that backs up its tables daily.

Tag policy example

The tag policy attached to the organization root might define a `CostCenter` tag with four compliant values. A separate tag policy attached to the account may restrict the `CostCenter`

key to only two of the four compliant values. The combination of these tag policies comprises the effective tag policy. The result is that only two of the four compliant tag values defined in the organization root tag policy are compliant for the account.

Chat applications policy example

Amazon Q Developer in chat applications will reevaluate any previously created Amazon Q Developer in chat applications configurations against the effective chat applications policies and deny any previously allowed actions if they are consistent with the permitted settings and guardrails in the effective policy. The effective policy for a member account defines the permitted settings and guardrails. For example, if a chat applications policy with deny access for public Slack channels is applied to a member account, then the existing Amazon Q Developer in chat applications configurations for public Slack channels in the member account will be disabled. Amazon Q Developer in chat applications will not deliver notifications and channel members will not be able to run any tasks in the blocked channel. The Amazon Q Developer in chat applications console will mark the affected channels as disabled with an appropriate error messaging next to it.

AI services opt-out example

The AI services opt-out policy attached to the organization root might specify that all accounts in the organization opt out of content use by all AWS machine learning services. A separate AI services opt-out policy attached directly to one member account specifies that it opts in to content use for only Amazon Rekognition. The combination of these AI services opt-out policies comprises the effective AI services opt-out policy. The result is that all accounts in the organization are opted out of all AWS services, with the exception of one account that opts in to Amazon Rekognition.

How to view the effective management policy

You can view the effective policy of a management policy type for an account from the AWS Management Console, AWS API, or AWS Command Line Interface.

Minimum permissions

To view the effective policy of a management policy type for an account, you must have permission to run the following actions:

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

AWS Management Console

To view the effective policy of a management policy type for an account

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page, choose the name of the account for which you want to view the effective policy. You might have to expand OUs (choose the ▶) to find the account that you want.
3. On the **Policies** tab, choose the management policy type you want to view the effective policy for.
4. Choose **View the effective policy for this AWS account**.

   The console displays the effective policy applied to the specified account.

   > **Note**
   >
   > You can't copy and paste an effective policy and use it as the JSON for another policy without significant changes. Policy documents must include the inheritance operators that specify how each setting is merged into the final effective policy.

AWS CLI & AWS SDKs

To view the effective policy of a management policy type for an account

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

- AWS CLI: describe-effective-policy

  The following example shows the effective AI services opt-out policy for an account.

```
$ aws organizations describe-effective-policy \
    --policy-type AISERVICES_OPT_OUT_POLICY \
    --target-id 123456789012
{
    "EffectivePolicy": {
```

```
        "PolicyContent": "{\"services\":{\"comprehend\":{\"opt_out_policy\":
\"optOut\"},   ....TRUNCATED FOR BREVITY....   "opt_out_policy\":\"optIn\"}}}",
        "LastUpdatedTimestamp": "2020-12-09T12:58:53.548000-08:00",
        "TargetId": "123456789012",
        "PolicyType": "AISERVICES_OPT_OUT_POLICY"
    }
}
```


- AWS SDKs: DescribeEffectivePolicy

Declarative policies

Declarative policies allow you to centrally declare and enforce your desired configuration for a given AWS service at scale across an organization. Once attached, the configuration is always maintained when the service adds new features or APIs. Use declarative policies to prevent noncompliant actions. For example, you can block public internet access to Amazon VPC resources across your organization.

The key benefits of using declarative policies are:

- **Ease of use**: You can enforce the baseline configuration for an AWS service with a few selections in the AWS Organizations and AWS Control Tower consoles or with a few commands using the AWS CLI & AWS SDKs.
- **Set once and forget**: The baseline configuration for an AWS service is always maintained, even when the service introduces new features or APIs. The baseline configuration is also maintained when new accounts are added to an organization or when new principals and resources are created.
- **Transparency**: The account status report allows you to review the current status of all attributes supported by declarative policies for the accounts in scope. You can also create customizable error messages, which can help administrators redirect end users to internal wiki pages or provide a descriptive message that can help end users understand why an action failed.

For a full list of supported AWS services and attributes, see Supported AWS services and attributes.

Topics

- How declarative policies work
- Custom error messages for declarative policies
- Account status report for declarative policies

- Supported AWS services and attributes
- Getting started with declarative policies
- Best practices for using declarative policies
- Generating the account status report for declarative policies
- Declarative policy syntax and examples

How declarative policies work

Declarative policies are enforced in the service's control plane, which is an important distinction from authorization policies such as service control policies (SCPs) and resource control policies (RCPs). While authorization policies regulate access to APIs, declarative policies are applied directly at the service level to enforce durable intent. This ensures that the baseline configuration is always enforced, even when new features or APIs are introduced by the service.

The following table helps illustrate this distinction and provides some use cases.

	Service control policies	**Resource control policies**	**Declarative policies**		
Why?	To centrally define and enforce consisten t access controls on principals (such as IAM users and IAM roles) at scale.	To centrally define and enforce consisten t access controls on resources at scale	To centrally define and enforce the baseline configura tion for AWS services at scale.		
How?	By controlli ng the maximum available	By controlli ng the maximum available	By enforcing the desired configura tion of an		

	Service control policies	Resource control policies	Declarative policies		
	access permissions of principals at an API level.	access permissions for resources at an API level.	AWS service without using API actions.		
Governs service-linked roles?	No	No	Yes		
Feedback mechanism	Non-customizable access denied SCP error.	Non-customizable access denied RCP error.	Customizable error message. For more information, see Custom error messages for declarative policies.		
Example policy	Deny access to AWS based on the requested AWS Region	Restrict access to only HTTPS connections to your resources	Allowed Images Settings		

After you have created and attached a declarative policy, it is applied and enforced across your organization. Declarative policies can be applied to an entire organization, organizational units (OUs), or accounts. Accounts joining an organization will automatically inherit the declarative policy in the organization. For more information, see Understanding management policy inheritance.

The *effective policy* is the set of rules that are inherited from the organization root and OUs along with those directly attached to the account. The effective policy specifies the final set of rules that apply to the account. For more information, see Viewing effective management policies.

If a declarative policy is detached, the attribute state will roll back to its previous state before the declarative policy was attached.

Custom error messages for declarative policies

Declarative policies allow you to create custom error messages. For example, if an API operation fails due to a declarative policy, you can set the error message or provide a custom URL, such as a link to an internal wiki or a link to a message that describes the failure. If you do not specify a custom error message, AWS Organizations provides the following default error message: `Example: This action is denied due to an organizational policy in effect.`

You can also audit the process of creating declarative policies, updating declarative policies, and deleting declarative policies with AWS CloudTrail. CloudTrail can flag API operation failures due to declarative policies. For more information, see Logging and monitoring.

> **Important**
>
> Do not include *personally identifiable information (PII)* or other sensitive information in a custom error message. PII includes general information that can be used to identify or locate an individual. It covers records such as financial, medical, educational, or employment. PII examples include addresses, bank account numbers, and phone numbers.

Account status report for declarative policies

The *account status report* allows you to review the current status of all attributes supported by declarative policies for the accounts in scope. You can choose the accounts and organizational units (OUs) to include in the report scope, or choose an entire organization by selecting the root.

This report helps you assess readiness by providing a Region breakdown and if the current state of an attribute is *uniform across accounts* (through the `numberOfMatchedAccounts`) or *inconsistent* (through the `numberOfUnmatchedAccounts`). You can also see the *most frequent value*, which is the configuration value that is most frequently observed for the attribute.

In Figure 1, there is a generated account status report, which shows uniformity across accounts for the following attributes: VPC Block Public Access and Image Block Public Access. This means that, for each attribute, all the accounts in scope have the same configuration for that attribute.

The generated account status report shows inconsistent accounts for the following attributes: Allowed Images Settings, Instance Metadata defaults, Serial Console Access, and Snapshot Block Public Access. In this example, each attribute with an inconsistent account is due to there being one account with a different configuration value.

If there is a most frequent value, that is displayed in its respective column. For more detailed information of what each attribute controls, see Declarative policy syntax and example policies.

You can also expand an attribute to see a Region breakdown. In this example, Image Block Public Access is expanded and in each Region, you can see that there is also uniformity across accounts.

The choice to attach a declarative policy for enforcing a baseline configuration depends on your specific use case. Use the account status report to help you assess your readiness before attaching a declarative policy.

For more information, see Generating the account status report.

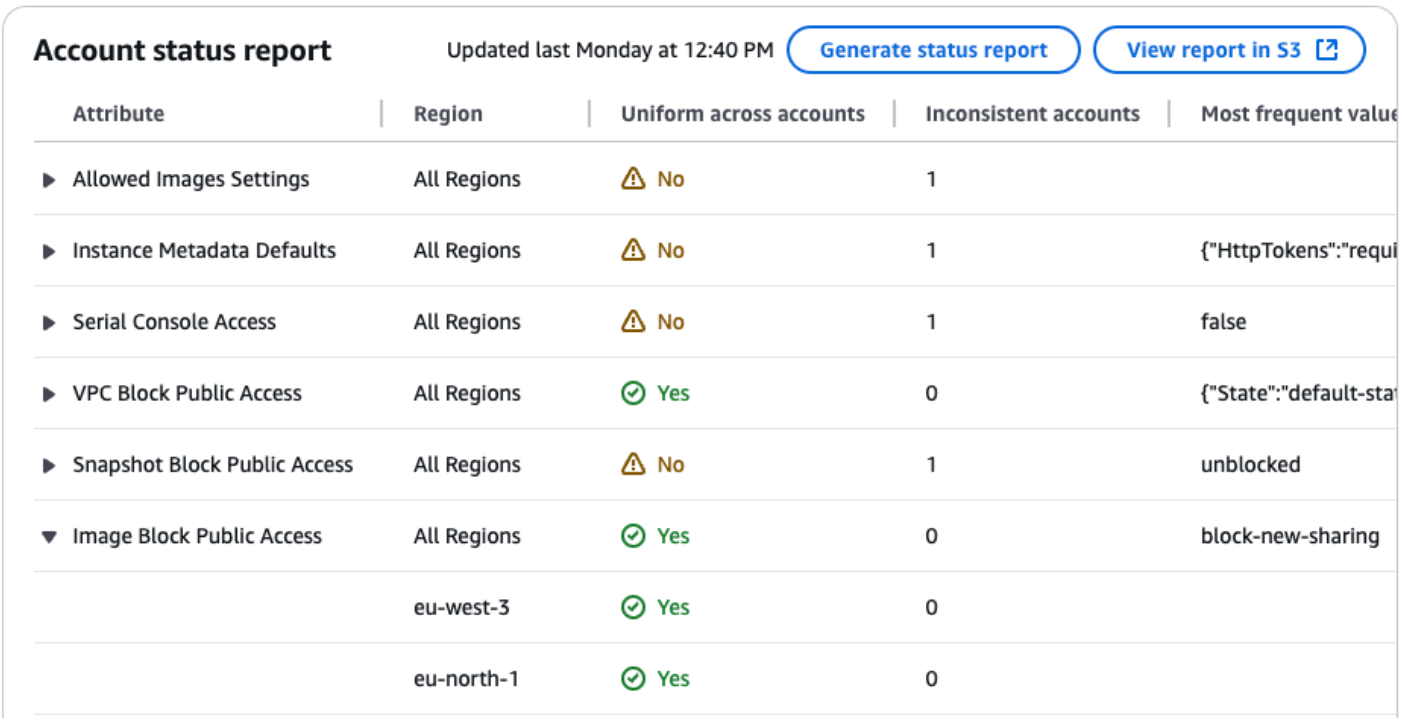


Figure 1: Example account status report with uniformity across accounts for VPC Block Public Access and Image Block Public Access.

Supported AWS services and attributes

Supported attributes for declarative policies for EC2

The following table displays the attributes supported for Amazon EC2 related services.

Declarative policies for EC2

AWS service	Attribute	Policy effect	Policy contents	More information
Amazon VPC	VPC Block Public Access	Controls if resources in Amazon VPCs and subnets can reach the internet through internet gateways (IGWs).	View policy	For more information, see Block public access to VPCs and subnets in the *Amazon VPC User Guide*.
Amazon EC2	Serial Console Access	Controls if the EC2 serial console is accessible.	View policy	For more information, see Configure access to the EC2 Serial Console in the *Amazon Elastic Compute Cloud User Guide*.
	Image Block Public Access	Controls if Amazon Machine Images (AMIs) are publicly sharable.	View policy	For more information, see Understand block public access for AMIs in the *Amazon Elastic Compute Cloud User Guide*.

AWS service	Attribute	Policy effect	Policy contents	More information
	Allowed Images Settings	Controls the discovery and use of Amazon Machine Images (AMI) in Amazon EC2 with Allowed AMIs.	View policy	For more information, see Amazon Machine Images (AMIs) in the *Amazon Elastic Compute Cloud User Guide*.
	Instance Metadata Defaults	Controls IMDS defaults for all new EC2 instances launches.	View policy	For more information, see Configure instance metadata options for new instances in the *Amazon Elastic Compute Cloud User Guide*.
Amazon EBS	Snapshot Block Public Access	Controls if Amazon EBS snapshots are publicly accessible.	View policy	For more information, see Block public access for Amazon EBS snapshots in the *Amazon Elastic Block Store User Guide*.

Getting started with declarative policies

Follow these steps to get started using declarative policies.

1. Learn about the permissions you must have to perform declarative policy tasks.
2. Enable declarative policies for your organization.

> **Note**
>
> **Enabling trust access is required**
> You must enable trusted access for the service where the declarative policy will enforce a baseline configuration. This creates a read-only service-linked role that is used to generate the account status report of what the existing configuration is for accounts across your organization.
> **Using the console**
> If you use the Organizations console, this step is a part of the process for enabling declarative policies.
> **Using the AWS CLI**
> If you use the AWS CLI, there are two separate APIs:
>
> - EnablePolicyType, which you use to enable declarative policies.
> - EnableAWSServiceAccess, which you use to enable trusted access.
>
> For more information on how to enable trusted access for a specific service with the AWS CLI see, AWS services that you can use with AWS Organizations.

3. Run the account status report.
4. Create a declarative policy.
5. Attach the declarative policy to your organization's root, OU, or account.
6. View the combined effective declarative policy that applies to an account.

For all of these steps, you sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.

Other information

- Learn declarative policy syntax and see example policies

Best practices for using declarative policies

AWS recommends the following best practices for using declarative policies.

Leverage readiness assessments

Use the declarative policy *account status report* to assess the current status of all attributes supported by declarative policies for the accounts in scope. You can choose the accounts and organizational units (OUs) to include in the report scope, or choose an entire organization by selecting the root.

This report helps you assess readiness by providing a Region breakdown and if the current state of an attribute is *uniform across accounts* (through the `numberOfMatchedAccounts`) or *inconsistent* (through the `numberOfUnmatchedAccounts`). You can also see the *most frequent value*, which is the configuration value that is most frequently observed for the attribute.

The choice to attach a declarative policy for enforcing a baseline configuration depends on your specific use case.

For more information and an illustrative example, see Account status report for declarative policies.

Start small and then scale

To simplify debugging, start with a test policy. Validate the behavior and impact of each change before making the next change. This approach reduces the number of variables you have to account for when an error or unexpected result occurs.

For example, you can start with a test policy attached to a single account in a noncritical test environment. After you have confirmed that it works to your specifications, you can then incrementally move the policy up the organization structure to more accounts and more organizational units (OUs).

Establish review processes

Implement processes to monitor for new declarative attributes, evaluate policy exceptions, and make adjustments to maintain alignment with your organizational security and operational requirements.

Validate changes using `DescribeEffectivePolicy`

After you make a change to a declarative policy, check the effective policies for representative accounts below the level where you made the change. You can view the effective policy by using the AWS Management Console, or by using the DescribeEffectivePolicy API operation or one of its AWS CLI or AWS SDK variants. Ensure that the change you made had the intended impact on the effective policy.

Communicate and train

Ensure your organizations understand the purpose and impact of your declarative policies. Provide clear guidance on the expected behaviors and how to handle failures due to policy enforcement.

Generating the account status report for declarative policies

The *account status report* allows you to review the current status of all attributes supported by declarative policies for the accounts in scope. You can choose the accounts and organizational units (OUs) to include in the report scope, or choose an entire organization by selecting the root.

This report helps you assess readiness by providing a Region breakdown and if the current state of an attribute is *uniform across accounts* (through the `numberOfMatchedAccounts`) or *inconsistent* (through the `numberOfUnmatchedAccounts`). You can also see the *most frequent value*, which is the configuration value that is most frequently observed for the attribute.

The choice to attach a declarative policy for enforcing a baseline configuration depends on your specific use case.

For more information and an illustrative example, see Account status report for declarative policies.

Prerequisites

Before you can generate an account status report, you must perform the following steps

1. The `StartDeclarativePoliciesReport` API can only be called by the management account or delegated administrators for an organization.
2. You must have an S3 bucket before generating the report (create a new one or use an existing one), it must be in the same Region in which the request is made, and it must have an appropriate S3 bucket policy. For a sample S3 policy, see *Sample Amazon S3 policy* under Examples in the *Amazon EC2 API Reference*
3. You must enable trusted access for the service where the declarative policy will enforce a baseline configuration. This creates a read-only service-linked role that is used to generate the account status report of what the existing configuration is for accounts across your organization.

   Using the console

   For the Organizations console, this step is a part of the process for enabling declarative policies.

   Using the AWS CLI

For the AWS CLI, use the EnableAWSServiceAccess API.

For more information on how to enable trusted access for a specific service with the AWS CLI see, AWS services that you can use with AWS Organizations.

4. Only one report per organization can be generated at a time. Attempting to generate a report while another is in progress will result in an error.

Access the compliance status report

Minimum permissions

To generate a compliance status report, you need permission to run the following actions:

- `ec2:StartDeclarativePoliciesReport`
- `ec2:DescribeDeclarativePoliciesReports`
- `ec2:GetDeclarativePoliciesReportSummary`
- `ec2:CancelDeclarativePoliciesReport`
- `organizations:DescribeAccount`
- `organizations:DescribeOrganization`
- `organizations:DescribeOrganizationalUnit`
- `organizations:ListAccounts`
- `organizations:ListDelegatedAdministrators`
- `organizations:ListAWSServiceAccessForOrganization`

AWS Management Console

Use the following procedure to generate an account status report.

To generate an account status report

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Policies** page, choose **Declarative policies for EC2**.

3. On the **Declarative policies for EC2** page, choose **View account status report** from the **Actions** dropdown menu.
4. On the **View account status report** page, choose **Generate status report**.
5. In the **Organizational structure** widget, specify which organizational units (OUs) you want to include in the report.
6. Choose **Submit**.

AWS CLI & AWS SDKs

To generate an account status report

Use the following operations to generate a compliance status report, check on its status, and view the report:

- `ec2:start-declarative-policies-report`: Generates an account status report. The report is generated asynchronously, and can take several hours to complete. For more information, see StartDeclarativePoliciesReport in the *Amazon EC2 API Reference*.
- `ec2:describe-declarative-policies-report`: Describes the metadata of an account status report, including the state of the report. For more information, see DescribeDeclarativePoliciesReports in the *Amazon EC2 API Reference*.
- `ec2:get-declarative-policies-report-summary`: Retrieves a summary of the account status report. For more information, see GetDeclarativePoliciesReportSummary in the *Amazon EC2 API Reference*.
- `ec2:cancel-declarative-policies-report`: Cancels the generation of an account status report. For more information, see CancelDeclarativePoliciesReport in the *Amazon EC2 API Reference*.

Declarative policy syntax and examples

This page describes declarative policy syntax and provides examples.

Considerations

- When you configure a service attribute using a declarative policy, it might impact multiple APIs. Any noncompliant actions will fail.
- Account administrators will not be able to modify the value of the service attribute at the individual account level.

Syntax for declarative policies

A declarative policy is a plaintext file that is structured according to the rules of JSON. The syntax for declarative policies follows the syntax for all management policy types. For a complete discussion of that syntax, see Policy syntax and inheritance for management policy types. This topic focuses on applying that general syntax to the specific requirements of the declarative policy type.

The following example shows basic declarative policy syntax:

```
{
    "ec2_attributes": {
        "exception_message": {
            "@@assign": "Your custom error message.https://myURL"
        },

        ...

        [Insert supported service attributes]

        ...
    }
}
```

- The `ec2_attributes` field key name. Declarative policies always start with a fixed key name for the given AWS service. It's the top line in the example policy above. Currently declarative policies only supported Amazon EC2 related services.
- Under `ec2_attributes`, you can use `exception_message` to set a custom error message. For more information, see Custom error messages for declarative policies.
- Under `ec2_attributes`, you can insert one or more of the supported declarative policies. For those schemas, see Supported declarative policies.

Supported declarative policies

The following are the AWS services and attributes that declarative policies support. In some of the following examples, the JSON whitespace formatting might be compressed to save space.

- VPC Block Public Access
- Serial Console Access

- Image Block Public Access
- Allowed Images Settings
- Instance Metadata Defaults
- Snapshot Block Public Access

VPC Block Public Access

Policy effect

Controls if resources in Amazon VPCs and subnets can reach the internet through internet gateways (IGWs). For more information, see Configuration for internet access in the *Amazon Virtual Private Cloud User Guide*.

Policy contents

```
"vpc_block_public_access": {
    "internet_gateway_block": { // (optional)
        "mode": { // (required)
            "@@assign": "block_ingress" // off | block_ingress | block_bidirectional
        },
        "exclusions_allowed": { // (required)
            "@@assign": "enabled" // enabled | disabled
        }
    }
}
```

The following are the available fields for this attribute:

- `"internet_gateway"`:
  - `"mode"`:
    - `"off"`: VPC BPA is not enabled.
    - `"block_ingress"`: All internet traffic to the VPCs (except for VPCs or subnets which are excluded) is blocked. Only traffic to and from NAT gateways and egress-only internet gateways is allowed because these gateways only allow outbound connections to be established.
    - `"block_bidirectional"`: All traffic to and from internet gateways and egress-only internet gateways (except for excluded VPCs and subnets) is blocked..

- `"exclusions_allowed"`: An exclusion is a mode that can be applied to a single VPC or subnet that exempts it from the account's VPC BPA mode and will allow bidirectional or egress-only access.
  - `"enabled"`: Exclusions can be created by the account.
  - `"disabled"`: Exclusions cannot be created by the account.

> **Note**
>
> You can use the attribute to configure if exclusions are allowed, but you cannot create exclusions with this attribute itself. To create exclusions, you must create them in the account that owns the VPC. For more information about creating VPC BPA exclusions, see Create and delete exclusions in the *Amazon VPC User Guide*.

Considerations

If you use this attribute in a declarative policy, you cannot use the following operations to modify the enforced configuration for the accounts in scope. This list is not exhaustive:

- `ModifyVpcBlockPublicAccessOptions`
- `CreateVpcBlockPublicAccessExclusion`
- `ModifyVpcBlockPublicAccessExclusion`

Serial Console Access

Policy effect

Controls if the EC2 serial console is accessible. For more information about the EC2 serial console, see EC2 Serial Console in the *Amazon Elastic Compute Cloud User Guide*.

Policy contents

```
"serial_console_access": {
    "status": { // (required)
        "@@assign": "enabled" // enabled | disabled
    }
}
```

The following are the available fields for this attribute:

- `"status"`:
  - `"enabled"`: EC2 serial console access is allowed.
  - `"disabled"`: EC2 serial console access is blocked.

Considerations

If you use this attribute in a declarative policy, you cannot use the following operations to modify the enforced configuration for the accounts in scope. This list is not exhaustive:

- `EnableSerialConsoleAccess`
- `DisableSerialConsoleAccess`

Image Block Public Access

Policy effect

Controls if Amazon Machine Images (AMIs) are publicly sharable. For more information about AMIs, see Amazon Machine Images (AMIs) in the *Amazon Elastic Compute Cloud User Guide*.

Policy contents

```
"image_block_public_access": {
    "state": { // (required)
        "@@assign": "block_new_sharing" // unblocked | block_new_sharing
    }
}
```

The following are the available fields for this attribute:

- `"state"`:
  - `"unblocked"`: No restrictions on the public sharing of AMIs.
  - `"block_new_sharing"`: Blocks new public sharing of AMIs. AMIs that were already publicly shared remain publicly available.

Considerations

If you use this attribute in a declarative policy, you cannot use the following operations to modify the enforced configuration for the accounts in scope. This list is not exhaustive:

- `EnableImageBlockPublicAccess`
- `DisableImageBlockPublicAccess`

Allowed Images Settings

Policy effect

Controls the discovery and use of Amazon Machine Images (AMI) in Amazon EC2 with Allowed AMIs.. For more information about AMIs, see Amazon Machine Images (AMIs) in the *Amazon Elastic Compute Cloud User Guide*.

Policy contents

The following are the available fields for this attribute:

```
"allowed_images_settings": {
    "state": { // (required)
        "@@assign": "enabled" // enabled | disabled | audit_mode
    },
    "image_criteria": { // (optional)
        "criteria_1": {
            "allowed_image_providers": { // limit 200
                "@@append": [
                    "amazon" // amazon | aws_marketplace | aws_backup_vault | 12
 digit account ID
                ]
            }
        }
    }
}
```

- `"state"`:
  - `"enabled"`: The attribute is active and enforced.
  - `"disabled"`: The attribute is inactive and not enforced.
  - `"audit_mode"`: The attribute is in audit mode. This means it will identify noncompliant images but not block their use.
- `"image_criteria"`: A list of `allowed_image_providers` objects that define the allowed AMI sources.

- `"allowed_image_providers"`: A comma-separated list of provider names or account IDs.

Considerations

If you use this attribute in a declarative policy, you cannot use the following operations to modify the enforced configuration for the accounts in scope. This list is not exhaustive:

- `EnableAllowedImagesSettings`
- `ReplaceImageCriteriaInAllowedImagesSettings`
- `DisableAllowedImagesSettings`

Instance Metadata Defaults

Policy effect

Controls IMDS defaults for all new EC2 instance launches. For more information about IMDS defaults, see IMDS in the *Amazon Elastic Compute Cloud User Guide*.

Policy contents

The following are the available fields for this attribute:

```
"instance_metadata_defaults": {
    "http_tokens": { // (required)
        "@@assign": "required" // no_preference | required | optional
    },
    "http_put_response_hop_limit": { // (required)
        "@@assign": "4" // -1 | 1 -> 64
    },
    "http_endpoint": { // (required)
        "@@assign": "enabled" // no_preference | enabled | disabled
    },
    "instance_metadata_tags": { // (required)
        "@@assign": "enabled" // no_preference | enabled | disabled
    }
}
```

- `"http_tokens"`:
  - `"no_preference"`: Other defaults apply. For example, AMI defaults if applicable.

- `"required"`: IMDSv2 must be used. IMDSv1 is not allowed.
- `"optional"`: Both IMDSv1 and IMDSv2 are allowed.

> **Note**
>
> **Metadata version**
> Before setting `http_tokens` to `required` (IMDSv2 must be used), make sure that none of your instances are making IMDSv1 calls.

- `"http_put_response_hop_limit"`:
  - `"Integer"`: Integer value from -1 to 64, representing the maximum number of hops the metadata token can travel. To indicate no preference, specify -1.

> **Note**
>
> **Hop limit**
> If `http_tokens` is set to `required`, it is recommended to set `http_put_response_hop_limit` to a minimum of 2. For more information, see Instance metadata access considerations in the *Amazon Elastic Compute Cloud User Guide*.

- `"http_endpoint"`:
  - `"no_preference"`: Other defaults apply. For example, AMI defaults if applicable.
  - `"enabled"`: The instance metadata service endpoint is accessible.
  - `"disabled"`: The instance metadata service endpoint is not accessible.
- `"instance_metadata_tags"`:
  - `"no_preference"`: Other defaults apply. For example, AMI defaults if applicable.
  - `"enabled"`: Instance tags can be accessed from instance metadata.
  - `"disabled"`: Instance tags cannot be accessed from instance metadata.

Snapshot Block Public Access

Policy effect

Controls if Amazon EBS snapshots are publicly accessible. For more information about EBS snapshots, see Amazon EBS snapshots in the *Amazon Elastic Block Store User Guide*.

Policy contents

```
"snapshot_block_public_access": {
    "state": { // (required)
        "@@assign": "block_new_sharing" // unblocked | block_new_sharing |
 block_all_sharing
    }
}
```

The following are the available fields for this attribute:

- `"state"`:
  - `"block_all_sharing"`: Blocks all public sharing of snapshots. Snapshots that were already publicly shared are treated as private and are no longer publicly available.
  - `"block_new_sharing"`: Blocks new public sharing of snapshots. Snapshots that were already publicly shared remain publicly available.
  - `"unblocked"`: No restrictions on the public sharing of snapshots.

Considerations

If you use this attribute in a declarative policy, you cannot use the following operations to modify the enforced configuration for the accounts in scope. This list is not exhaustive:

- `EnableSnapshotBlockPublicAccess`
- `DisableSnapshotBlockPublicAccess`

Backup policies

Backup policies allow you to centrally manage and apply backup plans to the AWS resources across an organization's accounts.

AWS Backup enables you to create backup plans that define how to back up your AWS resources. The rules in the plan include a variety of settings, such as the backup frequency, the time window during which the backup occurs, the AWS Region containing the resources to back up and the vault in which to store the backup. You can then apply a backup plan to groups of AWS resources identified by using tags. You must also identify an AWS Identity and Access Management (IAM) role that grants AWS Backup permission to perform the backup operation on your behalf.

Backup policies in AWS Organizations combine all of those pieces into JSON text documents. You can attach a backup policy to any of the elements in your organization's structure, such as the root, organizational units (OUs), and individual accounts. Organizations applies inheritance rules to combine the policies in the organization's root, any parent OUs, or attached to the account. This results in an effective backup policy for each account. This effective policy instructs AWS Backup how to automatically back up your AWS resources.

How backup policies work

Backup policies give you granular control over backing up your resources at whatever level your organization requires. For example, you can specify in a policy attached to the organization's root that all Amazon DynamoDB tables must be backed up. That policy can include a default backup frequency. You can then attach a backup policy to OUs that override the backup frequency according to the requirements of each OU. For example, the `Developers` OU might specify a backup frequency of once per week, while the `Production` OU specifies once per day.

You can create partial backup policies that individually include only part of the required information to successfully back up your resources. You can attach these policies to different parts of the organization tree, such as the root or a parent OU, with the intention of those partial policies being inherited by lower-level OUs and accounts. When Organizations combines all of the policies for an account by using inheritance rules, the resulting effective policy must have all the required elements. Otherwise, AWS Backup considers the policy not valid and does not back up the affected resources.

> **Important**
>
> AWS Backup can only perform a successful backup when it is invoked by a *complete* effective policy that has all of the required elements.
> Although a partial policy strategy as described earlier can work, if an effective policy for an account is incomplete, it results in errors or resources that are not successfully backed up. As an alternate strategy, consider requiring that all backup policies be complete and valid by themselves. Use *default* values supplied by policies attached higher in the hierarchy, and override them where needed in child policies by including inheritance child control operators.

The effective backup plan for each AWS account in the organization appears in the AWS Backup console as an immutable plan for that account. You can view it, but not change it. You can, however, add or remove backup plan tags using TagResource and UntagResource APIs.

When AWS Backup begins a backup based on a policy-created backup plan, you can see the status of the backup job in the AWS Backup console. A user in a member account can see the status and any errors for the backup jobs in that member account. If you also enable trusted service access with AWS Backup, a user in the organization's management account can see the status and errors for all backup jobs in the organization. For more information, see Enabling cross-account management in the *AWS Backup Developer Guide*.

Getting started with backup policies

Follow these steps to get started using backup policies.

1. Learn about the permissions you must have to perform backup policy tasks.
2. Learn about some best practices we recommend when using backup policies.
3. Enable backup policies for your organization.
4. Create a backup policy.
5. Attach the backup policy to your organization's root, OU, or account.
6. View the combined effective backup policy that applies to an account.

For all of these steps, you sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.

Other information

- Learn backup policy syntax and see example policies

Best practices for using backup policies

AWS recommends the following best practices for using backup policies.

Decide on a backup policy strategy

You can create backup policies in incomplete pieces that are inherited and merged to make a complete policy for each member account. If you do this, you risk ending up with an effective

policy that is not complete if you make a change at one level without carefully considering the change's impact on all accounts below that level. To prevent this, we recommend that you instead ensure that the backup policies you implement at all levels are complete by themselves. Treat the parent policies as default policies that can be overridden by settings specified in child policies. That way, even if a child policy doesn't exist, the inherited policy is complete and uses the default values. You can control which settings can be added to, changed, or removed by child policies by using the child control inheritance operators.

Validate changes to your backup policies checking using `GetEffectivePolicy`

After you make a change to a backup policy, check the effective policies for representative accounts below the level where you made the change. You can view the effective policy by using the AWS Management Console, or by using the GetEffectivePolicy API operation or one of its AWS CLI or AWS SDK variants. Ensure that the change you made had the intended impact on the effective policy.

Start simply and make small changes

To simplify debugging, start with simple policies and make changes one item at a time. Validate the behavior and impact of each change before making the next change. This approach reduces the number of variables you have to account for when an error or unexpected result does happen.

Store copies of your backups in other AWS Regions and accounts in your organization

To improve your disaster recovery position, you can store copies of your backups.

- **A different region** – If you store copies of the backup in additional AWS Regions, you help protect the backup against accidental corruption or deletion in the original Region. Use the `copy_actions` section of the policy to specify a vault in one or more Regions of the same account in which the backup plan runs. To do this, identify the account by using the `$account` variable when you specify the ARN of the backup vault in which to store the copy of the backup. The `$account` variable is automatically replaced at run time with the account ID in which the backup policy is running.
- **A different account** – If you store copies of the backup in additional AWS accounts, you add a security barrier that helps protect against a malicious actor who compromises one of your accounts. Use the `copy_actions` section of the policy to specify a vault in one or more accounts in your organization, separate from the account in which the backup plan runs . To do this, identify the account by using its actual account ID number when you specify the ARN of the backup vault in which to store the copy of the backup.

Limit the number of plans per policy

Policies that contain multiple plans are more complicated to troubleshoot because of the larger number of outputs that must all be validated. Instead, have each policy contain one and only one backup plan to simplify debugging and troubleshooting. You can then add additional policies with other plans to meet other requirements. This approach helps keep any issues with a plan isolated to one policy, and it prevents those issues from complicating the troubleshooting of issues with other policies and their plans.

Use stack sets to create the required backup vaults and IAM roles

Use AWS CloudFormation stack sets integration with Organizations to automatically create the required backup vaults and AWS Identity and Access Management (IAM) roles in each of the member accounts in your organization. You can create a stack set that includes the resources you want automatically available in every AWS account in your organization. This approach enables you to run your backup plans with assurance that the dependencies are already met. For more information, see Create a Stack Set with Self-Managed Permissions in the *AWS CloudFormation User Guide*.

Check your results by reviewing the first backup created in each account

When you make a change to a policy, check the next backup created after that change to ensure the change had the desired impact. This step goes beyond looking at the effective policy and ensures that AWS Backup interprets your policies and implements the backup plans the way you intended.

Using AWS CloudTrail events to monitor backup policies in your organization

You can use AWS CloudTrail events to monitor when backup policies are created, updated, or deleted from any accounts in your organization, or when there is an invalid organizational backup plan. For more information, see Logging cross-account management events in the *AWS Backup Developer Guide*.

Backup policy syntax and examples

This page describes backup policy syntax and provides examples.

Syntax for backup policies

A backup policy is a plaintext file that is structured according to the rules of JSON. The syntax for backup policies follows the syntax for all management policy types. For more information, see

Policy syntax and inheritance for management policy types. This topic focuses on applying that general syntax to the specific requirements of the backup policy type.

For more information about AWS Backup plans, see CreateBackupPlan in the *AWS Backup Developer Guide*.

Considerations

Policy syntax

Duplicate key names will be rejected in JSON.

Policies must specify the AWS Regions and resources to be backed up.

Policies must specify the IAM role that AWS Backup assumes.

Using `@@assign` operator at the same level can overwrite existing settings. For more information, see A child policy overrides settings in a parent policy.

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

For more information, see Inheritance operators and Backup policy examples.

IAM roles

The IAM role must exist when creating a backup plan for the first time.

The IAM role must have permission to access resources identified by tag query.

The IAM role must have permission to perform the backup.

Backup vaults

Vaults must exist in each specified AWS Regions before a backup plan can run.

Vaults must exist for each AWS account that receives the effective policy. For more information, see Backup vault creation and deletion in the *AWS Backup Developer Guide*.

We recommend that you use AWS CloudFormation stack sets and its integration with Organizations to automatically create and configure backup vaults and IAM roles for each member account in the organization. For more information, see Create a stack set with self-managed permissions in the *AWS CloudFormation User Guide*.

Quotas

For a list of quotas see, AWS Backup quotas in the *AWS Backup Developer Guide*.

Backup syntax: Overview

Backup policy syntax includes the following components:

```
{
    "plans": {
        "PlanName": {
            "rules": { ... },
            "regions": { ... },
            "selections": { ... },
            "advanced_backup_settings": { ... },
            "backup_plan_tags": { ... }
        }
    }
}
```

Backup policy elements

Element	Description	Required
rules	List of backup rules. Each rule defines when backups start and the execution window for the resources specified in the `regions` and `selections` elements.	Yes
regions	List of AWS Regions where a backup policy can protect resources.	Yes
selections	One or more resource types within the specified `regions` that the backup `rules` protect.	Yes
advanced_backup_settings	Configuration options for specific backup scenarios. Currently, the only advanced backup setting that is supported is enabling Microsoft Volume Shadow Copy Service (VSS) backups	No

Element	Description	Required
	for Windows or SQL Server running on an Amazon EC2 instance.	
backup_plan_tags	Tags you want to associate with a backup plan. Each tag is a label consisting of a user-defined key and value. Tags can help you manage, identify, organize, search for, and filter your backup plans.	No

Backup syntax: rules

The `rules` policy key specifies the scheduled backup tasks that AWS Backup performs on the selected resources.

Backup rule elements

Element	Description	Required
`schedule_expression`	Cron expression in UTC that specifies when AWS Backup initiates a backup job. For information about cron expression, see Using cron and rate expressions to schedule rules in the *User Guide*	Yes
`target_backup_vault_name`	Backup vault where backups are stored. Backup vaults are identified by names that are unique to the account used to create them and the AWS Region where they are created.	Yes
`start_backup_window_minutes`	Number of minutes to wait before canceling a backup job will be canceled if it doesn't start successfully. If this value is included, it must be at least 60 minutes to avoid errors.	No

Element	Description	Required
`complete_backup_window_minutes`	Nnumber of minutes after a backup job is successfully started before it must be completed or it will be canceled by AWS Backup.	No
`enable_continuous_backup`	Specifies whether AWS Backup creates continuous backups. `True` causes AWS Backup to create continuous backups capable of point-in-time restore (PITR). `False` (or not specified) causes AWS Backup to create snapshot backups. For more information about continuous backups, see Point-in-time recovery in the *AWS Backup Developer Guide*. **Note:** PITR-enabled backups have 35-day maximum retention.	No

Element	Description	Required
`lifecycle`	Specifies when AWS Backup transitions a backup to cold storage and when it expires. Resource types that can transition to cold storage are listed in the Feature availability by resource table Feature availability by resources in the *AWS Backup Developer Guide*. Each lifecycle contains the following elements: • `move_to_cold_storage_after_days` : Number of days after the backup occurs before AWS Backup moves the recovery point to cold storage. • `delete_after_days` : Number of days after a backup occurs before AWS Backup deletes the recovery point. • `opt_in_to_archive_for_supported_resources` : If this value is assigned as `true`, a backup plan transitions supported resources to archive (cold) storage tier in accordance with your lifecycle settings. **Note**: Backups transitioned to cold storage must be stored in cold storage for a minimum of 90 days. This means that the `delete_after_days` must be 90 days greater than `move_to_cold_storage_after_days` .	No

Element	Description	Required

`copy_actions`	Specifies whether AWS Backup copies a backup to one or more additional locations.
	
	Each copy action contains the following elements:
	
	• `target_backup_vault_arn` : Vault where AWS Backup stores an additional copy of the backup.
	  • Use `$account` for same-account copies
	  • Use actual account ID for cross-account copies
	• `lifecycle` : Specifies when AWS Backup transitions a backup to cold storage and when it expires.
	
	  Each lifecycle contains the following elements:
	  • `move_to_cold_storage_after_days` : Number of days after the backup occurs before AWS Backup moves the recovery point to cold storage.
	  • `delete_after_days` : Number of days after s backup occurs before AWS Backup deletes the recovery point.
	
	Note: Backups transitioned to cold storage must be stored in cold storage for a minimum of 90 days.
	
	This means that the `delete_after_days` must be 90 days greater than `move_to_cold_storage_after_days` .
		No

Element	Description	Required
`recovery_point_tags`	Tags that you want to assigned to resources that are restored from backup. Each tag contains the following elements: • `tag_key`: Tag name (case-sensitive) • `tag_value` : Tag value (case-sensitive)	No

Backup syntax: regions

The `regions` policy key specifies which AWS Regions that AWS Backup looks in to find the resources that match the conditions in the `selections` key.

Backup regions elements

Element	Description	Required
`regions`	Specifies the AWS Region codes. For example: `["us-east-1", "eu-north-1"]` .	Yes

Backup syntax: selections

The `selections` policy key specifies the resources that are backed up by the rules in a backup policy.

There are two mutually exclusive elements: `tags` and `resources`. An effective policy **must** `have` either tags or `resources` in the selection to be valid.

If you want a selection with both tag conditions and resource conditions, use the `resources` keys.

Backup selection elements: Tags

Element	Description	Required
`iam_role_arn`	IAM role that AWS Backup assumes to query, discover, and backup resources across the specified Regions.	Yes

Element	Description	Required
	The role must have sufficient permissions to query resources based on tag conditions and perform backup operations on the matched resources.	
`tag_key`	Tag key name to search for.	Yes
`tag_value`	Value that must be associated with the matching tag_key. AWS Backup includes the resource only if both tag_key and tag_value match (case sensitive).	Yes
`conditions`	Tag keys and values you want to include or exclude Use string_equals or string_not_equals to include or exclude tags of an exact match. Use string_like and string_not_like to include or exclude tags that contains or do not contain specific characters **Note:** Limited to 30 conditions for each selection.	No

Backup selection elements: Resources

Element	Description	Required
`iam_role_arn`	IAM role that AWS Backup assumes to query, discover, and backup resources across the specified Regions. The role must have sufficient permissions to query resources based on tag conditions and	Yes

Element	Description	Required
	perform backup operations on the matched resources. **Note:** In AWS GovCloud (US) Regions, you must add the name of the partition to the ARN. For example, "`arn:aws:ec2:*:*:volume/*` " must be "`arn:aws-us-gov:ec2:*:*:volume/*` ".	
`resource_types`	Resource types to include in a backup plan.	Yes
`not_resource_types`	Resource types to exclude from a backup plan.	No
`conditions`	Tag keys and values you want to include or exclude Use string_equals or string_not_equals to include or exclude tags of an exact match. Use string_like and string_not_like to include or exclude tags that contains or do not contain specific characters **Note:** Limited to 30 conditions for each selection.	No

Supported resource types

Organizations supports the following resource types for the `resource_types` and `not_resource_types` elements:

- AWS Backup gateway virtual machines: "`arn:aws:backup-gateway:*:*:vm/*`"
- AWS CloudFormation stacks: "`arn:aws:cloudformation:*:*:stack/*`"
- Amazon DynamoDB tables: "`arn:aws:dynamodb:*:*:table/*`"

- Amazon EC2 instances: `"arn:aws:ec2:*:*:instance/*"`
- Amazon EBS volumes: `"arn:aws:ec2:*:*:volume/*"`
- Amazon EFS file systems: `"arn:aws:elasticfilesystem:*:*:file-system/*"`
- Amazon Aurora/Amazon DocumentDB/Amazon Neptune clusters: `"arn:aws:rds:*:*:cluster:*"`
- Amazon RDS databases: `"arn:aws:rds:*:*:db:*"`
- Amazon Redshift clusters: `"arn:aws:redshift:*:*:cluster:*"`
- Amazon S3: `"arn:aws:s3:::*"`
- AWS Systems Manager for SAP HANA databases: `"arn:aws:ssm-sap:*:*:HANA/*"`
- AWS Storage Gateway gateways: `"arn:aws:storagegateway:*:*:gateway/*"`
- Amazon Timestream databases: `"arn:aws:timestream:*:*:database/*"`
- Amazon FSx file systems: `"arn:aws:fsx:*:*:file-system/*"`
- Amazon FSx volumes: `"arn:aws:fsx:*:*:volume/*"`

Code examples

For more information, see Specifying resources with the tags block and Specifying resources with the resources block.

Backup syntax: advanced backup settings

The `advanced_backup_settings` key specifies the configuration options for specific backup scenarios. Each setting contains the following elements:

Advanced backup settings elements

Element	Description	Required
`advanced_backup_settings`	Specifies settings for specific backup scenarios. This key contains one or more settings. Each setting is a JSON object string with the following elements: Currently the only advanced backup setting that is	No

Element	Description	Required
	supported is enabling Microsoft Volume Shadow Copy Service (VSS) backups for Windows or SQL Server running on an Amazon EC2 instance. Each advanced backup setting the following elements: • `Object key name`: String that specifies the type of resource to which the following advanced settings apply. The key name must be the `"ec2"` resource type • `Object value`: Dtring that contains one or more backup settings specific to the associated resource type. The value specifies that `"windows_vss"` support is either `enabled` or `disabled` for backups performed on the Amazon EC2 instances.	

Example:

```
"advanced_backup_settings": {
    "ec2": {
```

```
        "windows_vss": {
            "@@assign": "enabled"
        }
    }
},
```

Backup syntax: backup plan tags

The `backup_plan_tags` policy key specifies the tags that are attached to a backup plan itself. This does not impact the tags specified for `rules` or `selections`.

Backup plan tag elements

Element	Description	Required
`backup_plan_tags`	Each tag is a label consisting of a user-defined key and value: • `tag_key`: Tag key name to search for. The value is case sensitive. • `tag_value` : Value that is attached to the backup plan and associated with the `tag_key`. The value is case sensitive.	No

Backup policy examples

The example backup policies that follow are for information purposes only. In some of the following examples, the JSON whitespace formatting might be compressed to save space.

- Example 1: Policy assigned to a parent node
- Example 2: A parent policy is merged with a child policy
- Example 3: A parent policy prevents any changes by a child policy
- Example 4: A parent policy prevents changes to one backup plan by a child policy
- Example 5: A child policy overrides settings in a parent policy
- Example 6: Specifying resources with the tags block
- Example 7: Specifying resources with the resources block

Example 1: Policy assigned to a parent node

The following example shows a backup policy that is assigned to one of the parent nodes of an account.

Parent policy – This policy can be attached to the organization's root, or to any OU that is a parent of all of the intended accounts.

```
{
    "plans": {
        "PII_Backup_Plan": {
            "regions": {
                "@@assign": [
                    "ap-northeast-2",
                    "us-east-1",
                    "eu-north-1"
                ]
            },
            "rules": {
                "Hourly": {
                    "schedule_expression": {
                        "@@assign": "cron(0 5/1 ? * * *)"
                    },
                    "start_backup_window_minutes": {
                        "@@assign": "480"
                    },
                    "complete_backup_window_minutes": {
                        "@@assign": "10080"
                    },
                    "lifecycle": {
                        "move_to_cold_storage_after_days": {
                            "@@assign": "180"
                        },
                        "delete_after_days": {
                            "@@assign": "270"
                        },
                        "opt_in_to_archive_for_supported_resources": {
                            "@@assign": "false"
                        }
                    },
                    "target_backup_vault_name": {
                        "@@assign": "FortKnox"
                    },
```

```
                    "copy_actions": {
                        "arn:aws:backup:us-east-1:$account:backup-
vault:secondary_vault": {
                            "target_backup_vault_arn": {
                                "@@assign": "arn:aws:backup:us-east-1:$account:backup-
vault:secondary_vault"
                            },
                            "lifecycle": {
                                "move_to_cold_storage_after_days": {
                                    "@@assign": "30"
                                },
                                "delete_after_days": {
                                    "@@assign": "120"
                                },
                                "opt_in_to_archive_for_supported_resources": {
                                    "@@assign": "false"
                                }
                            }
                        },
                        "arn:aws:backup:us-west-1:111111111111:backup-
vault:tertiary_vault": {
                            "target_backup_vault_arn": {
                                "@@assign": "arn:aws:backup:us-
west-1:111111111111:backup-vault:tertiary_vault"
                            },
                            "lifecycle": {
                                "move_to_cold_storage_after_days": {
                                    "@@assign": "30"
                                },
                                "delete_after_days": {
                                    "@@assign": "120"
                                },
                                "opt_in_to_archive_for_supported_resources": {
                                    "@@assign": "false"
                                }
                            }
                        }
                    }
                }
            },
            "selections": {
                "tags": {
                    "datatype": {
                        "iam_role_arn": {
```

```
                        "@@assign": "arn:aws:iam::$account:role/MyIamRole"
                    },
                    "tag_key": {
                        "@@assign": "dataType"
                    },
                    "tag_value": {
                        "@@assign": [
                            "PII",
                            "RED"
                        ]
                    }
                }
            }
        },
        "advanced_backup_settings": {
            "ec2": {
                "windows_vss": {
                    "@@assign": "enabled"
                }
            }
        }
    }
}
```

If no other policies are inherited or attached to the accounts, the effective policy rendered in each applicable AWS account looks like the following example. The CRON expression causes the backup to run once an hour on the hour. The account ID 123456789012 will be the actual account ID for each account.

```
{
    "plans": {
        "PII_Backup_Plan": {
            "regions": [
                "us-east-1",
                "ap-northeast-3",
                "eu-north-1"
            ],
            "rules": {
                "hourly": {
                    "schedule_expression": "cron(0 0/1 ? * * *)",
                    "start_backup_window_minutes": "60",
                    "target_backup_vault_name": "FortKnox",
```

```
                    "lifecycle": {
                        "delete_after_days": "2",
                        "move_to_cold_storage_after_days": "180",
                        "opt_in_to_archive_for_supported_resources": "false"
                    },
                    "copy_actions": {
                        "arn:aws:backup:us-east-1:$account:backup-
vault:secondary_vault": {
                            "target_backup_vault_arn": {
                                "@@assign": "arn:aws:backup:us-east-1:$account:backup-
vault:secondary_vault"
                            },
                            "lifecycle": {
                                "delete_after_days": "28",
                                "move_to_cold_storage_after_days": "180",
                                "opt_in_to_archive_for_supported_resources": "false"
                            }
                        },
                        "arn:aws:backup:us-west-1:111111111111:backup-
vault:tertiary_vault": {
                            "target_backup_vault_arn": {
                                "@@assign": "arn:aws:backup:us-
west-1:111111111111:backup-vault:tertiary_vault"
                            },
                            "lifecycle": {
                                "delete_after_days": "28",
                                "move_to_cold_storage_after_days": "180",
                                "opt_in_to_archive_for_supported_resources": "false"
                            }
                        }
                    }
                }
            },
            "selections": {
                "tags": {
                    "datatype": {
                        "iam_role_arn": "arn:aws:iam::123456789012:role/MyIamRole",
                        "tag_key": "dataType",
                        "tag_value": [
                            "PII",
                            "RED"
                        ]
                    }
                }
```

```
            },
            "advanced_backup_settings": {
                "ec2": {
                    "windows_vss": "enabled"
                }
            }
        }
    }
}
```

Example 2: A parent policy is merged with a child policy

In the following example, an inherited parent policy and a child policy either inherited or directly attached to an AWS account merge to form the effective policy.

Parent policy – This policy can be attached to the organization's root or to any parent OU.

```
{
    "plans": {
       "PII_Backup_Plan": {
            "regions": { "@@append":[ "us-east-1", "ap-northeast-3", "eu-north-1" ] },
            "rules": {
                "Hourly": {
                    "schedule_expression": { "@@assign": "cron(0 0/1 ? * * *)" },
                    "start_backup_window_minutes": { "@@assign": "60" },
                    "target_backup_vault_name": { "@@assign": "FortKnox" },
                    "lifecycle": {
                        "move_to_cold_storage_after_days": { "@@assign": "28" },
                        "delete_after_days": { "@@assign": "180" },
                        "opt_in_to_archive_for_supported_resources": { "@@assign":
 "false" }
                    },
                    "copy_actions": {
                        "arn:aws:backup:us-east-1:$account:backup-
vault:secondary_vault" : {
                            "target_backup_vault_arn" : {
                                "@@assign" : "arn:aws:backup:us-east-1:$account:backup-
vault:secondary_vault"
                            },
                            "lifecycle": {
                                "move_to_cold_storage_after_days": { "@@assign":
 "28" },
                                "delete_after_days": { "@@assign": "180" },
```

```
                                "opt_in_to_archive_for_supported_resources":
 { "@@assign": "false" }
                            }
                        }
                    }
                }
            },
            "selections": {
                "tags": {
                    "datatype": {
                        "iam_role_arn": { "@@assign": "arn:aws:iam::$account:role/
MyIamRole" },
                        "tag_key": { "@@assign": "dataType" },
                        "tag_value": { "@@assign": [ "PII", "RED" ] }
                    }
                }
            }
        }
    }
}
```

Child policy – This policy can be attached directly to the account or to an OU any level below the one the parent policy is attached to.

```
{
    "plans": {
       "Monthly_Backup_Plan": {
            "regions": {
                "@@append":[ "us-east-1", "eu-central-1" ] },
            "rules": {
                "Monthly": {
                    "schedule_expression": { "@@assign": "cron(0 5 1 * ? *)" },
                    "start_backup_window_minutes": { "@@assign": "480" },
                    "target_backup_vault_name": { "@@assign": "Default" },
                    "lifecycle": {
                        "move_to_cold_storage_after_days": { "@@assign": "30" },
                        "delete_after_days": { "@@assign": "365" },
                        "opt_in_to_archive_for_supported_resources": { "@@assign":
 "false" }
                    },
                    "copy_actions": {
                        "arn:aws:backup:us-east-1:$account:backup-vault:Default" : {
                            "target_backup_vault_arn" : {
```

```
                                "@@assign" : "arn:aws:backup:us-east-1:$account:backup-
vault:Default"
                            },
                            "lifecycle": {
                                "move_to_cold_storage_after_days": { "@@assign":
 "30" },
                                "delete_after_days": { "@@assign": "365" },
                                "opt_in_to_archive_for_supported_resources":
 { "@@assign": "false" }
                            }
                        }
                    }
                },
                "selections": {
                    "tags": {
                        "MonthlyDatatype": {
                            "iam_role_arn": { "@@assign": "arn:aws:iam::$account:role/
MyMonthlyBackupIamRole" },
                            "tag_key": { "@@assign": "BackupType" },
                            "tag_value": { "@@assign": [ "MONTHLY", "RED" ] }
                        }
                    }
                }
            }
        }
}
```

Resulting effective policy – The effective policy applied to the accounts contains two plans, each with its own set of rules and set of resources to apply the rules to.

```
{
    "plans": {
       "PII_Backup_Plan": {
            "regions": [ "us-east-1", "ap-northeast-3", "eu-north-1" ],
            "rules": {
                "hourly": {
                    "schedule_expression": "cron(0 0/1 ? * * *)",
                    "start_backup_window_minutes": "60",
                    "target_backup_vault_name": "FortKnox",
                    "lifecycle": {
                        "delete_after_days": "2",
                        "move_to_cold_storage_after_days": "180",
```

```
                        "opt_in_to_archive_for_supported_resources": { "@@assign":
 "false" }
                    },
                    "copy_actions": {
                        "arn:aws:backup:us-east-1:$account:backup-
vault:secondary_vault" : {
                            "target_backup_vault_arn" : {
                                "@@assign" : "arn:aws:backup:us-east-1:$account:backup-
vault:secondary_vault"
                            },
                            "lifecycle": {
                                "move_to_cold_storage_after_days": "28",
                                "delete_after_days": "180",
                                "opt_in_to_archive_for_supported_resources":
 { "@@assign": "false" }
                            }
                        }
                    }
                },
                "selections": {
                    "tags": {
                        "datatype": {
                            "iam_role_arn": "arn:aws:iam::$account:role/MyIamRole",
                            "tag_key": "dataType",
                            "tag_value": [ "PII", "RED" ]
                        }
                    }
                }
            },
            "Monthly_Backup_Plan": {
                "regions": [ "us-east-1", "eu-central-1" ],
                "rules": {
                    "monthly": {
                        "schedule_expression": "cron(0 5 1 * ? *)",
                        "start_backup_window_minutes": "480",
                        "target_backup_vault_name": "Default",
                        "lifecycle": {
                            "delete_after_days": "365",
                            "move_to_cold_storage_after_days": "30",
                            "opt_in_to_archive_for_supported_resources": { "@@assign":
 "false" }
                        },
                        "copy_actions": {
```

```
                        "arn:aws:backup:us-east-1:$account:backup-vault:Default" : {
                            "target_backup_vault_arn": {
                                "@@assign" : "arn:aws:backup:us-east-1:$account:backup-
vault:Default"
                            },
                            "lifecycle": {
                                "move_to_cold_storage_after_days": "30",
                                "delete_after_days": "365",
                                "opt_in_to_archive_for_supported_resources":
 { "@@assign": "false" }
                            }
                        }
                    }
                }
            },
            "selections": {
                "tags": {
                    "monthlydatatype": {
                        "iam_role_arn": "arn:aws:iam::&ExampleAWSAccountNo3;:role/
MyMonthlyBackupIamRole",
                        "tag_key": "BackupType",
                        "tag_value": [ "MONTHLY", "RED" ]
                    }
                }
            }
        }
    }
}
```

Example 3: A parent policy prevents any changes by a child policy

In the following example, an inherited parent policy uses the child control operators to enforce all settings and prevents them from being changed or overridden by a child policy.

Parent policy – This policy can be attached to the organization's root or to any parent OU. The presence of `"@@operators_allowed_for_child_policies": ["@@none"]` at every node of the policy means that a child policy can't make changes of any kind to the plan. Nor can a child policy add additional plans to the effective policy. This policy becomes the effective policy for every OU and account under the OU to which it is attached.

```
{
    "plans": {
        "@@operators_allowed_for_child_policies": ["@@none"],
```

```
        "PII_Backup_Plan": {
            "@@operators_allowed_for_child_policies": ["@@none"],
            "regions": {
                "@@operators_allowed_for_child_policies": ["@@none"],
                "@@append": [
                    "us-east-1",
                    "ap-northeast-3",
                    "eu-north-1"
                ]
            },
            "rules": {
                "@@operators_allowed_for_child_policies": ["@@none"],
                "Hourly": {
                    "@@operators_allowed_for_child_policies": ["@@none"],
                    "schedule_expression": {
                        "@@operators_allowed_for_child_policies": ["@@none"],
                        "@@assign": "cron(0 0/1 ? * * *)"
                    },
                    "start_backup_window_minutes": {
                        "@@operators_allowed_for_child_policies": ["@@none"],
                        "@@assign": "60"
                    },
                    "target_backup_vault_name": {
                        "@@operators_allowed_for_child_policies": ["@@none"],
                        "@@assign": "FortKnox"
                    },
                    "lifecycle": {
                        "@@operators_allowed_for_child_policies": ["@@none"],
                        "move_to_cold_storage_after_days": {
                            "@@operators_allowed_for_child_policies": ["@@none"],
                            "@@assign": "28"
                        },
                        "delete_after_days": {
                            "@@operators_allowed_for_child_policies": ["@@none"],
                            "@@assign": "180"
                        },
                        "opt_in_to_archive_for_supported_resources": {
                            "@@operators_allowed_for_child_policies": ["@@none"],
                            "@@assign": "false"
                        }
                    },
                    "copy_actions": {
                        "@@operators_allowed_for_child_policies": ["@@none"],
```

```
                        "arn:aws:backup:us-east-1:$account:backup-
vault:secondary_vault": {
                            "@@operators_allowed_for_child_policies": ["@@none"],
                            "target_backup_vault_arn": {
                                "@@assign": "arn:aws:backup:us-east-1:$account:backup-
vault:secondary_vault",
                                "@@operators_allowed_for_child_policies": ["@@none"]
                            },
                            "lifecycle": {
                                "@@operators_allowed_for_child_policies": ["@@none"],
                                "delete_after_days": {
                                    "@@operators_allowed_for_child_policies":
 ["@@none"],
                                    "@@assign": "28"
                                },
                                "move_to_cold_storage_after_days": {
                                    "@@operators_allowed_for_child_policies":
 ["@@none"],
                                    "@@assign": "180"
                                },
                                 "opt_in_to_archive_for_supported_resources": {
                                    "@@operators_allowed_for_child_policies":
 ["@@none"],
                                    "@@assign": "false"
                                }
                            }
                        }
                    }
                }
            },
            "selections": {
                "@@operators_allowed_for_child_policies": ["@@none"],
                "tags": {
                    "@@operators_allowed_for_child_policies": ["@@none"],
                    "datatype": {
                        "@@operators_allowed_for_child_policies": ["@@none"],
                        "iam_role_arn": {
                            "@@operators_allowed_for_child_policies": ["@@none"],
                            "@@assign": "arn:aws:iam::$account:role/MyIamRole"
                        },
                        "tag_key": {
                            "@@operators_allowed_for_child_policies": ["@@none"],
                            "@@assign": "dataType"
                        },
```

```
                        "tag_value": {
                            "@@operators_allowed_for_child_policies": ["@@none"],
                            "@@assign": [
                                "PII",
                                "RED"
                            ]
                        }
                    }
                }
            },
            "advanced_backup_settings": {
                "@@operators_allowed_for_child_policies": ["@@none"],
                "ec2": {
                    "@@operators_allowed_for_child_policies": ["@@none"],
                    "windows_vss": {
                        "@@assign": "enabled",
                        "@@operators_allowed_for_child_policies": ["@@none"]
                    }
                }
            }
        }
    }
}
```

Resulting effective policy – If any child backup policies exist, they are ignored and the parent policy becomes the effective policy.

```
{
    "plans": {
        "PII_Backup_Plan": {
            "regions": [
                "us-east-1",
                "ap-northeast-3",
                "eu-north-1"
            ],
            "rules": {
                "hourly": {
                    "schedule_expression": "cron(0 0/1 ? * * *)",
                    "start_backup_window_minutes": "60",
                    "target_backup_vault_name": "FortKnox",
                    "lifecycle": {
                        "delete_after_days": "2",
                        "move_to_cold_storage_after_days": "180",
```

```
                        "opt_in_to_archive_for_supported_resources": "false"
                    },
                    "copy_actions": {
                        "target_backup_vault_arn": "arn:aws:backup:us-
east-1:123456789012:backup-vault:secondary_vault",
                        "lifecycle": {
                            "move_to_cold_storage_after_days": "28",
                            "delete_after_days": "180",
                            "opt_in_to_archive_for_supported_resources": "false"
                        }
                    }
                }
            },
            "selections": {
                "tags": {
                    "datatype": {
                        "iam_role_arn": "arn:aws:iam::123456789012:role/MyIamRole",
                        "tag_key": "dataType",
                        "tag_value": [
                            "PII",
                            "RED"
                        ]
                    }
                }
            },
            "advanced_backup_settings": {
                "ec2": {"windows_vss": "enabled"}
            }
        }
    }
}
```

Example 4: A parent policy prevents changes to one backup plan by a child policy

In the following example, an inherited parent policy uses the child control operators to enforce the settings for a single plan and prevents them from being changed or overridden by a child policy. The child policy can still add additional plans.

Parent policy – This policy can be attached to the organization's root or to any parent OU. This example is similar to the previous example with all child inheritance operators blocked, except at the `plans` top level. The `@@append` setting at that level enables child policies to add other plans to the collection in the effective policy. Any changes to the inherited plan are still blocked.

The sections in the plan are truncated for clarity.

```
{
    "plans": {
        "@@operators_allowed_for_child_policies": ["@@append"],
        "PII_Backup_Plan": {
            "@@operators_allowed_for_child_policies": ["@@none"],
            "regions": { ... },
            "rules": { ... },
            "selections": { ... }
        }
    }
}
```

Child policy – This policy can be attached directly to the account or to an OU any level below the one the parent policy is attached to. This child policy defines a new plan.

The sections in the plan are truncated for clarity.

```
{
    "plans": {
        "MonthlyBackupPlan": {
            "regions": { ... },
            "rules": { ... },
            "selections": { … }
        }
    }
}
```

Resulting effective policy – The effective policy includes both plans.

```
{
    "plans": {
        "PII_Backup_Plan": {
            "regions": { ... },
            "rules": { ... },
            "selections": { ... }
        },
        "MonthlyBackupPlan": {
            "regions": { ... },
            "rules": { ... },
            "selections": { … }
```

```
        }
    }
}
```

Example 5: A child policy overrides settings in a parent policy

In the following example, a child policy uses value-setting operators to override some of the settings inherited from a parent policy.

Parent policy – This policy can be attached to the organization's root or to any parent OU. Any of the settings can be overridden by a child policy because the default behavior, in the absence of a child-control operator that prevents it, is to allow the child policy to `@@assign`, `@@append`, or `@@remove`. The parent policy contains all of the required elements for a valid backup plan, so it backs up your resources successfully if it is inherited as is.

```
{
    "plans": {
        "PII_Backup_Plan": {
            "regions": {
                "@@append": [
                    "us-east-1",
                    "ap-northeast-3",
                    "eu-north-1"
                ]
            },
            "rules": {
                "Hourly": {
                    "schedule_expression": {"@@assign": "cron(0 0/1 ? * * *)"},
                    "start_backup_window_minutes": {"@@assign": "60"},
                    "target_backup_vault_name": {"@@assign": "FortKnox"},
                    "lifecycle": {
                        "delete_after_days": {"@@assign": "2"},
                        "move_to_cold_storage_after_days": {"@@assign": "180"},
                        "opt_in_to_archive_for_supported_resources": {"@@assign":
 false}
                    },
                    "copy_actions": {
                        "arn:aws:backup:us-east-1:$account:backup-vault:t2": {
                            "target_backup_vault_arn": {"@@assign": "arn:aws:backup:us-
east-1:$account:backup-vault:t2"},
                            "lifecycle": {
                                "move_to_cold_storage_after_days": {"@@assign": "28"},
                                "delete_after_days": {"@@assign": "180"},
```

```
                                "opt_in_to_archive_for_supported_resources":
 {"@@assign": false}
                            }
                        }
                    }
                }
            },
            "selections": {
                "tags": {
                    "datatype": {
                        "iam_role_arn": {"@@assign": "arn:aws:iam::$account:role/
MyIamRole"},
                        "tag_key": {"@@assign": "dataType"},
                        "tag_value": {
                            "@@assign": [
                                "PII",
                                "RED"
                            ]
                        }
                    }
                }
            }
        }
    }
}
```

Child policy – The child policy includes only the settings that need to be different from the inherited parent policy. There must be an inherited parent policy that provides the other required settings when merged into an effective policy. Otherwise, the effective backup policy contains a backup plan that is not valid and doesn't back up your resources as expected.

```
{
    "plans": {
        "PII_Backup_Plan": {
            "regions": {
                "@@assign": [
                    "us-west-2",
                    "eu-central-1"
                ]
            },
            "rules": {
                "Hourly": {
                    "schedule_expression": {"@@assign": "cron(0 0/2 ? * * *)"},
```

```
                    "start_backup_window_minutes": {"@@assign": "80"},
                    "target_backup_vault_name": {"@@assign": "Default"},
                    "lifecycle": {
                        "move_to_cold_storage_after_days": {"@@assign": "30"},
                        "delete_after_days": {"@@assign": "365"},
                        "opt_in_to_archive_for_supported_resources": {"@@assign":
 false}
                    }
                }
            }
        }
    }
}
```

Resulting effective policy – The effective policy includes settings from both policies, with the settings provided by the child policy overriding the settings inherited from the parent. In this example, the following changes occur:

- The list of Regions is replaced with a completely different list. If you wanted to add a Region to the inherited list, use `@@append` instead of `@@assign` in the child policy.
- AWS Backup performs every other hour instead of hourly.
- AWS Backup allows 80 minutes for the backup to start instead of 60 minutes.
- AWS Backup uses the `Default` vault instead of `FortKnox`.
- The lifecycle is extended for both the transfer to cold storage and the eventual deletion of the backup.

```
{
    "plans": {
        "PII_Backup_Plan": {
            "regions": [
                "us-west-2",
                "eu-central-1"
            ],
            "rules": {
                "hourly": {
                    "schedule_expression": "cron(0 0/2 ? * * *)",
                    "start_backup_window_minutes": "80",
                    "target_backup_vault_name": "Default",
                    "lifecycle": {
                        "delete_after_days": "365",
```

```
                        "move_to_cold_storage_after_days": "30",
                        "opt_in_to_archive_for_supported_resources": "false"

                    },
                    "copy_actions": {
                        "arn:aws:backup:us-east-1:$account:backup-
vault:secondary_vault": {
                            "target_backup_vault_arn": {"@@assign": "arn:aws:backup:us-
east-1:$account:backup-vault:secondary_vault"},
                            "lifecycle": {
                                "move_to_cold_storage_after_days": "28",
                                "delete_after_days": "180",
                                "opt_in_to_archive_for_supported_resources": "false"
                            }
                        }
                    }
                }
            },
            "selections": {
                "tags": {
                    "datatype": {
                        "iam_role_arn": "arn:aws:iam::$account:role/MyIamRole",
                        "tag_key": "dataType",
                        "tag_value": [
                            "PII",
                            "RED"
                        ]
                    }
                }
            }
        }
    }
}
```

Example 6: Specifying resources with the `tags` block

The following example includes all resources with the `tag_key` = “`env`” and and `tag_value` = `"prod"` and `"gamma"`. This example excludes resources with the `tag_key` = `"backup"` and the `tag_value` = `"false"`.

```
...
"selections":{
    "tags":{
```

```
        "selection_name":{
            "iam_role_arn": {"@@assign": "arn:aws:iam::$account:role/IAMRole"},
            "tag_key":{"@@assign": "env"},
            "tag_value":{"@@assign": ["prod", "gamma"]},
            "conditions":{
                "string_not_equals":{
                    "condition_name1":{
                        "condition_key": { "@@assign": "aws:ResourceTag/backup"  },
                        "condition_value": {  "@@assign": "false" }
                    }
                }
            }
        }
    }
},
...
```

Example 7: Specifying resources with the `resources` block

The following are examples of using the `resources` block to specify resources.

Example: Select all resources in my account

The Boolean logic is similar to what you might use in IAM policies. The `"resource_types"` block uses a Boolean `AND` to combine the resource types.

```
...
"resources":{
    "resource_selection_name":{
        "iam_role_arn":{"@@assign": "arn:aws:iam::$account:role/IAMRole"},
        "resource_types":{
            "@@assign": [
                "*"
            ]
        }
    }
},
...
```

Example: Select all resources in my account, but exclude Amazon EBS volumes

The Boolean logic is similar to what you might use in IAM policies. The `"resource_types"` and `"not_resource_types"` blocks use a Boolean `AND` to combine the resource types.

```
...
"resources":{
    "resource_selection_name":{
        "iam_role_arn":{"@@assign": "arn:aws:iam::$account:role/IAMRole"},
        "resource_types":{
            "@@assign": [
                "*"
            ]
        },
        "not_resource_types":{
            "@@assign": [
                "arn:aws:ec2:*:*:volume/*"
            ]
        }
    }
},
...
```

Example: Select all resources tagged with "backup" : "true", but exclude Amazon EBS volumes

The Boolean logic is similar to what you might use in IAM policies. The `"resource_types"` and `"not_resource_types"` blocks use a Boolean `AND` to combine the resource types. The `"conditions"` block uses a Boolean `AND`.

```
...
"resources":{
    "resource_selection_name":{
        "iam_role_arn":{"@@assign": "arn:aws:iam::$account:role/IAMRole"},
        "resource_types":{
            "@@assign": [
                "*"
            ]
        },
        "not_resource_types":{
            "@@assign": [
                "arn:aws:ec2:*:*:volume/*"
            ]
        },
        "conditions":{
            "string_equals":{
                "condition_name1":{
                    "condition_key": { "@@assign":"aws:ResourceTag/backup"},
                    "condition_value": {  "@@assign":"true" }
```

```
                }
            }
        }
    }
},
...
```

Example: Select all Amazon EBS volumes and Amazon RDS DB instances tagged with both "backup" : "true" and "stage" : "prod"

The Boolean logic is similar to what you might use in IAM policies. The `"resource_types"` block uses a Boolean AND to combine the resource types. The `"conditions"` block uses a Boolean AND to combine resource types and tag conditions.

```
...
"resources":{
    "resource_selection_name":{
        "iam_role_arn":{"@@assign": "arn:aws:iam::$account:role/IAMRole"},
        "resource_types":{
            "@@assign": [
                "arn:aws:ec2:*:*:volume/*",
                "arn:aws:rds:*:*:db:*"
            ]
        },
        "conditions":{
            "string_equals":{
                "condition_name1":{
                    "condition_key":{"@@assign":"aws:ResourceTag/backup"},
                    "condition_value":{"@@assign":"true"}
                },
                "condition_name2":{
                    "condition_key":{"@@assign":"aws:ResourceTag/stage"},
                    "condition_value":{"@@assign":"prod"}
                }
            }
        }
    }
},
...
```

Example: Select all Amazon EBS volumes and Amazon RDS instances tagged with "backup" : "true" but not "stage" : "test"

The Boolean logic is similar to what you might use in IAM policies. The `"resource_types"` block uses a Boolean AND to combine the resource types. The `"conditions"` block uses a Boolean AND to combine resource types and tag conditions.

```
...
"resources":{
    "resource_selection_name":{
        "iam_role_arn":{"@@assign": "arn:aws:iam::$account:role/IAMRole"},
        "resource_types":{
            "@@assign": [
                "arn:aws:ec2:*:*:volume/*",
                "arn:aws:rds:*:*:db:*"
            ]
        },
        "conditions":{
            "string_equals":{
                "condition_name1":{
                    "condition_key":{"@@assign":"aws:ResourceTag/backup"},
                    "condition_value":{"@@assign":"true"}
                  }
            },
            "string_not_equals":{
                "condition_name2":{
                    "condition_key":{"@@assign":"aws:ResourceTag/stage"},
                    "condition_value":{"@@assign":"test"}
                }
            }
        }
    }
},
...
```

Example: Select all resources tagged with "key1" and a value which begins with "include" but not with "key2" and value that contains the word "exclude"

The Boolean logic is similar to what you might use in IAM policies. The `"resource_types"` block uses a Boolean AND to combine the resource types. The `"conditions"` block uses a Boolean AND to combine resource types and tag conditions.

In this example, note the use of the wildcard character (`*`) in `include*`, `*exclude*`, and `arn:aws:rds:*:*:db:*`. You can use the wildcard character (`*`) at the start, end, and middle of a string.

```
...
"resources":{
    "resource_selection_name":{
        "iam_role_arn":{"@@assign": "arn:aws:iam::$account:role/IAMRole"},
        "resource_types":{
            "@@assign": [
                "*"
            ]
        },
        "conditions":{
            "string_like":{
                "condition_name1":{
                    "condition_key":{"@@assign":"aws:ResourceTag/key1"},
                    "condition_value":{"@@assign":"include*"}
                }
            },
            "string_not_like":{
                "condition_name2":{
                    "condition_key":{"@@assign":"aws:ResourceTag/key2"},
                    "condition_value":{"@@assign":"*exclude*"}
                }
            }
        }
    }
},
...
```

Example: Select all resources tagged with "backup" : "true" except Amazon FSx file systems and Amazon RDS resources

The Boolean logic is similar to what you might use in IAM policies. The `"resource_types"` and `"not_resource_types"` blocks use a Boolean `AND` to combine the resource types. The `"conditions"` block uses a Boolean `AND` to combine resource types and tag conditions.

```
...
"resources":{
    "resource_selection_name":{
        "iam_role_arn":{"@@assign": "arn:aws:iam::$account:role/IAMRole"},
```

```
            "resource_types":{
                "@@assign": [
                    "*"
                ]
            },
            "not_resource_types":{
                "@@assign":[
                    "arn:aws:fsx:*:*:file-system/*",
                    "arn:aws:rds:*:*:db:*"
                ]
            },
        "conditions":{
            "string_equals":{
                "condition_name1":{
                    "condition_key":{"@@assign":"aws:ResourceTag/backup"},
                    "condition_value":{"@@assign":"true"}
                }
            }
        }
    }
},
...
```

Tag policies

Tag policies allow you to standardize the tags attached to the AWS resources in an organization's accounts.

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 general information about tagging, including naming and usage conventions, see the *Tagging AWS Resources User Guide*.
- For a list of services that support using tags, see the *Resource Groups Tagging API Reference*.
- For information about using tags to categorize resources, see the *Best Practices for Tagging AWS Resources Whitepaper*.
- For information on tagging Organizations resources, see Tagging AWS Organizations resources.
- For information on tagging resources in other AWS services, see the documentation for that service.

What are tag policies?

Tag policies are a type of policy that can help you standardize tags across resources in your organization's accounts. In a tag policy, you specify tagging rules applicable to resources when they are tagged.

For example, a tag policy can specify that when the `CostCenter` tag is attached to a resource, it must use the case treatment and tag values that the tag policy defines. A tag policy can also specify that noncompliant tagging operations on specified resource types are *enforced*. In other words, noncompliant tagging requests on specified resource types are prevented from completing. Untagged resources or tags that aren't defined in the tag policy aren't evaluated for compliance with the tag policy.

Using tag policies involves working with multiple AWS services:

- Use **AWS Organizations** to manage *tag policies*. When you sign in to the organization's management account, you use Organizations to enable the tag policies feature. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account. Then you can create tag policies and attach them to the organization entities to put those tagging rules in effect.
- Use **AWS Resource Groups** to manage *compliance* with tag policies. When you sign in to an account in your organization, you use Resource Groups to find noncompliant tags on resources in the account. You can correct noncompliant tags in the AWS service where you created the resource. You can also use the Tag Editor and the Resource Groups Tagging API to tag and untag resources from multiples services.

If you sign in to the management account in your organization, you can view compliance information for all your organization's accounts.

Tag policies are available only in an organization that has all features enabled. For more information on what's required to use tag policies, see Prerequisites and permissions for management policies for AWS Organizations.

> **Important**
>
> To get started with tag policies, AWS strongly recommends that you follow the example workflow described in Getting started with tag policies before moving on to more advanced tag policies. It's best to understand the effects of attaching a simple tag policy to a single account before expanding tag policies to an entire OU or organization. It's especially important to understand a tag policy's effects before you *enforce* compliance with any tag policy. The tables on the Getting started with tag policies page also provide links to instructions for more advanced policy-related tasks.

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.

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 *Tagging AWS Resource 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. Also see the workflows described in Getting started with tag policies.

Consider creating an SCP to set guardrails around resource creation requests

Resources that have never had tags attached to them don't show as noncompliant in reports. Account administrators can still create untagged resources. In some cases, you can use a service control policy (SCP) to set guardrails around resource creation requests. For an example SCP, see Require a tag on specified created resources.

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 **ABAC (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 management policies for AWS Organizations

- Best practices for using tag policies

Using tag policies for the first time

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

Task	Account to sign in to	AWS service console to use
Step 1: Enable tag policies for your organization.	The organization's management account.[1]	AWS Organizations
Step 2: Create a tag policy. Keep your first tag policy simple. Enter one tag key in the case treatment you want to use and leave all other options at their defaults.	The organization's management account.[1]	AWS Organizations
Step 3: Attach a tag policy to a single member account that you can use for testing. You'll need to sign in to this account in the next step.	The organization's management account.[1]	AWS Organizations
Step 4: Create some resources with compliant tags and some with noncompliant tags.	The member account that you're using for testing purposes.	Any AWS service that you are comfortable with. For example, you can use AWS Secrets Manager and follow the procedure in Creating a Basic Secret to create secrets with compliant and non-compliant secrets.
Step 5: View the effective tag policy and evaluate the compliance status of the account.	The member account that you're using for testing purposes.	Resource Groups and the AWS service where the resource was created.

Task	Account to sign in to	AWS service console to use
		If you created resources with compliant and non-compliant tags, you should see the non-compliant tags in the results.
Step 6: Repeat the process of finding and correcting compliance issues until the resources in the test account are compliant with your tag policy.	The member account that you're using for testing purposes.	Resource Groups and the AWS service where the resource was created.
At any time, you can evaluate organization-wide compliance.	The organization's management account.[1]	Resource Groups

[1] You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.

Expanding use of tag policies

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

Advanced task	Account to sign in to	AWS service console to use
Create more advanced tag policies.	The organization's management account.[1]	AWS Organizations
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.		

Advanced task	Account to sign in to	AWS service console to use
You can use the informati on in Understanding management policy inheritan ce and Tag policy syntax to create more detailed tag policies.		
Attach tag policies to additional accounts or OUs. Check the effective tag policy for an account after you attach more policies to it or to any OU in which the account is a member.	The organization's management account.[1]	AWS Organizations
Create an SCP to require tags when anyone creates new resources. For an example, see Require a tag on specified created resources.	The organization's management account.[1]	AWS Organizations
Continue to evaluate the compliance status of the account against the effective tag policy as it changes. Correct noncompliant tags.	A member account with an effective tag policy.	Resource Groups and the AWS service where the resource was created.
Evaluate organization-wide compliance.	The organization's management account.[1]	Resource Groups

[1] You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.

Enforcing tag policies for the first time

To enforce tag policies for the first time, follow a workflow similar to using tag policies for the first time and use a test account.

> **Warning**
>
> Use caution in enforcing compliance. Make sure that you understand the effects of using tag policies and follow the recommended workflow. Test how enforcement works on a test account before expanding it to more accounts. Otherwise, you could prevent users in your organization's accounts from tagging the resources they need. For more information, see Understanding enforcement.

Enforcement tasks	Account to sign in to	AWS service console to use
Step 1: Create a tag policy. 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.	The organization's management account.[1]	AWS Organizations
Step 2: Attach a tag policy to a single, test account.	The organization's management account.[1]	AWS Organizations
Step 3: Try creating some resources with compliant tags, and some with noncompliant tags. You	The member account that you're using for testing purposes.	Any AWS service that you are comfortable with. For example, you can use AWS Secrets Manager and follow

Enforcement tasks	Account to sign in to	AWS service console to use
shouldn't be allowed to create a tag on a resource of the type specified in the tag policy with a noncompliant tag.		the procedure in Creating a Basic Secret to create secrets with compliant and non-compliant secrets.
Step 4: Evaluate the compliance status of the account against the effective tag policy and correct noncompliant tags.	The member account that you're using for testing purposes.	Resource Groups and the AWS service where the resource was created.
Step 5: Repeat the process of finding and correcting compliance issues until the resources in the test account are compliant with your tag policy.	The member account that you're using for testing purposes.	Resource Groups and the AWS service where the resource was created.
At any time, you can evaluate organization-wide compliance.	The organization's management account.[1]	Resource Groups

[1] You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.

Using Amazon EventBridge to monitor noncompliant tags

You can use Amazon EventBridge, formerly Amazon 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 EventBridge, see the *Amazon EventBridge 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:

- From the **Visual editor** tab, select **Prevent noncompliant operations for this tag**.
- From the **JSON** tab, use the `enforced_for` field. For information on tag policy syntax, see Tag policy syntax and examples.

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.

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. Resource types that support enforcing compliance are listed when you use the visual editor to build a tag policy.
- **Understand interactions with some services** – Some AWS services have container-like groupings of resources that automatically create resources for you, and tags can propagate from a resource in one service to another. For example, tags on Amazon EC2 Auto Scaling groups and Amazon EMR clusters can automatically propagate to the contained Amazon EC2 instances. You may have tag policies for Amazon EC2 that are more strict than for Auto Scaling groups or EMR clusters. If you enable enforcement, the tag policy prevents resources from being tagged and may block dynamic scaling and provisioning.

The following sections show how you can find non-compliant resources, and correct them to be compliant.

Topics

- Finding non-compliant resources for an account with AWS Organizations
- Correcting non-compliant tags in resources with AWS Organizations
- Generating an organization-wide compliance report with AWS Organizations
- Services and resource types that support enforcement

Finding non-compliant resources for an account with AWS Organizations

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

To find non-compliant resources for an account with a tag policy, run the following command to save the results to a file:

```
$ aws resourcegroupstaggingapi get-resources  --region us-east-1 \
    --include-compliance-details \
    --exclude-compliant-resources > outputfile.txt
```

Correcting non-compliant tags in resources with AWS Organizations

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

- Use the console or tagging API operations of the AWS service that created the non-compliant resources.
- Use the AWS Resource Groups TagResources and UntagResources operations to add tags that are compliant with the effective policy or to remove non-compliant tags.

Generating an organization-wide compliance report with AWS Organizations

At any time, you can generate a report that lists all tagged resources in the AWS accounts across your organization. The report shows whether each resource is compliant with the effective tag policy. Note that it can take up to 48 hours for changes you make to a tag policy or resources to be reflected in the organization-wide compliance report. For example, assume that you have a tag policy that defines a new standardized tag for a resource type. Resources of that type that don't have this tag are shown as compliant in the report for up to 48 hours.

You can generate the report from your organization's management account in the `us-east-1` Region, provided that it has access to an Amazon S3 bucket. The bucket must have an attached bucket policy as shown in Amazon S3 Bucket Policy for Storing Report. To generate the report, run the following command:

```
$ aws resourcegroupstaggingapi start-report-creation --region us-east-1
```

You can generate one report at a time.

This report can take some time to complete. You can check the status by running the following command:

```
$ aws resourcegroupstaggingapi describe-report-creation --region us-east-1
{
    "Status": "SUCCEEDED"
}
```

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

Services and resource types that support enforcement

The following services and resource types support enforcement with tag policies:

Service name	Resource type	JSON syntax
Amazon API Gateway	• API keys • Domain names • REST API operations • Stages	• "apigateway:apikeys" • "apigateway:domainnames" • "apigateway:restapis" • "apigateway:restapis/stages"
AWS Amplify	• Component • Theme	• "amplifyuibuilder:app/environment/components" • "amplifyuibuilder:app/environment/themes"
AWS AppConfig	• Application • Configuration Profile • Deployment • Deployment Strategy • Environment	• "appconfig:application" • "appconfig:application/configurationprofile" • "appconfig:application/environment/deployment" • "appconfig:deploymentstrategy" • "appconfig:application/environment"
AWS App Mesh	• All • Gateway route • Mesh • Route • Virtual gateway • Virtual node • Virtual router • Virtual service	• "appmesh:*" • "appmesh:mesh/virtualGateway/gatewayRoute" • "appmesh:mesh" • "appmesh:mesh/virtualRouter/route" • "appmesh:mesh/virtualGateway" • "appmesh:mesh/virtualNode" • "appmesh:mesh/virtualRouter"

Service name	Resource type	JSON syntax
		• "appmesh:mesh/virtualService"
Amazon Athena	• All • Workgroup	• "athena:*" • "athena:workgroup"
AWS Audit Manager	• Assessment • Assessment Framework • Control	• "auditmanager:assessment " • "auditmanager:assessmentFramework " • "auditmanager:control "
AWS Backup	• Backup plan • Vault • Gateway • Hyper Visor • VM	• "backup:backup-plan" • "backup:backup-vault" • "backup-gateway:gateway" • "backup-gateway:hypervisor" • "backup-gateway:vm"
AWS Batch	• Job • Job Definition • Job Queue	• "batch:job" • "batch:job-definition" • "batch:job-queue"
AWS BugBust	• Event	• "bugbust:event"
AWS Certificate Manager	• All • Certificates • Private Certificate Authority	• "acm:*" • "acm:certificate" • "acm-pca:certificate-authority"

Service name	Resource type	JSON syntax
Amazon Chime	• Application Instance • Channel • Media Pipeline • Meeting • SIP Media Applications • User Application Instance • Voice Connector	• `"chime:app-instance"` • `"chime:app-instance/channel"` • `"chime:media-pipeline"` • `"chime:meeting"` • `"chime:sma"` • `"chime:app-instance/user"` • `"chime:vc"`
AWS Clean Rooms	• Collaboration • Configured Table • Membership • Configured Table Association	• `"cleanrooms:collaboration"` • `"cleanrooms:configuredtable"` • `"cleanrooms:membership"` • `"cleanrooms:membership/configuredtableassociation"`
AWS Cloud9	• Environment	• `"cloud9:environment"`
Amazon CloudFront	• All • Distribution	• `"cloudfront:*"` • `"cloudfront:distribution"`
AWS CloudTrail	• All • Trail	• `"cloudtrail:*"` • `"cloudtrail:trail"`
Amazon CloudWatch	• All • Alarm • Contributor Insights Rule • Metric Stream	• `"cloudwatch:*"` • `"cloudwatch:alarm"` • `"cloudwatch:insight-rule"` • `"cloudwatch:metric-stream"`
Amazon CloudWatch Internet Monitor	• Monitor	• `"internetmonitor:monitor"`

Service name	Resource type	JSON syntax
Amazon CloudWatch Logs	• Destination • Log group	• "logs:destination" • "logs:log-group"
Amazon CloudWatch Observability Access Manager	• Link • Sink	• "oam:link" • "oam:sink"
AWS CodeBuild	• All • Project	• "codebuild:*" • "codebuild:project"
Amazon CodeCatalyst	• Connections	• "codecatalyst:connections"
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"

Service name	Resource type	JSON syntax
AWS Config	• All • Aggregation authorization • Config aggregator • Config rule	• `"config:*"` • `"config:aggregation-authorization"` • `"config:config-aggregator"` • `"config:config-rule"`
Amazon CodeGuru Reviewer	• Association	• `"codeguru-reviewer:association"`
Amazon CodeGuru Security	• Scan	• `"codeguru-security:scans"`
CodeConnections	• Connection • Host	• `"codestar-connections:connection"` • `"codestar-connections:host"`
Amazon Connect	• Contact Flow • Integration Association • Queue • Quick Connect • Routing Profile • User	• `"connect:instance/contact-flow"` • `"connect:instance/integration-association"` • `"connect:instance/queue"` • `"connect:instance/transfer-destination"` • `"connect:instance/routing-profile"` • `"connect:instance/agent"`
Amazon Connect Wisdom	• Assistant • Association • Content • Knowledge Base • Session	• `"wisdom:assistant"` • `"wisdom:association"` • `"wisdom:content"` • `"wisdom:knowledge-base"` • `"wisdom:session"`

Service name	Resource type	JSON syntax
AWS Database Migration Service	• All • Endpoint • ES • Rep • Subgrp • Task	• `"dms:*"` • `"dms:endpoint"` • `"dms:es"` • `"dms:rep"` • `"dms:subgrp"` • `"dms:task"`
Amazon Data Lifecycle Manager	• Policy	• `"dlm:policy"`
AWS Direct Connect	• All • Dxcon • Dxlag • Dxvif	• `"directconnect:*"` • `"directconnect:dxcon"` • `"directconnect:dxlag"` • `"directconnect:dxvif"`
Amazon DynamoDB	• All • Table	• `"dynamodb:*"` • `"dynamodb:table"`

Service name	Resource type	JSON syntax
Amazon EC2	• Capacity reservation • Capacity reservation fleet • Carrier gateway • Client VPN endpoint • CoIP pool • Customer gateway • Dedicated host • DHCP options • Egress-only internet gateway • Elastic IP • Event window • Export Image Task • Export Instance Task • Fleet • FPGA image • Host reservation • Image • Import Image Task • Import Snapshot Task • Instance • Instance Connect Endpoint • Internet gateway • IP Address Manager	• `"ec2:capacity-reservation"` • `"ec2:capacity-reservation-fleet"` • `"ec2:carrier-gateway"` • `"ec2:client-vpn-endpoint"` • `"ec2:coip-pool"` • `"ec2:customer-gateway"` • `"ec2:dedicated-host"` • `"ec2:dhcp-options"` • `"ec2:egress-only-internet-gateway"` • `"ec2:elastic-ip"` • `"ec2:instance-event-window"` • `"ec2:export-image-task"` • `"ec2:export-instance-task"` • `"ec2:fleet"` • `"ec2:fpga-image"` • `"ec2:host-reservation"` • `"ec2:image"` • `"ec2:import-image-task"` • `"ec2:import-snapshot-task"` • `"ec2:instance"` • `"ec2:instance-connect-endpoint"` • `"ec2:internet-gateway"` • `"ec2:ipam"` • `"ec2:ipam-external-resource-verification-token"` • `"ec2:ipam-pool"`

Service name	Resource type	JSON syntax
	• IP Address Manager External Resource Verification Token • IP Address Manager Pool • IP Address Manager Resource Discovery • IP Address Manager Resource Discovery Association • IP Address Manager Scope • IPv4 Pool • Key Pair • Launch template • Local Gateway Route Table • Local Gateway Route Table Virtual Interface Group Association • Local Gateway Route Table VPC Association • NAT gateway • Network ACL • Network interface • Network Insights Access Scope • Network Insights Access Scope Analysis	• `"ec2:ipam-resource-discovery"` • `"ec2:ipam-resource-discovery-association"` • `"ec2:ipam-scope"` • `"ec2:ipv4pool-ec2"` • `"ec2:key-pair"` • `"ec2:launch-template"` • `"ec2:local-gateway-route-table"` • `"ec2:local-gateway-route-table-virtual-interface-group-association"` • `"ec2:local-gateway-route-table-vpc-association"` • `"ec2:natgateway"` • `"ec2:network-acl"` • `"ec2:network-interface"` • `"ec2:network-insights-access-scope"` • `"ec2:network-insights-access-scope-analysis"` • `"ec2:network-insights-analysis"` • `"ec2:network-insights-path"` • `"ec2:placement-group"` • `"ec2:prefix-list"` • `"ec2:replace-root-volume-task"` • `"ec2:reserved-instances"` • `"ec2:route-table"` • `"ec2:security-group"`

Service name	Resource type	JSON syntax
	• Network Insights Analysis • Network Insights Path • Placement Group • Prefix List • Replace Root Volume Task • Reserved Instances • Route table • Security group • Snapshot • Spot Fleet Request • Spot Instances request • Subnet • Subnet CIDR Reservation • Traffic mirror filter • Traffic mirror session • Traffic mirror target • Transit Gateway • Transit Gateway Attachment • Transit Gateway Connect Peer • Transit Gateway Multicast Domain	• `"ec2:snapshot"` • `"ec2:spot-fleet-request"` • `"ec2:spot-instances-request"` • `"ec2:subnet"` • `"ec2:subnet-cidr-reservation"` • `"ec2:traffic-mirror-filter"` • `"ec2:traffic-mirror-session"` • `"ec2:traffic-mirror-target"` • `"ec2:transit-gateway"` • `"ec2:transit-gateway-attachment"` • `"ec2:transit-gateway-connect-peer"` • `"ec2:transit-gateway-multicast-domain"` • `"ec2:transit-gateway-policy-table"` • `"ec2:transit-gateway-route-table"` • `"ec2:transit-gateway-route-table-announcement"` • `"ec2:verified-access-endpoint"` • `"ec2:verified-access-group"` • `"ec2:verified-access-instance"` • `"ec2:verified-access-trust-provider"` • `"ec2:volume"` • `"ec2:vpc-flow-log"` • `"ec2:vpc"` • `"ec2:vpc-endpoint"`

Service name	Resource type	JSON syntax
	• Transit Gateway Policy Table • Transit Gateway Route Table • Transit Gateway Route Table Announcement • Verified Access Endpoint • Verified Access Group • Verified Access Instance • Verified Access Trust Provider • Volume • VPC Flow Log • VPC • VPC endpoint • VPC endpoint service • VPC peering connection • VPN connection • VPN gateway	• `"ec2:vpc-endpoint-service"` • `"ec2:vpc-peering-connection"` • `"ec2:vpn-connection"` • `"ec2:vpn-gateway"`
Amazon EC2 Recycle Bin	• Rule	• `"rbin:rule"`

Service name	Resource type	JSON syntax
AWS Elastic Beanstalk	• Application • Application version • Configuration template • Platform	• "elasticbeanstalk:application" • "elasticbeanstalk:applicationversion" • "elasticbeanstalk:configurationtemplate" • "elasticbeanstalk:platform"
Amazon Elastic Container Registry	• Repository	• "ecr:repository"
Amazon Elastic Container Service	• Capacity Provider • Cluster • Service • Task Definition • Task set	• "ecs:capacity-provider" • "ecs:cluster" • "ecs:service" • "ecs:task-definition" • "ecs:task-set"
Amazon Elastic File System	• All • File system	• "elasticfilesystem:*" • "elasticfilesystem:file-system"
Amazon Elastic Kubernetes Service	• Cluster	• "eks:cluster"
Amazon Elastic Search	• Domain	• "es:domain"
Amazon EMR	• Cluster • Editor	• "elasticmapreduce:cluster" • "elasticmapreduce:editor"
Amazon EMR Serverless	• Application	• "emr-serverless:applications"

Service name	Resource type	JSON syntax
AWS Entity Resolution	• Matching Workflow • Schema Mapping	• `"entityresolution:matchingworkflow"` • `"entityresolution:schemamapping"`
Amazon ElastiCache	• Cluster	• `"elasticache:cluster"`
Amazon EventBridge	• All • Event bus • Rule	• `"events:*"` • `"events:event-bus"` • `"events:rule"`
Amazon EventBridge Pipes	• Pipe	• `"pipes:pipe"`
Amazon EventBridge Scheduler	• Schedule Group	• `"scheduler:schedule-group"`
Amazon Fraud Detector	• Detector • Detector version • Model • Rule • Variable	• `"frauddetector:detector"` • `"frauddetector:detector-version"` • `"frauddetector:model"` • `"frauddetector:rule"` • `"frauddetector:variable"`
Amazon Global Accelerator	• Accelerator	• `"globalaccelerator:accelerator"`

Service name	Resource type	JSON syntax
Elastic Load Balancing	• All • Listener • Listener Rule • Load balancer • Target group	• "elasticloadbalancing:*" • "elasticloadbalancing:listener" • "elasticloadbalancing:listener-rule" • "elasticloadbalancing:loadbalancer" • "elasticloadbalancing:targetgroup"
Amazon FSx	• All • Backup • File system	• "fsx:*" • "fsx:backup" • "fsx:file-system"
Amazon GuardDuty	• Detector • Filter • IP Set • Threat Intel Set	• "guardduty:detector" • "guardduty:detector/filter" • "guardduty:detector/ipset" • "guardduty:detector/threatintelset"
AWS HealthLake	• Datastore	• "healthlake:datastore "

Service name	Resource type	JSON syntax
AWS HealthOmics	• Annotation Store • Annotation Store Version • Reference Store • Reference • Run • Run Group • Sequence Store • Read Set • Variant Store • Workflow	• `"omics:annotationStore"` • `"omics:annotationStore/version"` • `"omics:referenceStore"` • `"omics:referenceStore/reference"` • `"omics:run"` • `"omics:runGroup"` • `"omics:sequenceStore"` • `"omics:sequenceStore/readSet"` • `"omics:variantStore"` • `"omics:workflow"`
Amazon Inspector	• Filter	• `"inspector2:filter "`
AWS Identity and Access Management	• Instance Profile • MFA • OIDC Provider • Policy • SAML Provider • Server Certificate	• `"iam:instance-profile"` • `"iam:mfa"` • `"iam:oidc-provider"` • `"iam:policy"` • `"iam:saml-provider"` • `"iam:server-certificate"`
AWS IoT Analytics	• All • Channel • Dataset • Datastore • Pipeline	• `"iotanalytics:*"` • `"iotanalytics:channel"` • `"iotanalytics:dataset"` • `"iotanalytics:datastore"` • `"iotanalytics:pipeline"`
AWS IoT Events	• All • Detector model • Input	• `"iotevents:*"` • `"iotevents:detectorModel"` • `"iotevents:input"`

Service name	Resource type	JSON syntax
AWS IoT Fleet Hub	• Application	• `"iotfleethub:application"`
AWS IoT SiteWise	• Asset • Asset Model	• `"iotsitewise:asset"` • `"iotsitewise:asset-model "`
AWS IoT Greengrass	• Bulk Deployment • Connector Definition • Core Definition • Device Definition • Function Definition • Logger Definition • Resource Definition • Subscription Definition	• `"greengrass:bulk"` • `"greengrass:connectorsDefinition"` • `"greengrass:coresDefinition"` • `"greengrass:devicesDefinition"` • `"greengrass:functionsDefinition"` • `"greengrass:loggersDefinition"` • `"greengrass:resourcesDefinition"` • `"greengrass:subscriptionsDefinition"`
AWS Key Management Service	• All • Key	• `"kms:*"` • `"kms:key"`
Amazon Kinesis	• All • Application	• `"kinesisanalytics:*"` • `"kinesisanalytics:application"`
Amazon Data Firehose	• All • Delivery stream	• `"firehose:*"` • `"firehose:deliverystream"`
AWS Lambda	• All • Function	• `"lambda:*"` • `"lambda:function"`
Amazon Macie	• Custom Data Identifier	• `"macie2:custom-data-identifier"`
Amazon MediaStore	• Container	• `"mediastore:container"`

Service name	Resource type	JSON syntax
Amazon MQ	• Broker • Configuration	• "mq:broker" • "mq:configuration"
Amazon Network Firewall	• Firewall • Firewall Policy • Stateful Rule Group • Stateless Rule Group	• "network-firewall:firewall" • "network-firewall:firewall-policy" • "network-firewall:stateful-rulegroup" • "network-firewall:stateless-rulegroup"
Amazon OpenSearch Serverless	• Collection	• "aoss:collection"
AWS Organizations	• Account • Organizational Unit • Policy • Root	• "organizations:account" • "organizations:ou" • "organizations:policy" • "organizations:root"
Amazon Pinpoint SMS Voice V2	• Configuration Set • Opt Out List • Phone Number • Pool • Sender Id	• "sms-voice:configuration-set" • "sms-voice:opt-out-list" • "sms-voice:phone-number" • "sms-voice:pool" • "sms-voice:sender-id"

Service name	Resource type	JSON syntax
Amazon RDS	• Cluster parameter group • Cluster endpoint • Event subscription • DB option group • DB parameter group • DB proxy • DB proxy endpoint • Reserved DB instance • DB security group • DB subnet group • Target group	• `"rds:cluster-pg"` • `"rds:cluster-endpoint"` • `"rds:es"` • `"rds:og"` • `"rds:pg"` • `"rds:db-proxy"` • `"rds:db-proxy-endpoint"` • `"rds:ri"` • `"rds:secgrp"` • `"rds:subgrp"` • `"rds:target-group"`
Amazon Redshift	• All • Cluster • Event subscription • HSM client certificate • HSM configuration • Parameter group • Snapshot • Snapshot copy grant • Snapshot schedule • Subnet group	• `"redshift:*"` • `"redshift:cluster"` • `"redshift:eventsubscription"` • `"redshift:hsmclientcertificate"` • `"redshift:hsmconfiguration"` • `"redshift:parametergroup"` • `"redshift:snapshot"` • `"redshift:snapshotcopygrant"` • `"redshift:snapshotschedule"` • `"redshift:subnetgroup"`

Service name	Resource type	JSON syntax
Amazon Redshift Serverless	• Namespace • Workgroup	• "redshift-serverless:namespace" • "redshift-serverless:workgroup"
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 • Storage Lens • Storage Lens Group	• "s3:bucket" • "s3:storage-lens" • "s3:storage-lens-group"

Service name	Resource type	JSON syntax
Amazon SageMaker AI	• App Image Config • Artifact • Context • Training job • Processing job • Model package group • Human task UI • Model Package • Action • Pipeline • Experiment • Flow Definition • Project	• "sagemaker:app-image-config" • "sagemaker:artifact" • "sagemaker:context" • "sagemaker:training-job" • "sagemaker:processing-job " • "sagemaker:model-package-group" • "sagemaker:human-task-ui" • "sagemaker:model-package" • "sagemaker:action" • "sagemaker:pipeline" • "sagemaker:experiment" • "sagemaker:flow-definition" • "sagemaker:project"
AWS Secrets Manager	• All • Secret	• "secretsmanager:*" • "secretsmanager:secret"
AWS Security Lake	• Data Lake • Subscriber	• "securitylake:data-lake" • "securitylake:subscriber"
AWS Service Catalog	• Application • Attribute Group • Portfolio • Product	• "servicecatalog:applications" • "servicecatalog:attribute-groups " • "catalog:portfolio " • "catalog:product "
Amazon Simple Notification Service (SNS)	• Topic	• "sns:topic"

Service name	Resource type	JSON syntax
Amazon Simple Queue Service (SQS)	• Queue	• "sqs:queue"
Amazon States Language	• All • Activity • State Machine	• "states:*" • "states:activity " • "states:stateMachine "
AWS Step Functions	• Activity	• "states:activity"
AWS Storage Gateway	• All • Gateway • Share • Tape • Volume	• "storagegateway:*" • "storagegateway:gateway" • "storagegateway:share" • "storagegateway:tape" • "storagegateway:gateway/volume"
AWS Systems Manager	• Association • Automation execution • Document • Maintenance Window • Managed instance • Ops item • Patch baseline • Contacts	• "ssm:association" • "ssm:automation-execution" • "ssm:document" • "ssm:maintenancewindow" • "ssm:managed-instance" • "ssm:opsitem" • "ssm:patchbaseline" • "ssm-contacts:contact"
Amazon Textract	• Adapters • Versions	• "textract:adapters" • "textract:adapters/versions"
AWS Transfer Family	• Server • User • Workflow	• "transfer:server" • "transfer:user" • "transfer:workflow"

Service name	Resource type	JSON syntax
Amazon Well-Architected	• Workload	• `"wellarchitected:workload"`
AWS Wickr	• Network	• `"wickr:network"`
Amazon WorkSpaces	• All • Connection Alias • Directory • WorkSpace • WorkSpaces bundle • WorkSpaces image • WorkSpaces IP group	• `"workspaces:*"` • `"workspaces:connectionalias"` • `"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 syntax for tag policies follows the syntax for management policy types. For a complete discussion of that syntax, see Understanding management policy inheritance. This topic focuses on applying that general syntax to the specific requirements of the tag policy type.

The following tag policy shows basic tag policy syntax:

```
{
    "tags": {
        "costcenter": {
            "tag_key": {
                "@@assign": "CostCenter"
            },
            "tag_value": {
                "@@assign": [
                    "100",
```

```
                    "200"
                ]
            },
            "enforced_for": {
                "@@assign": [
                    "secretsmanager:*"
                ]
            }
        }
    }
}
```

Tag policy syntax includes the following elements:

- The `tags` field key name. Tag policies always start with this fixed key name. It's the top line in the example policy above.
- A *policy key* that uniquely identifies the policy statement. It must match the value for the *tag key*, except for the case treatment. The policy value is 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 the `CostCenter` tag passed on all AWS Secrets Manager resources must be compliant with this policy.

> **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. In this example, `@@assign` is used to assign strings to `tag_key`, `tag_value`, and `enforced_for`. For more information about operators, see Inheritance operators.
- – You can use the `*` wildcard in tag values and `enforced_for` fields.
  - You can use only one wildcard per tag value. For example, `*@example.com` is allowed, but `*@*.com` is not.
  - For `enforced_for`, you can use `<service>:*` with some services to enable enforcement for all resources for that service. For a list of services and resource types that support `enforced_for`, see Services and resource types that support enforcement.

    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 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 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. The examples in this guide show tag policies formatted with extra white space to improve their readability. However, to save space if your policy size approaches the maximum size, you can delete any white space. Examples of white space include space characters and line breaks that are outside quotation marks.
- Untagged resources don't appear as noncompliant in results.

Example 1: Define organization-wide tag key case

The following example shows a tag policy that only defines two tag keys and the capitalization that you want accounts in your organization to standardize on.

Policy A – organization root tag policy

```
{
    "tags": {
        "CostCenter": {
            "tag_key": {
                "@@assign": "CostCenter",
                "@@operators_allowed_for_child_policies": ["@@none"]
            }
        },
        "Project": {
            "tag_key": {
                "@@assign": "Project",
                "@@operators_allowed_for_child_policies": ["@@none"]
            }
        }
    }
}
```

This tag policy defines two tag keys: `CostCenter` and `Project`. Attaching this tag policy to the organization root has the following effects:

- All accounts in your organization inherit this tag policy.
- All accounts in your organization must use the defined case treatment for compliance. Resources with `CostCenter` and `Project` tags are compliant. Resources with alternate case treatment for the tag key (for example, `costcenter`, `Costcenter`, or `COSTCENTER`) are noncompliant.

- The `@@operators_allowed_for_child_policies": ["@@none"]` lines lock down the tag keys. Tag policies that are attached lower in the organization tree (child policies) can't use value-setting operators to change the tag key, including its case treatment.
- As with all tag policies, untagged resources or tags that aren't defined in the tag policy aren't evaluated for compliance with the tag policy.

AWS recommends that you use this example as a guide in creating a similar tag policy for tag keys that you want to use. Attach it to the organization root. Then create a tag policy similar to the next example, which only defines the acceptable values for the defined tag keys.

Next step: Define values

Assume that you attached the previous tag policy to the organization root. Next, you can create a tag policy like the following and attach it to an account. This policy defines acceptable values for the `CostCenter` and `Project` tag keys.

Policy B – account tag policy

```
{
    "tags": {
        "CostCenter": {
            "tag_value": {
                "@@assign": [
                    "Production",
                    "Test"
                ]
            }
        },
        "Project": {
            "tag_value": {
                "@@assign": [
                    "A",
                    "B"
                ]
            }
        }
    }
}
```

If you attach Policy A to the organization root and Policy B to an account, the policies combine to create the following effective tag policy for the account:

Policy A + Policy B = effective tag policy for account

```
{
    "tags": {
        "Project": {
            "tag_value": [
                "A",
                "B"
            ],
            "tag_key": "Project"
        },
        "CostCenter": {
            "tag_value": [
                "Production",
                "Test"
            ],
            "tag_key": "CostCenter"
        }
    }
}
```

For more information about policy inheritance, including examples of how the inheritance operators work and example effective tag policies, see Understanding management policy inheritance.

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 are allowed in child tag policies. Therefore, any `Color` tags on resources in affected accounts are considered non-compliant. However, the `enforced_for` option actually prevents affected accounts from tagging ***only*** Amazon DynamoDB tables with the `Color` tag.

```
{
    "tags": {
        "Color": {
            "tag_key": {
                "@@operators_allowed_for_child_policies": [
                    "@@none"
                ],
```

```
                "@@assign": "Color"
            },
            "tag_value": {
                "@@operators_allowed_for_child_policies": [
                    "@@none"
                ],
                "@@assign": []
            },
            "enforced_for": {
                "@@assign": [
                    "dynamodb:table"
                ]
            }
        }
    }
}
```

Supported Regions

Tag policy features are available in the following Regions:

Region name	Region parameter
US East (N. Virginia) Region[1]	**us-east-1**
US East (Ohio) Region	us-east-2
US West (N. California) Region	us-west-1
US West (Oregon) Region	us-west-2
Africa (Cape Town) Region[2]	af-south-1
Asia Pacific (Hong Kong) Region[2]	ap-east-1
Asia Pacific (Mumbai) Region	ap-south-1
Asia Pacific (Hyderabad)[2]	ap-south-2
Asia Pacific (Tokyo) Region	ap-northeast-1
Asia Pacific (Seoul) Region	ap-northeast-2

Region name	Region parameter
Asia Pacific (Osaka) Region	`ap-northeast-3`
Asia Pacific (Singapore) Region	`ap-southeast-1`
Asia Pacific (Sydney) Region	`ap-southeast-2`
Asia Pacific (Jakarta) Region[2]	`ap-southeast-3`
Asia Pacific (Malaysia) Region	`ap-southeast-5`
Asia Pacific (Melbourne)[2]	`ap-southeast-4`
Asia Pacific (Thailand)	`ap-southeast-7`
Canada (Central) Region	`ca-central-1`
Canada West (Calgary)[2]	`ca-west-1`
China (Beijing) Region	`cn-north-1`
China (Ningxia) Region	`cn-northwest-1`
Europe (Frankfurt) Region	`eu-central-1`
Europe (Zurich) Region[2]	`eu-central-2`
Europe (Milan) Region[2]	`eu-south-1`
Europe (Spain)[2]	`eu-south-2`
Europe (Ireland) Region	`eu-west-1`
Europe (London) Region	`eu-west-2`
Europe (Paris) Region	`eu-west-3`
Europe (Stockholm) Region	`eu-north-1`
Mexico (Central) Region	`mx-central-1`

Region name	Region parameter
Middle East (Bahrain) Region[2]	`me-south-1`
South America (São Paulo) Region	`sa-east-1`
Israel (Tel Aviv)[2]	`il-central-1`
AWS GovCloud (US-East) Region	`us-gov-east-1`
AWS GovCloud (US-West) Region	`us-gov-west-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
- 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 report storage in the *Tagging AWS Resources User Guide*.

[2]These Regions must be manually enabled. To learn more about enabling and disabling AWS Regions, see Specify which AWS Regions your account can use in the *AWS Account Management Reference Guide*. The Resource Groups console isn't available in these Regions.

Chat applications policies

Chat applications policies in AWS Organizations enable you to control access to your organization's accounts from chat applications such as Slack and Microsoft Teams.

Amazon Q Developer in chat applications is an AWS service that enables DevOps and software development teams to use messaging program chat rooms to monitor and respond to operational events in their AWS Cloud. Amazon Q Developer in chat applications processes AWS service notifications from Amazon Simple Notification Service (Amazon SNS), and forwards them to chat rooms so teams can analyze and act on them immediately, regardless of location.

How chat applications policies work

Using chat applications policies, the management account or delegated administrator of an organization can do the following across an organization:

- Enforce which supported chat applications (Amazon Chime, Microsoft Teams, and Slack) can be used.
- Restrict chat client access to specific workspaces (Slack) and teams (Microsoft Teams).
- Restrict Slack channel visibility to either public or private channels.
- Set and enforce specific role settings.

Chat applications policies restrict and take precedence over account level settings such as role settings and channel guardrail policies. You can access and modify chat applications policies from the Amazon Q Developer in chat applications or the Organizations console.

After the policies are attached to accounts and organizational units (OU), any current and future Amazon Q Developer in chat applications configurations for the accounts in scope will automatically comply with the governance and permissions settings. For more information, see Understanding management policy inheritance.

If you try to perform an action restricted by a chat applications policy, an error message will notify you that the action is not allowed due to the chat applications policy with the recommendation to contact the management account or delegated administrator of your organization.

> **Note**
>
> Chat applications policies are validated at runtime. This means that existing resources are continuously checked for compliance. There is no overlap with existing IAM permissions since runtime-based IAM permissions for sending notifications or interacting with Amazon Q Developer in chat applications are not currently supported.

Getting started with chat applications policies

Follow these steps to get started using chat applications policies.

1. Learn about the permissions you must have to perform chat applications policy tasks.
2. Enable chat applications policies for your organization.
3. Create a chat applications policy.
4. Attach the chat applications policy to your organization's root, OU, or account.
5. View the combined effective chat applications policy that applies to an account.

For all of these steps, you sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.

Other information

- Learn chat applications policy syntax and see example policies

Chat applications policy syntax and examples

This topic describes chat applications policy syntax and provides examples.

Syntax for chat applications policies

A chat applications policy is a plaintext file that is structured according to the rules of JSON. The syntax for chat applications policies follows the syntax for management policy types. For a complete discussion of that syntax, see Understanding management policy inheritance. This topic focuses on applying that general syntax to the specific requirements of the chat applications policy type.

The following example shows the basic syntax for a chat applications policy:

```
{
    "chatbot":{
       "platforms":{
          "slack":{
             "client":{
                "@@assign":"enabled" // enabled | disabled
             },
             "workspaces": { // limit 255
                   "@@assign":[
                      "Slack-Workspace-Id"
                   ]
             },
             "default":{
                "supported_channel_types":{
                   "@@assign":[
                      "private" // public | private
                   ]
                },
                "supported_role_settings":{
                   "@@assign":[
                      "user_role" // user_role | channel_role
                   ]
                }
             },
             "overrides":{ // limit 255
                "Slack-Workspace-Id":{
                   "supported_channel_types":{
                      "@@assign":[
                         "public" // public | private
                      ]
                   },
                   "supported_role_settings":{
                      "@@assign":[
                         "user_role" // user_role | channel_role
                      ]
                   }
                }
             }
          },
          "microsoft_teams":{
             "client":{
                "@@assign":"enabled"
             },
```

```
            "tenants":{ // limit 36
               "Microsoft-Teams-Tenant-Id":{ // limit 36
                  "@@assign":[
                     "Microsoft-Teams-Team-Id"
                  ]
               }
            },
            "default":{
               "supported_role_settings":{
                  "@@assign":[
                     "user_role" // user_role | channel_role
                  ]
               }
            },
            "overrides":{ // limit 36
               "Microsoft-Teams-Tenant-Id":{ // limit 36
                  "Microsoft-Teams-Team-Id":{
                     "supported_role_settings":{
                        "@@assign":[
                           "user_role" // user_role | channel_role
                        ]
                     }
                  }
               }
            }
         },
         "chime":{
           "client":{
              "@@assign":"disabled" // enabled | disabled
           }
         }
      },
      "default":{
         "client":{
            "@@assign":"disabled" // enabled | disabled
         }
      }
   }
}
```

This chat applications policy includes the following elements:

- The `chatbot` field key name. Chat applications policies always start with this fixed key name. It's the top line in this example policy.
- Under `chatbot`, there is a `platforms` block, which contains the configuration for the different supported chat applications: Slack, Microsoft Teams, and Amazon Chime.
  - For Slack, the following fields are available:
    - `"client"`:
      - `"enabled"`: The Slack client is enabled. Slack integrations are allowed.
      - `"disabled"`: The Slack client is disabled. Slack integrations are not allowed.
    - `"workspaces"`: Comma-separated listed of allowed Slack workspaces. In this example, the allowed Slack workspaces are *`Slack-Workspace-Id1`* and *`Slack-Workspace-Id2`*.
    - `"default"`: The default settings for Slack workspaces.
      - `"supported_channel_types"`:
        - `"public"`: Slack workspaces in scope allow public Slack channels by default.
        - `"private"`: Slack workspaces in scope allow private Slack channels by default.
      - `supported_role_settings`:
        - `"user_role"`: Slack workspaces in scope allow User level IAM roles by default.
        - `"channel_role"`: Slack workspaces in scope allow Channel level IAM roles by default.
    - `"overrides"`: The override settings for the Slack workspaces.
      - *`Slack-Workspace-Id2`*: Comma-separated listed of Slack workspaces where the override setting apply. In this example, the Slack workspace is *`Slack-Workspace-Id2`*.
        - `"supported_channel_types"`:
          - `"public"`: Override setting whether Slack workspaces in scope allow public Slack channels.
          - `"private"`: Override setting whether Slack workspaces in scope allow private Slack channels.
        - `supported_role_settings`:
          - `"user_role"`: Override setting whether Slack workspaces in scope allow User level IAM roles.
          - `"channel_role"`: Override setting whether Slack workspaces in scope allow Channel level IAM roles.
  - For Microsoft Teams, the following fields are available:
    - `"client"`:

        - `"enabled"`: The Microsoft Teams client is enabled. Microsoft Teams integrations are allowed.
        - `"disabled"`: The Microsoft Teams client is disabled. Microsoft Teams integrations are not allowed.
    - `"tenants"`: Comma-separated listed of allowed Microsoft Teams tenants. In this example, the allowed tenant is *`Microsoft-Teams-Tenant-Id`*.
        - *`Microsoft-Teams-Tenant-Id`*: Comma-separated list of allowed teams within the tenant. In this example, the allowed team is *`Microsoft-Teams-Team-Id`*.
    - `"default"`: The default settings for the teams within the tenant.
        - `supported_role_settings`:
            - `"user_role"`: Teams in scope allow User level IAM roles by default.
            - `"channel_role"`: Teams in scope allow Channel level IAM roles by default.
    - `"overrides"`: The override settings for the Microsoft Teams tenants.
        - *`Microsoft-Teams-Tenant-Id`*: Comma-separated listed of tenants where the override setting apply. In this example, the tenant is *`Microsoft-Teams-Tenant-Id`*.
            - *`Microsoft-Teams-Team-Id`*: Comma-separate listed of teams within the tenant. In this example, the allowed team is *`Microsoft-Teams-Team-Id`*.
                - `supported_role_settings`:
                    - `"user_role"`: Override setting whether the teams in scope allow User level IAM roles.
                    - `"channel_role"`: Override setting whether the teams in scope allow Channel level IAM roles.
- For Amazon Chime, the following fields are available:
    - `"client"`:
        - `"enabled"`: The Amazon Chime client is enabled. Amazon Chime integrations are allowed.
        - `"disabled"`: The Amazon Chime client is disabled. Amazon Chime integrations are not allowed.
- Under `chatbot`, there is a `default` block which disables Amazon Q Developer in chat applications across the organization unless overridden at a lower level. This default also disables any new chat application that Amazon Q Developer in chat applications supports. For example, if

Amazon Q Developer in chat applications supports a new chat application, this default disables that newly supported chat application as well.

> **Note**
>
> For more information about Channel level IAM roles and User level IAM roles, see Understanding Amazon Q Developer in chat applications permissions in the *Amazon Q Developer in chat applications Administrator Guide*.

Chat applications policy examples

The example policies that follow are for information purposes only.

Example 1: Allow only private Slack Channels in a specific workspace, disable Microsoft Teams, all authentication modes supported

The following policy is focused on controlling the allowed configurations for Slack and Microsoft Teams chatbot integrations.

```
{
   "chatbot": {
      "platforms": {
         "slack": {
            "client": {
               "@@assign": "enabled"
            },
            "workspaces": {
               "@@assign": [
                  "Slack-Workspace-Id"
               ]
            },
            "default": {
               "supported_channel_types": {
                  "@@assign": [
                     "private"
                  ]
               },
               "supported_role_settings": {
                  "@@assign": [
                     "channel_role",
```

```
                    "user_role"
                ]
             }
          }
       },
       "microsoft_teams": {
          "client": {
             "@@assign": "disabled"
          }
       },
       "chime":{
          "client":{
             "@@assign":"disabled"
          }
       },
       "default":{
          "client":{
             "@@assign":"disabled"
          }
       }
    }
 }
}
```

For Slack

- The Slack client is enabled.
- Only the specific Slack workspace *`Slack-Workspace-Id`* is allowed.
- The default settings are to allow only private Slack channels, Channel level IAM roles, and User level IAM roles.

For Microsoft Team

- The Microsoft Teams client is disabled.

For Amazon Chime

- The Amazon Chime client is disabled.

Additional details

- The `default` block at the bottom sets the client to be disabled, which disables Amazon Q Developer in chat applications across the organization unless overridden at a lower level. This default also disables any new chat application that Amazon Q Developer in chat applications supports. For example, if Amazon Q Developer in chat applications supports a new chat application, this default disables that newly supported chat application as well.

Example 2: Allow only Slack integrations with User Level IAM roles

The following policy takes a more permissive approach to Slack, allowing all Slack workspaces but restricting the authentication mode to only User level IAM roles.

```
{
   "chatbot":{
      "platforms":{
         "slack":{
            "client":{
               "@@assign":"enabled"
            },
            "workspaces":
               {
                  "@@assign":[
                     "*"
                  ]
               },
            "default":{
               "supported_role_settings":{
                  "@@assign":[
                     "user_role"
                  ]
               }
            }
         },
         "microsoft_teams":{
            "client":{
               "@@assign":"disabled"
            }
         },
         "chime":{
            "client":{
               "@@assign":"disabled"
            }
         }
```

```
      },
      "default":{
         "client":{
            "@@assign":"disabled"
         }
      }
   }
}
```

For Slack

- The Slack client is enabled.
- No specific Slack workspaces are defined using the wildcard "*", so all workspaces are permitted.
- The default settings are to allow only User level IAM roles.

For Microsoft Team

- The Microsoft Teams client is disabled.

For Amazon Chime

- The Amazon Chime client is disabled.

Additional details

- The `default` block at the bottom sets the client to be disabled, which disables Amazon Q Developer in chat applications across the organization unless overridden at a lower level. This default also disables any new chat application that Amazon Q Developer in chat applications supports. For example, if Amazon Q Developer in chat applications supports a new chat application, this default disables that newly supported chat application as well.

Example 3: Allow only Microsoft Teams integrations in a specific Tenants

The following example policy locks down the organization to only allow Microsoft Teams chatbot integrations within the specified tenant, while completely blocking Slack integrations.

```
{
```

```
   "chatbot":{
      "platforms":{
         "slack":{
            "client": {
               "@@assign": "disabled"
            },
         },
         "microsoft_teams":{
            "client": {
               "@@assign": "enabled"
            },
            "tenants":{
               "Microsoft-Teams-Tenant-Id":{
                  "@@assign":[
                     "*"
                  ]
               }
            }
         },
         "chime": {
            "client":{
               "@@assign": "disabled"
            }
         }
      }
   }
}
```

For Slack

- The Slack client is disabled.

For Microsoft Team

- Only the specific tenant *Microsoft-Teams-Tenant-Id* is permitted, using the wildcard "*" to allow all teams within that tenant.

For Amazon Chime

- The Amazon Chime client is disabled.

Additional details

- The `default` block at the bottom sets the client to be disabled, which disables Amazon Q Developer in chat applications across the organization unless overridden at a lower level. This default also disables any new chat application that Amazon Q Developer in chat applications supports. For example, if Amazon Q Developer in chat applications supports a new chat application, this default disables that newly supported chat application as well.

Example 4: Allows restricted Amazon Q Developer in chat applications access for Slack workspaces and a Microsoft Teams tenant

The following policy allows restricted Amazon Q Developer in chat applications access for selected Slack workspaces and a Microsoft Teams tenant.

```
{
    "chatbot":{
       "platforms":{
          "slack":{
             "client":{
                "@@assign":"enabled"
             },
             "workspaces": {
                   "@@assign":[
                      "Slack-Workspace-Id1",
                      "Slack-Workspace-Id2"
                   ]
             },
             "default":{
                "supported_channel_types":{
                   "@@assign":[
                      "private"
                   ]
                },
                "supported_role_settings":{
                   "@@assign":[
                      "user_role"
                   ]
                }
             },
             "overrides":{
                "Slack-Workspace-Id2":{
                   "supported_channel_types":{
```

```
                  "@@assign":[
                     "public",
                     "private"
                  ]
               },
               "supported_role_settings":{
                  "@@assign":[
                     "channel_role",
                     "user_role"
                  ]
               }
            }
         }
      },
      "microsoft_teams":{
         "client":{
            "@@assign":"enabled"
         },
         "tenants":{
            "Microsoft-Teams-Tenant-Id":{
               "@@assign":[
                  "Microsoft-Teams-Team-Id"
               ]
            }
         },
         "default":{
            "supported_role_settings":{
               "@@assign":[
                  "user_role"
               ]
            }
         },
         "overrides":{
            "Microsoft-Teams-Tenant-Id":{
               "Microsoft-Teams-Team-Id":{
                  "supported_role_settings":{
                     "@@assign":[
                        "channel_role",
                        "user_role"
                     ]
                  }
               }
            }
         }
```

```
            }
        },
        "default":{
            "client":{
                "@@assign":"disabled"
            }
        }
    }
}
```

For Slack

- The Slack client is enabled.
- The allowed Slack workspaces are *Slack-Workspace-Id1* and *Slack-Workspace-Id2*.
- The default settings for Slack are to only allow private channels and User level IAM roles.
- There is an override for the workspace *Slack-Workspace-Id2* that allows both public and private channels as well as both Channel level IAM roles and User level IAM roles.

For Microsoft Team

- The Microsoft Teams is enabled.
- The allowed Teams tenants are *Microsoft-Teams-Tenant-Id* with the team *Microsoft-Teams-Team-Id*.
- The default settings are to only allow User level IAM roles.
- There is an override for the tenant *Microsoft-Teams-Tenant-Id* that allows both Channel level IAM roles and User level IAM roles for the team *Microsoft-Teams-Team-Id*.

Additional details

- The `default` block at the bottom sets the client to be disabled, which disables Amazon Q Developer in chat applications across the organization unless overridden at a lower level. This means Amazon Chime is disabled in this example. This default also disables any new chat application that Amazon Q Developer in chat applications supports. For example, if Amazon Q Developer in chat applications supports a new chat application, this default disables that newly supported chat application as well.

AI services opt-out policies

AI services opt-out policies allow you to control data collection for AWS AI services for all the accounts in an organization.

AWS AI services might use and store customer content for service improvement. *Service improvement* is the use and storage of content that is not personal data to develop and improve AWS and affiliate machine-learning and artificial intelligence technologies. For this purpose, we might store content in an AWS Region outside of the AWS Region where you are using the service. As an AWS customer, you can opt out of having your content used for service improvements at any time.

You can create opt-out policies for an individual AI service, or for all services supported by AI services opt-out policies. You can also query the effective policy applicable to each account to see the effects of your setting choices.

For more detailed information, see AWS Machine Learning and Artificial Intelligence Services in the AWS Service Terms. For a list of services supported by AI services opt-out policies, see List of supported AI services.

Topics

- Considerations when using AI services opt-out policies
- Getting started with AI services opt-out policies
- Opt out from all supported AWS AI services
- AI services opt-out policy syntax and examples

Considerations when using AI services opt-out policies

Opting out deletes all of the associated historical content

When you opt out of content use by an AWS AI service, that service deletes all of the associated historical content that was shared with AWS before you set the option. This deletion is limited to data stored that is not required to provide service functions.

For example, you use a service while opted in. That service might store copies of that your content for service improvement. You opt out. Any copies that have been stored by the service for service improvement are deleted, but any data that is used to provide the service to you is not deleted.

Getting started with AI services opt-out policies

Follow these steps to get started using Artificial Intelligence (AI) services opt-out policies.

1. Learn about the permissions you must have to perform backup policy tasks.
2. Enable AI services opt-out policies for your organization.
3. Create an AI services opt-out policy.
4. Attach the AI services opt-out policy to your organization's root, OU, or account.
5. View the combined effective AI services opt-out policy that applies to an account.

For all of these steps, you sign in as an AWS Identity and Access Management (IAM) user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.

Other information

- Learn policy syntax for AI services opt-out policies and see policy examples

Opt out from all supported AWS AI services

In this topic:

- You can opt out with a one button selection in the AWS Organizations console.
- You can opt out by attaching the provided example policy using the AWS CLI & AWS SDKs.
- You can view a list of AWS services supported by the AI services opt-out policy.

Opt out from all supported AI services

You can opt your organization out of having its content used for service improvement by creating and attaching an AI services opt-out policy. This policy applies to all current and future supported AWS AI services. Member accounts cannot update the policy.

AWS Management Console

To opt out from all AI services

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

2. On the **AI services opt-out policies** page, choose **Opt out from all services**.
3. On the **Opt out from all services** confirmation page, choose **Opt out from all services**.

AWS CLI & AWS SDKs

To opt out from all AI services

1. Copy "Example 1: Opt out of all AI services for all accounts in the organization" in AI services opt-out examples.
2. Follow the instruction in Attaching and detaching AI services opt-out.

> **Note**
>
> Additional steps are required to opt out from Amazon Monitron. For more information, see AWS Service Terms.

List of services supported by the AI services opt-out policy

The following is a list of AWS services supported by the AI services opt-out policy:

- AWS Supply Chain
- AWS Database Migration Service
- Amazon Chime SDK voice analytics
- Amazon CloudWatch
- Amazon CodeGuru Profiler
- Amazon CodeWhisperer (now part of Amazon Q Developer)
- Amazon Comprehend
- Amazon Connect
- Amazon Connect Optimization
- Amazon Connect Contact Lens
- Amazon DataZone
- AWS Entity Resolution
- Amazon Fraud Detector

- AWS Glue
- Amazon GuardDuty
- Amazon Lex
- Amazon Polly
- Amazon Q
- Amazon QuickSight
- Amazon Rekognition
- Amazon Security Lake
- Amazon Textract
- Amazon Transcribe
- Amazon Translate

AI services opt-out policy syntax and examples

This topic describes Artificial Intelligence (AI) services opt-out policy syntax and provides examples.

Syntax for AI services opt-out policies

An AI services opt-out policy is a plaintext file that is structured according to the rules of JSON. The syntax for AI services opt-out policies follows the syntax for management policy types. For a complete discussion of that syntax, see Understanding management policy inheritance. This topic focuses on applying that general syntax to the specific requirements of the AI services opt-out policy type.

> **Important**
>
> The capitalization of the values discussed in this section are important. Enter the values with upper and lower case letters as shown in this topic. The policies do not work if you use unexpected capitalization.

The following policy shows the basic AI services opt-out policy syntax. If this example was attached directly to an account, that account would be explicitly opted out of one service and opted in to another. Other services could be opted in or opted out by policies inherited from higher levels (OU or root policies).

```
{
    "services": {
        "rekognition": {
            "opt_out_policy": {
                "@@assign": "optOut"
            }
        },
        "lex": {
            "opt_out_policy": {
                "@@assign": "optIn"
            }
        }
    }
}
```

Imagine the following example policy attached to the organization's root. It sets the default for the organization to opt out of all AI services. This automatically includes any AI services not otherwise explicitly exempted, including any AI services that AWS might deploy in the future. You can attach child policies to OUs or directly to accounts to override this setting for any AI service except Amazon Comprehend. The second entry in the following example uses `@@operators_allowed_for_child_policies` set to none to prevent it from being overridden. The third entry in the example makes an organization-wide exemption for Amazon Rekognition. It opts in the entire organization for that service, but the policy does allow child policies to override where appropriate.

```
{
    "services": {
        "default": {
            "opt_out_policy": {
                "@@assign": "optOut"
            }
        },
        "comprehend": {
            "opt_out_policy": {
                "@@operators_allowed_for_child_policies": ["@@none"],
                "@@assign": "optOut"
            }
        },
        "rekognition": {
            "opt_out_policy": {
                "@@assign": "optIn"
```

```
            }
        }
    }
}
```

AI services opt-out policy syntax includes the following elements:

- The `services` element. An AI services opt-out policy is identified by this fixed name as the outermost JSON containing element.

  An AI services opt-out policy can have one or more statements under the `services` element. Each statement contains the following elements:

  - A *service name key* that identifies an AWS AI service. The following key names are valid values for this field:
    - **`default`** – represents **all** currently available AI services and implicitly and automatically includes any AI services that might be added in the future.
    - `awssupplychain`
    - `dms`
    - `chimesdkvoiceanalytics`
    - `cloudwatch`
    - `codeguruprofiler`
    - `codewhisperer`
    - `comprehend`
    - `connectamd`
    - `connectoptimization`
    - `contactlens`
    - `datazone`
    - `entityresolution`
    - `frauddetector`
    - `glue`
    - `guardduty`
    - `lex`
    - `polly`
    - `q`

- `quicksightq`
- `rekognition`
- `securitylake`
- `textract`
- `transcribe`
- `translate`

Each policy statement identified by a service name key can contain the following elements:

- The `opt_out_policy` key. This key must be present. This is the only key you can place under a service name key.

  The `opt_out_policy` key can contain ***only*** the `@@assign` operator with one of the following values:

  - `optOut` – you choose to opt out of content use for the specified AI service.
  - `optIn` – you choose to opt in to content use for the specified AI service.

  > **Notes**
  >
  > - You can't use the `@@append` and `@@remove` inheritance operators in AI services opt-out policies.
  > - You can't use the `@@enforced_for` operator in AI services opt-out policies.

- At any level, you can specify the `@@operators_allowed_for_child_policies` operator to control what child policies can do to override settings imposed by parent policies. You can specify one of the following values:
  - `@@assign` – child policies of this policy can use the `@@assign` operator to override the inherited value with a different value.
  - `@@none` – child policies of this policy can't change the value.

  The behavior of the `@@operators_allowed_for_child_policies` depends on where you place it. You can use the following locations:

  - Under the `services` key – controls whether a child policy can add to or change the list of services in the effective policy.
  - Under the key for a specific AI service or the `default` key - controls whether a child policy can add to or change the list of keys under this specific entry.

- Under the `opt_out_policies` key for a specific service – controls whether a child policy can change only the setting for this specific service.

AI services opt-out policy examples

The example policies that follow are for information purposes only.

Example 1: Opt out of all AI services for all accounts in the organization

The following example shows a policy that you could attach to your organization's root to opt out of AI services for accounts in your organization.

> **Tip**
>
> If you copy the following example using the copy button in the example's upper-right corner, the copy doesn't include the line numbers. It's ready to paste.

```
    | {
    |     "services": {
[1] |         "@@operators_allowed_for_child_policies": ["@@none"],
    |         "default": {
[2] |             "@@operators_allowed_for_child_policies": ["@@none"],
    |             "opt_out_policy": {
[3] |                 "@@operators_allowed_for_child_policies": ["@@none"],
    |                 "@@assign": "optOut"
    |             }
    |         }
    |     }
    | }
```

- [1] – The `"@@operators_allowed_for_child_policies": ["@@none"]` that is under `services` prevents any child policy from adding any new sections for individual services other than the `default` section that is already there. `Default` is the placeholder that represents "all AI services".
- [2] – The `"@@operators_allowed_for_child_policies": ["@@none"]` that is under `default` prevents any child policy from adding any new sections other than the `opt_out_policy` section that is already there.

- [3] – The `"@@operators_allowed_for_child_policies": ["@@none"]` that is under `opt_out_policy` prevents child policies from changing the value of the `optOut` setting or adding any additional settings.

Example 2: Set an organization default setting for all services, but allow child policies to override the setting for individual services

The following example policy sets an organization-wide default for all AI services. The value for `default` prevents a child policy from change the `optOut` value for service `default`, the placeholder for all AI services. If this policy is applied as a parent policy by attaching it to the root or to an OU, child policies can still change the opt-out setting for individual services, as shown in the second policy.

- Because there is no `"@@operators_allowed_for_child_policies": ["@@none"]` under the `services` key, child policies can add new sections for individual services.
- The `"@@operators_allowed_for_child_policies": ["@@none"]` that is under `default` prevents any child policy from adding any new sections other than the `opt_out_policy` section that is already there.
- The `"@@operators_allowed_for_child_policies": ["@@none"]` that is under `opt_out_policy` prevents child policies from changing the value of the `optOut` setting or adding any additional settings.

Organization root userAI services opt-out parent policy

```
{
    "services": {
        "default": {
            "@@operators_allowed_for_child_policies": ["@@none"],
            "opt_out_policy": {
                "@@operators_allowed_for_child_policies": ["@@none"],
                "@@assign": "optOut"
            }
        }
    }
}
```

The following example policy assumes that the previous example policy is attached to either the organization root or to a parent OU, and that you attach this example to an account affected by

the parent policy. It overrides the default opt-out setting and explicitly opts in to only the Amazon Lex service.

AI services opt-out child policy

```
{
    "services": {
        "lex": {
            "opt_out_policy": {
                "@@assign": "optIn"
            }
        }
    }
}
```

The resulting effective policy for the AWS account is that the account opts in to only Amazon Lex, and opts out of all other AWS AI services because of the inherited `default` opt-out setting from the parent policy.

Example 3: Define an organization-wide AI services opt-out policy for a single service

The following example shows an AI services opt-out policy that defines an `optOut` setting for a single AI service. If this policy is attached to the organization's root, it prevents any child policy from overriding the `optOut` setting for this one service. Other services are not addressed by this policy, but could be affected by child policies in other OUs or accounts.

```
{
    "services": {
        "rekognition": {
            "opt_out_policy": {
                "@@assign": "optOut",
                "@@operators_allowed_for_child_policies": ["@@none"]
            }
        }
    }
}
```

Delegated administrator for AWS Organizations

We recommend that you use the AWS Organizations management account and its users and roles only for tasks that must be performed by that account. We also recommend that you store

your AWS resources in other member accounts in the organization and keep them out of the management account. This is because security features like Organizations service control policies (SCPs) do not restrict users or roles in the management account.

From the organization's management account, you can delegate policy management for Organizations to specified member accounts to perform policy actions that are by default available only to the management account.

For example resource-based delegation policies, see Resource-based policy examples for AWS Organizations.

Topics

- Create a resource-based delegation policy with AWS Organizations
- Update a resource-based delegation policy with AWS Organizations
- View a resource-based delegation policy with AWS Organizations
- Delete a resource-based delegation policy with AWS Organizations

Create a resource-based delegation policy with AWS Organizations

From the management account, create a resource-based delegation policy for your organization and add a statement that specifies which member accounts can perform actions on policies. You can add multiple statements in the policy to denote a different set of permissions to member accounts.

Minimum permissions

To create the resource-based delegation policy, you need permissions to run the following actions:

- `organizations:PutResourcePolicy`
- `organizations:DescribeResourcePolicy`

Additionally, you must grant roles and users in the delegated administrator account the corresponding IAM permissions to the required actions. Without IAM permissions, it is assumed that the calling principal doesn't have the required permissions to manage AWS Organizations policies.

AWS Management Console

Add statements to the resource-based delegation policy in the AWS Management Console using one of the following methods:

- **JSON policy** – Paste and customize an example resource-based delegation policy to use in your account, or type your own JSON policy document in the JSON editor.
- **Visual editor** – Construct a new delegation policy in the visual editor, which guides you in creating a delegation policy without having to write JSON syntax.

Use the JSON policy editor to create a delegation policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. Choose **Settings**.
3. In the **Delegated administrator for AWS Organizations** section, choose **Delegate** to create the Organizations delegation policy.
4. Enter a JSON policy document. For details about the IAM policy language, see IAM JSON policy reference.
5. Resolve any security warnings, errors, or general warnings generated during policy validation, and then choose **Create policy** to save your work.

Use the visual editor to create a delegation policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. Choose **Settings**.
3. In the **Delegated administrator for AWS Organizations** section, choose **Delegate** to create the Organizations delegation policy.
4. On the **Create Delegation policy** page, choose **Add new statement**.
5. Set **Effect** to `Allow`.
6. Add `Principal` to define the member accounts to which you want to delegate.

7. From the list of **Actions**, choose the actions you want to delegate. You can use **Filter actions** to narrow down the choices.
8. To specify if the delegated member account can attach policies to the organization root or organizational units (OUs), set `Resources`. You must also select `policy` as a resource type. You can specify resources in the following ways:
   - Choose **Add a resource** and construct the Amazon Resource Name (ARN) by following the prompts in the dialog box.
   - List resource ARNs manually in the editor. For more information about ARN syntax, see Amazon Resource Name (ARN) in the AWS General Reference Guide. For information about using ARNs in the resource element of a policy, see IAM JSON policy elements: Resource.
9. Choose **Add a condition** to specify other conditions, including the policy type you want to delegate. Choose the condition's **Condition key**, **Tag key**, **Qualifier**, and **Operator**, and then type a **`Value`**. When you're finished, choose **Add condition**. For more information about the **Condition** element, see IAM JSON policy elements: Condition in the IAM JSON policy reference.
10. To add more permission blocks, choose **Add new statement**. For each block, repeat steps 5 through 9.
11. Resolve any security warnings, errors, or general warnings generated during policy validation, and then choose **Create policy** to save your work.

AWS CLI & AWS SDKs

Create a delegation policy

You can use the following command to create a delegation policy:

- AWS CLI: put-resource-policy

  The following example creates a delegation policy.

```
$ aws organizations put-resource-policy --content
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Fully_manage_backup_policies",
```

```
            "Effect": "Allow",
            "Principal": {
                "AWS": "135791357913"
            },
            "Action": [
                "organizations:DescribeOrganization",
                "organizations:ListAccounts",
                "organizations:CreatePolicy",
                "organizations:DescribePolicy",
                "organizations:UpdatePolicy",
                "organizations:DeletePolicy",
                "organizations:AttachPolicy",
                "organizations:DetachPolicy"
            ],
            "Resource": [
                "arn:aws:organizations::246802468024:root/o-abcdef/r-pqrstu",
                "arn:aws:organizations::246802468024:ou/o-abcdef/*",
                "arn:aws:organizations::246802468024:account/o-abcdef/*",
                "arn:aws:organizations::246802468024:organization/policy/
backup_policy/*",
            ],
            "Condition": {
                "StringLikeIfExists": {
                    "organizations:PolicyType": [
                        "BACKUP_POLICY"
                    ]
                }
            }
        }
    ]
}
```

- AWS SDK: PutResourcePolicy

Supported delegation policy actions

The following actions are supported for delegation policy:

- `AttachPolicy`
- `CreatePolicy`
- `DeletePolicy`

- DescribeAccount
- DescribeCreateAccountStatus
- DescribeEffectivePolicy
- DescribeHandshake
- DescribeOrganization
- DescribeOrganizationalUnit
- DescribePolicy
- DescribeResourcePolicy
- DetachPolicy
- DisablePolicyType
- EnablePolicyType
- ListAccounts
- ListAccountsForParent
- ListAWSServiceAccessForOrganization
- ListChildren
- ListCreateAccountStatus
- ListDelegatedAdministrators
- ListDelegatedServicesForAccount
- ListHandshakesForAccount
- ListHandshakesForOrganization
- ListOrganizationalUnitsForParent
- ListParents
- ListPolicies
- ListPoliciesForTarget
- ListRoots
- ListTagsForResource
- ListTargetsForPolicy
- TagResource

- `UntagResource`
- `UpdatePolicy`

Supported condition keys

Only condition keys supported by AWS Organizations can be used for delegation policy. For more information, see Condition keys for AWS Organizations in the *Service Authorization Reference*.

Update a resource-based delegation policy with AWS Organizations

From the management account, update a resource-based delegation policy for your organization and add a statement that specifies which member accounts can perform actions on policies. You can add multiple statements in the policy to denote a different set of permissions to member accounts.

Minimum permissions

To update the resource-based delegation policy, you need permissions to run the following actions:

- `organizations:PutResourcePolicy`
- `organizations:DescribeResourcePolicy`

Additionally, you must grant roles and users in the delegated administrator account the corresponding IAM permissions to the required actions. Without IAM permissions, it is assumed that the calling principal doesn't have the required permissions to manage AWS Organizations policies.

AWS Management Console

Add statements to the resource-based delegation policy in the AWS Management Console using one of the following methods:

- **JSON policy** – Paste and customize an example resource-based delegation policy to use in your account, or type your own JSON policy document in the JSON editor.
- **Visual editor** – Construct a new delegation policy in the visual editor, which guides you in creating a delegation policy without having to write JSON syntax.

Use the JSON policy editor to update a delegation policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. Choose **Settings**.
3. In the **Delegated administrator for AWS Organizations** section, choose **Edit** to update the Organizations delegation policy.
4. Enter a JSON policy document. For details about the IAM policy language, see IAM JSON policy reference.
5. Resolve any security warnings, errors, or general warnings generated during policy validation, and then choose **Create policy**.

Use the visual editor update a delegation policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. Choose **Settings**.
3. In the **Delegated administrator for AWS Organizations** section, choose **Edit** to update the Organizations delegation policy.
4. On the **Create Delegation policy** page, choose **Add new statement**.
5. Set **Effect** to `Allow`.
6. Add `Principal` to define the member accounts to which you want to delegate.
7. From the list of **Actions**, choose the actions you want to delegate. You can use **Filter actions** to narrow down the choices.
8. To specify if the delegated member account can attach policies to the organization root or organizational units (OUs), set `Resources`. You must also select `policy` as a resource type. You can specify resources in the following ways:

   - Choose **Add a resource** and construct the Amazon Resource Name (ARN) by following the prompts in the dialog box.
   - List resource ARNs manually in the editor. For more information about ARN syntax, see Amazon Resource Name (ARN) in the AWS General Reference Guide. For information

about using ARNs in the resource element of a policy, see IAM JSON policy elements: Resource.

9. Choose **Add a condition** to specify other conditions, including the policy type you want to delegate. Choose the condition's **Condition key**, **Tag key**, **Qualifier**, and **Operator**, and then type a **`Value`**. When you're finished, choose **Add condition**. For more information about the **Condition** element, see IAM JSON policy elements: Condition in the IAM JSON policy reference.
10. To add more permission blocks, choose **Add new statement**. For each block, repeat steps 5 through 9.
11. Resolve any security warnings, errors, or general warnings generated during policy validation, and then choose **Save policy**.

AWS CLI & AWS SDKs

Create or update a delegation policy

You can use the following command to create or update a delegation policy:

- AWS CLI: put-resource-policy

  The following example creates or updates the delegation policy.

```
$ aws organizations put-resource-policy --content
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Fully_manage_backup_policies",
            "Effect": "Allow",
            "Principal": {
                "AWS": "135791357913"
            },
            "Action": [
                "organizations:DescribeOrganization",
                "organizations:ListAccounts",
                "organizations:CreatePolicy",
                "organizations:DescribePolicy",
                "organizations:UpdatePolicy",
                "organizations:DeletePolicy",
                "organizations:AttachPolicy",
```

```
                "organizations:DetachPolicy"
            ],
            "Resource": [
                "arn:aws:organizations::246802468024:root/o-abcdef/r-pqrstu",
                "arn:aws:organizations::246802468024:ou/o-abcdef/*",
                "arn:aws:organizations::246802468024:account/o-abcdef/*",
                "arn:aws:organizations::246802468024:organization/policy/
backup_policy/*",
            ],
            "Condition": {
                "StringLikeIfExists": {
                    "organizations:PolicyType": [
                        "BACKUP_POLICY"
                    ]
                }
            }
        }
    ]
}
```

- AWS SDK: PutResourcePolicy

Supported delegation policy actions

The following actions are supported for delegation policy:

- `AttachPolicy`
- `CreatePolicy`
- `DeletePolicy`
- `DescribeAccount`
- `DescribeCreateAccountStatus`
- `DescribeEffectivePolicy`
- `DescribeHandshake`
- `DescribeOrganization`
- `DescribeOrganizationalUnit`
- `DescribePolicy`
- `DescribeResourcePolicy`

- DetachPolicy
- DisablePolicyType
- EnablePolicyType
- ListAccounts
- ListAccountsForParent
- ListAWSServiceAccessForOrganization
- ListChildren
- ListCreateAccountStatus
- ListDelegatedAdministrators
- ListDelegatedServicesForAccount
- ListHandshakesForAccount
- ListHandshakesForOrganization
- ListOrganizationalUnitsForParent
- ListParents
- ListPolicies
- ListPoliciesForTarget
- ListRoots
- ListTagsForResource
- ListTargetsForPolicy
- TagResource
- UntagResource
- UpdatePolicy

Supported condition keys

Only condition keys supported by AWS Organizations can be used for delegation policy. For more information, see Condition keys for AWS Organizations in the *Service Authorization Reference*.

View a resource-based delegation policy with AWS Organizations

From the management account, view your organization's resource-based delegation policy to understand which delegated administrators have access to manage which policy types.

Minimum permissions

To view the resource-based delegation policy, you need permissions to run the following action: `organizations:DescribeResourcePolicy`.

AWS Management Console

To view a delegation policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. Choose **Settings**.
3. In the **Delegated administrator for AWS Organizations** section, scroll to view the full delegation policy.

AWS CLI & AWS SDKs

View a delegation policy

You can use the following command to view a delegation policy:

- AWS CLI: describe-resource-policy

  The following example retrieves the policy.

  ```
  $ aws organizations describe-resource-policy
  ```

- AWS SDK: DescribeResourcePolicy

Delete a resource-based delegation policy with AWS Organizations

When you no longer need to delegate the management of policies in your organization, you can delete the resource-based delegation policy from the organization's management account.

Important

If you delete your resource-based delegation policy, you can't recover it.

Minimum permissions

To delete the resource-based delegation policy, you need permissions to run the following action: `organizations:DeleteResourcePolicy`.

AWS Management Console

To delete a delegation policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. Choose **Settings**.
3. In the **Delegated administrator for AWS Organizations** section, choose **Delete**.
4. In the **Delete policy** confirmation dialog box, type **`delete`**. Then, choose **Delete policy**.

AWS CLI & AWS SDKs

Delete a delegation policy

You can use the following command to delete a delegation policy:

- AWS CLI: delete-resource-policy

  The following example deletes the policy.

  ```
  $ aws organizations delete-resource-policy
  ```

- AWS SDK: DeleteResourcePolicy

Enabling a policy type

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

Minimum permissions

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

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

AWS Management Console

To enable a policy type

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

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

AWS CLI & AWS SDKs

To enable a policy type

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

- AWS CLI: enable-policy-type

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

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

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

- AWS SDKs: EnablePolicyType

Disabling a policy type

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

Considerations

Disabled policies are detached from all entities but not deleted

When you disable a policy type, all policies of the specified type are automatically detached from all entities in the organization root. The policies are ***not*** deleted.

(Service control policy type only) All entities in the root are initially attached to only the default `FullAWSAccess` SCP

(Service control policy type only) If you re-enable the SCP policy type later, all entities in the organization root are initially attached to only the default `FullAWSAccess` SCP. Attachments of

SCPs to entities are lost when the SCPs are disabled in the organization. If you later want to re-enable SCPs, you must reattach them to the organization's root, OUs, and accounts, as appropriate.

Disable a policy type

Minimum permissions

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

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

AWS Management Console

To disable a policy type

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

   The list of available policies of the specified type disappears.

AWS CLI & AWS SDKs

To disable a policy type

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

- AWS CLI: disable-policy-type

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

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

The list of `PolicyTypes` in the output no longer includes the specified policy type.

- AWS SDKs: DisablePolicyType

Creating organization policies with AWS Organizations

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

This topic describes how to create policies with AWS Organizations. A *policy* defines the controls that you want to apply to a group of AWS accounts.

Topics

- Create a service control policy (SCP)
- Create a resource control policy (RCP)
- Create a declarative policy
- Create a backup policy
- Create a tag policy
- Create a chat applications policy
- Create an AI services opt-out policy

Create a service control policy (SCP)

Minimum permissions

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

- `organizations:CreatePolicy`

AWS Management Console

To create a service control policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Service control policies** page, choose **Create policy**.
3. On the **Create new service control policy** page, enter a **Policy name** and an optional **Policy description**.
4. (Optional) Add one or more tags by choosing **Add tag** and then entering a key and an optional value. Leaving the value blank sets it to an empty string; it isn't `null`. You can attach up to 50 tags to a policy. For more information, see Tagging AWS Organizations resources.

   > **Note**
   >
   > In most of the steps that follow, we discuss using the controls on the right side of the JSON editor to construct the policy, element by element. Alternatively, you can, at any time, simply enter text in the JSON editor on the left side of the window. You can directly type, or use copy and paste.

5. To build the policy, your next steps vary depending on whether you want to add a statement that denies or allows access. For more information, see SCP evaluation. If you use `Deny` statements, you have additional control because you can restrict access to specific resources, define conditions for when SCPs are in effect, and use the NotAction element. For details about syntax, see SCP syntax.

   To add a statement that *denies* access:

   a. In the right **Edit statement** pane of the editor, under **Add actions**, choose an AWS service.

As you choose options on the right, the JSON editor updates to show the corresponding JSON policy on left.

b. After you select a service, a list opens that contains the available actions for that service. You can choose **All actions**, or choose one or more individual actions that you want to deny.

   The JSON on the left updates to include the actions you selected.

   > **Note**
   >
   > If you select an individual action and then also go back and also select **All actions**, the expected entry for *`servicename`*`:*` is added to the JSON, but the individual actions that you previously selected are left in the JSON and not removed.

c. If you want to add actions from additional services, you can choose **All services** at the top of the **Statement** box, and then repeat the previous two steps as needed.

d. Specify resources to include in the statement.

   - Next to **Add a resource**, choose **Add**.
   - In the **Add resource** dialog, choose the service whose resources you want to control from the list. You can select from among only those services you selected in the previous step.
   - Under **Resource type**, choose the type of resource you want to control.
   - Finally, complete the Amazon Resource Name (ARN) in **Resource ARN** to identify the specific resource to which you want to control access. You must replace all placeholders that are surrounded by curly braces `{}`. You can specify wild cards (`*`) where that resource type's ARN syntax permits. See the documentation for a specific resource type for information about where you can use wild cards.
   - Save your addition to the policy by choosing **Add resource**. The `Resource` element in the JSON reflects your additions or changes. The **Resource** element is required.

> **Tip**
>
> If you want to specify all resources for the selected service, either choose the **All resources** option in the list, or edit the `Resource` statement directly in the JSON to read `"Resource":"*"`.

e. (Optional) To specify conditions that limit when a policy statement is in effect, next to **Add condition**, choose **Add**.

- **Condition key** – From the list you can choose any condition key that is available for all AWS services (for example, `aws:SourceIp`) or a service-specific key for only one of the services that you selected for this statement.
- **Qualifier** – (Optional) When the request has more than one values for a multivalued context key, you can specify a qualifier for testing requests against the values. For more information see, Single-valued vs. multivalued context keys in the *IAM User Guide*. To check if a request can have multiple values, see the Actions, resources, and condition keys for AWS services in the *Service Authorization Reference*.
  - **Default** – Tests a single value in the request against the condition key value in the policy. The condition returns true if the value in the request matches the value in the policy. If the policy specifies more than one value then they are treated as an "or" test, and the condition returns true if the request values matches any of the policy values.
  - **For any value in a request** – When the request can have multiple values, this option tests whether *at least one* of the request values matches at least one of the condition key values in the policy. The condition returns true if any one of the key values in the request matches any one of the condition values in the policy. For no matching key or a null dataset, the condition returns false.
  - **For all values in a request** – When the request can have multiple values, this option tests whether *every* request value matches a condition key value in the policy. The condition returns true if every key value in the request matches at least one value in the policy. It also returns true if there are no keys in the request, or if the key values resolve to a null data set, such as an empty string.
- **Operator** – The operator specifies the type of comparison to make. The options that are presented depend on the data type of the condition key. For example, the `aws:CurrentTime` global condition key lets you pick from any of the date

comparison operators, or `Null`, which you can use to test whether the value is present in the request.

For any condition operator except the `Null` test, you can choose the IfExists option.

- **Value** – (Optional) Specify one or more values for which you want to test the request.

Choose **Add condition**.

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

6. To add a statement that *allows* access:

   a. In the JSON editor on the left, change the line `"Effect": "Deny"` to `"Effect": "Allow"`.

   As you choose options on the right, the JSON editor updates to show the corresponding JSON policy on the left.

   b. After you select a service, a list opens that contains the available actions for that service. You can choose **All actions**, or choose one or more individual actions that you want to allow.

   The JSON on the left updates to include the actions you selected.

   > **Note**
   >
   > If you select an individual action and then also go back and also select **All actions**, the expected entry for *`servicename`*`:*` is added to the JSON, but the individual actions that you previously selected are left in the JSON and not removed.

   c. If you want to add actions from additional services, you can choose **All services** at the top of the **Statement** box, and then repeat the previous two steps as needed.

7. (Optional) To add another statement to the policy, choose **Add statement** and use the visual editor to build the next statement.

8. When you're finished adding statements, choose **Create policy** to save the completed SCP.

Your new SCP appears in the list of the organization's policies. You can now attach your SCP to the root, OUs, or accounts.

AWS CLI & AWS SDKs

To create a service control policy

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

- AWS CLI: create-policy

  The following example assumes that you have a file named `Deny-IAM.json` with the JSON policy text in it. It uses that file to create a new service control policy.

```
$ aws organizations create-policy \
    --content file://Deny-IAM.json \
    --description "Deny all IAM actions" \
    --name DenyIAMSCP \
    --type SERVICE_CONTROL_POLICY
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
service_control_policy/p-i9j8k7l6m5",
            "Name": "DenyIAMSCP",
            "Description": "Deny all IAM actions",
            "Type": "SERVICE_CONTROL_POLICY",
            "AwsManaged": false
        },
         "Content": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":
\"Statement1\",\"Effect\":\"Deny\",\"Action\":[\"iam:*\"],\"Resource\":[\"*\"]}]}"
    }
}
```

- AWS SDKs: CreatePolicy

> **Note**
>
> SCPs don't take effect on the management account and in a few other situations. For more information, see Tasks and entities not restricted by SCPs.

Create a resource control policy (RCP)

Minimum permissions

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

- `organizations:CreatePolicy`

AWS Management Console

To create a resource control policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Resource control policy** page, choose **Create policy**.
3. On the **Create new resource control policy** page, enter a **Policy name** and an optional **Policy description**.
4. (Optional) Add one or more tags by choosing **Add tag** and then entering a key and an optional value. Leaving the value blank sets it to an empty string; it isn't `null`. You can attach up to 50 tags to a policy. For more information, see Tagging AWS Organizations resources.

   Note

   In most of the steps that follow, we discuss using the controls on the right side of the JSON editor to construct the policy, element by element. Alternatively, you can, at any time, simply enter text in the JSON editor on the left side of the window. You can directly type, or use copy and paste.

5. To add a statement:

   a. In the right **Edit statement** pane of the editor, under **Add actions**, choose an AWS service.

   As you choose options on the right, the JSON editor updates to show the corresponding JSON policy on left.

b. After you select a service, a list opens that contains the available actions for that service. You can choose **All actions**, or choose one or more individual actions that you want to deny.

   The JSON on the left updates to include the actions you selected.

   > **Note**
   >
   > If you select an individual action and then also go back and also select **All actions**, the expected entry for *`servicename`*`:*` is added to the JSON, but the individual actions that you previously selected are left in the JSON and not removed.

c. If you want to add actions from additional services, you can choose **All services** at the top of the **Statement** box, and then repeat the previous two steps as needed.

d. Specify resources to include in the statement.

   - Next to **Add a resource**, choose **Add**.
   - In the **Add resource** dialog, choose the service whose resources you want to control from the list. You can select from among only those services you selected in the previous step.
   - Under **Resource type**, choose the type of resource you want to control.
   - Complete the Amazon Resource Name (ARN) in **Resource ARN** to identify the specific resource to which you want to control access. You must replace all placeholders that are surrounded by curly braces `{}`. You can specify wild cards (`*`) where that resource type's ARN syntax permits. See the documentation for a specific resource type for information about where you can use wild cards.
   - Save your addition to the policy by choosing **Add resource**. The `Resource` element in the JSON reflects your additions or changes. The **Resource** element is required.

   > **Tip**
   >
   > If you want to specify all resources for the selected service, either choose the **All resources** option in the list, or edit the `Resource` statement directly in the JSON to read `"Resource":"*"`.

e. (Optional) To specify conditions that limit when a policy statement is in effect, next to **Add condition**, choose **Add**.

- **Condition key** – From the list you can choose any condition key that is available for all AWS services (for example, `aws:SourceIp`) or a service-specific key for only one of the services that you selected for this statement.
- **Qualifier** – (Optional) When the request has more than one values for a multivalued context key, you can specify a qualifier for testing requests against the values. For more information see, Single-valued vs. multivalued context keys in the *IAM User Guide*. To check if a request can have multiple values, see the Actions, resources, and condition keys for AWS services in the *Service Authorization Reference*.
  - **Default** – Tests a single value in the request against the condition key value in the policy. The condition returns true if the value in the request matches the value in the policy. If the policy specifies more than one value then they are treated as an "or" test, and the condition returns true if the request values matches any of the policy values.
  - **For any value in a request** – When the request can have multiple values, this option tests whether *at least one* of the request values matches at least one of the condition key values in the policy. The condition returns true if any one of the key values in the request matches any one of the condition values in the policy. For no matching key or a null dataset, the condition returns false.
  - **For all values in a request** – When the request can have multiple values, this option tests whether *every* request value matches a condition key value in the policy. The condition returns true if every key value in the request matches at least one value in the policy. It also returns true if there are no keys in the request, or if the key values resolve to a null data set, such as an empty string.
- **Operator** – The operator specifies the type of comparison to make. The options that are presented depend on the data type of the condition key. For example, the `aws:CurrentTime` global condition key lets you pick from any of the date comparison operators, or `Null`, which you can use to test whether the value is present in the request.

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

Choose **Add condition**.

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

f. (Optional) To use the `NotAction` element to deny access to all actions ***except*** those specified, replace `Action` in the left pane with `NotAction`, just after the `"Effect": "Deny",` element. For more information, see IAM JSON Policy Elements: NotAction in the *IAM User Guide*.

6. (Optional) To add another statement to the policy, choose **Add statement** and use the visual editor to build the next statement.
7. When you're finished adding statements, choose **Create policy** to save the completed RCP.

Your new RCP appears in the list of the organization's policies. You can now attach your RCP to the root, OUs, or accounts.

AWS CLI & AWS SDKs

To create a resource control policy

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

- AWS CLI: create-policy

  The following example assumes that you have a file named `Deny-IAM.json` with the JSON policy text in it. It uses that file to create a new resource control policy.

```
$ aws organizations create-policy \
    --content file://Deny-IAM.json \
    --description "Deny all IAM actions" \
    --name DenyIAMRCP \
    --type RESOURCE_CONTROL_POLICY
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
resource_control_policy/p-i9j8k7l6m5",
            "Name": "DenyIAMRCP",
            "Description": "Deny all IAM actions",
```

```
            "Type": "RESOURCE_CONTROL_POLICY",
            "AwsManaged": false
        },
         "Content": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":
\"Statement1\",\"Effect\":\"Deny\",\"Action\":[\"iam:*\"],\"Resource\":[\"*\"]}]}"
    }
}
```

- AWS SDKs: CreatePolicy

> **Note**
>
> RCPs don't take effect on the management account and in a few other situations. For more information, see Resources and entities not restricted by RCPs.

Create a declarative policy

> **Minimum permissions**
>
> To create a declarative policy, you need permission to run the following action:
>
> - `organizations:CreatePolicy`

AWS Management Console

To create a declarative policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Declarative policies** page, choose **Create policy**.
3. On the **Create new declarative policy for EC2** page, enter a **Policy name** and an optional **Policy description**.
4. (Optional) You can add one or more tags to the policy by choosing **Add tag** and then entering a key and an optional value. Leaving the value blank sets it to an empty string; it

isn't `null`. You can attach up to 50 tags to a policy. For more information, see Tagging AWS Organizations resources.

5. You can build the policy using the **Visual editor** as described in this procedure. You can also enter or paste policy text in the **JSON** tab. For information about declarative policy syntax, see Declarative policy syntax and examples.

   If you choose to use the **Visual editor**, select the service attribute you want to include in your declarative policy. For more information, see Supported AWS services and attributes.

6. Choose **Add service attribute**, and configure the attribute to your specifications. For more detailed information on the each effect, see Declarative policy syntax and examples.

7. When you're finished editing your policy, choose **Create policy** at the lower-right corner of the page.

AWS CLI & AWS SDKs

To create a declarative policy

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

- AWS CLI: create-policy

  1. Create a declarative policy like the following, and store it in a text file.

```
{
    "ec2_attributes": {
        "image_block_public_access": {
            "state": {
                "@@assign": "block_new_sharing"
            }
        }
    }
}
```

     This declarative policy specifies that all accounts affected by the policy are must be configured so that new Amazon Machine Images (AMIs) are not publicly sharable. For information about declarative policy syntax, see Declarative policy syntax and examples.

  2. Import the JSON policy file to create a new policy in the organization. In this example, the previous JSON file was named `policy.json`.

```
$ aws organizations create-policy \
    --type DECLARATIVE_POLICY_EC2 \
    --name "MyTestPolicy" \
    --description "My test policy" \
    --content file://policy.json
{
    "Policy": {
        "Content": "{"ec2_attributes":{"image_block_public_access":{"state":
{"@@assign":"block_new_sharing"}}}}".
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5"
            "Arn": "arn:aws:organizations::o-aa111bb222:policy/
declarative_policy_ec2/p-i9j8k7l6m5",
            "Description": "My test policy",
            "Name": "MyTestPolicy",
            "Type": "DECLARATIVE_POLICY_EC2"
        }
    }
}
```

- AWS SDKs: CreatePolicy

What to do next

After you create a declarative policy, assess readiness using the account status report. You can then enforce your baseline configurations. To do that, you can attach the policy to the organization root, organizational units (OUs), AWS accounts within your organization, or a combination of all of those.

Create a backup policy

Minimum permissions

To create a backup policy, you need permission to run the following action:

- `organizations:CreatePolicy`

AWS Management Console

You can create a backup policy in the AWS Management Console in one of two ways:

- A visual editor that lets you choose options and generates the JSON policy text for you.
- A text editor that lets you directly create the JSON policy text yourself.

The visual editor makes the process easy, but it limits your flexibility. It's a great way to create your first policies and get comfortable with using them. After you understand how they work and have started to be limited by what the visual editor provides, you can add advanced features to your policies by editing the JSON policy text yourself. The visual editor uses only the @@assign value-setting operator, and it doesn't provide any access to the child control operators. You can add the child control operators only if you manually edit the JSON policy text.

To create a backup policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Backup policies** page, choose **Create policy**.
3. On the **Create policy** page, enter a **Policy name** and an optional **Policy description**.
4. (Optional) You can add one or more tags to the policy by choosing **Add tag** and then entering a key and an optional value. Leaving the value blank sets it to an empty string; it isn't `null`. You can attach up to 50 tags to a policy. For more information about tagging, see Tagging AWS Organizations resources.
5. You can build the policy using the **Visual editor** as described in this procedure. You can also enter or paste policy text in the **JSON** tab. For information about backup policy syntax, see Backup policy syntax and examples.

   If you choose to use the **Visual editor**, select the backup options appropriate for your scenario. A backup plan consists of three parts. For more information about these backup plan elements, see Creating a backup plan and Assigning resources in the *AWS Backup Developer Guide*.

   a. Backup plan general details

      - The **Backup plan name** can consist of only alphanumeric, hyphen, and underline characters.
      - You must select at least one **Backup plan region** from the list. The plan can back up resources in only the selected AWS Regions.

b. One or more backup rules that specify how and when AWS Backup is to operate. Each backup rule defines the following items:

   - A schedule that includes the frequency of the backup and the time window in which the backup can occur.
   - The name of the backup vault to use. The **Backup vault name** can consist of only alphanumeric, hyphen, and underline characters. The backup vault must exist before the plan can successfully run. Create the vault using the AWS Backup console or AWS CLI commands.
   - (Optional) One or more **Copy to region** rules to also copy the backup to vaults in other AWS Regions.
   - One or more tag key and value pairs to attach to the backup recovery points created each time this backup plan runs.
   - Lifecycle options that specify when the backup transitions to cold storage, and when the backup expires.

   Choose **Add rule** to add each rule you need to the plan.

   For more information about backup rules, see Backup Rules in the *AWS Backup Developer Guide*.

c. A resource assignment that specifies which resources that AWS Backup should backup with this plan. The assignment is made by specifying tag pairs that AWS Backup uses to find and match resources

   - The **Resource assignment name** can consist of only alphanumeric, hyphen, and underline characters.
   - Specify the **IAM role** for AWS Backup to use to perform the backup by its name.

     In the console, you don't specify the entire Amazon Resource Name (ARN). You must include both the role name and its prefix that specifies the type of role. The prefixes are typically `role` or `service-role` , and they are separated from the role name by a forward slash ('/'). For example, you might enter `role/MyRoleName` or `service-role/MyManagedRoleName`. This is converted to a full ARN for you when stored in the underlying JSON.

> **Important**
>
> The specified IAM role must already exist in the account the policy is applied to. If it does not, the backup plan might successfully start backup jobs, but those backup jobs will fail.

- Specify one or more **Resource tag key** and **Tag values** pairs to identify resources that you want backed up. If there is more than one tag value, separate the values with commas.

Choose **Add assignment** to add each configured resource assignment to the backup plan.

For more information, see Assign Resources to a Backup Plan in the *AWS Backup Developer Guide*.

6. When you're finished creating your policy, choose **Create policy**. The policy appears in your list of available backup policies.

AWS CLI & AWS SDKs

To create a backup policy

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

- AWS CLI: create-policy

Create a backup plan as JSON text similar to the following, and store it in a text file. For complete rules for the syntax, see Backup policy syntax and examples.

```
{
    "plans": {
        "PII_Backup_Plan": {
            "regions": { "@@assign": [ "ap-northeast-2", "us-east-1", "eu-
north-1" ] },
            "rules": {
                "Hourly": {
                    "schedule_expression": { "@@assign": "cron(0 5/1 ? * * *)" },
                    "start_backup_window_minutes": { "@@assign": "480" },
                    "complete_backup_window_minutes": { "@@assign": "10080" },
```

```
                    "lifecycle": {
                        "move_to_cold_storage_after_days": { "@@assign": "180" },
                        "delete_after_days": { "@@assign": "270" }
                    },
                    "target_backup_vault_name": { "@@assign": "FortKnox" },
                    "copy_actions": {
                        "arn:aws:backup:us-east-1:$account:backup-vault:secondary-
vault": {
                            "lifecycle": {
                                "move_to_cold_storage_after_days": { "@@assign":
 "10" },
                                "delete_after_days": { "@@assign": "100" }
                            }
                        }
                    }
                }
            },
            "selections": {
                "tags": {
                    "datatype": {
                        "iam_role_arn": { "@@assign": "arn:aws:iam::$account:role/
MyIamRole" },
                        "tag_key": { "@@assign": "dataType" },
                        "tag_value": { "@@assign": [ "PII" ] }
                    }
                }
            }
        }
    }
}
```

This backup plan specifies that AWS Backup should back up all resources in the affected AWS accounts that are in the specified AWS Regions and that have the tag `dataType` with a value of `PII`.

Next, import the JSON policy file backup plan to create a new backup policy in the organization. Note the policy ID at the end of the policy ARN in the output.

```
$ aws organizations create-policy \
    --name "MyBackupPolicy" \
    --type BACKUP_POLICY \
    --description "My backup policy" \
```

```
    --content file://policy.json{
    "Policy": {
        "PolicySummary": {
            "Arn": "arn:aws:organizations::o-aa111bb222:policy/backup_policy/p-
i9j8k7l6m5",
            "Description": "My backup policy",
            "Name": "MyBackupPolicy",
            "Type": "BACKUP_POLICY"
        }
        "Content": "...a condensed version of the JSON policy document you
 provided in the file...",
    }
}
```

- AWS SDKs: CreatePolicy

Create a tag policy

Minimum permissions

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

- `organizations:CreatePolicy`

You can create a tag policy in the AWS Management Console in one of two ways:

- A visual editor that lets you choose options and generates the JSON policy text for you.
- A text editor that lets you directly create the JSON policy text yourself.

The visual editor makes the process easy, but it limits your flexibility. It's a great way to create your first policies and get comfortable with using them. After you understand how they work and have started to be limited by what the visual editor provides, you can add advanced features to your policies by editing the JSON policy text yourself. The visual editor uses only the @@assign value-setting operator, and it doesn't provide any access to the child control operators. You can add the child control operators only if you manually edit the JSON policy text.

AWS Management Console

You can create a tag policy in the AWS Management Console in one of two ways:

- A visual editor that lets you choose options and generates the JSON policy text for you.
- A text editor that lets you directly create the JSON policy text yourself.

The visual editor makes the process easy, but it limits your flexibility. It's a great way to create your first policies and get comfortable with using them. After you understand how they work and have started to be limited by what the visual editor provides, you can add advanced features to your policies by editing the JSON policy text yourself. The visual editor uses only the @@assign value-setting operator, and it doesn't provide any access to the child control operators. You can add the child control operators only if you manually edit the JSON policy text.

To create a tag policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Tag policies** page, choose **Create policy**.
3. On the **Create policy** page, enter a **Policy name** and an optional **Policy description**.
4. (Optional) You can add one or more tags to the policy object itself. These tags are not part of the policy. To do this, choose **Add tag** and then enter a key and an optional value. Leaving the value blank sets it to an empty string; it isn't `null`. You can attach up to 50 tags to a policy. For more information, see Tagging AWS Organizations resources.
5. You can build the tag policy using the **Visual editor** as described in this procedure. You can also type or paste a tag policy in the **JSON** tab. For information about tag policy syntax, see Tag policy syntax.

   If you choose to use the **Visual editor**, specify the following:

6. For **New tag key 1**, specify the name of a tag key to add.
7. For **Compliance Options** you can select the following options:

   a. **Use the capitalization that you've specified above for the tag key** — leave this option cleared (the default) to specify that the inherited parent tag policy, if any exists, should define the case treatment for the tag key.

Enable this option if you want to mandate a specific capitalization for the tag key using this policy. If you select this option, the capitalization you specified for **Tag Key** overrides the case treatment specified in an inherited parent policy.

If a parent policy doesn't exist and you don't enable this option, only tag keys in all lowercase characters are considered compliant. For more information about inheritance from parent policies, see Understanding management policy inheritance.

> **Tip**
>
> Consider using the example tag policy shown in Example 1: Define organization-wide tag key case 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.

b. **Specify allowed values for this tag key** — enable this option if you want to add allowed values for this tag key to any values inherited from a parent policy.

By default, this option is cleared, which means that only those values defined in and inherited from a parent policy are considered compliant. If a parent policy doesn't exist and you don't specify tag values then any value (including no value at all) is considered compliant.

To update the list of acceptable tag values, select **Specify allowed values for this tag key** and then choose **Specify values**. When prompted, enter the new values (one value per box), and then choose **Save changes**.

8. For **Resource types to enforce**, you can select **Prevent noncompliant operations for this tag**.

We recommend that you leave this option cleared (the default) unless you are experienced with using tag policies. Make sure that you have reviewed the recommendations in Understanding enforcement, and test thoroughly. Otherwise, you could prevent users in your organization's accounts from tagging the resources they need.

If you do want to enforce compliance with this tag key, select the check box and then **Specify resource types**. When prompted, select the resource types to include in the policy. Then choose **Save changes**.

> **Important**
>
> When you select this option, any operations that manipulate tags for resources of the specified types succeed only if the operation results in tags that are compliant with the policy.

9. (Optional) To add another tag key to this tag policy, choose **Add tag key**. Then perform steps 6–9 to define the tag key.
10. When you're finished building your tag policy, choose **Save changes**.

AWS CLI & AWS SDKs

To create a tag policy

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

- AWS CLI: create-policy

You can use any text editor to create a tag policy. Use JSON syntax and save the tag policy as a file with any name and extension in a location of your choosing. Tag policies can have a maximum of 2,500 characters, including spaces. For information about tag policy syntax, see Tag policy syntax.

To create a tag policy

1. Create a tag policy in a text file that looks similar to the following:

   Contents of `testpolicy.json`:

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

This tag policy defines the `CostCenter` tag key. The tag can accept any value or no value. A policy like this means that a resource that has the CostCenter tag attached with or without a value is compliant.

2. Create a policy that contains the policy content from the file. Extra white space in the output has been truncated for readability.

```
$ aws organizations create-policy \
    --name "MyTestTagPolicy" \
    --description "My Test policy" \
    --content file://testpolicy.json \
    --type TAG_POLICY
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-a1b2c3d4e5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
tag_policy/p-a1b2c3d4e5",
            "Name": "MyTestTagPolicy",
            "Description": "My Test policy",
            "Type": "TAG_POLICY",
            "AwsManaged": false
        },
        "Content": "{\n\"tags\":{\n\"CostCenter\":{\n\"tag_key\":{\n\"@@assign
\":\"CostCenter\"\n}\n}\n}\n\n"
    }
}
```

- AWS SDKs: CreatePolicy

Create a chat applications policy

Minimum permissions

To create a chat applications policy, you need permission to run the following action:

- `organizations:CreatePolicy`

AWS Management Console

You can create a chat applications policy in the AWS Management Console in one of two ways:

- A visual editor that lets you choose options and generates the JSON policy text for you.
- A text editor that lets you directly create the JSON policy text yourself.

The visual editor makes the process easy, but it limits your flexibility. It's a great way to create your first policies and get comfortable with using them. After you understand how they work and have started to be limited by what the visual editor provides, you can add advanced features to your policies by editing the JSON policy text yourself. The visual editor uses only the @@assign value-setting operator, and it doesn't provide any access to the child control operators. You can add the child control operators only if you manually edit the JSON policy text.

To create a chat applications policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Chatbot policies** page, choose **Create policy**.
3. On the **Create new chat applications policy** page, enter a **Policy name** and an optional **Policy description**.
4. (Optional) You can add one or more tags to the policy by choosing **Add tag** and then entering a key and an optional value. Leaving the value blank sets it to an empty string; it isn't `null`. You can attach up to 50 tags to a policy. For more information, see Tagging AWS Organizations resources.
5. You can build the policy using the **Visual editor** as described in this procedure. You can also enter or paste policy text in the **JSON** tab. For information about chat applications policy syntax, see Chat applications policy syntax and examples.

   If you choose to use the **Visual editor**, configure your chat applications policy by specifying access controls for chat clients.

   a. Choose one of the following for **Set Amazon Chime chat client access**
      - Deny chime access.
      - Allow Chime access.

b. Choose on the following for **Set Microsoft Teams chat client access**

- Deny access to all Teams
- Allow access to all Teams
- Restrict access to named Teams

c. Choose one of the following for **Set Slack chat client access**

- Deny access to all Slack workspaces
- Allow access to all Slack workspaces
- Restrict access to named Slack worksapces

> **Note**
>
> In addition, you can select **Limit Amazon Q Developer in chat applications usage to only private Slack channels**.

d. Select the following options for **Set IAM permissions types**

- **Enable Channel level IAM role** — All channel members share IAM role permissions to run tasks in a channel. A channel role is appropriate if channel members require the same permissions.
- **Enable User level IAM role** — Channel members must choose an IAM user role to perform actions (Requires Console access to choose roles). User roles are apporopriate if channel members require different permissions and can choose their user roles.

6. When you're finished creating your policy, choose **Create policy**. The policy appears in your list of chatbot backup policies.

AWS CLI & AWS SDKs

To create a chat applications policy

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

- AWS CLI: create-policy

  You can use any text editor to create a chat applications policy. Use JSON syntax and save the chat applications policy as a file with any name and extension in a location of your choosing. Chat applications policies can have a maximum of ? characters, including spaces. For information about tag policy syntax, see Chat applications policy syntax and examples.

To create a chat applications policy

1. Create a chat applications policy in a text file that looks similar to the following:

   Contents of `testpolicy.json`:

```
{
   "chatbot": {
      "platforms": {
         "slack": {
            "client": {
               "@@assign": "enabled"
            },
            "workspaces": {
               "@@assign": [
                  "Slack-Workspace-Id"
               ]
            },
            "default": {
               "supported_channel_types": {
                  "@@assign": [
                     "private"
                  ]
               }
            }
         },
         "microsoft_teams": {
            "client": {
               "@@assign": "disabled"
            }
         }
      }
   }
}
```

   This chat applications policy allows only private Slack channels in a specific workspace, disables Microsoft Teams, and supports all role settings.

2. Create a policy that contains the policy content from the file. Extra white space in the output has been truncated for readability.

```
$ aws organizations create-policy \
```

```
    --name "MyTestChatbotPolicy" \
    --description "My Test policy" \
    --content file://testpolicy.json \
    --type CHATBOT_POLICY
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-a1b2c3d4e5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
chatbot_policy/p-a1b2c3d4e5",
            "Name": "MyTestChatApplicationsPolicy",
            "Description": "My Test policy",
            "Type": "CHATBOT_POLICY",
            "AwsManaged": false
        },
        "Content": "{"chatbot":{"platforms":{"slack":{"client":
{"@@assign":"enabled"},"workspaces":{"@@assign":["Slack-Workspace-
Id"]},"supported_channel_types":{"@@assign":["private"]}},"microsoft_teams":
{"client":{"@@assign":"disabled"}}}}}"
    }
}
```

- AWS SDKs: CreatePolicy

Create an AI services opt-out policy

Minimum permissions

To create an AI services opt-out policy, you need permission to run the following action:

- `organizations:CreatePolicy`

AWS Management Console

To create an AI services opt-out policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AI services opt-out policies** page, choose **Create policy**.

3. On the **Create new AI services opt-out policy** page, enter a **Policy name** and an optional **Policy description**.
4. (Optional) You can add one or more tags to the policy by choosing **Add tag** and then entering a key and an optional value. Leaving the value blank sets it to an empty string; it isn't `null`. You can attach up to 50 tags to a policy. For more information, see Tagging AWS Organizations resources.
5. Enter or paste the policy text in the **JSON** tab. For information about AI services opt-out policy syntax, see AI services opt-out policy syntax and examples. For example policies that you can use as a starting point, see AI services opt-out policy examples.
6. When you're finished editing your policy, choose **Create policy** at the lower-right corner of the page.

AWS CLI & AWS SDKs

To create an AI services opt-out policy

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

- AWS CLI: create-policy

1. Create an AI services opt-out policy like the following, and store it in a text file. Note that "`optOut`" and "`optIn`" are case-sensitive.

```
{
    "services": {
        "default": {
            "opt_out_policy": {
                "@@assign": "optOut"
            }
        },
        "rekognition": {
            "opt_out_policy": {
                "@@assign": "optIn"
            }
        }
    }
}
```

This AI services opt-out policy specifies that all accounts affected by the policy are opted out of all AI services except for Amazon Rekognition.

2. Import the JSON policy file to create a new policy in the organization. In this example, the previous JSON file was named `policy.json`.

```
$ aws organizations create-policy \
    --type AISERVICES_OPT_OUT_POLICY \
    --name "MyTestPolicy" \
    --description "My test policy" \
    --content file://policy.json
{
    "Policy": {
        "Content": "{\"services\":{\"default\":{\"opt_out_policy\":{\"@@assign
\":\"optOut\"}},\"rekognition\":{\"opt_out_policy\":{\"@@assign\":\"optIn
\"}}}}",
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5"
            "Arn": "arn:aws:organizations::o-aa111bb222:policy/
aiservices_opt_out_policy/p-i9j8k7l6m5",
            "Description": "My test policy",
            "Name": "MyTestPolicy",
            "Type": "AISERVICES_OPT_OUT_POLICY"
        }
    }
}
```

- AWS SDKs: CreatePolicy

Updating organization policies with AWS Organizations

When your policy requirements change, you can update an existing policy.

This topic describes how to update policies with AWS Organizations. A *policy* defines the controls that you want to apply to a group of AWS accounts.

Topics

- Update a service control policy (SCP)
- Update a resource control policy (RCP)
- Update a declarative policy

- Update a backup policy
- Update a tag policy
- Update a chat applications policy
- Update an AI services opt-out policy

Update a service control policy (SCP)

When you sign in to your organization's management account, you can rename or change the contents of a policy. Changing the contents of an SCP immediately affects any users, groups, and roles in all attached accounts.

Minimum permissions

To update an SCP, you need permission to run the following actions:

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

AWS Management Console

To update a policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Service control policies** page, choose the name of the policy that you want to update.
3. On the policy's detail page, choose **Edit policy**.
4. Make any or all of the following changes:
   - You can rename the policy by entering a new name in **Policy name**.
   - You can change the description by entering new text in **Policy description**.

- You can edit the policy text by editing the policy in JSON format in the left pane. Alternatively, you can choose a statement in the editor on the right, and also alter its elements by using the controls. For more details about each control, see the Creating an SCP procedure earlier in this topic.

5. When you're finished, choose **Save changes**.

AWS CLI & AWS SDKs

To update a policy

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

- AWS CLI: update-policy

  The following example renames a policy.

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --name "MyRenamedPolicy"
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
service_control_policy/p-i9j8k7l6m5",
            "Name": "MyRenamedPolicy",
            "Description": "Blocks all IAM actions",
            "Type": "SERVICE_CONTROL_POLICY",
            "AwsManaged": false
        },
        "Content": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":
\"Statement1\",\"Effect\":\"Deny\",\"Action\":[\"iam:*\"],\"Resource\":[\"*\"]}]}"
    }
}
```

  The following example adds or changes the description for a service control policy.

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --description "My new policy description"
{
```

```
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
service_control_policy/p-i9j8k7l6m5",
            "Name": "MyRenamedPolicy",
            "Description": "My new policy description",
            "Type": "SERVICE_CONTROL_POLICY",
            "AwsManaged": false
        },
        "Content": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":
\"Statement1\",\"Effect\":\"Deny\",\"Action\":[\"iam:*\"],\"Resource\":[\"*\"]}]}"
    }
}
```

The following example changes the policy document of the SCP by specifying a file that contains the new JSON policy text.

```
$ aws organizations update-policy \
    --policy-id p-zlfw1r64
    --content file://MyNewPolicyText.json
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
service_control_policy/p-i9j8k7l6m5",
            "Name": "MyRenamedPolicy",
            "Description": "My new policy description",
            "Type": "SERVICE_CONTROL_POLICY",
            "AwsManaged": false
        },
        "Content": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":
\"AModifiedPolicy\",\"Effect\":\"Deny\",\"Action\":[\"iam:*\"],\"Resource\":[\"*
\"]}]}"
    }
}
```

- AWS SDKs: UpdatePolicy

Update a resource control policy (RCP)

When you sign in to your organization's management account, you can rename or change the contents of a policy. Changing the contents of an RCP immediately affects any resources in all attached accounts.

Minimum permissions

To update an RCP, you need permission to run the following actions:

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

AWS Management Console

To update a policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Resource control policy** page, choose the name of the policy that you want to update.
3. On the policy's detail page, choose **Edit policy**.
4. Make any or all of the following changes:
   - You can rename the policy by entering a new name in **Policy name**.
   - You can change the description by entering new text in **Policy description**.
   - You can edit the policy text by editing the policy in JSON format in the left pane. Alternatively, you can choose a statement in the editor on the right, and also alter its elements by using the controls. For more details about each control, see the Creating an RCP procedure earlier in this topic.
5. When you're finished, choose **Save changes**.

AWS CLI & AWS SDKs

To update a policy

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

- AWS CLI: update-policy

  The following example renames a policy.

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --name "MyRenamedPolicy"
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
service_control_policy/p-i9j8k7l6m5",
            "Name": "MyRenamedPolicy",
            "Description": "Blocks all IAM actions",
            "Type": "SERVICE_CONTROL_POLICY",
            "AwsManaged": false
        },
        "Content": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":
\"Statement1\",\"Effect\":\"Deny\",\"Action\":[\"iam:*\"],\"Resource\":[\"*\"]}]}"
    }
}
```

  The following example adds or changes the description for a resource control policy.

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --description "My new policy description"
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
service_control_policy/p-i9j8k7l6m5",
            "Name": "MyRenamedPolicy",
            "Description": "My new policy description",
            "Type": "SERVICE_CONTROL_POLICY",
```

```
            "AwsManaged": false
        },
        "Content": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":
\"Statement1\",\"Effect\":\"Deny\",\"Action\":[\"iam:*\"],\"Resource\":[\"*\"]}]}"
    }
}
```

The following example changes the policy document of the RCP by specifying a file that contains the new JSON policy text.

```
$ aws organizations update-policy \
    --policy-id p-zlfw1r64
    --content file://MyNewPolicyText.json
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
service_control_policy/p-i9j8k7l6m5",
            "Name": "MyRenamedPolicy",
            "Description": "My new policy description",
            "Type": "SERVICE_CONTROL_POLICY",
            "AwsManaged": false
        },
        "Content": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":
\"AModifiedPolicy\",\"Effect\":\"Deny\",\"Action\":[\"iam:*\"],\"Resource\":[\"*
\"]}]}"
    }
}
```

- AWS SDKs: UpdatePolicy

Update a declarative policy

Minimum permissions

To update a declarative policy, you must have permission to run the following actions:

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

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

AWS Management Console

To update a declarative policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Declarative policies** page, choose the name of the policy that you want to update.
3. On the policy's detail page, choose **Edit policy**.
4. You can enter a new **Policy name**, **Policy description**, or edit the **JSON** policy text. For information about declarative policy syntax, see Declarative policy syntax and examples.
5. When you're finished updating the policy, choose **Save changes**.

AWS CLI & AWS SDKs

To update a policy

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

- AWS CLI: update-policy

  The following example renames a declarative policy.

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --name "Renamed policy"
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
declarative_policy_ec2/p-i9j8k7l6m5",
            "Name": "Renamed policy",
            "Type": "DECLARATIVE_POLICY_EC2",
            "AwsManaged": false
```

```
        },
        "Content": "{"ec2-configuration":{"ec2_attributes":
{"image_block_public_access":{"state":{"@@assign":"block_new_sharing"}}}}".
    }
}
```

The following example adds or changes the description for a declarative policy.

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --description "My new description"
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
declarative_policy_ec2/p-i9j8k7l6m5",
            "Name": "Renamed policy",
            "Description": "My new description",
            "Type": "DECLARATIVE_POLICY_EC2",
            "AwsManaged": false
        },
        "Content": "{"ec2_attributes":{"image_block_public_access":{"state":
{"@@assign":"block_new_sharing"}}}}".
    }
}
```

- AWS SDKs: UpdatePolicy

Update a backup policy

When you sign in to your organization's management account, you can edit a policy that requires changes in your organization.

Minimum permissions

To update a backup policy, you must have permission to run the following actions:

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

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

AWS Management Console

To update a backup policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Backup policies** page, choose the name of the policy that you want to update.
3. Choose **Edit policy**.
4. You can enter a new **Policy name**, **Policy description**. You can change the policy content by using either the **Visual editor** or by directly editing the **JSON**.
5. When you're finished updating the policy, choose **Save changes**.

AWS CLI & AWS SDKs

To update a backup policy

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

- AWS CLI: update-policy

  The following example renames a backup policy.

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --name "Renamed policy"
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
backup_policy/p-i9j8k7l6m5",
            "Name": "Renamed policy",
            "Type": "BACKUP_POLICY",
            "AwsManaged": false
        },
```

```
        "Content": "{\"plans\":{\"TestBackupPlan\":{\"regions\":{\"@@assign\":
  ....TRUNCATED FOR BREVITY....   "@@assign\":[\"Yes\"]}}}}}}"
    }
}
```

The following example adds or changes the description for a backup policy.

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --description "My new description"
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
backup_policy/p-i9j8k7l6m5",
            "Name": "Renamed policy",
            "Description": "My new description",
            "Type": "BACKUP_POLICY",
            "AwsManaged": false
        },
       "Content": "{\"plans\":{\"TestBackupPlan\":{\"regions\":{\"@@assign\":
  ....TRUNCATED FOR BREVITY....   "@@assign\":[\"Yes\"]}}}}}}"
    }
}
```

The following example changes the JSON policy document attached to a backup policy. In this example, the content is taken from a file called `policy.json` with the following text:

```
{
    "plans": {
        "PII_Backup_Plan": {
            "regions": { "@@assign": [ "ap-northeast-2", "us-east-1", "eu-
north-1" ] },
            "rules": {
                "Hourly": {
                    "schedule_expression": { "@@assign": "cron(0 5/1 ? * * *)" },
                    "start_backup_window_minutes": { "@@assign": "480" },
                    "complete_backup_window_minutes": { "@@assign": "10080" },
                    "lifecycle": {
                        "move_to_cold_storage_after_days": { "@@assign": "180" },
                        "delete_after_days": { "@@assign": "270" },
```

```
                        "opt_in_to_archive_for_supported_resources": {"@@assign":
 false}
                    },
                    "target_backup_vault_name": { "@@assign": "FortKnox" },
                    "copy_actions": {
                        "arn:aws:backup:us-east-1:$account:backup-vault:secondary-
vault": {
                            "lifecycle": {
                                "move_to_cold_storage_after_days": { "@@assign":
 "10" },
                                "delete_after_days": { "@@assign": "100" },
                                "opt_in_to_archive_for_supported_resources":
 {"@@assign": false}
                            }
                        }
                    }
                }
            },
            "selections": {
                "tags": {
                    "datatype": {
                        "iam_role_arn": { "@@assign": "arn:aws:iam::$account:role/
MyIamRole" },
                        "tag_key": { "@@assign": "dataType" },
                        "tag_value": { "@@assign": [ "PII" ] }
                    }
                }
            }
        }
    }
}
```

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --content file://policy.json
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
backup_policy/p-i9j8k7l6m5",
            "Name": "Renamed policy",
            "Description": "My new description",
```

```
            "Type": "BACKUP_POLICY",
            "AwsManaged": false
        },
         "Content": "{\"plans\":{\"TestBackupPlan\":{\"regions\":{\"@@assign\":
  ....TRUNCATED FOR BREVITY....   "@@assign\":[\"Yes\"]}}}}}}"
}
```

- AWS SDKs: UpdatePolicy

Update a tag policy

Minimum permissions

To update a tag policy, you must have permission to run the following actions:

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

AWS Management Console

To update a tag policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Tag policies page** page, choose the tag policy that you want to update.
3. Choose **Edit policy**.
4. You can enter a new **Policy name**, **Policy description**. You can change the policy content by using either the **Visual editor** or by editing the **JSON**.
5. When you're finished updating the tag policy, choose **Save changes**.

AWS CLI & AWS SDKs

To update a policy

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

- AWS CLI: update-policy

  The following example renames a tag policy.

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --name "Renamed tag policy"
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
tag_policy/p-i9j8k7l6m5",
            "Name": "Renamed tag policy",
            "Type": "TAG_POLICY",
            "AwsManaged": false
        },
        "Content": "{\n\"tags\":{\n\"CostCenter\":{\n\"tag_key\":{\n\"@@assign\":
\"CostCenter\"\n}\n}\n}\n}\n\n"
    }
}
```

  The following example adds or changes the description for a tag policy.

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --description "My new tag policy description"
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
tag_policy/p-i9j8k7l6m5",
            "Name": "Renamed tag policy",
            "Description": "My new tag policy description",
            "Type": "TAG_POLICY",
            "AwsManaged": false
        },
       "Content": "{\n\"tags\":{\n\"CostCenter\":{\n\"tag_key\":{\n\"@@assign\":
\"CostCenter\"\n}\n}\n}\n}\n\n"
    }
```

```
}
```

The following example changes the JSON policy document attached to an AI services opt-out policy. In this example, the content is taken from a file called `policy.json` with the following text:

```
{
  "tags": {
    "Stage": {
      "tag_key": {
        "@@assign": "Stage"
      },
      "tag_value": {
        "@@assign": [
          "Production",
          "Test"
        ]
      }
    }
  }
}
```

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --content file://policy.json
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
tag_policy/p-i9j8k7l6m5",
            "Name": "Renamed tag policy",
            "Description": "My new tag policy description",
            "Type": "TAG_POLICY",
            "AwsManaged": false
        },
         "Content": "{\"tags\":{\"Stage\":{\"tag_key\":{\"@@assign\":\"Stage
\"},\"tag_value\":{\"@@assign\":[\"Production\",\"Test\"]},\"enforced_for\":
{\"@@assign\":[\"ec2:instance\"]}}}}"
}
```

- AWS SDKs: UpdatePolicy

Update a chat applications policy

Minimum permissions

To update a chat applications policy, you must have permission to run the following actions:

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

AWS Management Console

To update a chat applications policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Chatbot policies** **page** page, choose the chat applications policy that you want to update.
3. Choose **Edit policy**.
4. You can enter a new **Policy name**, **Policy description**. You can change the policy content by using either the **Visual editor** or by editing the **JSON**.
5. When you're finished updating the tag policy, choose **Save changes**.

AWS CLI & AWS SDKs

To update a policy

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

- AWS CLI: update-policy

  The following example renames a chat applications policy.

```
$ aws organizations update-policy \
```

```
    --policy-id p-i9j8k7l6m5 \
    --name "Renamed chat applications policy"
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
chatbot_policy/p-i9j8k7l6m5",
            "Name": "Renamed chat applications policy",
            "Type": "CHATBOT_POLICY",
            "AwsManaged": false
        },
        "Content": "{"chatbot":{"platforms":{"slack":{"client":
{"@@assign":"enabled"},"workspaces":{"@@assign":["Slack-Workspace-Id"]},"default":
{"supported_channel_types":{"@@assign":["private"]}}},"microsoft_teams":{"client":
{"@@assign":"disabled"}}}}}"
    }
}
```

- AWS SDKs: UpdatePolicy

Update an AI services opt-out policy

Minimum permissions

To update an AI services opt-out policy, you must have permission to run the following actions:

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

AWS Management Console

To update an AI services opt-out policy

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

2. On the **AI services opt-out policies** page, choose the name of the policy that you want to update.
3. On the policy's detail page, choose **Edit policy**.
4. You can enter a new **Policy name**, **Policy description**, or edit the **JSON** policy text. For information about AI services opt-out policy syntax, see AI services opt-out policy syntax and examples. For example policies that you can use as a starting point, see AI services opt-out policy examples.
5. When you're finished updating the policy, choose **Save changes**.

AWS CLI & AWS SDKs

To update a policy

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

- AWS CLI: update-policy

  The following example renames an AI services opt-out policy.

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --name "Renamed policy"
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
aiservices_opt_out_policy/p-i9j8k7l6m5",
            "Name": "Renamed policy",
            "Type": "AISERVICES_OPT_OUT_POLICY",
            "AwsManaged": false
        },
        "Content": "{\"services\":{\"default\":{\"opt_out_policy\":
  ....TRUNCATED FOR BREVITY...   :{\"@@assign\":\"optIn\"}}}}"
    }
}
```

  The following example adds or changes the description for an AI services opt-out policy.

```
$ aws organizations update-policy \
```

```
    --policy-id p-i9j8k7l6m5 \
    --description "My new description"
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
aiservices_opt_out_policy/p-i9j8k7l6m5",
            "Name": "Renamed policy",
            "Description": "My new description",
            "Type": "AISERVICES_OPT_OUT_POLICY",
            "AwsManaged": false
        },
        "Content": "{\"services\":{\"default\":{\"opt_out_policy\":
  ....TRUNCATED FOR BREVITY...   :{\"@@assign\":\"optIn\"}}}}"
    }
}
```

The following example changes the JSON policy document attached to an AI services opt-out policy. In this example, the content is taken from a file called `policy.json` with the following text:

```
{
    "services": {
        "default": {
            "opt_out_policy": {
                "@@assign": "optOut"
            }
        },
        "comprehend": {
            "opt_out_policy": {
                "@@operators_allowed_for_child_policies": ["@@none"],
                "@@assign": "optOut"
            }
        },
        "rekognition": {
            "opt_out_policy": {
                "@@assign": "optIn"
            }
        }
    }
}
```

```
$ aws organizations update-policy \
    --policy-id p-i9j8k7l6m5 \
    --content file://policy.json
{
    "Policy": {
        "PolicySummary": {
            "Id": "p-i9j8k7l6m5",
            "Arn": "arn:aws:organizations::123456789012:policy/o-aa111bb222/
aiservices_opt_out_policy/p-i9j8k7l6m5",
            "Name": "Renamed policy",
            "Description": "My new description",
            "Type": "AISERVICES_OPT_OUT_POLICY",
            "AwsManaged": false
        },
         "Content": "{\n\"services\": {\n\"default\": {\n\"   ....TRUNCATED FOR
 BREVITY....    ": \"optIn\"\n}\n}\n}\n}\n"}
}
```

- AWS SDKs: UpdatePolicy

Editing tags attached to organization policies with AWS Organizations

This topic describes how to edit tags attached policies with AWS Organizations. A *policy* defines the controls that you want to apply to a group of AWS accounts.

Topics

- Edit tags attached to a service control policy (SCP)
- Edit tags attached to a resource control policy (RCP)
- Edit tags attached to an declarative policy
- Edit tags attached to a backup policy
- Edit tags attached to a tag policy
- Edit tags attached to a chat applications policy
- Edit tags attached to an AI services opt-out policy

Edit tags attached to a service control policy (SCP)

When you sign in to your organization's management account, you can add or remove the tags attached to an SCP. For more information about tagging, see Tagging AWS Organizations resources.

Minimum permissions

To edit the tags attached to an SCP in your organization, you must have the following permissions:

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

AWS Management Console

To edit the tags attached to an SCP

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Service control policies** page choose the name of the policy with the tags that you want to edit.
3. On the policy details page, choose the **Tags** tab, and then choose**Manage tags**.
4. Make any or all of the following changes:
   - Change the value of a tag by entering a new value over the old one. You can't directly modify the tag key. To change a key, you must delete the tag with the old key and then add a tag with the new key.
   - Remove an existing tag by choosing **Remove**.

- Add a new tag key and value pair. Choose **Add tag**, then enter the new key name and optional value in the provided boxes. If you leave the **Value** box empty, the value is an empty string; it isn't `null`.

5. When you're finished, choose **Save changes**.

AWS CLI & AWS SDKs

To edit the tags attached to an SCP

You can use one of the following commands to edit the tags attached to an SCP:

- AWS CLI: tag-resource and untag-resource
- AWS SDKs: TagResource and UntagResource

Edit tags attached to a resource control policy (RCP)

When you sign in to your organization's management account, you can add or remove the tags attached to an RCP. For more information about tagging, see Tagging AWS Organizations resources.

Minimum permissions

To edit the tags attached to an RCP in your AWS organization, you must have the following permissions:

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

AWS Management Console

To edit the tags attached to an RCP

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Resource control policy** page, choose the name of the policy with the tags that you want to edit.
3. On the policy details page, choose the **Tags** tab, and then choose **Manage tags**.
4. Make any or all of the following changes:
   - Change the value of a tag by entering a new value over the old one. You can't directly modify the tag key. To change a key, you must delete the tag with the old key and then add a tag with the new key.
   - Remove an existing tag by choosing **Remove**.
   - Add a new tag key and value pair. Choose **Add tag**, then enter the new key name and optional value in the provided boxes. If you leave the **Value** box empty, the value is an empty string; it isn't `null`.
5. When you're finished, choose **Save changes**.

AWS CLI & AWS SDKs

To edit the tags attached to an RCP

You can use one of the following commands to edit the tags attached to an RCP:

- AWS CLI: tag-resource and untag-resource
- AWS SDKs: TagResource and UntagResource

Edit tags attached to an declarative policy

When you sign in to your organization's management account, you can add or remove the tags attached to a declarative policy. For more information about tagging, see Tagging AWS Organizations resources.

Minimum permissions

To edit the tags attached to a declarative policy in your AWS organization, you must have the following permissions:

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

AWS Management Console

To edit the tags attached to a declarative policy

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

AWS CLI & AWS SDKs

To edit the tags attached to a declarative policy

You can use one of the following commands to edit the tags attached to a declarative policy:

- AWS CLI: tag-resource and untag-resource
- AWS SDKs: TagResource and UntagResource

Edit tags attached to a backup policy

When you sign in to your organization's management account, you can add or remove the tags attached to a backup policy. For more information about tagging, see Tagging AWS Organizations resources.

Minimum permissions

To edit the tags attached to a backup policy in your organization, you must have the following permissions:

- `organizations:DescribeOrganization` (console only – to navigate to the policy)
- `organizations:DescribePolicy` (console only – to navigate to the policy)
- `organizations:TagResource`
- `organizations:UntagResource`

AWS Management Console

To edit the tags attached to an backup policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. **Backup policies** page
3. Choose the name of the policy with the tags that you want to edit.

   The policy detail page appears.
4. On the **Tags** tab, choose **Manage tags**.

5. You can perform any of these actions on this page:

   - Edit the value for any tag by entering a new value over the old one. You can't modify the key. To change a key, you must delete the tag with the old key and add a tag with the new key.
   - Remove an existing tag by choosing **Remove**.
   - Add a new tag key and value pair. Choose **Add tag**, then enter the new key name and optional value in the provided boxes. If you leave the **Value** box empty, the value is an empty string; it isn't `null`.
6. Choose **Save changes** after you've made all the additions, removals, and edits you want to make.

AWS CLI & AWS SDKs

To edit the tags attached to a backup policy

You can use one of the following commands to edit the tags attached to a backup policy:

- AWS CLI: tag-resource and untag-resource
- AWS SDKs: TagResource and UntagResource

Edit tags attached to a tag policy

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

Minimum permissions

To edit the tags attached to a tag policy in your organization, you must have the following permissions:

- `organizations:DescribeOrganization` (console only – to navigate to the policy)
- `organizations:DescribePolicy` (console only – to navigate to the policy)
- `organizations:TagResource`
- `organizations:UntagResource`

AWS Management Console

To edit the tags attached to a tag policy

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

AWS CLI & AWS SDKs

To edit the tags attached to a tag policy

You can use one of the following commands to edit the tags attached to a tag policy:

- AWS CLI: tag-resource and untag-resource
- AWS SDKs: TagResource and UntagResource

Edit tags attached to a chat applications policy

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

Minimum permissions

To edit the tags attached to a chat applications policy in your organization, you must have the following permissions:

- `organizations:DescribeOrganization` (console only – to navigate to the policy)
- `organizations:DescribePolicy` (console only – to navigate to the policy)
- `organizations:TagResource`
- `organizations:UntagResource`

AWS Management Console

To edit the tags attached to an chat applications policy

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

AWS CLI & AWS SDKs

To edit the tags attached to a chat applications policy

You can use one of the following commands to edit the tags attached to a chat applications policy:

- AWS CLI: tag-resource and untag-resource
- AWS SDKs: TagResource and UntagResource

Edit tags attached to an AI services opt-out policy

When you sign in to your organization's management account, you can add or remove the tags attached to an AI services opt-out policy. For more information about tagging, see Tagging AWS Organizations resources.

Minimum permissions

To edit the tags attached to an AI services opt-out policy in your organization, you must have the following permissions:

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

AWS Management Console

To edit the tags attached to an AI services opt-out policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AI services opt-out policies** page, choose the name of the policy with the tags that you want to edit.
3. On the chosen policy's detail page, choose the **Tags** tab, and then choose **Manage tags**.
4. You can perform any of these actions on this page:

- Edit the value for any tag by entering a new value over the old one. You can't modify the key. To change a key, you must delete the tag with the old key and add a tag with the new key.
- Remove an existing tag by choosing **Remove**.
- Add a new tag key and value pair. Choose **Add tag**, then enter the new key name and optional value in the provided boxes. If you leave the **Value** box empty, the value is an empty string; it isn't `null`.

5. Choose **Save changes** after you've made all the additions, removals, and edits you want to make.

AWS CLI & AWS SDKs

To edit the tags attached to a AI services opt-out policy

You can use one of the following commands to edit the tags attached to a AI services opt-out policy:

- AWS CLI: tag-resource and untag-resource
- AWS SDKs: TagResource and UntagResource

Attaching organization policies with AWS Organizations

This topic describes how to attach policies with AWS Organizations. A *policy* defines the controls that you want to apply to a group of AWS accounts.

Topics

- Attach policies with AWS Organizations

Attach policies with AWS Organizations

Minimum permissions

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

- `organizations:AttachPolicy`

Minimum permissions

To attach an authorization policy (SCP or RCP) to a root, OU, or account, you need permission to run the following action:

- `organizations:AttachPolicy` with a `Resource` element in the same policy statement that includes "*" or the Amazon Resource Name (ARN) of the specified policy and the ARN of the root, OU, or account that you want to attach the policy to

AWS Management Console

Service control policies (SCPs)

You can attach an SCP by either navigating to the policy or to the root, OU, or account that you want to attach the policy to.

To attach an SCP by navigating to the root, OU, or account

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page, navigate to and then choose the check box next to the root, OU, or account that you want to attach an SCP to. You might have to expand OUs (choose the ▶) to find the OU or account that you want.
3. In the **Policies** tab, in the entry for **Service control policies**, choose **Attach**.
4. Find the policy that you want and choose **Attach policy**.

   The list of attached SCPs on the **Policies** tab is updated to include the new addition. The policy change takes effect immediately, affecting the permissions of IAM users and roles in the attached account or all accounts under the attached root or OU.

To attach an SCP by navigating to the policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Service control policies** page, choose the name of the policy that you want to attach.
3. On the **Targets** tab, choose **Attach**.
4. Choose the radio button next to the root, OU, or account that you want to attach the policy to. You might have to expand OUs (choose the ) to find the OU or account that you want.
5. Choose **Attach policy**.

   The list of attached SCPs on the **Targets** tab is updated to include the new addition. The policy change takes effect immediately, affecting the permissions of IAM users and roles in the attached account or all accounts under the attached root or OU.

Resource control policies (RCPs)

You can attach an RCP by either navigating to the policy or to the root, OU, or account that you want to attach the policy to.

To attach an RCP by navigating to the root, OU, or account

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page, navigate to and then choose the check box next to the root, OU, or account that you want to attach an RCP to. You might have to expand OUs (choose the ) to find the OU or account that you want.
3. In the **Policies** tab, in the entry for **Resource control policies**, choose **Attach**.
4. Find the policy that you want and choose **Attach policy**.

The list of attached RCPs on the **Policies** tab is updated to include the new addition. The policy change takes effect immediately, affecting the permissions of resources in the attached account or all accounts under the attached root or OU.

To attach an RCP by navigating to the policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Resource control policy** page, choose the name of the policy that you want to attach.
3. On the **Targets** tab, choose **Attach**.
4. Choose the radio button next to the root, OU, or account that you want to attach the policy to. You might have to expand OUs (choose the ) to find the OU or account that you want.
5. Choose **Attach policy**.

   The list of attached RCPs on the **Targets** tab is updated to include the new addition. The policy change takes effect immediately, affecting the permissions of resources in the attached account or all accounts under the attached root or OU.

Declarative policies

You can attach a declarative policy by either navigating to the policy or to the root, OU, or account that you want to attach the policy to.

To attach a declarative policy by navigating to the root, OU, or account

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page, navigate to and then choose the name of the root, OU, or account that you want to attach a policy to. You might have to expand OUs (choose the ) to find the OU or account that you want.

3. In the **Policies** tab, in the entry for **Declarative policies**, choose **Attach**.
4. Find the policy that you want and choose **Attach policy**.

   The list of attached declarative policies on the **Policies** tab is updated to include the new addition. The policy change takes effect immediately.

To attach a declarative policy by navigating to the policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Declarative policies** page, choose the name of the policy that you want to attach.
3. On the **Targets** tab, choose **Attach**.
4. Choose the radio button next to the root, OU, or account that you want to attach the policy to. You might have to expand OUs (choose the ▶) to find the OU or account that you want.
5. Choose **Attach policy**.

   The list of attached declarative policies on the **Targets** tab is updated to include the new addition. The policy change takes effect immediately.

Backup policies

You can attach a backup policy by either navigating to the policy or to the root, OU, or account that you want to attach the policy to.

To attach a backup policy by navigating to the root, OU, or account

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page, navigate to and then choose the name of the root, OU, or account that you want to attach a policy to. You might have to expand OUs (choose the ▶) to find the OU or account that you want.
3. In the **Policies** tab, in the entry for **Backup policies**, choose **Attach**.

4. Find the policy that you want and choose **Attach policy**.

   The list of attached backup policies on the **Policies** tab is updated to include the new addition. The policy change takes effect immediately.

To attach a backup policy by navigating to the policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Backup policies** page, choose the name of the policy that you want to attach.
3. On the **Targets** tab, choose **Attach**.
4. Choose the radio button next to the root, OU, or account that you want to attach the policy to. You might have to expand OUs (choose the ▶) to find the OU or account that you want.
5. Choose **Attach policy**.

   The list of attached backup policies on the **Targets** tab is updated to include the new addition. The policy change takes effect immediately.

Tag policies

You can attach a tag policy by either navigating to the policy or to the root, OU, or account that you want to attach the policy to.

To attach a tag policy by navigating to the root, OU, or account

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page, navigate to and then choose the name of the root, OU, or account that you want to attach a policy to. You might have to expand OUs (choose the ▶) to find the OU or account that you want.
3. In the **Policies** tab, in the entry for **Tag policies**, choose **Attach**.
4. Find the policy that you want and choose **Attach policy**.

The list of attached tag policies on the **Policies** tab is updated to include the new addition. The policy change takes effect immediately.

To attach a tag policy by navigating to the policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Tag policies** page, choose the name of the policy that you want to attach.
3. On the **Targets** tab, choose **Attach**.
4. Choose the radio button next to the root, OU, or account that you want to attach the policy to. You might have to expand OUs (choose the) to find the OU or account that you want.
5. Choose **Attach policy**.

   The list of attached tag policies on the **Targets** tab is updated to include the new addition. The policy change takes effect immediately.

Chat applications policies

You can attach a chat applications policy by either navigating to the policy or to the root, OU, or account that you want to attach the policy to.

To attach a chat applications policy by navigating to the root, OU, or account

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page, navigate to and then choose the name of the root, OU, or account that you want to attach a policy to. You might have to expand OUs (choose the) to find the OU or account that you want.
3. In the **Policies** tab, in the entry for **Chat applications policies**, choose **Attach**.
4. Find the policy that you want and choose **Attach policy**.

The list of attached chat applications policies on the **Policies** tab is updated to include the new addition. The policy change takes effect immediately.

To attach a chat applications policy by navigating to the policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Chatbot policies** page, choose the name of the policy that you want to attach.
3. On the **Targets** tab, choose **Attach**.
4. Choose the radio button next to the root, OU, or account that you want to attach the policy to. You might have to expand OUs (choose the ) to find the OU or account that you want.
5. Choose **Attach policy**.

   The list of attached chat applications policies on the **Targets** tab is updated to include the new addition. The policy change takes effect immediately.

AI services opt-out policies

You can attach an AI services opt-out policy by either navigating to the policy or to the root, OU, or account that you want to attach the policy to.

To attach an AI services opt-out policy by navigating to the root, OU, or account

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page, navigate to and then choose the name of the root, OU, or account that you want to attach a policy to. You might have to expand OUs (choose the ) to find the OU or account that you want.
3. In the **Policies** tab, in the entry for **AI service opt-out policies**, choose **Attach**.
4. Find the policy that you want and choose **Attach policy**.

The list of attached AI services opt-out policies on the **Policies** tab is updated to include the new addition. The policy change takes effect immediately.

To attach an AI services opt-out policy by navigating to the policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AI services opt-out policies** page, choose the name of the policy that you want to attach.
3. On the **Targets** tab, choose **Attach**.
4. Choose the radio button next to the root, OU, or account that you want to attach the policy to. You might have to expand OUs (choose the ▶) to find the OU or account that you want.
5. Choose **Attach policy**.

   The list of attached AI services opt-out policies on the **Targets** tab is updated to include the new addition. The policy change takes effect immediately.

AWS CLI & AWS SDKs

To attach a policy

The following code examples show how to use `AttachPolicy`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
```

```
    using System.Threading.Tasks;
    using Amazon.Organizations;
    using Amazon.Organizations.Model;

    /// <summary>
    /// Shows how to attach an AWS Organizations policy to an organization,
    /// an organizational unit, or an account.
    /// </summary>
    public class AttachPolicy
    {
        /// <summary>
        /// Initializes the Organizations client object and then calls the
        /// AttachPolicyAsync method to attach the policy to the root
        /// organization.
        /// </summary>
        public static async Task Main()
        {
            IAmazonOrganizations client = new AmazonOrganizationsClient();
            var policyId = "p-00000000";
            var targetId = "r-0000";

            var request = new AttachPolicyRequest
            {
                PolicyId = policyId,
                TargetId = targetId,
            };

            var response = await client.AttachPolicyAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully attached Policy ID {policyId} to
Target ID: {targetId}.");
            }
            else
            {
                Console.WriteLine("Was not successful in attaching the policy.");
            }
        }
    }
```

- For API details, see AttachPolicy in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

To attach a policy to a root, OU, or account

Example 1

The following example shows how to attach a service control policy (SCP) to an OU:

```
aws organizations attach-policy
                --policy-id p-examplepolicyid111
                --target-id ou-examplerootid111-exampleouid111
```

Example 2

The following example shows how to attach a service control policy directly to an account:

```
aws organizations attach-policy
                --policy-id p-examplepolicyid111
                --target-id 333333333333
```

- For API details, see AttachPolicy in *AWS CLI Command Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
def attach_policy(policy_id, target_id, orgs_client):
    """
    Attaches a policy to a target. The target is an organization root, account,
 or
    organizational unit.
```

```
    :param policy_id: The ID of the policy to attach.
    :param target_id: The ID of the resources to attach the policy to.
    :param orgs_client: The Boto3 Organizations client.
    """
    try:
        orgs_client.attach_policy(PolicyId=policy_id, TargetId=target_id)
        logger.info("Attached policy %s to target %s.", policy_id, target_id)
    except ClientError:
        logger.exception(
            "Couldn't attach policy %s to target %s.", policy_id, target_id
        )
        raise
```

- For API details, see AttachPolicy in *AWS SDK for Python (Boto3) API Reference*.

The policy change takes effect immediately, affecting the permissions of IAM users and roles in the attached account or all accounts under the attached root or OU

Detaching organization policies with AWS Organizations

This topic describes how to detach policies with AWS Organizations. A *policy* defines the controls that you want to apply to a group of AWS accounts.

Topics

- Detach policies with AWS Organizations

Detach policies with AWS Organizations

Minimum permissions

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

- `organizations:DetachPolicy`

> **Note**
>
> You can't detach the last authorization policy (SCP or RCP) from a root, an OU, or an account. There must be at least one SCP and RCP attached to every root, OU, and account at all times.

AWS Management Console

Service control policies (SCPs)

You can detach an SCP by either navigating to the policy or to the root, OU, or account that you want to detach the policy from.

To detach an SCP by navigating to the root, OU, or account it's attached to

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page navigate to the Root, OU, or account that you want to detach a policy from. You might have to expand OUs (choose the ▶) to find the OU or account that you want. Choose the name of the Root, OU, or account.
3. On the **Policies** tab, choose the radio button next to the SCP that you want to detach, and then choose **Detach**.
4. In the confirmation dialog box, choose **Detach policy**.

   The list of attached SCPs is updated. The policy change caused by detaching the SCP takes effect immediately. For example, detaching an SCP immediately affects the permissions of IAM users and roles in the formerly attached account or accounts under the formerly attached organization root or OU.

To detach an SCP by navigating to the policy

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

2. On the **Service control policies** page, choose the name of the policy that you want to detach from a root, OU, or account.
3. On the **Targets** tab, choose the radio button next to the root, OU, or account that you want to detach the policy from. You might have to expand OUs (choose the ▶) to find the OU or account that you want.
4. Choose **Detach**.
5. In the confirmation dialog box, choose **Detach**.

   The list of attached SCPs is updated. The policy change caused by detaching the SCP takes effect immediately. For example, detaching an SCP immediately affects the permissions of IAM users and roles in the formerly attached account or accounts under the formerly attached organization root or OU.

Resource control policies (RCPs)

You can detach an RCP by either navigating to the policy or to the root, OU, or account that you want to detach the policy from. After you detach an RCP from an entity, that RCP no longer applies to any resources that were affected by the now detached entity.

> **Note**
>
> **You cannot detach the `RCPFullAWSAccess` policy**
> The `RCPFullAWSAccess` policy is automatically attached to the root, every OU, and every account in your organization. You cannot detach this policy.

To detach an RCP by navigating to the root, OU, or account it's attached to

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

3. On the **Policies** tab, choose the radio button next to the RCP that you want to detach, and then choose **Detach**.
4. In the confirmation dialog box, choose **Detach policy**.

   The list of attached RCPs is updated. The policy change caused by detaching the RCP takes effect immediately. For example, detaching an RCP 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 RCP by navigating to the policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Resource control policy** page, choose the name of the policy that you want to detach from a root, OU, or account.
3. On the **Targets** tab, choose the radio button next to the root, OU, or account that you want to detach the policy from. You might have to expand OUs (choose the ▶) to find the OU or account that you want.
4. Choose **Detach**.
5. In the confirmation dialog box, choose **Detach**.

   The list of attached RCPs is updated. The policy change caused by detaching the RCP takes effect immediately. For example, detaching an RCP immediately affects the permissions of IAM users and roles in the formerly attached account or accounts under the formerly attached organization root or OU.

Declarative policies

You can detach a declarative policy by either navigating to the policy or to the root, OU, or account that you want to detach the policy from.

To detach a declarative policy by navigating to the root, OU, or account it's attached to

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page navigate to the Root, OU, or account that you want to detach a policy from. You might have to expand OUs (choose the ▶) to find the OU or account that you want. Choose the name of the Root, OU, or account.
3. On the **Policies** tab, choose the radio button next to the declarative policy that you want to detach, and then choose **Detach**.
4. In the confirmation dialog box, choose **Detach policy**.

   The list of attached declarative policies is updated. The policy change takes effect immediately.

To detach a declarative policy by navigating to the policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Declarative policies** page, choose the name of the policy that you want to detach from a root, OU, or account.
3. On the **Targets** tab, choose the radio button next to the root, OU, or account that you want to detach the policy from. You might have to expand OUs (choose the ▶) to find the OU or account that you want.
4. Choose **Detach**.
5. In the confirmation dialog box, choose **Detach**.

   The list of attached declarative policies is updated. The policy change takes effect immediately.

Backup policies

You can detach a backup policy by either navigating to the policy or to the root, OU, or account that you want to detach the policy from.

To detach a backup policy by navigating to the root, OU, or account it's attached to

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page navigate to the Root, OU, or account that you want to detach a policy from. You might have to expand OUs (choose the) to find the OU or account that you want. Choose the name of the Root, OU, or account.
3. On the **Policies** tab, choose the radio button next to the backup policy that you want to detach, and then choose **Detach**.
4. In the confirmation dialog box, choose **Detach policy**.

   The list of attached backup policies is updated. The policy change takes effect immediately.

To detach a backup policy by navigating to the policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Backup policies** page, choose the name of the policy that you want to detach from a root, OU, or account.
3. On the **Targets** tab, choose the radio button next to the root, OU, or account that you want to detach the policy from. You might have to expand OUs (choose the) to find the OU or account that you want.
4. Choose **Detach**.
5. In the confirmation dialog box, choose **Detach**.

   The list of attached backup policies is updated. The policy change takes effect immediately.

Tag policies

You can detach a tag policy by either navigating to the policy or to the root, OU, or account that you want to detach the policy from.

To detach a tag policy by navigating to the root, OU, or account it's attached to

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page navigate to the Root, OU, or account that you want to detach a policy from. You might have to expand OUs (choose the ▶) to find the OU or account that you want. Choose the name of the Root, OU, or account.
3. On the **Policies** tab, choose the radio button next to the tag policy that you want to detach, and then choose **Detach**.
4. In the confirmation dialog box, choose **Detach policy**.

   The list of attached tag policies is updated. The policy change takes effect immediately.

To detach a tag policy by navigating to the policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Tag policies** page, choose the name of the policy that you want to detach from a root, OU, or account.
3. On the **Targets** tab, choose the radio button next to the root, OU, or account that you want to detach the policy from. You might have to expand OUs (choose the ▶) to find the OU or account that you want.
4. Choose **Detach**.
5. In the confirmation dialog box, choose **Detach**.

   The list of attached tag policies is updated. The policy change takes effect immediately.

Chat applications policies

You can detach a chat applications policy by either navigating to the policy or to the root, OU, or account that you want to detach the policy from.

To detach a chat applications policy by navigating to the root, OU, or account it's attached to

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page navigate to the Root, OU, or account that you want to detach a policy from. You might have to expand OUs (choose the ) to find the OU or account that you want. Choose the name of the Root, OU, or account.
3. On the **Policies** tab, choose the radio button next to the chat applications policy that you want to detach, and then choose **Detach**.
4. In the confirmation dialog box, choose **Detach policy**.

   The list of attached chat applications policies is updated. The policy change takes effect immediately.

To detach a chat applications policy by navigating to the policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Chatbot policies** page, choose the name of the policy that you want to detach from a root, OU, or account.
3. On the **Targets** tab, choose the radio button next to the root, OU, or account that you want to detach the policy from. You might have to expand OUs (choose the ) to find the OU or account that you want.
4. Choose **Detach**.
5. In the confirmation dialog box, choose **Detach**.

The list of attached chat applications policies is updated. The policy change takes effect immediately.

AI services opt-out policies

You can detach an AI services opt-out policy by either navigating to the policy or to the root, OU, or account that you want to detach the policy from.

To detach an AI services opt-out policy by navigating to the root, OU, or account it's attached to

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AWS accounts** page navigate to the Root, OU, or account that you want to detach a policy from. You might have to expand OUs (choose the ) to find the OU or account that you want. Choose the name of the Root, OU, or account.
3. On the **Policies** tab, choose the radio button next to the AI services opt-out policy that you want to detach, and then choose **Detach**.
4. In the confirmation dialog box, choose **Detach policy**.

   The list of attached AI services opt-out policies is updated. The policy change takes effect immediately.

To detach an AI services opt-out policy by navigating to the policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AI services opt-out policies** page, choose the name of the policy that you want to detach from a root, OU, or account.
3. On the **Targets** tab, choose the radio button next to the root, OU, or account that you want to detach the policy from. You might have to expand OUs (choose the ) to find the OU or account that you want.

4. Choose **Detach**.
5. In the confirmation dialog box, choose **Detach**.

   The list of attached AI services opt-out policies is updated. The policy change takes effect immediately.

AWS CLI & AWS SDKs

To attach a policy

The following code examples show how to use `DetachPolicy`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to detach a policy from an AWS Organizations organization,
/// organizational unit, or account.
/// </summary>
public class DetachPolicy
{
    /// <summary>
    /// Initializes the Organizations client object and uses it to call
    /// DetachPolicyAsync to detach the policy.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();
```

```
            var policyId = "p-00000000";
            var targetId = "r-0000";

            var request = new DetachPolicyRequest
            {
                PolicyId = policyId,
                TargetId = targetId,
            };

            var response = await client.DetachPolicyAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully detached policy with Policy Id:
{policyId}.");
            }
            else
            {
                Console.WriteLine("Could not detach the policy.");
            }
        }
    }
```

- For API details, see DetachPolicy in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

To detach a policy from a root, OU, or account

The following example shows how to detach a policy from an OU:

```
aws organizations  detach-policy  --target-id ou-examplerootid111-exampleouid111
 --policy-id p-examplepolicyid111
```

- For API details, see DetachPolicy in *AWS CLI Command Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
def detach_policy(policy_id, target_id, orgs_client):
    """
    Detaches a policy from a target.

    :param policy_id: The ID of the policy to detach.
    :param target_id: The ID of the resource where the policy is currently
 attached.
    :param orgs_client: The Boto3 Organizations client.
    """
    try:
        orgs_client.detach_policy(PolicyId=policy_id, TargetId=target_id)
        logger.info("Detached policy %s from target %s.", policy_id, target_id)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from target %s.", policy_id, target_id
        )
        raise
```

- For API details, see DetachPolicy in *AWS SDK for Python (Boto3) API Reference*.

The policy change takes effect immediately, affecting the permissions of IAM users and roles and resources, if applicable, in the attached account or all accounts under the attached root or OU.

Getting information about your organization's policies

This topic 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.

Topics

- Listing all policies
- Listing the policies attached to a root, OU, or account
- Listing all roots, OUs, and accounts that a policy is attached to
- Getting details about a policy

Listing all policies

Minimum permissions

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

- `organizations:ListPolicies`

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

AWS Management Console

To list all of the policies in your organization

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

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

AWS CLI & AWS SDKs

The following code examples show how to use `ListPolicies`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to list the AWS Organizations policies associated with an
/// organization.
/// </summary>
public class ListPolicies
{
    /// <summary>
    /// Initializes an Organizations client object, and then calls its
    /// ListPoliciesAsync method.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        // The value for the Filter parameter is required and must must be
        // one of the following:
        //     AISERVICES_OPT_OUT_POLICY
        //     BACKUP_POLICY
        //     SERVICE_CONTROL_POLICY
        //     TAG_POLICY
        var request = new ListPoliciesRequest
        {
            Filter = "SERVICE_CONTROL_POLICY",
```

```
                MaxResults = 5,
            };

            var response = new ListPoliciesResponse();
            try
            {
                do
                {
                    response = await client.ListPoliciesAsync(request);
                    response.Policies.ForEach(p => DisplayPolicies(p));
                    if (response.NextToken is not null)
                    {
                        request.NextToken = response.NextToken;
                    }
                }
                while (response.NextToken is not null);
            }
            catch (AWSOrganizationsNotInUseException ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        /// <summary>
        /// Displays information about the Organizations policies associated
        /// with an organization.
        /// </summary>
        /// <param name="policy">An Organizations policy summary to display
        /// information on the console.</param>
        private static void DisplayPolicies(PolicySummary policy)
        {
            string policyInfo = $"{policy.Id}
{policy.Name}\t{policy.Description}";

            Console.WriteLine(policyInfo);
        }
    }
```

- For API details, see ListPolicies in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

To retrieve a list of all policies in an organization of a certain type

The following example shows you how to get a list of SCPs, as specified by the filter parameter:

```
aws organizations list-policies --filter SERVICE_CONTROL_POLICY
```

The output includes a list of policies with summary information:

```
{
        "Policies": [
                {
                        "Type": "SERVICE_CONTROL_POLICY",
                        "Name": "AllowAllS3Actions",
                        "AwsManaged": false,
                        "Id": "p-examplepolicyid111",
                        "Arn": "arn:aws:organizations::111111111111:policy/
service_control_policy/p-examplepolicyid111",
                        "Description": "Enables account admins to delegate
 permissions for any S3 actions to users and roles in their accounts."
                },
                {
                        "Type": "SERVICE_CONTROL_POLICY",
                        "Name": "AllowAllEC2Actions",
                        "AwsManaged": false,
                        "Id": "p-examplepolicyid222",
                        "Arn": "arn:aws:organizations::111111111111:policy/
service_control_policy/p-examplepolicyid222",
                        "Description": "Enables account admins to delegate
 permissions for any EC2 actions to users and roles in their accounts."
                },
                {
                        "AwsManaged": true,
                        "Description": "Allows access to every operation",
                        "Type": "SERVICE_CONTROL_POLICY",
                        "Id": "p-FullAWSAccess",
                        "Arn": "arn:aws:organizations::aws:policy/
service_control_policy/p-FullAWSAccess",
                        "Name": "FullAWSAccess"
```

```
                }
        ]
}
```

- For API details, see ListPolicies in *AWS CLI Command Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
def list_policies(policy_filter, orgs_client):
    """
    Lists the policies for the account, limited to the specified filter.

    :param policy_filter: The kind of policies to return.
    :param orgs_client: The Boto3 Organizations client.
    :return: The list of policies found.
    """
    try:
        response = orgs_client.list_policies(Filter=policy_filter)
        policies = response["Policies"]
        logger.info("Found %s %s policies.", len(policies), policy_filter)
    except ClientError:
        logger.exception("Couldn't get %s policies.", policy_filter)
        raise
    else:
        return policies
```

- For API details, see ListPolicies in *AWS SDK for Python (Boto3) API Reference*.

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

Minimum permissions

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

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

AWS Management Console

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

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

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

AWS CLI & AWS SDKs

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

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

- AWS CLI: list-policies-for-target

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

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

- AWS SDKs: ListPoliciesForTarget

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

Minimum permissions

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

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

AWS Management Console

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

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

3. Choose the **Targets** tab, to display a table of every root, OU, and account that the chosen policy is attached to.

AWS CLI & AWS SDKs

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

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

- AWS CLI: list-targets-for-policy

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

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

```
            "Type": "ROOT"
        }
    ]
}
```

- AWS SDKs: ListTargetsForPolicy

Getting details about a policy

Minimum permissions

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

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

AWS Management Console

To get details about a policy

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

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

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

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

AWS CLI & AWS SDKs

The following code examples show how to use `DescribePolicy`.

CLI

AWS CLI

To get information about a policy

The following example shows how to request information about a policy:

```
aws organizations describe-policy --policy-id p-examplepolicyid111
```

The output includes a policy object that contains details about the policy:

```
{
        "Policy": {
                "Content": "{\n  \"Version\": \"2012-10-17\",\n  \"Statement
\": [\n    {\n      \"Effect\": \"Allow\",\n      \"Action\": \"*\",\n
 \"Resource\": \"*\"\n    }\n  ]\n}",
                "PolicySummary": {
                        "Arn": "arn:aws:organizations::111111111111:policy/o-
exampleorgid/service_control_policy/p-examplepolicyid111",
                        "Type": "SERVICE_CONTROL_POLICY",
                        "Id": "p-examplepolicyid111",
                        "AwsManaged": false,
                        "Name": "AllowAllS3Actions",
                        "Description": "Enables admins to delegate S3
 permissions"
                }
        }
}
```

- For API details, see DescribePolicy in *AWS CLI Command Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
def describe_policy(policy_id, orgs_client):
    """
    Describes a policy.

    :param policy_id: The ID of the policy to describe.
    :param orgs_client: The Boto3 Organizations client.
    :return: The description of the policy.
    """
    try:
        response = orgs_client.describe_policy(PolicyId=policy_id)
        policy = response["Policy"]
        logger.info("Got policy %s.", policy_id)
    except ClientError:
        logger.exception("Couldn't get policy %s.", policy_id)
        raise
    else:
        return policy
```

- For API details, see DescribePolicy in *AWS SDK for Python (Boto3) API Reference*.

Deleting organization policies with AWS Organizations

When you no longer need a policy and after you detach it from all organizational units (OUs) and accounts, you can delete it.

This topic describes how to delete policies with AWS Organizations. A *policy* defines the controls that you want to apply to a group of AWS accounts.

Topics

- Delete policies with AWS Organizations

Delete policies with AWS Organizations

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

Before you can delete a policy, you must first detach it from all attached entities.

> **Note**
>
> - You can't delete any AWS managed SCP such as the SCP named `FullAWSAccess`.
> - You can't delete any AWS managed RCP such as the RCP named `RCPFullAWSAccess`.

> **Minimum permissions**
>
> To delete a policy, you need permission to run the following action:
>
> - `organizations:DeletePolicy`

AWS Management Console

Service control policies (SCPs)

To delete an SCP

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Service control policies** page, choose the name of the SCP that you want to delete.
3. You must first detach the policy that you want to delete from all roots, OUs, and accounts. Choose the **Targets** tab, choose the radio button next to each root, OU, or account that is shown in the **Targets** list, and then choose **Detach**. In the confirmation dialog box, choose **Detach**. Repeat until you remove all targets.

4. Choose **Delete** at the top of the page.
5. On the confirmation dialog box, enter the name of the policy, and then choose **Delete**.

Resource control policies (RCPs)

To delete an RCP

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Resource control policies** page, choose the name of the RCP that you want to delete.
3. You must first detach the policy that you want to delete from all roots, OUs, and accounts. Choose the **Targets** tab, choose the radio button next to each root, OU, or account that is shown in the **Targets** list, and then choose **Detach**. In the confirmation dialog box, choose **Detach**. Repeat until you remove all targets.
4. Choose **Delete** at the top of the page.
5. On the confirmation dialog box, enter the name of the policy, and then choose **Delete**.

Declarative policies

To delete a declarative policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Declarative policies** page, choose the name of the policy that you want to delete.
3. You must first detach the policy that you want to delete from all roots, OUs, and accounts. Choose the **Targets** tab, choose the radio button next to each root, OU, or account that is shown in the **Targets** list, and then choose **Detach**. In the confirmation dialog box, choose **Detach**. Repeat until you remove all targets.
4. Choose **Delete** at the top of the page.
5. On the confirmation dialog box, enter the name of the policy, and then choose **Delete**.

Backup policies

To delete a backup policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Backup policies** page, choose the name of the backup policy that you want to delete.
3. You must first detach the backup policy that you want to delete from all roots, OUs, and accounts. Choose the **Targets** tab, choose the radio button next to each root, OU, or account that is shown in the **Targets** list, and then choose **Detach**. In the confirmation dialog box, choose **Detach**. Repeat until you remove all targets.
4. Choose **Delete** at the top of the page.
5. On the confirmation dialog box, enter the name of the policy, and then choose **Delete**.

Tag policies

To delete a tag policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Tag policies page**, choose the policy that you want to delete.
3. You must first detach the policy that you want to delete from all roots, OUs, and accounts. Choose the **Targets** tab, choose the radio button next to each root, OU, or account that's shown in the **Targets** list, and then choose **Detach**. In the confirmation dialog box, choose **Detach**. Repeat until you remove all targets.
4. Choose **Delete** at the top of the page.
5. On the confirmation dialog box, enter the name of the policy, and then choose **Delete**.

Chat applications policies

To delete a chat applications policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Chatbot policies** page, choose the name of the policy that you want to delete.
3. You must first detach the policy that you want to delete from all roots, OUs, and accounts. Choose the **Targets** tab, choose the radio button next to each root, OU, or account that is shown in the **Targets** list, and then choose **Detach**. In the confirmation dialog box, choose **Detach**. Repeat until you remove all targets.
4. Choose **Delete** at the top of the page.
5. On the confirmation dialog box, enter the name of the policy, and then choose **Delete**.

AI services opt-out policies

To delete an AI services opt-out policy

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **AI services opt-out policies** page, choose the name of the policy that you want to delete.
3. You must first detach the policy that you want to delete from all roots, OUs, and accounts. Choose the **Targets** tab, choose the radio button next to each root, OU, or account that is shown in the **Targets** list, and then choose **Detach**. In the confirmation dialog box, choose **Detach**. Repeat until you remove all targets.
4. Choose **Delete** at the top of the page.
5. On the confirmation dialog box, enter the name of the policy, and then choose **Delete**.

AWS CLI & AWS SDKs

To delete an a policy

The following code examples show how to use `DeletePolicy`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Deletes an existing AWS Organizations policy.
/// </summary>
public class DeletePolicy
{
    /// <summary>
    /// Initializes the Organizations client object and then uses it to
    /// delete the policy with the specified policyId.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var policyId = "p-00000000";

        var request = new DeletePolicyRequest
        {
            PolicyId = policyId,
        };

        var response = await client.DeletePolicyAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully deleted Policy: {policyId}.");
        }
```

```
            else
            {
                Console.WriteLine($"Could not delete Policy: {policyId}.");
            }
        }
    }
```

- For API details, see DeletePolicy in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

To delete a policy

The following example shows how to delete a policy from an organization. The example assumes that you previously detached the policy from all entities:

```
aws organizations delete-policy --policy-id p-examplepolicyid111
```

- For API details, see DeletePolicy in *AWS CLI Command Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
def delete_policy(policy_id, orgs_client):
    """
    Deletes a policy.

    :param policy_id: The ID of the policy to delete.
    :param orgs_client: The Boto3 Organizations client.
```

```
    """
    try:
        orgs_client.delete_policy(PolicyId=policy_id)
        logger.info("Deleted policy %s.", policy_id)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_id)
        raise
```

- For API details, see DeletePolicy in *AWS SDK for Python (Boto3) API Reference*.

Tagging AWS Organizations resources

A *tag* is a custom attribute label that you add to an AWS resource to make it easier to identify, organize, and search for resources. Each tag has two parts:

- A *tag key* (for example, `CostCenter`, `Environment`, or `Project`). Tag keys can be up to 128 characters in length and are case sensitive.
- A *tag value* (for example, `111122223333` or `Production`). Tag values can be up to 256 characters in length, and like tag keys, are case sensitive. You can set the value of a tag to an empty string, but you can't set the value of a tag to null. Omitting the tag value is the same as using an empty string.

For more information about what characters are allowed in a tag key or value, see the Tags parameter of the Tag API in the *Resource Groups Tagging API Reference*.

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

> **Tip**
>
> Use tag policies to help standardize your implementation of tags across the resources in your organization's accounts.

Topics

- Considerations
- Using tags
- Adding, updating, and removing tags

Considerations

AWS Organizations supports the following tagging operations when you are logged in to the management account:

You can add tags to the following organization resources

- AWS accounts
- Organizational units
- The organization's root
- Policies

You can add tags at the following times

- When you create the resource — Specify the tags in either the Organizations console, or use the `Tags` parameter with one of the `Create` API operations. This isn't applicable to the organization's root.
- After you create the resource — Use the Organizations console, or call the `TagResource` operation.

Other considerations

You can view the tags on any of the taggable resources in AWS Organizations by using the console or by calling the `ListTagsForResource` operation.

You can remove tags from a resource by specifying the keys to remove by using the console or by calling the `UntagResource` operation.

Using tags

Tags help you to organize resources in your organization by enabling you to group them by whatever categories are useful to you. For example, you can assign a "Department" tag that tracks the owning department. You can assign an "Environment" tag to track whether a given resource is part of your alpha, beta, gamma, or production environments.

You can also use tags to:

- Enforce tagging standards on your resources.
- Control who can access your resources.

Adding, updating, and removing tags

When you sign in to your organization's management account, you can add tags to the resources in your organization.

Adding tags to a resource when you create it

Minimum permissions

To add tags to a resource when you create it, you need the following permissions:

- Permission to create a resource of the specified type
- `organizations:TagResource`
- `organizations:ListTagsForResource` – required only when using the Organizations console

You can include tag keys and values that are attached to the following resources as you create them.

- AWS account
  - Created account
  - Invited account
- Organizational unit (OU)
- Policy
  - Service control policy
  - Resource control policy
  - Declarative policy
  - Backup policy
  - Tag policy
  - Chat applications policy
  - AI services opt-out policy

The organization root is created when you initially create the organization, so you can only add tags to it as an existing resource.

Adding or updating tags for an existing resource

You can also add new tags or update the values of tags attached to existing resources.

Minimum permissions

To add or update tags to resources in your organization, you need the following permissions:

- `organizations:TagResource`
- `organizations:ListTagsForResource` – required only when using the Organizations console

To remove tags from resources in your organization, you need the following permissions:

- `organizations:UntagResource`

AWS Management Console

To add, update, or remove tags for an existing resource

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. Navigate to and choose the account, Root, OU, or policy, and click on its name to open its detail page.
3. On the **Tags** tab, choose **Manage tags**.
4. You can add new tags, modify the values of existing tags, or remove tags.

   To add a tag, choose **Add tag**, and then enter a **Key** and, optionally, a **Value** for the tag.

   To remove a tag, choose **Remove**.

   Tag keys and values are case sensitive. Use the capitalization that you want to standardize on. You must also comply with the requirements of any tag policies that apply.
5. Repeat the previous step as many times as you need.
6. Choose **Save changes**.

AWS CLI & AWS SDKs

To add or update tags to an existing resource

You can use one of the following commands to add tags to the taggable resources in your organization:

- AWS CLI: tag-resource
- AWS SDKs: TagResource

To delete tags from a resource in your organization

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

- AWS CLI: untag-resource
- AWS SDKs: UntagResource

Using AWS Organizations with other AWS services

You can use *trusted access* to enable a supported AWS service that you specify, called the *trusted service*, to perform tasks in your organization and its accounts on your behalf. This involves granting permissions to the trusted service but does *not* otherwise affect the permissions for users or roles. When you enable access, the trusted service can create an IAM role called a *service-linked role* in every account in your organization whenever that role is needed. That role has a permissions policy that allows the trusted service to do the tasks that are described in that service's documentation. This enables you to specify settings and configuration details that you would like the trusted service to maintain in your organization's accounts on your behalf. The trusted service only creates service-linked roles when it needs to perform management actions on accounts, and not necessarily in all accounts of the organization.

Important

We ***strongly recommend*** that, when the option is available, you enable and disable trusted access by using ***only*** the trusted service's console, or its AWS CLI or API operation equivalents. This lets the trusted service perform any required initialization when enabling trusted access, such as creating any required resources and any required clean up of resources when disabling trusted access.
For information about how to enable or disable trusted service access to your organization using the trusted service, see the **Learn more** link under the **Supports Trusted Access** column at AWS services that you can use with AWS Organizations.
If you disable access by using the Organizations console, CLI commands, or API operations, it causes the following actions to occur:

- The service can no longer create a service-linked role in the accounts in your organization. This means that the service can't perform operations on your behalf on any new accounts in your organization. The service can still perform operations in older accounts until the service completes its clean-up from AWS Organizations.
- The service can no longer perform tasks in the member accounts in the organization, unless those operations are explicitly permitted by the IAM policies that are attached to your roles. This includes any data aggregation from the member accounts to the management account, or to a delegated administrator account, where relevant.

- Some services detect this and clean up any remaining data or resources related to the integration, while other services stop accessing the organization but leave any historical data and configuration in place to support a possible re-enabling of the integration.

Instead, using the other service's console or commands to disable the integration ensures that the other service can clean up any resources that are required only for the integration. How the service cleans up its resources in the organization's accounts depends on that service. For more information, see the documentation for the other AWS service.

Permissions required to enable trusted access

Trusted access requires permissions for two services: AWS Organizations and the trusted service. To enable trusted access, choose one of the following scenarios:

- If you have credentials with permissions in both AWS Organizations and the trusted service, enable access by using the tools (console or AWS CLI) provided by the trusted service. This allows the service to enable trusted access in AWS Organizations on your behalf and to create any resources required for the service to operate in your organization.

  The minimum permissions for these credentials are the following:

  - `organizations:EnableAWSServiceAccess`. You can also use the `organizations:ServicePrincipal` condition key with this operation to limit requests that those operations make to a list of approved service principal names. For more information, see Condition keys.
  - `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.

2. The person who has credentials with permissions in the trusted service enables that service to work with AWS Organizations. This instructs the service to perform any required initialization, such as creating any resources that are required for the trusted service to operate in the organization. For more information, see the service-specific instructions at AWS services that you can use with AWS Organizations.

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) in AWS Organizations.
> You can apply SCPs to only member accounts. SCPs don't apply to the management account. We recommend that you don't run services in the management account. Instead, run them in member accounts where you can control the security by using SCPs.

- If you have credentials with permissions in both AWS Organizations and the trusted service, disable access by using the tools (console or AWS CLI) that are available for the trusted service. The service then cleans up by removing resources that are no longer required and by disabling trusted access for the service in AWS Organizations on your behalf.

  The minimum permissions for these credentials are the following:

- `organizations:DisableAWSServiceAccess`. You can also use the `organizations:ServicePrincipal` condition key with this operation to limit requests that those operations make to a list of approved service principal names. For more information, see Condition keys.
- `organizations:ListAWSServiceAccessForOrganization` – Required if you use the AWS Organizations console.
- The minimum permissions required by the trusted service depend on the service. For more information, see the trusted service's documentation.

- If the credentials with permissions in AWS Organizations aren't the credentials with permissions in the trusted service, perform these steps in the following order:
  1. The person with permissions in the trusted service first disables access using that service. This instructs the trusted service to clean up by removing the resources required for trusted access. For more information, see the service-specific instructions at AWS services that you can use with AWS Organizations.
  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.

How to enable or disable trusted access

If you have permissions only for AWS Organizations and you want to enable or disable trusted access to your organization on behalf of the administrator of the other AWS service, use the following procedure.

> **Important**
>
> We ***strongly recommend*** that, when the option is available, you enable and disable trusted access by using ***only*** the trusted service's console, or its AWS CLI or API operation

equivalents. This lets the trusted service perform any required initialization when enabling trusted access, such as creating any required resources and any required clean up of resources when disabling trusted access.
For information about how to enable or disable trusted service access to your organization using the trusted service, see the **Learn more** link under the **Supports Trusted Access** column at AWS services that you can use with AWS Organizations.
If you disable access by using the Organizations console, CLI commands, or API operations, it causes the following actions to occur:

- The service can no longer create a service-linked role in the accounts in your organization. This means that the service can't perform operations on your behalf on any new accounts in your organization. The service can still perform operations in older accounts until the service completes its clean-up from AWS Organizations.
- The service can no longer perform tasks in the member accounts in the organization, unless those operations are explicitly permitted by the IAM policies that are attached to your roles. This includes any data aggregation from the member accounts to the management account, or to a delegated administrator account, where relevant.
- Some services detect this and clean up any remaining data or resources related to the integration, while other services stop accessing the organization but leave any historical data and configuration in place to support a possible re-enabling of the integration.

Instead, using the other service's console or commands to disable the integration ensures that the other service can clean up any resources that are required only for the integration. How the service cleans up its resources in the organization's accounts depends on that service. For more information, see the documentation for the other AWS service.

AWS Management Console

To enable trusted service access

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for the service that you want to enable, and choose its name.

3. Choose **Enable trusted access**.
4. In the confirmation dialog box, check the box to **Show the option to enable trusted access**, enter **`enable`** in the box, and then choose **Enable trusted access**.
5. If you are *enabling* access, tell the administrator of the other AWS service that they can now enable the other service to work with AWS Organizations.

To disable trusted service access

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for the service that you want to disable, and choose its name.
3. Wait until the administrator of the other service tells you that the service is disabled and that its resources have been cleaned up.
4. In the confirmation dialog box, enter **`disable`** in the box, and then choose **Disable trusted access**.

AWS CLI, AWS API

To enable or disable trusted service access

You can use the following AWS CLI commands or API operations to enable or disable trusted service access:

- AWS CLI: AWS organizations enable-aws-service-access
- AWS CLI: AWS organizations disable-aws-service-access
- AWS API: EnableAWSServiceAccess
- AWS API: DisableAWSServiceAccess

AWS Organizations and service-linked roles

AWS Organizations uses IAM service-linked roles to enable trusted services to perform tasks on your behalf in your organization's member accounts. When you configure a trusted service and authorize it to integrate with your organization, that service can request that AWS Organizations

create a service-linked role in its member account. The trusted service does this asynchronously as needed and not necessarily in all accounts in the organization at the same time. The service-linked role has predefined IAM permissions that allow the trusted service to perform only specific tasks within that account. In general, AWS manages all service-linked roles, which means that you typically can't alter the roles or the attached policies.

To make all of this possible, when you create an account in an organization or you accept an invitation to join your existing account to an organization, AWS Organizations provisions the member account with a service-linked role named `AWSServiceRoleForOrganizations`. Only the AWS Organizations service itself can assume this role. The role has permissions that allow AWS Organizations to create service-linked roles for other AWS services. This service-linked role is present in all organizations.

Although we don't recommend it, if your organization has only consolidated billing features enabled, the service-linked role named `AWSServiceRoleForOrganizations` is never used, and you can delete it. If you later want to enable all features in your organization, the role is required, and you must restore it. The following checks occur when you begin the process to enable all features:

- **For each member account that was *invited to join* the organization** – The account administrator receives a request to agree to enable all features. To successfully agree to the request, the administrator must have both `organizations:AcceptHandshake` *and* `iam:CreateServiceLinkedRole` permissions if the service-linked role (`AWSServiceRoleForOrganizations`) doesn't already exist. If the `AWSServiceRoleForOrganizations` role already exists, the administrator needs only the `organizations:AcceptHandshake` permission to agree to the request. When the administrator agrees to the request, AWS Organizations creates the service-linked role if it doesn't already exist.
- **For each member account that was *created* in the organization** – The account administrator receives a request to recreate the service-linked role. (The administrator of the member account doesn't receive a request to enable all features because the administrator of the management account (formerly known as the "master account") is considered the owner of the created member accounts.) AWS Organizations creates the service-linked role when the member account administrator agrees to the request. The administrator must have both `organizations:AcceptHandshake` *and* `iam:CreateServiceLinkedRole` permissions to successfully accept the handshake.

After you enable all features in your organization, you no longer can delete the `AWSServiceRoleForOrganizations` service-linked role from any account.

> **Important**
>
> AWS Organizations SCPs never affect service-linked roles. These roles are exempt from any SCP restrictions.

Using the AWSServiceRoleForDeclarativePoliciesEC2Report service-linked role

The`AWSServiceRoleForDeclarativePoliciesEC2Report` service-linked role is used by Organizations to describe account attribute states for member accounts to create Declarative Policies reports. The role's permissions are defined in the AWS managed policy: DeclarativePoliciesEC2Report.

AWS services that you can use with AWS Organizations

With AWS Organizations you can perform account management activities at scale by consolidating multiple AWS accounts into a single organization. Consolidating accounts simplifies how you use other AWS services. You can leverage the multi-account management services available in AWS Organizations with select AWS services to perform tasks on all accounts that are members of your organization.

The following table lists AWS services that you can use with AWS Organizations, and the benefit of using each service on an organization-wide level.

Trusted access – You can enable a compatible AWS service to perform operations across all of the AWS accounts in your organization. For more information, see Using AWS Organizations with other AWS services.

Delegated administrator for AWS services – A compatible AWS service can register an AWS member account in the organization as an administrator for the organization's accounts in that service. For more information, see Delegated administrator for AWS services that work with Organizations.

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
AWS Account Management Manage the details and metadata for all of the AWS accounts for your organization.	Manage account details, alternate contacts, and Regions for all of the AWS accounts in your organization.	 Yes Learn more	 Yes Learn more	
AWS Application Migration Service AWS Application Migration Service allows companies to lift-and-shift to AWS a large number of physical, virtual, or cloud servers without compatibility issues,	You can manage large-scale migrations across multiple accounts.	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
performance disruption, or long cutover windows.				
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.	 Yes Learn more	 No	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
AWS Audit Manager Automate the continuous collection of evidence to help you audit your use of cloud services.	Continuously audit your AWS use across multiple accounts in your organization to simplify how you assess risk and compliance.	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
AWS Backup Manage and monitor backups across all of the accounts in your organization.	You can configure and manage backup plans for your entire organization, or for groups of accounts in your organization units (OUs). You can centrally monitor backups for all of your accounts.	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
AWS Billing and Cost Management Provides an overview of your AWS cloud financial management data and to help you make faster and more informed decisions.	Allows split cost allocation data to retrieve AWS Organizations information, if applicable, and collect telemetry data for the split cost allocation data services that you have opted into. For more information, see What is AWS Billing	 Yes Learn more	 No	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
	and Cost Management? in the *Billing and Cost Management user guide*.			

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
AWS CloudFormation Stacksets Create, update, or delete stacks across multiple accounts and Regions with a single operation.	A user in the management account or a delegated administrator account can create a stack set with service-managed permissions that deploys stack instances to accounts in your organization.	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
AWS CloudTrail Enable governance, compliance, and operational and risk auditing of your account.	A user in a management account or delegated administrator account can create an organization trail or event data store that logs all events for all accounts in the organization.	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
Amazon CloudWatch Monitor your AWS resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables that you can measure for your resources and applications.	Integrating with Organizations has two benefits in CloudWatch. First, by integrating with Organizations, you can use CloudWatch to discover and understand the state of telemetry configuration for your AWS resources from a central view	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
	in the CloudWatch console. Second, when you can use Network Flow Monitor in CloudWatch to get visibility into network performance metrics, by integrating with Organizations, you can view network performance information for resources			

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
	in multiple accounts instead of just one account.			

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
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*.	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
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 and conformance packs across all AWS accounts in your organization. You can use a delegated	 Yes Learn more	 Yes Learn more: Config rules Conformance packs Multi-account multi-region data aggregation	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
	administrator account to aggregate resource configuration and compliance data from all member accounts of an organization in AWS Organizations. For more information, see Register a delegated administrator in the AWS Config Developer Guide.			

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
AWS Control Tower Set up and govern a secure, compliant, multi-account AWS environment.	You can set up a landing zone, a multi-account environment for all of your AWS resources. This environment includes an organization and organization entities. You can use this environment to enforce compliance regulations on	 Yes Learn more	 No	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
	all of your AWS accounts. For more information, see How AWS Control Tower and Manage Accounts Through AWS Organizations in the *AWS Control Tower User Guide*.			

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
AWS Cost Optimization Hub Gather cost recommendations across AWS optimization products.	You can easily identify, filter, and aggregate AWS cost optimization recommendations across your AWS Organizations member accounts and AWS Regions. For more information, see Cost Optimization Hub in the *Cost Optimization Hub*	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
	user guide.			
Amazon Detective Generate visualizations from your log data to analyze, investigate, and quickly identify the root cause of security findings or suspicious activities.	You can integrate Amazon Detective with AWS Organizations to ensure that your Detective behavior graph provides visibility into the activity for all of your organization accounts.	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
Amazon DevOps Guru Analyze operational data and application metrics and events to identify behaviors that deviate from normal operating patterns. Users are notified when DevOps Guru detects an operational issue or risk.	You can integrate with AWS Organizations to manage insights from all accounts across your entire organization. You delegate an administrator to view, sort, and filter insights from all accounts to obtain organization-wide health of all monitored	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
	applications.			
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.	 Yes Learn more	 No	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
Amazon EventBridge Monitor your AWS resources and the applications that you run on AWS in real time.	You can enable sharing of all Amazon EventBridge events, formerly Amazon CloudWatch Events, across all accounts in your organization. For more information, see Sending and receiving Amazon EventBridge events between AWS accounts in the	 No	 No	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
	Amazon EventBridge User Guide.			
Amazon Elastic Compute Cloud Amazon VPC IP Address Manager (IPAM) provides on-demand, scalable computing capacity in the AWS Cloud.	Enable the Organizations admin to create a report of what the existing configuration is for accounts across their organization when using the declarative policies feature.	 Yes Learn more	 No	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
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 the accounts in your organization.	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
Amazon GuardDuty GuardDuty is a continuous security monitoring service that analyzes and processes information from a variety of data sources. It uses threat intelligence feeds and machine learning to identify unexpected and potentially unauthorized and malicious activity within your AWS environment.	You can designate a member account to view and manage GuardDuty for all of the accounts in your organization. Adding member accounts automatically enables GuardDuty for those accounts in the selected AWS Region. You can also	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
	automate GuardDuty activation for new accounts added to your organization. For more information, see GuardDuty and Organizations in the *Amazon GuardDuty User Guide*.			

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
AWS Health Get visibility into events that might affect your resource performance or availability issues for AWS services.	You can aggregate AWS Health events across accounts in your organization.	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
AWS Identity and Access Management Securely control access to AWS resources.	You can use service last accessed data in IAM to help you better understand AWS activity across your organization. You can use this data to create and update service control policies (SCPs) that restrict access to only the AWS	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
	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.* IAM root access management lets you centrally manage root user credentials and perform privileged			

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
	tasks on member accounts..			
IAM Access Analyzer Analyze resource-based policies in your AWS environment to identify any policies that grant access to a principal outside of your zone of trust.	You can designate a member account to be an administrator for IAM Access Analyzer. For more information, see Enabling Access Analyzer in the *IAM User Guide*.	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
Amazon Inspector Automatically scan your AWS workloads for vulnerabilities to discover Amazon EC2 instances and container images that reside in Amazon ECR for software vulnerabilities and unintended network exposure.	Delegate an administrator to enable or disable scans for member accounts, view aggregated finding data from the entire organization, create and manage suppression rules. For more information, see Managing multiple accounts with AWS Organizations	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
	in the *Amazon Inspector User Guide*.			
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.	 Yes Learn more	 Learn more	Yes

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
Amazon Macie Discovers and classifies your business-critical content using machine learning to help you meet data security and privacy requirements. It continuously evaluates your content stored in Amazon S3 and notifies you of potential issues.	You can configure Amazon Macie for all of the accounts in your organization to get a consolidated view of all of your data in Amazon S3, across all accounts from a designated Macie administrator account. You can configure Macie to automatically	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
	protect resources in new accounts as your organization grows. You are alerted to remediate policy misconfigurations across S3 buckets throughout your organization.			

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
AWS Managed Services (AMS) Self-Service Reporting (SSR) Collects data from various native AWS services and provides access to reports on major AMS offerings. SSR provides the information that you can use to support operations, configuration management, asset management, security management, and compliance.	You can enable Aggregated SSR, a feature that allows customers to view consolidated Self-service reports across your organization through either your management account or a delegated administrator account.	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
AWS Marketplace A curated digital catalog that you can use to find, buy, deploy, and manage third-party software, data, and services that you need to build solutions and run your businesses.	You can share licenses for your AWS Marketplace subscriptions and purchases across the accounts in your organization.	 Yes Learn more	 No	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
AWS Marketplace Private Marketplace Provides you with a broad catalog of products available in AWS Marketplace, along with fine-grained control of those products.	Enables you to create multiple private marketplace experiences that are associated with your entire organization, one or more OUs, or one or more accounts in your organization, each with its own set of approved products. Your AWS	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
	administrators can also apply company branding to each private marketplace experience with your company or team's logo, messaging, and color scheme.			

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
AWS Marketplace procurement insights dashboard Enables you to view agreements and cost-analysis data for all your AWS Marketplace purchases across the AWS accounts in your organization.	AWS Marketplace procurement insights dashboard listens to organization changes, such as an account joining the organization, and aggregates data for their corresponding agreements to build their dashboards.	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
AWS Network Manager Enables you to centrally manage your AWS Cloud WAN core network and your AWS Transit Gateway network across AWS accounts, Regions, and on-premises locations.	You can centrally manage and monitor your global networks with transit gateways and their attached resources in multiple AWS accounts within your organization.	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
Amazon Q Developer Amazon Q Developer is a generative AI powered conversational assistant that can help you understand, build, extend, and operate AWS applications.	The paid subscription version of Amazon Q Developer requires Organizations integration.	 Yes Learn more	 No	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
AWS Resource Access Manager 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	 Yes Learn more	 No	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
	reservations, see the *Amazon EC2 User Guide* or the *Amazon EC2 User Guide*. For a list of shareable resources, see Shareable Resources in the *AWS RAM User Guide*.			

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
AWS Resource Explorer Explore your resources using an internet search engine-like experience.	Enable multi-account search.	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
AWS Security Hub View your security state in AWS and check your environment against security industry standards and best practices.	You can automatically enable Security Hub for all of your organization's accounts, including new accounts as they are added. This increases the coverage for Security Hub checks and findings, which provides a more	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
	accurate picture of your overall security posture.			
Amazon S3 Storage Lens Get visibility into your Amazon S3 storage usage and activity metrics with actionable recommendations to optimize storage.	Configure Amazon S3 Storage Lens to gain visibility into Amazon S3 storage usage and activity trends, and recommendations for all member accounts in your organization.	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
AWS Security Incident Response AWS security service that provides 24/7 live, human-assisted security incident support to help customers respond rapidly to cybersecurity incidents such as credential theft and ransomware attacks.	Security coverage for the entire organization.	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
Amazon Security Lake Amazon Security Lake centralizes security data from cloud, on-premises, and custom sources into a data lake that's stored in your account.	Create a data lake that collects logs and events across your accounts.	 Yes Learn more	 Yes Learn more	
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.	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
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.	 Yes Learn more	 No	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
AWS IAM Identity Center Provide single sign-on access for all of your accounts and cloud applications.	Users can sign in to the AWS access portal with their corporate credentials and access resources in their assigned management account or member accounts.	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
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. You can manage change templates, approvals and reporting for all member accounts in your organization from a	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
	delegated administrator account by using Systems Manager Change Manager.			
AWS User Notifications A central location for your AWS notifications.	You can configure and view notifications centrally across accounts in your organization.	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
Tag policies Use standardize tags across resources in your organization's accounts.	You can create tag policies to define tagging rules for specific resources and resource types and attach those policies to organization units and accounts to enforce those rules.	 Yes Learn more	 No	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
AWS Trusted Advisor Trusted Advisor inspects your AWS environment and makes recommendations when opportunities exist to save money, to improve system availability and performance, or to help close security gaps.	Run Trusted Advisor checks for all of the AWS accounts in your organization.	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
AWS Well-Architected Tool The AWS Well-Architected Tool helps you document the state of your workloads and compares them to the latest AWS architectural best practices.	Enables both AWS WA Tool and Organizations customers to simplify the process of sharing AWS WA Tool resources with other members of their organization.	 Yes Learn more	 No	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
Amazon VPC IP Address Manager (IPAM) IPAM is a VPC feature that makes it easier for you to plan, track, and monitor IP addresses for your AWS workloads.	Monitor IP address usage throughout your organization and share IP address pools across member accounts.	 Yes Learn more	 Yes Learn more	

AWS service	Benefits of using with AWS Organizations	Supports trusted access	Supports delegated administrator	
Amazon VPC Reachability Analyzer Reachability Analyzer is a configuration analysis tool that enables you to perform connectivity testing between a source resource and a destination resource in your virtual private clouds (VPCs).	Trace paths across accounts in your organizations.	 Yes Learn more	 Yes Learn more	

AWS Account Management and AWS Organizations

AWS Account Management helps you manage the account information and metadata for all of the AWS accounts in your organization. You can set, modify, or delete the alternate contact information for each of your organization's member accounts. For more information, see Using AWS Account Management in your organization in the *AWS Account Management User Guide*.

Use the following information to help you integrate AWS Account Management with AWS Organizations.

To enable trusted access with Account Management

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

Account Management requires trusted access to AWS Organizations before you can designate a member account to be the delegated administrator for this service for your organization.

You can only enable trusted access using the Organizations tools.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Account Management** in the list of services.
4. Choose **Enable trusted access**.
5. In the **Enable trusted access for AWS Account Management** dialog box, type **enable** to confirm, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Account Management that they can now enable that service to work with AWS Organizations from the service console .

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable AWS Account Management as a trusted service with Organizations.

```
$ aws organizations enable-aws-service-access \
    --service-principal account.amazonaws.com
```

This command produces no output when successful.

- AWS API: EnableAWSServiceAccess

To disable trusted access with Account Management

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

Only an administrator in the AWS Organizations management account can disable trusted access with AWS Account Management.

You can only disable trusted access using the Organizations tools.

You can disable trusted access by using either the AWS Organizations console, by running an Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Account Management** in the list of services.
4. Choose **Disable trusted access**.
5. In the **Disable trusted access for AWS Account Management** dialog box, type **disable** to confirm, and then choose **Disable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Account Management that they can now disable that service from working with AWS Organizations using the service console or tools .

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS Account Management as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal account.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for Account Management

When you designate a member account to be a delegated administrator for the organization, users and roles from the designated account can manage the AWS account metadata for other member accounts in the organization. If you don't enable a delegated admin account, then these tasks can be performed only by the organization's management account. This helps you to separate management of the organization from management of your account details.

Minimum permissions

Only a user or role in the Organizations management account can configure a member account as a delegated administrator for Account Management in the organization

For general instructions on how to configure a delegation policy, see Create a resource-based delegation policy with AWS Organizations.

AWS CLI, AWS API

If you want to configure a delegated administrator account using the AWS CLI or one of the AWS SDKs, you can use the following commands:

- AWS CLI:

```
$  aws organizations register-delegated-administrator \
    --account-id 123456789012 \
    --service-principal account.amazonaws.com
```

- AWS SDK: Call the Organizations `RegisterDelegatedAdministrator` operation and the member account's ID number and identify the account service principal `account.amazonaws.com` as parameters.

AWS Application Migration Service (Application Migration Service) and AWS Organizations

AWS Application Migration Service simplifies, expedites, and reduces the cost of migrating applications to AWS. By integrating with Organizations, you can use the global view feature to manage large-scale migrations across multiple accounts. For more information see Setting up your AWS Organizations in the *Application Migration Service user guide*.

Use the following information to help you integrate AWS Application Migration Service with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Application Migration Service to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Application Migration Service and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForApplicationMigrationService`

Service principals used by Application Migration Service

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Application Migration Service grant access to the following service principals:

- `mgn.amazonaws.com`

Enabling trusted access with Application Migration Service

When you enable trusted access with Application Migration Service you can use the global view feature, which allows you to manage large-scale migrations across multiple accounts. Global view provides visibility and the ability to perform specific actions on source servers, apps, and waves in different AWS accounts. For more information, see Setting up your AWS Organizations in the *AWS Application Migration Service user guide*.

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can enable trusted access using either the AWS Application Migration Service console or the AWS Organizations console.

> **Important**
>
> We strongly recommend that whenever possible, you use the AWS Application Migration Service console or tools to enable integration with Organizations. This lets AWS Application Migration Service perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by AWS Application Migration Service. For more information, see this note.
>
> If you enable trusted access by using the AWS Application Migration Service console or tools then you don't need to complete these steps.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

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

3. Choose **AWS Application Migration Service** in the list of services.
4. Choose **Enable trusted access**.
5. In the **Enable trusted access for AWS Application Migration Service** dialog box, type **enable** to confirm, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Application Migration Service that they can now enable that service to work with AWS Organizations from the service console .

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable AWS Application Migration Service as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal mgn.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access with Application Migration Service

Only an administrator in the Organizations management account can disable trusted access with Application Migration Service.

You can disable trusted access using either the AWS Application Migration Service or the AWS Organizations tools.

> **Important**
>
> We strongly recommend that whenever possible, you use the AWS Application Migration Service console or tools to disable integration with Organizations. This lets AWS Application Migration Service perform any clean up that it requires, such as deleting

resources or access roles that are no longer needed by the service. Proceed with these steps only if you can't disable integration using the tools provided by AWS Application Migration Service.
If you disable trusted access by using the AWS Application Migration Service console or tools then you don't need to complete these steps.

You can disable trusted access by using either the AWS Organizations console, by running an Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Application Migration Service** in the list of services.
4. Choose **Disable trusted access**.
5. In the **Disable trusted access for AWS Application Migration Service** dialog box, type **disable** to confirm, and then choose **Disable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Application Migration Service that they can now disable that service from working with AWS Organizations using the service console or tools .

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS Application Migration Service as a trusted service with Organizations.

```
$ aws organizations disable-aws-service-access \
    --service-principal mgn.amazonaws.com
```

This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for Application Migration Service

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for Application Migration Service that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of Application Migration Service. For more information see Setting up your AWS Organizations in the *Application Migration Service user guide*.

Minimum permissions

Only a user or role in the Organizations management account can configure a member account as a delegated administrator for Application Migration Service in the organization

AWS CLI, AWS API

If you want to configure a delegated administrator account using the AWS CLI or one of the AWS SDKs, you can use the following commands:

- AWS CLI:

```
$ aws organizations register-delegated-administrator \
    --account-id 123456789012 \
    --service-principal mgn.amazonaws.com
```

- AWS SDK: Call the Organizations `RegisterDelegatedAdministrator` operation and the member account's ID number and identify the account service `mgn.amazonaws.com` as parameters.

Disabling a delegated administrator for Application Migration Service

Only an administrator in the Organizations management account can remove a delegated administrator for Application Migration Service. You can remove the delegated administrator using the Organizations `DeregisterDelegatedAdministrator` CLI or SDK operation.

AWS Artifact and AWS Organizations

AWS Artifact is a service that allows you to download AWS security compliance reports such as ISO and PCI reports. Using AWS Artifact, a user in the organization's management account can automatically accept agreements on behalf of all member accounts in an organization, even as new reports and accounts are added. Member account users can view and download agreements. For more information, see Managing an agreement for multiple accounts in AWS Artifact in the *AWS Artifact User Guide*.

Use the following information to help you integrate AWS Artifact with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows AWS Artifact to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between AWS Artifact and Organizations, or if you remove the member account from the organization.

Although you can delete or modify this role if you remove the member account from the organization, we do not recommend it.

Modifying the role is discouraged because it can lead to security issues such as the cross-service confused deputy. To learn more about protection against confused deputy, see Cross-service deputy prevention in the *AWS Artifact User Guide*.

- `AWSServiceRoleForArtifact`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by AWS Artifact grant access to the following service principals:

- `artifact.amazonaws.com`

Enabling trusted access with AWS Artifact

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can only enable trusted access using the Organizations tools.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Artifact** in the list of services.
4. Choose **Enable trusted access**.
5. In the **Enable trusted access for AWS Artifact** dialog box, type **enable** to confirm, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Artifact that they can now enable that service to work with AWS Organizations from the service console .

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable AWS Artifact as a trusted service with Organizations.

```
$ aws organizations enable-aws-service-access \
```

```
    --service-principal artifact.amazonaws.com
```

This command produces no output when successful.

- AWS API: EnableAWSServiceAccess

Disabling trusted access with AWS Artifact

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

Only an administrator in the AWS Organizations management account can disable trusted access with AWS Artifact.

You can only disable trusted access using the Organizations tools.

AWS Artifact requires trusted access with AWS Organizations to work with organization agreements. If you disable trusted access using AWS Organizations while you are using AWS Artifact for organization agreements, it stops functioning because it cannot access the organization. Any organization agreements that you accept in AWS Artifact remain, but can't be accessed by AWS Artifact. The AWS Artifact role that AWS Artifact creates remains. If you then re-enable trusted access, AWS Artifact continues to operate as before, without the need for you to reconfigure the service.

A standalone account that is removed from an organization no longer has access to any organization agreements.

You can disable trusted access by using either the AWS Organizations console, by running an Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Artifact** in the list of services.

4. Choose **Disable trusted access**.
5. In the **Disable trusted access for AWS Artifact** dialog box, type **disable** to confirm, and then choose **Disable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Artifact that they can now disable that service from working with AWS Organizations using the service console or tools .

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS Artifact as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal artifact.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

AWS Audit Manager and AWS Organizations

AWS Audit Manager helps you continuously audit your AWS usage to simplify how you assess risk and compliance with regulations and industry standards. Audit Manager automates evidence collection to make it easier to assess if your policies, procedures, and activities are operating effectively. When it is time for an audit, Audit Manager helps you manage stakeholder reviews of your controls and helps you build audit-ready reports with much less manual effort.

When you integrate Audit Manager with AWS Organizations, you can gather evidence from a broader source by including multiple AWS accounts from your organization within the scope of your assessments.

For more information, see Enable AWS Organizations in the *Audit Manager User Guide*.

Use the following information to help you integrate AWS Audit Manager with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Audit Manager to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Audit Manager and Organizations, or if you remove the member account from the organization.

For more information about how Audit Manager uses this role, see Using service-linked roles in the *AWS Audit Manager Users Guide*.

- `AWSServiceRoleForAuditManager`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Audit Manager grant access to the following service principals:

- `auditmanager.amazonaws.com`

To enable trusted access with Audit Manager

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

Audit Manager requires trusted access to AWS Organizations before you can designate a member account to be the delegated administrator for your organization.

You can enable trusted access using either the AWS Audit Manager console or the AWS Organizations console.

> **Important**
>
> We strongly recommend that whenever possible, you use the AWS Audit Manager console or tools to enable integration with Organizations. This lets AWS Audit Manager perform

any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by AWS Audit Manager. For more information, see this note.
If you enable trusted access by using the AWS Audit Manager console or tools then you don't need to complete these steps.

To enable trusted access using the Audit Manager console

For instructions about enabling trusted access, see Setting Up in the *AWS Audit Manager User Guide*.

Note

If you configure a delegated administrator using the AWS Audit Manager console, then AWS Audit Manager automatically enables trusted access for you.

You can enable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To enable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable AWS Audit Manager as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal auditmanager.amazonaws.com
  ```

  This command produces no output when successful.

- AWS API: EnableAWSServiceAccess

To disable trusted access with Audit Manager

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

Only an administrator in the AWS Organizations management account can disable trusted access with AWS Audit Manager.

You can only disable trusted access using the Organizations tools.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS Audit Manager as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal auditmanager.amazonaws.com
  ```

  This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for Audit Manager

When you designate a member account to be a delegated administrator for the organization, users and roles from that account can perform administrative actions for Audit Manager that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of Audit Manager.

Minimum permissions

Only a user or role in the Organizations management account with the following permission can configure a member account as a delegated administrator for Audit Manager in the organization: `audit-manager:RegisterAccount`

For instruction about enabling a delegated administrator account for Audit Manager, see Setting Up in the *AWS Audit Manager User Guide*.

If you configure a delegated administrator using the AWS Audit Manager console, then Audit Manager automatically enables trusted access for you.

AWS CLI, AWS API

If you want to configure a delegated administrator account using the AWS CLI or one of the AWS SDKs, you can use the following commands:

- AWS CLI:

```
$  aws audit-manager register-account \
    --delegated-admin-account 123456789012
```

- AWS SDK: Call the `RegisterAccount` operation and provide `delegatedAdminAccount` as a parameter to delegate the administrator account.

AWS Backup and AWS Organizations

AWS Backup is a service that allows you to manage and monitor the AWS Backup jobs in your organization. Using AWS Backup, if you sign-in as a user in the organization's management account, you can enable organization-wide backup protection and monitoring. It helps you to achieve compliance by using backup policies to centrally apply AWS Backup plans to resources across all of the accounts in your organization. When you use both AWS Backup and AWS Organizations together, you can get the following benefits:

Protection

You can enable the backup policy type in your organization and then create backup policies to attach to the organization's root, OUs, or accounts. A backup policy combines an AWS Backup plan with the other details required to apply the plan automatically to your accounts.Policies that are directly attached to an account are merged with policies inherited from the organization's root and any parent OUs to create an *effective policy* that applies to the account. The policy includes the ID of an IAM role that has permissions to run AWS Backup on the resources in your accounts. AWS Backup uses the IAM role to perform the backup on your behalf as specified by the backup plan in the effective policy.

Monitoring

When you enable trusted access for AWS Backup in your organization, you can use the AWS Backup console to view details about the backup, restore, and copy jobs in any of the accounts in your organization. For more information, see Monitor your backup jobs in the *AWS Backup Developer Guide*.

For more information about AWS Backup, see the *AWS Backup Developer Guide*.

Use the following information to help you integrate AWS Backup with AWS Organizations.

Enabling trusted access with AWS Backup

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can enable trusted access using either the AWS Backup console or the AWS Organizations console.

Important

We strongly recommend that whenever possible, you use the AWS Backup console or tools to enable integration with Organizations. This lets AWS Backup perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by AWS Backup. For more information, see this note.
If you enable trusted access by using the AWS Backup console or tools then you don't need to complete these steps.

To enabled trusted access using AWS Backup, see Enabling backup in multiple AWS accounts in the *AWS Backup Developer Guide*.

Disabling trusted access with AWS Backup

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

AWS Backup requires trusted access with AWS Organizations to enable monitoring of backup, restore, and copy jobs across your organization's accounts. If you disable trusted access AWS Backup, you lose the ability to view jobs outside of the current account. The AWS Backup role that AWS Backup creates remains. If you later re-enable trusted access, AWS Backup continues to operate as before, without the need for you to reconfigure the service.

You can only disable trusted access using the Organizations tools.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS Backup as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal backup.amazonaws.com
  ```

  This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for AWS Backup

See Delegated administrator in the *AWS Backup Developer Guide*.

AWS Billing and Cost Management and AWS Organizations

AWS Billing and Cost Management provides a suite of features to help you set up your billing, retrieve and pay invoices, and analyze, organize, plan, and optimize your costs. When you use Billing and Cost Management with AWS Organizations you allow split cost allocation data to retrieve AWS Organizations information, if applicable, and collect telemetry data for the split cost allocation data services that you opted into.

Use the following information to help you integrate AWS Billing and Cost Management with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Billing and Cost Management to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Billing and Cost Management and Organizations, or if you remove the member account from the organization.

For more information, see Service-linked role permissions for Billing and Cost Management in the *Billing and Cost Management User Guide*.

- `AWSServiceRoleForSplitCostAllocationData`

Service principals used by Billing and Cost Management

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Billing and Cost Management grant access to the following service principals:

Billing and Cost Management uses the `billing-cost-management.amazonaws.com` service principal.

Enabling trusted access with Billing and Cost Management

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

With trusted access enabled via management account, customers can take advantage of the split cost allocation data feature under Billing and Cost Management. When customers enable split cost allocation data for Amazon Elastic Kubernetes Service with Amazon Managed Service for Prometheus, trusted access is invoked to create service-linked roles for all member accounts within the Organization. This allows split cost allocation data to collect telemetry data from customers' Amazon Managed Service for Prometheus work spaces and perform cost allocation based on those metrics.

You can only enable trusted access using the Organizations tools.

You can enable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To enable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable AWS Billing and Cost Management as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal  billing-cost-management.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

You can only disable trusted access using the Organizations tools.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS Billing and Cost Management as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal  billing-cost-management.amazonaws.com
  ```

  This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

AWS CloudFormation StackSets and AWS Organizations

AWS CloudFormation StackSets enables you to create, update, or delete stacks across multiple AWS accounts and AWS Regions with a single operation. StackSets integration with AWS Organizations enables you to create stack sets with service-managed permissions, using a service-linked role that has the relevant permission in each member account. This lets you deploy stack instances to member accounts in your organization. You don't have to create the necessary AWS Identity and Access Management roles; StackSets creates the IAM role in each member account on your behalf.

You can also choose to enable automatic deployments to accounts that are added to your organization in the future. With auto deployment enabled, roles and deployment of associated stack set instances are automatically added to all accounts added in the future to that OU.

With trusted access between StackSets and Organizations enabled, the management account has permissions to create and manage stack sets for your organization. The management account can register up to five member accounts as delegated administrators. With trusted access enabled, delegated administrators also have permissions to create and manage stack sets for your organization. Stack sets with service-managed permissions are created in the management account, including stack sets that are created by delegated administrators.

> **Important**
>
> Delegated administrators have full permissions to deploy to accounts in your organization. The management account cannot limit delegated administrator permissions to deploy to specific OUs or to perform specific stack set operations.

For more information about integrating StackSets with Organizations, see Working with AWS CloudFormation StackSets in the *AWS CloudFormation User Guide*.

Use the following information to help you integrate AWS CloudFormation StackSets with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows AWS CloudFormation Stacksets to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between AWS CloudFormation Stacksets and Organizations, or if you remove the member account from the organization.

- Management account: `AWSServiceRoleForCloudFormationStackSetsOrgAdmin`

To create the service-linked role `AWSServiceRoleForCloudFormationStackSetsOrgMember` for the member accounts in your organization, you need to create a stack set in the management account first. This creates a stack set instance, which then creates the role in the member accounts.

- Member accounts: `AWSServiceRoleForCloudFormationStackSetsOrgMember`

For more details about creating stack sets, see Working with AWS CloudFormation StackSets in the *AWS CloudFormation User Guide*.

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by AWS CloudFormation Stacksets grant access to the following service principals:

- Management account: `stacksets.cloudformation.amazonaws.com`

  You can modify or delete this role only if you disabled trusted access between StackSets and Organizations.
- Member accounts: `member.org.stacksets.cloudformation.amazonaws.com`

  You can modify or delete this role from an account only if you first disable trusted access between StackSets and Organizations, or if you first remove the account from the target organization or organizational unit (OU).

Enabling trusted access with AWS CloudFormation Stacksets

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

Only an administrator in the Organizations management account has permissions to enable trusted access with another AWS service. You can enable trusted access using either the AWS CloudFormation console or the Organizations console.

You can only enable trusted access using AWS CloudFormation StackSets.

To enable trusted access using the AWS CloudFormation Stacksets console, see Enable Trusted Access with AWS Organizations in the AWS CloudFormation User Guide.

Disabling trusted access with AWS CloudFormation Stacksets

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

Only an administrator in an Organizations management account has permissions to disable trusted access with another AWS service. You can disable trusted access only by using the Organizations console. If you disable trusted access with Organizations while you are using StackSets, all previously created stack instances are retained. However, stack sets deployed using the service-linked role's permissions can no longer perform deployments to accounts managed by Organizations.

You can disable trusted access using either the AWS CloudFormation console or the Organizations console.

> **Important**
>
> If you disable trusted access programmatically (e.g with AWS CLI or with an API), be aware that this will remove the permission. It is better to disable trusted access with the AWS CloudFormation console.

You can disable trusted access by using either the AWS Organizations console, by running an Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS CloudFormation StackSets** in the list of services.
4. Choose **Disable trusted access**.
5. In the **Disable trusted access for AWS CloudFormation StackSets** dialog box, type **disable** to confirm, and then choose **Disable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS CloudFormation StackSets that they can now disable that service from working with AWS Organizations using the service console or tools .

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS CloudFormation StackSets as a trusted service with Organizations.

```
$ aws organizations disable-aws-service-access \
    --service-principal stacksets.cloudformation.amazonaws.com
```

This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for AWS CloudFormation Stacksets

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for AWS CloudFormation Stacksets that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of AWS CloudFormation Stacksets.

For instructions on how to designate a member account as a delegated administrator of AWS CloudFormation Stacksets in the organization, see Register a delegated administrator in the *AWS CloudFormation User Guide*.

AWS CloudTrail and AWS Organizations

AWS CloudTrail is an AWS service that helps you enable governance, compliance, and operational and risk auditing of your AWS account. Using AWS CloudTrail, a user in a management account can create an organization trail that logs all events for all AWS accounts in that organization. Organization trails are automatically applied to all member accounts in the organization. Member accounts can see the organization trail, but can't modify or delete it. By default, member accounts don't have access to the log files for the organization trail in the Amazon S3 bucket. This helps you uniformly apply and enforce your event logging strategy across the accounts in your organization.

For more information, see Creating a Trail for an Organization in the *AWS CloudTrail User Guide*.

Use the following information to help you integrate AWS CloudTrail with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows CloudTrail to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between CloudTrail and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForCloudTrail`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by CloudTrail grant access to the following service principals:

- `cloudtrail.amazonaws.com`

Enabling trusted access with CloudTrail

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

If you enable trusted access by creating a trail from the AWS CloudTrail console, trusted access is configured automatically for you (recommended). You can also enable trusted access using the AWS Organizations console. You must sign in with your AWS Organizations management account to create an organization trail.

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*.

> **Important**
>
> We strongly recommend that whenever possible, you use the AWS CloudTrail console or tools to enable integration with Organizations.

You can enable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To enable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable AWS CloudTrail as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal cloudtrail.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access with CloudTrail

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

AWS CloudTrail requires trusted access with AWS Organizations to work with organization trails and organization event data stores. If you disable trusted access using AWS Organizations while you're using AWS CloudTrail, all organization trails for member accounts are deleted because CloudTrail can't access the organization. All management account organization trails and organization event data stores are converted to account-level trails and event data stores. The `AWSServiceRoleForCloudTrail` role created for integration between CloudTrail and AWS Organizations stays in the account. If you re-enable trusted access, CloudTrail will not take action on existing trails and event data stores. The management account must update any account-level trails and event data stores to apply them to the organization.

To convert an account-level trail or event data store to an organization trail or organization event data store, do the following:

- From the CloudTrail console, update the trail or event data store and choose the **Enable for all accounts in my organization** option.
- From the AWS CLI, do the following:
  - To update a trail, run the update-trail command and include the `--is-organization-trail` parameter.
  - To update an event data store, run the update-event-data-store command and include the `--organization-enabled` parameter.

Only an administrator in the AWS Organizations management account can disable trusted access with AWS CloudTrail. You can disable trusted access only with the Organizations tools, using either the AWS Organizations console, running an Organizations AWS CLI command, or calling an Organizations API operation in one of the AWS SDKs.

You can disable trusted access by using either the AWS Organizations console, by running an Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS CloudTrail** in the list of services.
4. Choose **Disable trusted access**.
5. In the **Disable trusted access for AWS CloudTrail** dialog box, type **disable** to confirm, and then choose **Disable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS CloudTrail that they can now disable that service from working with AWS Organizations using the service console or tools .

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS CloudTrail as a trusted service with Organizations.

```
$ aws organizations disable-aws-service-access \
```

```
    --service-principal cloudtrail.amazonaws.com
```

This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for CloudTrail

When you use CloudTrail with Organizations, you can register any account within the organization to act as a CloudTrail delegated administrator to manage the organization's trails and event data stores on behalf of the organization. A delegated administrator is a member account in an organization that can perform the same administrative tasks in CloudTrail as the management account.

Minimum permissions

Only an administrator in the Organizations management account can register a delegated administrator for CloudTrail.

You can register a delegated administrator account using the CloudTrail console, or by using the Organizations `RegisterDelegatedAdministrator` CLI or SDK operation. To register a delegated administrator using the CloudTrail console, see Add a CloudTrail delegated administrator.

Disabling a delegated administrator for CloudTrail

Only an administrator in the Organizations management account can remove a delegated administrator for CloudTrail. You can remove the delegated administrator using either the CloudTrail console, or by using the Organizations `DeregisterDelegatedAdministrator` CLI or SDK operation. For information on how to remove a delegated administrator using the CloudTrail console, see Remove a CloudTrail delegated administrator .

Amazon CloudWatch and AWS Organizations

You can use AWS Organizations for Amazon CloudWatch for the following use cases:

- Discover and understand the state of telemetry configuration for your AWS resources from a central view in the CloudWatch console. This simplifies the process of auditing your telemetry

collection configurations for multiple resource types across your AWS organization or account. You must turn on trusted access to use telemetry config across your organization.

For more information, see Auditing CloudWatch telemetry configurations in the *Amazon CloudWatch User Guide*.

- Work with multiple accounts in Network Flow Monitor, a feature of Amazon CloudWatch Network Monitoring. Network Flow Monitor provides near real-time visibility into network performance for traffic between Amazon EC2 instances. After you turn on trusted access to integrate with Organizations, you can create a monitor to visualize network performance details across multiple accounts.

  For more information, see Initialize Network Flow Monitor for multi-account monitoring in the *Amazon CloudWatch User Guide*.

Use the following information to help you integrate Amazon CloudWatch with AWS Organizations.

Service-linked roles created when you enable integration

Create the following service-linked role in your organization's management account. The service-linked role is automatically created in member accounts when you enable trusted access. This role allows CloudWatch to perform supported operations within your organization's accounts in your organization. You can delete or modify this role only if you disable trusted access between CloudWatch and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForObservabilityAdmin`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by CloudWatch grant access to the following service principals:

- `observabilityadmin.amazonaws.com`
- `networkflowmonitor.amazonaws.com`
- `topology.networkflowmonitor.amazonaws.com`

Enabling trusted access with CloudWatch

For information about the permissions that you need to turn on trusted access, see Permissions required to enable trusted access.

You can enable trusted access using either the Amazon CloudWatch console or the AWS Organizations console.

> **Important**
>
> We strongly recommend that whenever possible, you use the Amazon CloudWatch console or tools to enable integration with Organizations. This lets Amazon CloudWatch perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by Amazon CloudWatch. For more information, see this note.
> If you enable trusted access by using the Amazon CloudWatch console or tools then you don't need to complete these steps.

To turn on trusted access using the CloudWatch console

See Turning on CloudWatch telemetry auditing in the *Amazon CloudWatch User Guide*.

When you turn on trusted access in CloudWatch, you enable telemetry auditing and you can work with multiple accounts in Network Flow Monitor.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **Amazon CloudWatch** in the list of services.
4. Choose **Enable trusted access**.

5. In the **Enable trusted access for Amazon CloudWatch** dialog box, type **enable** to confirm, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of Amazon CloudWatch that they can now enable that service to work with AWS Organizations from the service console .

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable Amazon CloudWatch as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal observabilityadmin.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Turn off trusted access with CloudWatch

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

You can disable trusted access using either the Amazon CloudWatch or the AWS Organizations tools.

> **Important**
>
> We strongly recommend that whenever possible, you use the Amazon CloudWatch console or tools to disable integration with Organizations. This lets Amazon CloudWatch perform any clean up that it requires, such as deleting resources or access roles that are no longer needed by the service. Proceed with these steps only if you can't disable integration using the tools provided by Amazon CloudWatch.

If you disable trusted access by using the Amazon CloudWatch console or tools then you don't need to complete these steps.

To turn off trusted access using the CloudWatch console

See Turning off CloudWatch telemetry auditing in the *Amazon CloudWatch User Guide*

When you turn off trusted access in CloudWatch, telemetry auditing is no longer active and you can no longer work with multiple accounts in Network Flow Monitor.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable Amazon CloudWatch as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal observabilityadmin.amazonaws.com
  ```

  This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Registering a delegated administrator account for CloudWatch

When you register a member account as a delegated administrator account for the organization, users and roles from that account can perform administrative actions for CloudWatch that otherwise can be performed only by users or roles signed in with the organization's management account. Using a delegated administrator account helps you to separate management of the organization from management of features in CloudWatch.

Minimum permissions

Only an administrator in the Organizations management account can register a member account as a delegated administrator account for CloudWatch in the organization.

You can register a delegated administrator account using the CloudWatch console, or by using the Organizations `RegisterDelegatedAdministrator` API operation with the AWS Command Line Interface or an SDK.

For information on how to register a delegated administrator account by using the CloudWatch console, see Turning on CloudWatch telemetry auditing in the *Amazon CloudWatch User Guide*.

When you register a delegated administrator account in CloudWatch, you can use the account for management operations with telemetry auditing and with Network Flow Monitor.

Deregister a delegated administrator for CloudWatch

Minimum permissions

Only an administrator signed in with the Organizations management account can deregister a delegated administrator account for CloudWatch in the organization.

You can deregister the delegated administrator account by using either the CloudWatch console, or by using the Organizations `DeregisterDelegatedAdministrator` API operation with the AWS Command Line Interface or an SDK. For more information, see Deregistering a delegated administrator account in the *Amazon CloudWatch User Guide*.

When you deregister a delegated administrator account in CloudWatch, you can no longer use the account for management operations with telemetry auditing and with Network Flow Monitor.

AWS Compute Optimizer and AWS Organizations

AWS Compute Optimizer is a service that analyzes the configuration and utilization metrics of your AWS resources. Resource examples include Amazon Elastic Compute Cloud (Amazon EC2) instances and Auto Scaling groups. Compute Optimizer reports whether your resources are optimal and generates optimization recommendations to reduce the cost and improve the performance of your

workloads. For more information about Compute Optimizer, see the AWS Compute Optimizer User Guide.

Use the following information to help you integrate AWS Compute Optimizer with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Compute Optimizer to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Compute Optimizer and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForComputeOptimizer`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Compute Optimizer grant access to the following service principals:

- `compute-optimizer.amazonaws.com`

Enabling trusted access with Compute Optimizer

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can enable trusted access using either the AWS Compute Optimizer console or the AWS Organizations console.

> **Important**
>
> We strongly recommend that whenever possible, you use the AWS Compute Optimizer console or tools to enable integration with Organizations. This lets AWS Compute Optimizer perform any configuration that it requires, such as creating resources needed by

the service. Proceed with these steps only if you can't enable integration using the tools provided by AWS Compute Optimizer. For more information, see this note.
If you enable trusted access by using the AWS Compute Optimizer console or tools then you don't need to complete these steps.

To enable trusted access using the Compute Optimizer console

You must sign in to the Compute Optimizer console using your organization's management account. Opt-in on behalf of your organization by following the instructions at Opting in your Account in the *AWS Compute Optimizer User Guide*.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Compute Optimizer** in the list of services.
4. Choose **Enable trusted access**.
5. In the **Enable trusted access for AWS Compute Optimizer** dialog box, type **enable** to confirm, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Compute Optimizer that they can now enable that service to work with AWS Organizations from the service console .

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

Run the following command to enable AWS Compute Optimizer as a trusted service with Organizations.

```
$ aws organizations enable-aws-service-access \
    --service-principal compute-optimizer.amazonaws.com
```

This command produces no output when successful.

- AWS API: EnableAWSServiceAccess

Disabling trusted access with Compute Optimizer

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

Only an administrator in the AWS Organizations management account can disable trusted access with AWS Compute Optimizer.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS Compute Optimizer as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal compute-optimizer.amazonaws.com
  ```

  This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for Compute Optimizer

When you designate a member account to be a delegated administrator for the organization, users and roles from the designated account can manage the AWS account metadata for other member accounts in the organization. If you don't enable a delegated admin account, then these tasks can be performed only by the organization's management account. This helps you to separate management of the organization from management of your account details.

Minimum permissions

Only a user or role in the Organizations management account can configure a member account as a delegated administrator for Compute Optimizer in the organization

For instructions about enabling a delegated administrator account for Compute Optimizer, see https://docs.aws.amazon.com/compute-optimizer/latest/ug/delegate-administrator-account.html in the *AWS Compute Optimizer User Guide*.

AWS CLI, AWS API

If you want to configure a delegated administrator account using the AWS CLI or one of the AWS SDKs, you can use the following commands:

- AWS CLI:

```
$  aws organizations register-delegated-administrator \
    --account-id 123456789012 \
    --service-principal compute-optimizer.amazonaws.com
```

- AWS SDK: Call the Organizations `RegisterDelegatedAdministrator` operation and the member account's ID number and identify the account service principal `account.amazonaws.com` as parameters.

Disabling a delegated administrator for Compute Optimizer

Only an administrator in the organization management account can configure a delegated administrator for Compute Optimizer.

To disable the delegated admin Compute Optimizer account using the Compute Optimizer console, see https://docs.aws.amazon.com/compute-optimizer/latest/ug/delegate-administrator-account.html in the *AWS Compute Optimizer User Guide*.

To remove a delegated administrator using the AWS AWS CLI, see deregister-delegated-administrator in the *AWS AWS CLI Command Reference*.

AWS Config and AWS Organizations

Multi-account, multi-region data aggregation in AWS Config enables you to aggregate AWS Config data from multiple accounts and AWS Regions into a single account. Multi-account, multi-region data aggregation is useful for central IT administrators to monitor compliance for multiple AWS accounts in the enterprise. An aggregator is a resource type in AWS Config that collects AWS Config data from multiple source accounts and Regions. Create an aggregator in the Region where you want to see the aggregated AWS Config data. While creating an aggregator, you can choose to add either individual account IDs or your organization. For more information about AWS Config, see the AWS Config Developer Guide.

You can also use AWS Config APIs to manage AWS Config rules across all AWS accounts in your organization. For more information, see Enabling AWS Config Rules Across All Accounts in Your Organization in the *AWS Config Developer Guide*.

Use the following information to help you integrate AWS Config with AWS Organizations.

Service-linked roles

The following service-linked role allows AWS Config to perform supported operations within the accounts in your organization.

- `AWSServiceRoleForConfig`

Learn more about creating this role in Permissions for the IAM Role Assigned to AWS Config in the *AWS Config Developer Guide*

Learn more about how AWS Config uses service-linked roles in Using Service-Linked Roles for AWS Config in the *AWS Config Developer Guide*

You can delete or modify this role only if you disable trusted access between AWS Config and Organizations, or if you remove the member account from the organization.

Enabling trusted access with AWS Config

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can enable trusted access using either the AWS Config console or the AWS Organizations console.

> **Important**
>
> We strongly recommend that whenever possible, you use the AWS Config console or tools to enable integration with Organizations. This lets AWS Config perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by AWS Config. For more information, see this note.
> If you enable trusted access by using the AWS Config console or tools then you don't need to complete these steps.

To enable trusted access using the AWS Config console

To enable trusted access to AWS Organizations using AWS Config, create a multi-account aggregator and add the organization. For information on how to configure a multi-account aggregator, see Creating Aggregators in the *AWS Config Developer Guide*.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Config** in the list of services.
4. Choose **Enable trusted access**.

5. In the **Enable trusted access for AWS Config** dialog box, type **enable** to confirm, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Config that they can now enable that service to work with AWS Organizations from the service console .

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable AWS Config as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal config.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access with AWS Config

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

You can only disable trusted access using the Organizations tools.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

Run the following command to disable AWS Config as a trusted service with Organizations.

```
$ aws organizations disable-aws-service-access \
    --service-principal config.amazonaws.com
```

This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

AWS Cost Optimization Hub and AWS Organizations

AWS Cost Optimization Hub is an AWS Billing and Cost Management feature that helps you consolidate and prioritize cost optimization recommendations across your AWS accounts and AWS Regions, so that you can get the most out of your AWS spend. When you use Cost Optimization Hub with AWS Organizations you can easily identify, filter, and aggregate AWS cost optimization recommendations across your Organizations member accounts and AWS Regions.

For more information, see Cost Optimization Hub in the *AWS Cost Management User Guide*.

Use the following information to help you integrate AWS Cost Optimization Hub with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Cost Optimization Hub to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Cost Optimization Hub and Organizations, or if you remove the member account from the organization.

For more information, see Service-linked role permissions for Cost Optimization Hub in the *AWS Cost Management User Guide*.

- `AWSServiceRoleForCostOptimizationHub`

Service principals used by Cost Optimization Hub

Cost Optimization Hub uses the `cost-optimization-hub.bcm.amazonaws.com` service principal.

Enabling trusted access with Cost Optimization Hub

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

When you opt in using your organization's management account and include all member accounts within the organization, trusted access for Cost Optimization Hub is automatically enabled in your organization account.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Cost Optimization Hub** in the list of services.
4. Choose **Enable trusted access**.
5. In the **Enable trusted access for AWS Cost Optimization Hub** dialog box, type **enable** to confirm, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Cost Optimization Hub that they can now enable that service to work with AWS Organizations from the service console .

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable AWS Cost Optimization Hub as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal cost-optimization-hub.bcm.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

You can only disable trusted access using the Organizations tools.

> **Important**
>
> If you disable Cost Optimization Hub trusted access after you opt in, Cost Optimization Hub denies access to recommendations for your organization's member accounts. Moreover, the member accounts within the organization aren't opted in to Cost Optimization Hub. Learn more in Cost Optimization Hub and Organizations trusted access in the *AWS Cost Management User Guide*.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS Cost Optimization Hub as a trusted service with Organizations.

```
$ aws organizations disable-aws-service-access \
    --service-principal cost-optimization-hub.bcm.amazonaws.com
```

This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for Cost Optimization Hub

When you designate a member account to be a delegated administrator for the organization, the designated account can retrieve Cost Optimization Hub recommendations for all accounts under your organization and manage Cost Optimization Hub preferences, giving you greater flexibility to centrally identify resource optimization opportunities.

Minimum permissions

Only a user or role in the Organizations management account with the following permission can configure a member account as a delegated administrator for Cost Optimization Hub in the organization:

For instructions about enabling a delegated administrator account for Cost Optimization Hub, see Delegate an administrator account in the *AWS Cost Management User Guide*.

Disabling a delegated administrator for Cost Optimization Hub

Only an administrator in the Organizations management account can remove a delegated administrator for Cost Optimization Hub.

To disable the delegated admin Cost Optimization Hub account using the Cost Optimization Hub console, see Delegate an administrator account in the *AWS Cost Management User Guide*.

To remove a delegated administrator using the AWS CLI, see `deregister-delegated-administrator` in the *AWS Config CLI Reference*.

AWS Control Tower and AWS Organizations

AWS Control Tower offers a straightforward way to set up and govern an AWS multi-account environment, following prescriptive best practices. AWS Control Tower orchestration extends the

capabilities of AWS Organizations. AWS Control Tower applies preventive and detective controls (guardrails) to help keep your organizations and accounts from divergence from best practices (drift).

AWS Control Tower orchestration extends the capabilities of AWS Organizations.

For more information, see the *AWS Control Tower user guide*.

Use the following information to help you integrate AWS Control Tower with AWS Organizations.

Roles needed for integration

The `AWSControlTowerExecution` role must be present in all enrolled accounts. It allows AWS Control Tower to manage your individual accounts and report information about them to your Audit and Log Archive accounts.

To learn more about roles used by AWS Control Tower, see How AWS Control Tower works with roles to create and manage accounts and Using Identity-Based Policies (IAM Policies) for AWS Control Tower.

Service principals used by AWS Control Tower

AWS Control Tower uses the `controltower.amazonaws.com` service principal.

Enabling trusted access with AWS Control Tower

AWS Control Tower uses trusted access to detect drift for preventive controls, and to track account and OU changes that cause drift.

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can only enable trusted access using the Organizations tools.

To enable trusted access from the Organizations console, choose **`Enable access`** next to **AWS Control Tower**.

You can enable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To enable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable AWS Control Tower as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal controltower.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access with AWS Control Tower

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

You can only disable trusted access using the Organizations tools.

> **Important**
>
> Disabling AWS Control Tower's trusted access causes drift in your AWS Control Tower Landing Zone. The only way to fix the drift is to use AWS Control Tower's Landing Zone repair. Re-enabling trusted access in Organizations does not fix the drift. Learn more about drift in the *AWS Control Tower user guide*.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS Control Tower as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal controltower.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

Amazon Detective and AWS Organizations

Amazon Detective uses your log data to generate visualizations that allow you to analyze, investigate, and identify the root cause of security findings or suspicious activity.

Using AWS Organizations allows you to ensure that your Detective behavior graph provides visibility into the activity for all of your organization accounts.

When you grant trusted access to Detective, the Detective service can react automatically to changes in the organization membership. The delegated administrator can enable any organization account as a member account in the behavior graph. Detective also can automatically enable new organization accounts as member accounts. Organization accounts cannot disassociate themselves from the behavior graph.

For more information, see Using Amazon Detective in your organization in the *Amazon Detective Administration Guide*.

Use the following information to help you integrate Amazon Detective with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Detective to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Detective and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForDetective`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Detective grant access to the following service principals:

- `detective.amazonaws.com`

To enable trusted access with Detective

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

> **Note**
>
> When you designate a delegated administrator for Amazon Detective, Detective automatically enables trusted access for Detective for your organization.
> Detective requires trusted access to AWS Organizations before you can designate a member account to be the delegated administrator for this service for your organization.

You can only enable trusted access using the Organizations tools.

You can enable trusted access by using the AWS Organizations console.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **Amazon Detective** in the list of services.
4. Choose **Enable trusted access**.

5. In the **Enable trusted access for Amazon Detective** dialog box, type **enable** to confirm it, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of Amazon Detective that they can now enable that service to work with AWS Organizations from the service console .

To disable trusted access with Detective

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

Only an administrator in the AWS Organizations management account can disable trusted access with Amazon Detective.

You can only disable trusted access using the Organizations tools.

You can disable trusted access by using the AWS Organizations console.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **Amazon Detective** in the list of services.
4. Choose **Disable trusted access**.
5. In the **Disable trusted access for Amazon Detective** dialog box, type **disable** to confirm, and then choose **Disable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of Amazon Detective that they can now disable that service from working with AWS Organizations using tthe service console or tools;.

Enabling a delegated administrator account for Detective

The delegated administrator account for Detective is the administrator account for a Detective behavior graph. The delegated administrator determines which organization accounts to enable

and disable as member accounts in that behavior graph. The delegated administrator can configure Detective to automatically enable new organization accounts as member accounts as they are added to the organization. For information on how a delegated administrator manages organization accounts, see Managing organization accounts as member accounts in the *Amazon Detective Administration Guide*.

Only an administrator in the organization management account can configure a delegated administrator for Detective.

You can specify a delegated administrator account from the Detective console or API, or by using the Organizations CLI or SDK operation.

> **Minimum permissions**
>
> Only a user or role in the Organizations management account can configure a member account as a delegated administrator for Detective in the organization

To configure a delegated administrator using the Detective console or API, see Designating a Detective administrator account for an organization in the *Amazon Detective Administration Guide*.

AWS CLI, AWS API

If you want to configure a delegated administrator account using the AWS CLI or one of the AWS SDKs, you can use the following commands:

- AWS CLI:

```
$ aws organizations register-delegated-administrator \
    --account-id 123456789012 \
    --service-principal detective.amazonaws.com
```

- AWS SDK: Call the Organizations `RegisterDelegatedAdministrator` operation and the member account's ID number and identify the account service principal `account.amazonaws.com` as parameters.

Disabling a delegated administrator for Detective

You can remove the delegated administrator using either the Detective console or API, or by using the Organizations `DeregisterDelegatedAdministrator` CLI or SDK operation. For

information on how to remove a delegated administrator using the Detective console or API, or the Organizations API, see Designating a Detective administrator account for an organization in the *Amazon Detective Administration Guide*.

Amazon DevOps Guru and AWS Organizations

Amazon DevOps Guru analyzes operational data and application metrics and events to identify behaviors that deviate from normal operating patterns. Users are notified when DevOps Guru detects an operational issue or risk.

Using DevOps Guru enables multi-account support with AWS Organizations, so you can designate a member account to manage insights across your entire organization. This delegated administrator can then view, sort, and filter insights from all accounts within your organization to develop a holistic view of the health of all monitored applications within your organization without the need for any additional customization.

For more information, see Monitor accounts across your organization in the *Amazon DevOps Guru User Guide*.

Use the following information to help you integrate Amazon DevOps Guru with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows DevOps Guru to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between DevOps Guru and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForDevOpsGuru`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by DevOps Guru grant access to the following service principals:

- `devops-guru.amazonaws.com`

For more information, see Using service-linked roles for DevOps Guru in the *Amazon DevOps Guru User Guide*.

To enable trusted access with DevOps Guru

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

> **Note**
>
> When you designate a delegated administrator for Amazon DevOps Guru, DevOps Guru automatically enables trusted access for DevOps Guru for your organization.
> DevOps Guru requires trusted access to AWS Organizations before you can designate a member account to be the delegated administrator for this service for your organization.

> **Important**
>
> We strongly recommend that whenever possible, you use the Amazon DevOps Guru console or tools to enable integration with Organizations. This lets Amazon DevOps Guru perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by Amazon DevOps Guru. For more information, see this note.

You can enable trusted access by using either the AWS Organizations console or the DevOps Guru console.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for **Amazon DevOps Guru**, choose the service's name, and then choose **Enable trusted access**.
3. In the confirmation dialog box, enable **Show the option to enable trusted access**, enter **`enable`** in the box, and then choose **Enable trusted access**.

4. If you are the administrator of only AWS Organizations, tell the administrator of Amazon DevOps Guru that they can now enable that service using its console to work with AWS Organizations.

DevOps Guru console

To enable trusted service access using the DevOps Guru console

1. Sign in as administrator in the management account and open DevOps Guru console: Amazon DevOps Guru console
2. Choose **Enable trusted access**.

To disable trusted access with DevOps Guru

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

Only an administrator in the AWS Organizations management account can disable trusted access with Amazon DevOps Guru.

You can only disable trusted access using the Organizations tools.

You can disable trusted access by using the AWS Organizations console.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **Amazon DevOps Guru** in the list of services.
4. Choose **Disable trusted access**.
5. In the **Disable trusted access for Amazon DevOps Guru** dialog box, type **disable** to confirm, and then choose **Disable trusted access**.

6. If you are the administrator of only AWS Organizations, tell the administrator of Amazon DevOps Guru that they can now disable that service from working with AWS Organizations using tthe service console or tools;.

Enabling a delegated administrator account for DevOps Guru

The delegated administrator account for DevOps Guru can see the insights data from all the member accounts which are onboarded to DevOps Guru from the organization. For information on how a delegated administrator manages organization accounts, see Monitor accounts across your organization in the *Amazon DevOps Guru User Guide*.

Only an administrator in the organization management account can configure a delegated administrator for DevOps Guru.

You can specify a delegated administrator account from the DevOps Guru console, or by using the Organizations `RegisterDelegatedAdministrator` CLI or SDK operation.

> **Minimum permissions**
>
> Only a user or role in the Organizations management account can configure a member account as a delegated administrator for DevOps Guru in the organization

DevOps Guru console

To configure a delegated administrator in the DevOps Guru console

1. Sign in as administrator in the management account and open DevOps Guru console: Amazon DevOps Guru console
2. Choose **Register delegated administrator**. You can choose either Management account or any member account as the delegated admin.

AWS CLI, AWS API

If you want to configure a delegated administrator account using the AWS CLI or one of the AWS SDKs, you can use the following commands:

- AWS CLI:

```
$ aws organizations register-delegated-administrator \
    --account-id 123456789012 \
    --service-principal devops-guru.amazonaws.com
```

- AWS SDK: Call the Organizations `RegisterDelegatedAdministrator` operation and the member account's ID number and identify the account service principal `account.amazonaws.com` as parameters.

Disabling a delegated administrator for DevOps Guru

You can remove the delegated administrator using either the DevOps Guru console, or by using the Organizations `DeregisterDelegatedAdministrator` CLI or SDK operation. For information on how to remove a delegated administrator using the DevOps Guru console, see Monitor accounts across your organization in the *Amazon DevOps Guru User Guide*.

AWS Directory Service and AWS Organizations

AWS Directory Service for Microsoft Active Directory, or AWS Managed Microsoft AD, lets you run Microsoft Active Directory (AD) as a managed service. AWS Directory Service makes it easy to set up and run directories in the AWS Cloud or connect your AWS resources with an existing on-premises Microsoft Active Directory. AWS Managed Microsoft AD also integrates tightly with AWS Organizations to allow seamless directory sharing across multiple AWS accounts and any VPC in a Region. For more information, see the AWS Directory Service Administration Guide.

To share an AWS Directory Service across an organization, the organization must have **All features** enabled, and the directory must be in the organization management account.

Use the following information to help you integrate AWS Directory Service with AWS Organizations.

Enabling trusted access with AWS Directory Service

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can enable trusted access using either the AWS Directory Service console or the AWS Organizations console.

Important

We strongly recommend that whenever possible, you use the AWS Directory Service console or tools to enable integration with Organizations. This lets AWS Directory Service perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by AWS Directory Service. For more information, see this note.
If you enable trusted access by using the AWS Directory Service console or tools then you don't need to complete these steps.

To enable trusted access using the AWS Directory Service console

To share a directory, which automatically enables trusted access, see Share Your Directory in the *AWS Directory Service Administration Guide*. For step-by-step instructions, see Tutorial: Sharing Your AWS Managed Microsoft AD Directory.

You can enable trusted access by using the AWS Organizations console.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Directory Service** in the list of services.
4. Choose **Enable trusted access**.
5. In the **Enable trusted access for AWS Directory Service** dialog box, type **enable** to confirm it, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Directory Service that they can now enable that service to work with AWS Organizations from the service console .

Disabling trusted access with AWS Directory Service

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

If you disable trusted access using AWS Organizations while you are using AWS Directory Service, all previously shared directories continue to operate as normal. However, you can no longer share new directories within the organization until you enable trusted access again.

You can only disable trusted access using the Organizations tools.

You can disable trusted access by using the AWS Organizations console.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Directory Service** in the list of services.
4. Choose **Disable trusted access**.
5. In the **Disable trusted access for AWS Directory Service** dialog box, type **disable** to confirm, and then choose **Disable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Directory Service that they can now disable that service from working with AWS Organizations using tthe service console or tools;.

Amazon Elastic Compute Cloud and AWS Organizations

Amazon Elastic Compute Cloud provides on-demand, scalable computing capacity in the AWS Cloud. When you use Amazon EC2 with Organizations; you enable the Organizations admin to create a report of what the existing configuration is for accounts across their organization after using Amazon EC2's Declarative Policies feature.

Use the following information to help you integrate Amazon Elastic Compute Cloud with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Amazon EC2 to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Amazon EC2 and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForDeclarativePoliciesEC2Report`

Service principals used by Amazon EC2

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Amazon EC2 grant access to the following service principals:

- `ec2.amazonaws.com`

Enabling trusted access with Amazon EC2

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

To enable the Organizations admin to create a report of what the existing configuration is for accounts across their organization, you must enable trusted access.

You can only enable trusted access using the Organizations tools.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

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

3. Choose **Amazon Elastic Compute Cloud** in the list of services.
4. Choose **Enable trusted access**.
5. In the **Enable trusted access for Amazon Elastic Compute Cloud** dialog box, type **enable** to confirm, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of Amazon Elastic Compute Cloud that they can now enable that service to work with AWS Organizations from the service console .

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable Amazon Elastic Compute Cloud as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal ec2.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

You can only disable trusted access using the Organizations tools.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable Amazon Elastic Compute Cloud as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal ec2.amazonaws.com
  ```

  This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

AWS Firewall Manager and AWS Organizations

AWS Firewall Manager is a security management service you use to centrally configure and manage firewall rules and other protections across the AWS accounts and applications in your organization. Using Firewall Manager, you can roll out AWS WAF rules, create AWS Shield Advanced protections, configure and audit Amazon Virtual Private Cloud (Amazon VPC) security groups, and deploy AWS Network Firewalls. Use Firewall Manager to set up your protections just once and have them automatically applied across all accounts and resources within your organization, even as new resources and accounts are added. For more information about AWS Firewall Manager, see the *AWS Firewall Manager Developer Guide*.

Use the following information to help you integrate AWS Firewall Manager with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Firewall Manager to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Firewall Manager and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForFMS`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Firewall Manager grant access to the following service principals:

- `fms.amazonaws.com`

Enabling trusted access with Firewall Manager

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can enable trusted access using either the AWS Firewall Manager console or the AWS Organizations console.

> **Important**
>
> We strongly recommend that whenever possible, you use the AWS Firewall Manager console or tools to enable integration with Organizations. This lets AWS Firewall Manager perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by AWS Firewall Manager. For more information, see this note.
> If you enable trusted access by using the AWS Firewall Manager console or tools then you don't need to complete these steps.

You must sign in with your AWS Organizations management account and configure an account within the organization as the AWS Firewall Manager administrator account. For more information, see Set the AWS Firewall Manager Administrator Account in the *AWS Firewall Manager Developer Guide*.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Firewall Manager** in the list of services.
4. Choose **Enable trusted access**.
5. In the **Enable trusted access for AWS Firewall Manager** dialog box, type **enable** to confirm, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Firewall Manager that they can now enable that service to work with AWS Organizations from the service console .

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable AWS Firewall Manager as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal fms.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access with Firewall Manager

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

You can disable trusted access using either the AWS Firewall Manager or the AWS Organizations tools.

> **Important**
>
> We strongly recommend that whenever possible, you use the AWS Firewall Manager console or tools to disable integration with Organizations. This lets AWS Firewall Manager perform any clean up that it requires, such as deleting resources or access roles that are no longer needed by the service. Proceed with these steps only if you can't disable integration using the tools provided by AWS Firewall Manager.
> If you disable trusted access by using the AWS Firewall Manager console or tools then you don't need to complete these steps.

To disable trusted access using the Firewall Manager console

You can change or revoke the AWS Firewall Manager administrator account by following the instructions in Designating a Different Account as the AWS Firewall Manager Administrator Account in the *AWS Firewall Manager Developer Guide*.

If you revoke the administrator account, you must sign in to the AWS Organizations management account and set a new administrator account for AWS Firewall Manager.

You can disable trusted access by using either the AWS Organizations console, by running an Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Firewall Manager** in the list of services.
4. Choose **Disable trusted access**.
5. In the **Disable trusted access for AWS Firewall Manager** dialog box, type **disable** to confirm, and then choose **Disable trusted access**.

6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Firewall Manager that they can now disable that service from working with AWS Organizations using the service console or tools .

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS Firewall Manager as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal fms.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for Firewall Manager

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for Firewall Manager that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of Firewall Manager.

> **Minimum permissions**
>
> Only a user or role in the Organizations management account can configure a member account as a delegated administrator for Firewall Manager in the organization.

For instructions on how to designate a member account as the Firewall Manager administrator for the organization, see Set the AWS Firewall Manager Administrator Account in the *AWS Firewall Manager Developer Guide*.

Amazon GuardDuty and AWS Organizations

Amazon GuardDuty is a continuous security monitoring service that analyzes and processes a variety data sources, using threat intelligence feeds and machine learning to identify unexpected and potentially unauthorized and malicious activity within your AWS environment. This can include issues like escalations of privileges, uses of exposed credentials, communication with malicious IP addresses, URLs, or domains, or presence of malware on your Amazon Elastic Compute Cloud instances and container workloads.

You can help simplify management of GuardDuty by using Organizations to manage GuardDuty across all of the accounts in your organization.

For more information, see Managing GuardDuty accounts with AWS Organizations in the *Amazon GuardDuty User Guide*

Use the following information to help you integrate Amazon GuardDuty with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked roles are automatically created in your organization's management account when you enable trusted access. These roles allow GuardDuty to perform supported operations within your organization's accounts in your organization. You can delete a role only if you disable trusted access between GuardDuty and Organizations, or if you remove the member account from the organization.

- The `AWSServiceRoleForAmazonGuardDuty` service-linked role is automatically created in accounts that have integrated GuardDuty with Organizations. For more information, see Managing GuardDuty accounts with Organizations in the *Amazon GuardDuty User Guide*
- The `AmazonGuardDutyMalwareProtectionServiceRolePolicy` service-linked role is automatically created in accounts that have enabled GuardDuty Malware Protection. For more information, see Service-linked role permissions for GuardDuty Malware Protection in the *Amazon GuardDuty User Guide*

Service principals used by the service-linked roles

- `guardduty.amazonaws.com`, used by the `AWSServiceRoleForAmazonGuardDuty` service-linked role.

- `malware-protection.guardduty.amazonaws.com`, used by the `AmazonGuardDutyMalwareProtectionServiceRolePolicy` service-linked role.

Enabling trusted access with GuardDuty

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can only enable trusted access using Amazon GuardDuty.

Amazon GuardDuty requires trusted access to AWS Organizations before you can designate a member account to be the GuardDuty administrator for your organization. If you configure a delegated administrator using the GuardDuty console, then GuardDuty automatically enables trusted access for you.

However, if you want to configure a delegated administrator account using the AWS CLI or one of the AWS SDKs, then you must explicitly call the EnableAWSServiceAccess operation and provide the service principal as a parameter. Then you can call EnableOrganizationAdminAccount to delegate the GuardDuty administrator account.

Disabling trusted access with GuardDuty

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

You can only disable trusted access using the Organizations tools.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable Amazon GuardDuty as a trusted service with Organizations.

```
$ aws organizations disable-aws-service-access \
    --service-principal guardduty.amazonaws.com
```

This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for GuardDuty

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for GuardDuty that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of GuardDuty.

> **Minimum permissions**
>
> For information about the permissions required to designate a member account as a delegated administrator, see Permissions required to designate a delegated administrator in the *Amazon GuardDuty User Guide*

To designate a member account as a delegated administrator for GuardDuty

See Designate a delegated administrator and add member accounts (console) and Designate a delegated administrator and add member accounts (API)

AWS Health and AWS Organizations

AWS Health provides ongoing visibility into your resource performance and the availability of your AWS services and accounts. AWS Health delivers events when your AWS resources and services are impacted by an issue or will be affected by upcoming changes. After you enable organizational view, a user in the organization's management account can aggregate AWS Health events across all accounts in the organization. Organizational view only shows AWS Health events delivered after the feature is enabled and retains them for 90 days.

You can enable organizational view by using the AWS Health console, the AWS Command Line Interface (AWS CLI), or the AWS Health API.

For more information, see Aggregating AWS Health events in the *AWS Health User Guide*.

Use the following information to help you integrate AWS Health with AWS Organizations.

Service-linked roles for integration

The `AWSServiceRoleForHealth_Organizations` service-linked role allows AWS Health to perform supported operations within your organization's accounts in your organization.

This role is created automatically in your organization's management account when you enable trusted access by calling the EnableHealthServiceAccessForOrganization API operation. Otherwise, create the role using the AWS Health console, API, or CLI, as described in Creating a service-linked role in the IAM User Guide.

You can delete or modify this role only if you disable trusted access between AWS Health and Organizations, or if you remove the member account from the organization.

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by AWS Health grant access to the following service principals:

- `health.amazonaws.com`

Enabling trusted access with AWS Health

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

When you the enable organizational view feature for AWS Health, trusted access is also enabled for you automatically.

You can enable trusted access using either the AWS Health console or the AWS Organizations console.

> **Important**
>
> We strongly recommend that whenever possible, you use the AWS Health console or tools to enable integration with Organizations. This lets AWS Health perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps

only if you can't enable integration using the tools provided by AWS Health. For more information, see this note.
If you enable trusted access by using the AWS Health console or tools then you don't need to complete these steps.

To enable trusted access using the AWS Health console

You can enable trusted access by using AWS Health and one of the following options:

- Use the AWS Health console. For more information, see Organizational view (console) in the *AWS Health User Guide*.
- Use the AWS CLI. For more information, see Organizational view (CLI) in the *AWS Health User Guide*.
- Call the EnableHealthServiceAccessForOrganization API operation.

You can enable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To enable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable AWS Health as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal health.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access with AWS Health

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

After you disable the organizational view feature, AWS Health stops aggregating events for all other accounts in your organization. This also disables trusted access for you automatically.

You can disable trusted access using either the AWS Health or the AWS Organizations tools.

> **Important**
>
> We strongly recommend that whenever possible, you use the AWS Health console or tools to disable integration with Organizations. This lets AWS Health perform any clean up that it requires, such as deleting resources or access roles that are no longer needed by the service. Proceed with these steps only if you can't disable integration using the tools provided by AWS Health.
> If you disable trusted access by using the AWS Health console or tools then you don't need to complete these steps.

To disable trusted access using the AWS Health console

You can disable trusted access with one of the following options:

- Use the AWS Health console. For more information, see Disabling organizational view (console) in the *AWS Health User Guide*.
- Use the AWS CLI. For more information, see Disabling organizational view (CLI) in the *AWS Health User Guide*.
- Call the DisableHealthServiceAccessForOrganization API operation.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS Health as a trusted service with Organizations.

```
$ aws organizations disable-aws-service-access \
```

```
    --service-principal health.amazonaws.com
```

This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for AWS Health

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for AWS Health that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of AWS Health.

To designate a member account as a delegated administrator for AWS Health

See Register a delegated administrator for your organizational view

To remove a delegated administrator for AWS Health

See Remove a delegated administrator from your organizational view

AWS Identity and Access Management and AWS Organizations

AWS Identity and Access Management is a web service for securely controlling access to AWS services.

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) 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*.

IAM lets you centrally manage root user credentials and perform privileged tasks on member accounts. After you enable root access management, which enables trusted access for IAM in AWS Organizations, you can centrally secure the root user credentials of member accounts. Member accounts can't sign in to their root user or perform password recovery for their root user. The management account or a delegated administrator account for IAM can also perform some privileged tasks on member accounts using short-term root access. Short-term privileged sessions give you temporary credentials that you can scope to take privileged actions on a member account in your organization.

For more information, see Centrally manage root access for member accounts in the *IAM User Guide*.

Use the following information to help you integrate AWS Identity and Access Management with AWS Organizations.

Enabling trusted access with IAM

When you enable root access management, trusted access is enabled for IAM in AWS Organizations.

Disabling trusted access with IAM

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

Only an administrator in the AWS Organizations management account can disable trusted access with AWS Identity and Access Management.

You can only disable trusted access using the Organizations tools.

You can disable trusted access by using either the AWS Organizations console, by running an Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Identity and Access Management** in the list of services.
4. Choose **Disable trusted access**.
5. In the **Disable trusted access for AWS Identity and Access Management** dialog box, type **disable** to confirm, and then choose **Disable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Identity and Access Management that they can now disable that service from working with AWS Organizations using the service console or tools .

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS Identity and Access Management as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal iam.amazonaws.com
  ```

  This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for IAM

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform privileged tasks on member accounts that otherwise can be performed only by users or roles in the organization's management account. For more information, see Perform a privileged task on an Organizations member account in the IAM User Guide.

Only an administrator in the organization management account can configure a delegated administrator for IAM.

You can specify a delegated administrator account from the IAM console or API, or by using the Organizations CLI or SDK operation.

Disabling a delegated administrator for IAM

Only an administrator in either the Organizations management account or the IAM delegated admin account can remove a delegated administrator account from the organization. You can disable delegated administration using the Organizations `DeregisterDelegatedAdministrator` CLI or SDK operation.

Amazon Inspector and AWS Organizations

Amazon Inspector is an automated vulnerability management service that continually scans Amazon EC2 and container workloads for software vulnerabilities and unintended network exposure.

Using Amazon Inspector you can manage multiple accounts that are associated through AWS Organizations by simply delegating an administrator account for Amazon Inspector. The delegated administrator manages Amazon Inspector for the organization and is granted special permissions to perform tasks on behalf of your organization such as:

- Enable or disable scans for member accounts
- View aggregated finding data from the entire organization
- Create and manage suppression rules

For more information, see Managing multiple accounts with AWS Organizations in the *Amazon Inspector User Guide*.

Use the following information to help you integrate Amazon Inspector with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Amazon Inspector to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Amazon Inspector and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForAmazonInspector2`

For more information, see Using service-linked roles with Amazon Inspector in the *Amazon Inspector User Guide*.

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Amazon Inspector grant access to the following service principals:

- `inspector2.amazonaws.com`

To enable trusted access with Amazon Inspector

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

Amazon Inspector requires trusted access to AWS Organizations before you can designate a member account to be the delegated administrator for this service for your organization.

When you designate a delegated administrator for Amazon Inspector, Amazon Inspector automatically enables trusted access for Amazon Inspector for your organization.

However, if you want to configure a delegated administrator account using the AWS CLI or one of the AWS SDKs, then you must explicitly call the `EnableAWSServiceAccess` operation and provide the service principal as a parameter. Then you can call `EnableDelegatedAdminAccount` to delegate the Inspector administrator account.

You can enable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To enable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable Amazon Inspector as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal inspector2.amazonaws.com
  ```

  This command produces no output when successful.

- AWS API: EnableAWSServiceAccess

> **Note**
>
> If you are using the `EnableAWSServiceAccess` API, you need to also call `EnableDelegatedAdminAccount` to delegate the Inspector administrator account.

To disable trusted access with Amazon Inspector

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

Only an administrator in the AWS Organizations management account can disable trusted access with Amazon Inspector.

You can only disable trusted access using the Organizations tools.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable Amazon Inspector as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal inspector2.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for Amazon Inspector

With Amazon Inspector you can manage multiple accounts in an organization using a delegated administrator with AWS Organizations service.

The AWS Organizations management account designates an account within the organization as the delegated administrator account for Amazon Inspector. The delegated administrator manages Amazon Inspector for the organization and is granted special permissions to perform tasks on behalf of your organization such as: enable or disable scans for member accounts, view aggregated finding data from the entire organization, and create and manage suppression rules

For information on how a delegated administrator manages organization accounts, see Understanding the relationship between administrator and member accounts in the *Amazon Inspector User Guide*.

Only an administrator in the organization management account can configure a delegated administrator for Amazon Inspector.

You can specify a delegated administrator account from the Amazon Inspector console or API, or by using the Organizations CLI or SDK operation.

Minimum permissions

Only a user or role in the Organizations management account can configure a member account as a delegated administrator for Amazon Inspector in the organization

To configure a delegated administrator using the Amazon Inspector console, see Step 1: Enable Amazon Inspector - Multi-account environment in the *Amazon Inspector User Guide*.

Note

You must call `inspector2:enableDelegatedAdminAccount` in each region where you use Amazon Inspector.

AWS CLI, AWS API

If you want to configure a delegated administrator account using the AWS CLI or one of the AWS SDKs, you can use the following commands:

- AWS CLI:

```
$ aws organizations register-delegated-administrator \
```

```
    --account-id 123456789012 \
    --service-principal inspector2.amazonaws.com
```

- AWS SDK: Call the Organizations `RegisterDelegatedAdministrator` operation and the member account's ID number and identify the account service principal `account.amazonaws.com` as parameters.

Disabling a delegated administrator for Amazon Inspector

Only an administrator in the AWS Organizations management account can remove a delegated administrator account from the organization.

You can remove the delegated administrator using either the Amazon Inspector console or API, or by using the Organizations `DeregisterDelegatedAdministrator` CLI or SDK operation. To remove a delegated administrator using the Amazon Inspector console, see Removing a delegated administrator in the *Amazon Inspector User Guide*.

AWS License Manager and AWS Organizations

AWS License Manager streamlines the process of bringing software vendor licenses to the cloud. As you build out cloud infrastructure on AWS, you can save costs by using bring-your-own-license (BYOL) opportunities—that is, by repurposing your existing license inventory for use with cloud resources. With rule-based controls on the consumption of licenses, administrators can set hard or soft limits on new and existing cloud deployments, stopping noncompliant server usage before it happens.

For more information about License Manager, see the License Manager User Guide.

By linking License Manager with AWS Organizations, you can:

- Enable cross-account discovery of computing resources throughout your organization.
- View and manage commercial Linux subscriptions that you own and run on AWS. For more information see Linux subscriptions in AWS License Manager.

Use the following information to help you integrate AWS License Manager with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked roles are automatically created in your organization's management account when you enable trusted access. These roles allow License Manager to perform supported operations within your organization's accounts in your organization.

You can delete or modify roles only if you disable trusted access between License Manager and Organizations, or if you remove the member account from the organization.

- `AWSLicenseManagerMasterAccountRole`
- `AWSLicenseManagerMemberAccountRole`
- `AWSServiceRoleForAWSLicenseManagerRole`
- `AWSServiceRoleForAWSLicenseManagerLinuxSubscriptionsService`

For more information, see License Manager–Management account role, License Manager–Member account role, and License Manager–Linux subscriptions role.

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by License Manager grant access to the following service principals:

- `license-manager.amazonaws.com`
- `license-manager.member-account.amazonaws.com`
- `license-manager-linux-subscriptions.amazonaws.com`

Enabling trusted access with License Manager

You can only enable trusted access using AWS License Manager.

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

To enable trusted access with License Manager

You must sign in to the License Manager console using your AWS Organizations management account and associate it with your License Manager account. For more information, see Settings in AWS License Manager.

Disabling trusted access with License Manager

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

You can only disable trusted access using the Organizations tools.

You can disable trusted access by running an Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  You can run the following command to disable AWS License Manager as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal license-manager.amazonaws.com
  ```

  This command produces no output when successful.

  To disable trusted access for Linux subscriptions use:

  ```
  $  aws organizations disable-aws-service-access \
      --service-principal license-manager-linux-subscriptions.amazonaws.com
  ```

- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for License Manager

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for License Manager that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of License Manager.

To delegate a member account as an administrator for License Manager, follow the steps at Register a delegated administrator in the *License Manager User Guide*.

AWS Managed Services (AMS) Self-Service Reporting (SSR) and AWS Organizations

AWS Managed Services (AMS) Self-Service Reporting (SSR) collects data from various native AWS services and provides access to reports on major AMS offerings. SSR provides the information that you can use to support operations, configuration management, asset management, security management, and compliance.

After you integrate with AWS Organizations, you can enable Aggregated self-service reporting (SSR). This is an AMS feature that allows Advanced and Accelerate customers to view their existing Self-service reports aggregated at the organization level, cross-account. This gives you visibility into key operational metrics such as patch compliance, backup coverage, and incidents across all AMS-managed accounts within AWS Organizations.

Use the following information to help you integrate AWS Managed Services (AMS) Self-Service Reporting (SSR) with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows AMS to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between AMS and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForManagedServices_SelfServiceReporting`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by AMS grant access to the following service principals:

- `selfservicereporting.managedservices.amazonaws.com`

Enabling trusted access with AMS

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can enable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To enable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable AWS Managed Services (AMS) Self-Service Reporting (SSR) as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal selfservicereporting.managedservices.amazonaws.com
  ```

  This command produces no output when successful.

- AWS API: EnableAWSServiceAccess

Disabling trusted access with AMS

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

You can only disable trusted access using the Organizations tools.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS Managed Services (AMS) Self-Service Reporting (SSR) as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal selfservicereporting.managedservices.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for AMS

Delegated administrator accounts can view AMS reports (such as patch and backup) across all the accounts in a single aggregated view in the AMS console.

You can add a delegated administrator using either the AMS console or API, or by using the Organizations `RegisterDelegatedAdministrator` CLI or SDK operation.

Disabling a delegated administrator for AMS

Only an administrator in the organization management account can configure a delegated administrator for AMS.

You can remove the delegated administrator using either the AMS console or API, or by using the Organizations `DeregisterDelegatedAdministrator` CLI or SDK operation.

Amazon Macie and AWS Organizations

Amazon Macie is a fully managed data security and data privacy service that uses machine learning and pattern matching to discover, monitor, and help you protect your sensitive data in Amazon Simple Storage Service (Amazon S3). Macie automates the discovery of sensitive data, such as personally identifiable information (PII) and intellectual property, to provide you with a better understanding of the data that your organization stores in Amazon S3.

For more information, see Managing Amazon Macie accounts with AWS Organizations in the *Amazon Macie User Guide*.

Use the following information to help you integrate Amazon Macie with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created for your organization's delegated Macie administrator account when you enable trusted access. This role allows Macie to perform supported operations for the accounts in your organization.

You can delete this role only if you disable trusted access between Macie and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleRorAmazonMacie`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Macie grant access to the following service principals:

- `macie.amazonaws.com`

Enabling trusted access with Macie

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can enable trusted access using either the Amazon Macie console or the AWS Organizations console.

> **Important**
>
> We strongly recommend that whenever possible, you use the Amazon Macie console or tools to enable integration with Organizations. This lets Amazon Macie perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by Amazon Macie. For more information, see this note.
> If you enable trusted access by using the Amazon Macie console or tools then you don't need to complete these steps.

To enable trusted access using the Macie console

Amazon Macie requires trusted access to AWS Organizations to designate a member account to be the Macie administrator for your organization. If you configure a delegated administrator using the Macie management console, then Macie automatically enables trusted access for you.

For more information, see Integrating and configuring an organization in Amazon Macie in the *Amazon Macie User Guide*.

You can enable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To enable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable Amazon Macie as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal macie.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Enabling a delegated administrator account for Macie

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for Macie that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of Macie.

Minimum permissions

Only a user or role in the Organizations management account with the following permissions can configure a member account as a delegated administrator for Macie in the organization:

- `organizations:EnableAWSServiceAccess`

- `macie:EnableOrganizationAdminAccount`

To designate a member account as a delegated administrator for Macie

Amazon Macie requires trusted access to AWS Organizations to designate a member account to be the Macie administrator for your organization. If you configure a delegated administrator using the Macie management console, then Macie automatically enables trusted access for you.

For more information, see https://docs.aws.amazon.com/macie/latest/user/macie-organizations.html#register-delegated-admin

AWS Marketplace and AWS Organizations

AWS Marketplace is a curated digital catalog that you can use to find, buy, deploy, and manage third-party software, data, and services that you need to build solutions and run your businesses.

AWS Marketplace creates and manages licenses using AWS License Manager for your purchases in AWS Marketplace. When you share (grant access to) your licenses with other accounts in your organization, AWS Marketplace creates and manages new licenses for those accounts.

For more information, see Service-linked roles for AWS Marketplace in the *AWS Marketplace Buyer Guide*.

Use the following information to help you integrate AWS Marketplace with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows AWS Marketplace to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between AWS Marketplace and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForMarketplaceLicenseManagement`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by AWS Marketplace grant access to the following service principals:

- `license-management.marketplace.amazonaws.com`

Enabling trusted access with AWS Marketplace

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can enable trusted access using either the AWS Marketplace console or the AWS Organizations console.

> **Important**
>
> We strongly recommend that whenever possible, you use the AWS Marketplace console or tools to enable integration with Organizations. This lets AWS Marketplace perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by AWS Marketplace. For more information, see this note.
> If you enable trusted access by using the AWS Marketplace console or tools then you don't need to complete these steps.

To enable trusted access using the AWS Marketplace console

See Creating a service-linked role for AWS Marketplace in the *AWS Marketplace Buyer Guide*.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Marketplace** in the list of services.
4. Choose **Enable trusted access**.
5. In the **Enable trusted access for AWS Marketplace** dialog box, type **enable** to confirm, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Marketplace that they can now enable that service to work with AWS Organizations from the service console .

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable AWS Marketplace as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal license-management.marketplace.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access with AWS Marketplace

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can only enable trusted access using the Organizations tools.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS Marketplace as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal license-management.marketplace.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

AWS Marketplace Private Marketplace and AWS Organizations

AWS Marketplace is a curated digital catalog that you can use to find, buy, deploy, and manage third-party software, data, and services that you need to build solutions and run your businesses. A private marketplace provides you with a broad catalog of products available in AWS Marketplace, along with fine-grained control of those products.

AWS Marketplace Private Marketplace enables you to create multiple private marketplace experiences that are associated with your entire organization, one or more OUs, or one or more accounts in your organization, each with its own set of approved products. Your AWS administrators can also apply company branding to each private marketplace experience with your company or team's logo, messaging, and color scheme.

For more information, see Using roles to configure Private Marketplace in AWS Marketplace in the *AWS Marketplace Buyer Guide*.

Use the following information to help you integrate AWS Marketplace Private Marketplace with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access using the AWS Marketplace Private Marketplace console. This role allows Private Marketplace to perform supported operations within your organization's accounts in your organization. You can delete or modify this role only if you disable trusted access between AWS Marketplace Private Marketplace and Organizations and disassociate all private marketplace experiences in your organization.

If you enable trusted access directly from the Organizations console, CLI or SDK, the service-linked role is not created automatically.

- `AWSServiceRoleForPrivateMarketplaceAdmin`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Private Marketplace grant access to the following service principals:

- `private-marketplace.marketplace.amazonaws.com`

Enabling trusted access with Private Marketplace

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can enable trusted access using either the AWS Marketplace Private Marketplace console or the AWS Organizations console.

Important

We strongly recommend that whenever possible, you use the AWS Marketplace Private Marketplace console or tools to enable integration with Organizations. This lets AWS Marketplace Private Marketplace perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by AWS Marketplace Private Marketplace. For more information, see this note.

If you enable trusted access by using the AWS Marketplace Private Marketplace console or tools then you don't need to complete these steps.

To enable trusted access using the Private Marketplace console

See Getting started with Private Marketplace in the *AWS Marketplace Buyer Guide*.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Marketplace Private Marketplace** in the list of services.
4. Choose **Enable trusted access**.
5. In the **Enable trusted access for AWS Marketplace Private Marketplace** dialog box, type **enable** to confirm, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Marketplace Private Marketplace that they can now enable that service to work with AWS Organizations from the service console .

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable AWS Marketplace Private Marketplace as a trusted service with Organizations.

```
$ aws organizations enable-aws-service-access \
```

```
    --service-principal private-marketplace.marketplace.amazonaws.com
```

This command produces no output when successful.

- AWS API: EnableAWSServiceAccess

Disabling trusted access with Private Marketplace

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can only disable trusted access using the Organizations tools.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS Marketplace Private Marketplace as a trusted service with Organizations.

```
$ aws organizations disable-aws-service-access \
    --service-principal private-marketplace.marketplace.amazonaws.com
```

  This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for Private Marketplace

The management account administrator can delegate Private Marketplace administrative permissions to a designated member account known as delegated administrator. To register an account as a delegated administrator for the private marketplace, the management account administrator must ensure that trusted access and the service-linked role are enabled, choose **Register a new administrator**, provide the 12-digit AWS account number, and choose **Submit**.

Management accounts and delegated administrator accounts can perform Private Marketplace administrative tasks, such as creating experiences, updating branding settings, associating or disassociating audiences, adding or removing products, and approving or declining pending requests.

To configure a delegated administrator using the Private Marketplace console, see Creating and managing a private marketplace in the *AWS Marketplace Buyer Guide*.

You can also configure a delegated administrator by using the Organizations `RegisterDelegatedAdministrator` API. For more information, see RegisterDelegatedAdministrator in the *Organizations Command Reference*.

Disabling a delegated administrator for Private Marketplace

Only an administrator in the organization management account can configure a delegated administrator for Private Marketplace.

You can remove the delegated administrator using either the Private Marketplace console or API, or by using the Organizations `DeregisterDelegatedAdministrator` CLI or SDK operation.

To disable the delegated admin Private Marketplace account using the Private Marketplace console, see Creating and managing a private marketplace in the *AWS Marketplace Buyer Guide*

AWS Marketplace procurement insights dashboard and AWS Organizations

You use the AWS Marketplace procurement insights dashboard to view agreements and cost-analysis data for all of the AWS accounts in your organization. When integrated with Organizations, AWS Marketplace procurement insights dashboard listens to organization changes, such as an account joining the organization, and aggregates data for their corresponding agreements to build their dashboards.

For more information, see Procurement insights in the *AWS Marketplace Buyer Guide*.

Use the following information to help you integrate AWS Marketplace procurement insights dashboard with AWS Organizations.

Service-linked roles and managed policies created when you enable integration

When you activate the AWS Marketplace procurement insights dashboard dashboard the `AWSServiceRoleForProcurementInsightsPolicy` service-linked role and the `AWSServiceRoleForProcurementInsightsPolicy` AWS managed policy are created.

Enabling trusted access with AWS Marketplace procurement insights

Enabling trusted access grants the AWS Marketplace procurement insights dashboard the ability to integrate with the customer's Organizations service. AWS Marketplace procurement insights dashboard listens to organization changes, such as an account joining the organization, and aggregates data for their corresponding agreements to build their dashboards.

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can enable trusted access using either the AWS Marketplace procurement insights dashboard console or the AWS Organizations console.

> **Important**
>
> We strongly recommend that whenever possible, you use the AWS Marketplace procurement insights dashboard console or tools to enable integration with Organizations. This lets AWS Marketplace procurement insights dashboard perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by AWS Marketplace procurement insights dashboard. For more information, see this note.
> If you enable trusted access by using the AWS Marketplace procurement insights dashboard console or tools then you don't need to complete these steps.

To enable trusted access by enabling the AWS Marketplace procurement insights dashboard

See Enabling the AWS Marketplace procurement insights dashboard in the *AWS Marketplace Buyer Guide*.

To enable trusted access using Organizations tools

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Marketplace procurement insights dashboard** in the list of services.
4. Choose **Enable trusted access**.
5. In the **Enable trusted access for AWS Marketplace procurement insights dashboard** dialog box, type **enable** to confirm, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Marketplace procurement insights dashboard that they can now enable that service to work with AWS Organizations from the service console .

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable AWS Marketplace procurement insights dashboard as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal procurement-insights.marketplace.amazonaws.com
  ```

  This command produces no output when successful.

- AWS API: EnableAWSServiceAccess

Disabling trusted access with AWS Marketplace procurement insights

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can only disable trusted access using the Organizations tools.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS Marketplace procurement insights dashboard as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal procurement-insights.marketplace.amazonaws.com
  ```

  This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for AWS Marketplace procurement insights

To configure a delegated administrator in the AWS Marketplace procurement insights console, see See Registering delegated administrators> in the *AWS Marketplace Buyer Guide*.

You can also configure a delegated administrator by using the Organizations `RegisterDelegatedAdministrator` API. For more information, see RegisterDelegatedAdministrator in the *Organizations Command Reference*.

Disabling a delegated administrator for AWS Marketplace procurement insights

Only an administrator in the organization management account can configure a delegated administrator for AWS Marketplace procurement insights.

To remove a delegated administrator through the AWS Marketplace procurement insights console, see Deregistering delegated administrators in the *AWS Marketplace Buyer Guide*.

You can also remove the delegated administrator by using the Organizations `DeregisterDelegatedAdministrator` CLI or SDK operation.

AWS Network Manager and AWS Organizations

Network Manager enables you to centrally manage your AWS Cloud WAN core network and your AWS Transit Gateway network across AWS accounts, Regions, and on-premises locations. With multi-account support you can create a single global network for any of your AWS accounts, and register transit gateways from multiple accounts to the global network using the Network Manager console.

With trusted access between Network Manager and Organizations enabled, the registered delegated administrators and the management accounts can leverage the service-linked role deployed in the member accounts to describe resources attached to your global networks. From the Network Manager console the registered delegated administrators and the management accounts can assume the custom IAM roles deployed in the member accounts: `CloudWatch-CrossAccountSharingRole` for multi-account monitoring and eventing, and `IAMRoleForAWSNetworkManagerCrossAccountResourceAccess` for the console switch role access for viewing and managing multi-account resources)

Important

- We strongly recommend using the Network Manager console to manage multi-account settings (enable/disable trusted access and register/deregister delegated administrators). Managing these settings from the console automatically deploys and manages all required service-linked roles and custom IAM roles to the member accounts needed for multi-account access.
- When you enable trusted access for Network Manager in the Network Manager console, the console also enables AWS CloudFormation StackSets service. Network Manager uses StackSets to deploy custom IAM roles needed for multi-account management.

For more information about integrating Network Manager with Organizations, see Manage multiple accounts in Network Manager with AWS Organizations in the *Amazon VPC User Guide*.

Use the following information to help you integrate AWS Network Manager with AWS Organizations.

Service-linked roles created when you enable integration

When you enable trusted access, the following service-linked roles are automatically created in the listed organization accounts. These roles allow Network Manager to perform supported operations within the accounts in your organization. If you disable trusted access, Network Manager will not delete these roles from accounts in your organization. You can manually delete them using the IAM console.

Management account

- `AWSServiceRoleForNetworkManager`
- `AWSServiceRoleForCloudFormationStackSetsOrgAdmin`
- `AWSServiceRoleForCloudWatchCrossAccount`

Member accounts

- `AWSServiceRoleForNetworkManager`
- `AWSServiceRoleForCloudFormationStackSetsOrgMember`

When you register a member account as a delegated administrator, the following additional role is automatically created in the delegated administrator account:

- `AWSServiceRoleForCloudWatchCrossAccount`

Service principals used by the service-linked roles

The service-linked roles can only be assumed by the service principals authorized by the trust relationships defined for the role.

- For the `AWSServiceRoleForNetworkManager service-linked` role, `networkmanager.amazonaws.com` is the only service principal that has access.
- For the `AWSServiceRoleForCloudFormationStackSetsOrgMember` service-linked role, `member.org.stacksets.cloudformation.amazonaws.com` is the only service principal that has access.
- For the `AWSServiceRoleForCloudFormationStackSetsOrgAdmin` service-linked role, `stacksets.cloudformation.amazonaws.com` is the only service principal that has access.

- For the `AWSServiceRoleForCloudWatchCrossAccount` service-linked role, `cloudwatch-crossaccount.amazonaws.com` is the only service principal that has access.

Deleting these roles will impair multi-account functionality for Network Manager.

Enabling trusted access with Network Manager

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

Only an administrator in the Organizations management account has permissions to enable trusted access with another AWS service. Be sure to use the Network Manager *console* to enable trusted access, to avoid permissions issues. For more information, see Manage multiple accounts in Network Manager with AWS Organizations in the *Amazon VPC User Guide*.

Disabling trusted access with Network Manager

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

Only an administrator in an Organizations management account has permissions to disable trusted access with another AWS service.

> **Important**
>
> We strongly recommend using the Network Manager console to disable trusted access. If you disable trusted access in any other way, such as using AWS CLI, with an API, or with the AWS CloudFormation console, deployed AWS CloudFormation StackSets and custom IAM roles may not be properly cleaned up. To disable trusted service access, sign in to the Network Manager console.

Enabling a delegated administrator account for Network Manager

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for Network Manager that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of Network Manager.

For instructions on how to designate a member account as a delegated administrator of Network Manager in the organization, see Register a delegated administrator in the *Amazon VPC User Guide*.

Amazon Q Developer and AWS Organizations

Amazon Q Developer is a generative AI powered conversational assistant that can help you understand, build, extend, and operate AWS applications. It is also a general purpose, machine learning-powered code generator that provides you with code recommendations in real time. The paid subscription version of Amazon Q Developer requires Organizations integration. For more information see Account, IAM Identity Center, and Organizations setup in the *Amazon Q user guide*.

Use the following information to help you integrate Amazon Q Developer with AWS Organizations.

Service-linked roles

The `AWSServiceRoleForAmazonQDeveloper` service-linked role allows Amazon Q Developer to perform supported operations within your organization. Create the role using the Amazon Q Developer console, API, or CLI, as described in Creating a service-linked role in the IAM User Guide.

If you are using a member account, then you can delete or modify this role only if you disable trusted access between Amazon Q Developer and Organizations, or if you remove the member account from the organization.

Service principals used by Amazon Q Developer

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Amazon Q Developer grant access to the following service principals:

- `q.amazonaws.com`

Enabling trusted access with Amazon Q Developer

Amazon Q Developer Pro uses trusted access to share the settings made in the Organizations management account with member accounts in the same organization.

For example, the Amazon Q Developer Pro administrator, working in the Organizations management account, may enable suggestions with code references. If trusted access is enabled,

then suggestions with code references will also be enabled for all member accounts in that organization.

You can only enable trusted access using Amazon Q Developer.

To enable trusted access for Amazon Q Developer, use this procedure.

1. On the Amazon Q Developer **Settings** page, under **Member account settings**, choose **Edit**.
2. In the pop-up window, select **On**.
3. Choose **Save**.

For more information, see Enabling trusted access in the *Amazon Q Developer user guide*.

Disabling trusted access with Amazon Q Developer

You can only disable trusted access using the Amazon Q Developer tools.

To disable trusted access for Amazon Q Developer, use this procedure.

1. On the Amazon Q Developer **Settings** page, under **Member account settings**, choose **Edit**.
2. In the pop-up window, select **Off**.
3. Choose **Save**.

For more information, see Enabling trusted access in the *Amazon Q Developer user guide*.

AWS Resource Access Manager and AWS Organizations

AWS Resource Access Manager (AWS RAM) enables you to share specified AWS resources that you own with other AWS accounts. It's a centralized service that provides a consistent experience for sharing different types of AWS resources across multiple accounts.

For more information about AWS RAM, see the *AWS RAM User Guide*.

Use the following information to help you integrate AWS Resource Access Manager with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows AWS RAM to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between AWS RAM and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForResourceAccessManager`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by AWS RAM grant access to the following service principals:

- `ram.amazonaws.com`

Enabling trusted access with AWS RAM

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can enable trusted access using either the AWS Resource Access Manager console or the AWS Organizations console.

> **Important**
>
> We strongly recommend that whenever possible, you use the AWS Resource Access Manager console or tools to enable integration with Organizations. This lets AWS Resource Access Manager perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by AWS Resource Access Manager. For more information, see this note. If you enable trusted access by using the AWS Resource Access Manager console or tools then you don't need to complete these steps.

To enable trusted access using the AWS RAM console or CLI

See Enable Sharing with AWS Organizations in the *AWS RAM User Guide*.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Resource Access Manager** in the list of services.
4. Choose **Enable trusted access**.
5. In the **Enable trusted access for AWS Resource Access Manager** dialog box, type **enable** to confirm, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Resource Access Manager that they can now enable that service to work with AWS Organizations from the service console .

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable AWS Resource Access Manager as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal ram.amazonaws.com
  ```

  This command produces no output when successful.

- AWS API: EnableAWSServiceAccess

Disabling trusted access with AWS RAM

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

You can disable trusted access using either the AWS Resource Access Manager or the AWS Organizations tools.

> **Important**
>
> We strongly recommend that whenever possible, you use the AWS Resource Access Manager console or tools to disable integration with Organizations. This lets AWS Resource Access Manager perform any clean up that it requires, such as deleting resources or access roles that are no longer needed by the service. Proceed with these steps only if you can't disable integration using the tools provided by AWS Resource Access Manager.
> If you disable trusted access by using the AWS Resource Access Manager console or tools then you don't need to complete these steps.

To disable trusted access using the AWS Resource Access Manager console or CLI

See Enable Sharing with AWS Organizations in the *AWS RAM User Guide*.

You can disable trusted access by using either the AWS Organizations console, by running an Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Resource Access Manager** in the list of services.
4. Choose **Disable trusted access**.
5. In the **Disable trusted access for AWS Resource Access Manager** dialog box, type **disable** to confirm, and then choose **Disable trusted access**.

6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Resource Access Manager that they can now disable that service from working with AWS Organizations using the service console or tools .

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS Resource Access Manager as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal ram.amazonaws.com
  ```

  This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

AWS Resource Explorer and AWS Organizations

AWS Resource Explorer is a resource search and discovery service. With Resource Explorer, you can explore your resources, such as Amazon Elastic Compute Cloud instances, Amazon Kinesis Data Streams, or Amazon DynamoDB tables, using an internet search engine-like experience. You can search for your resources using resource metadata such as names, tags, and IDs. Resource Explorer works across AWS Regions in your account to simplify your cross-Region workloads.

When you integrate Resource Explorer with AWS Organizations, you can gather evidence from a broader source by including multiple AWS accounts from your organization within the scope of your assessments.

Use the following information to help you integrate AWS Resource Explorer with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Resource Explorer to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Resource Explorer and Organizations, or if you remove the member account from the organization.

For more information about how Resource Explorer uses this role, see Using service-linked roles in the *AWS Resource Explorer Users Guide*.

- `AWSServiceRoleForResourceExplorer`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Resource Explorer grant access to the following service principals:

- `resource-explorer-2.amazonaws.com`

To enable trusted access with AWS Resource Explorer

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

Resource Explorer requires trusted access to AWS Organizations before you can designate a member account to be the delegated administrator for your organization.

You can enable trusted access using either the Resource Explorer console or the Organizations console. We strongly recommend that whenever possible, you use the Resource Explorer console or tools to enable integration with Organizations. This lets AWS Resource Explorer perform any configuration that it requires, such as creating resources needed by the service.

To enable trusted access using the Resource Explorer console

For instructions about enabling trusted access, see Prerequisites to using Resource Explorer in the *AWS Resource Explorer User Guide*.

> **Note**
>
> If you configure a delegated administrator using the AWS Resource Explorer console, then AWS Resource Explorer automatically enables trusted access for you.

You can enable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To enable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable AWS Resource Explorer as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal resource-explorer-2.amazonaws.com
  ```

  This command produces no output when successful.

- AWS API: EnableAWSServiceAccess

To disable trusted access with Resource Explorer

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

Only an administrator in the AWS Organizations management account can disable trusted access with AWS Resource Explorer.

You can disable trusted access using either the AWS Resource Explorer or the AWS Organizations tools.

Important

We strongly recommend that whenever possible, you use the AWS Resource Explorer console or tools to disable integration with Organizations. This lets AWS Resource Explorer perform any clean up that it requires, such as deleting resources or access roles that are no longer needed by the service. Proceed with these steps only if you can't disable integration using the tools provided by AWS Resource Explorer.
If you disable trusted access by using the AWS Resource Explorer console or tools then you don't need to complete these steps.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS Resource Explorer as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal resource-explorer-2.amazonaws.com
  ```

  This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for Resource Explorer

Use your delegated administrator account to create multi-account resource views and scope it to an organizational unit or your entire organization. You can share multi-account views with any account in your organization via AWS Resource Access Manager by creating resource shares.

Minimum permissions

Only a user or role in the Organizations management account with the following permission can configure a member account as a delegated administrator for Resource Explorer in the organization: `resource-explorer:RegisterAccount`

For instruction about enabling a delegated administrator account for Resource Explorer, see Setting Up in the *AWS Resource Explorer User Guide*.

If you configure a delegated administrator using the AWS Resource Explorer console, then Resource Explorer automatically enables trusted access for you.

AWS CLI, AWS API

If you want to configure a delegated administrator account using the AWS CLI or one of the AWS SDKs, you can use the following commands:

- AWS CLI:

```
$ aws organizations register-delegated-administrator \
    --account-id 123456789012 \
    --service-principal resource-explorer-2.amazonaws.com
```

- AWS SDK: Call the Organizations `RegisterDelegatedAdministrator` operation and the member account's ID number and identify the account service `resource-explorer-2.amazonaws.com` as parameters.

Disabling a delegated administrator for Resource Explorer

Only an administrator in the Organizations management account or in the Resource Explorer delegated administrator account can remove a delegated administrator for Resource Explorer. You can disable trusted access using the Organizations `DeregisterDelegatedAdministrator` CLI or SDK operation.

AWS Security Hub and AWS Organizations

AWS Security Hub provides you with a comprehensive view of your security state in AWS and helps you check your environment against security industry standards and best practices.

Security Hub collects security data from across your AWS accounts, the AWS services you use, and supported third-party partner products. It helps you to analyze your security trends and identify the highest priority security issues.

When you use both Security Hub and AWS Organizations together, you can automatically enable Security Hub for all of your accounts, including new accounts as they are added. This increases the coverage for Security Hub checks and findings, which provides a more comprehensive and accurate picture of your overall security posture.

For more information about Security Hub, see the *AWS Security Hub User Guide*.

Use the following information to help you integrate AWS Security Hub with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Security Hub to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Security Hub and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForSecurityHub`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Security Hub grant access to the following service principals:

- `securityhub.amazonaws.com`

Enabling trusted access with Security Hub

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

When you designate a delegated administrator for Security Hub, Security Hub automatically enables trusted access for Security Hub in your organization.

Disabling trusted access with Security Hub

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access in the *AWS Organizations User Guide*.

Before you disable trusted access, we recommend working with the delegated administrator for your organization to disable Security Hub in member accounts and to clean up Security Hub resources in those accounts.

You can disable trusted access by using the AWS Organizations console, Organizations API, or the AWS CLI. Only an administrator of the Organizations management account can disable trusted access with Security Hub.

For instructions on disabling trusted access with Security Hub, see Disabling Security Hub integration with AWS Organizations.

Enabling a delegated administrator for Security Hub

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for Security Hub that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of Security Hub.

For information, see Designating a Security Hub administrator account in the *AWS Security Hub User Guide*.

To designate a member account as a delegated administrator for Security Hub

1. Sign in with your Organizations management account.
2. Perform one of the following:
   - If your management account does not have Security Hub enabled, then on the Security Hub console, choose **Go to Security Hub**.
   - If your management account does have Security Hub enabled, then on the Security Hub console, under **General** choose **Settings**.
3. Under **Delegated Administrator**, enter the account ID.

Disabling a delegated administrator for Security Hub

Only the organization management account can remove the delegated Security Hub administrator account.

To change the delegated Security Hub administrator, you must first remove the current delegated administrator account and then designate a new one.

If you use the Security Hub console to remove the delegated administrator in one Region, it is automatically removed in all Regions.

The Security Hub API only removes the delegated Security Hub administrator account from the Region where the API call or command is issued. You must repeat the action in other Regions.

If you use the Organizations API to remove the delegated Security Hub administrator account, it is automatically removed in all Regions.

For instructions on disabling the delegated Security Hub administrator, see Removing or changing the delegated administrator.

Amazon S3 Storage Lens and AWS Organizations

By giving Amazon S3 Storage Lens trusted access to your organization, you allow it to collect and aggregate metrics across all of the AWS accounts in your organization. S3 Storage Lens does this by accessing the list of accounts that belong to your organization and collects and analyzes the storage and usage and activity metrics for all of them.

For more information, see the Using service-linked roles for Amazon S3 Storage Lens in the *Amazon S3 Storage Lens User Guide*.

Use the following information to help you integrate Amazon S3 Storage Lens with AWS Organizations.

Service-linked role created when you enable integration

The following service-linked role is automatically created in your organization's delegated administrator account when you enable trusted access and the Storage Lens configuration has been applied to your organization. This role allows Amazon S3 Storage Lens to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Amazon S3 Storage Lens and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForS3StorageLens`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Amazon S3 Storage Lens grant access to the following service principals:

- `storage-lens.s3.amazonaws.com`

Enabling trusted access with Amazon S3 Storage Lens

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can enable trusted access using either the Amazon S3 Storage Lens console or the AWS Organizations console.

> **Important**
>
> We strongly recommend that whenever possible, you use the Amazon S3 Storage Lens console or tools to enable integration with Organizations. This lets Amazon S3 Storage Lens perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by Amazon S3 Storage Lens. For more information, see this note.
> If you enable trusted access by using the Amazon S3 Storage Lens console or tools then you don't need to complete these steps.

To enable trusted access using the Amazon S3 console

See Enabling trusted access for S3 Storage Lens in the *Amazon Simple Storage Service User Guide*.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **Amazon S3 Storage Lens** in the list of services.
4. Choose **Enable trusted access**.
5. In the **Enable trusted access for Amazon S3 Storage Lens** dialog box, type **enable** to confirm, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of Amazon S3 Storage Lens that they can now enable that service to work with AWS Organizations from the service console .

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable Amazon S3 Storage Lens as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal storage-lens.s3.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: EnableAWSServiceAccess

Disabling trusted access with Amazon S3 Storage Lens

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

You can only disable trusted access using the Amazon S3 Storage Lens tools.

You can disable trusted access using the Amazon S3 console, the AWS CLI or any of the AWS SDKs.

To disable trusted access using the Amazon S3 console

See Disabling trusted access for S3 Storage Lens in the *Amazon Simple Storage Service User Guide*.

Enabling a delegated administrator account for Amazon S3 Storage Lens

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for Amazon S3 Storage Lens that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of Amazon S3 Storage Lens.

> **Minimum permissions**
>
> Only a user or role in the Organizations management account with the following permission can configure a member account as a delegated administrator for Amazon S3 Storage Lens in the organization:
> `organizations:RegisterDelegatedAdministrator`
> `organizations:DeregisterDelegatedAdministrator`

Amazon S3 Storage Lens supports a maximum of 5 delegated administrator accounts in your organization.

To designate a member account as a delegated administrator for Amazon S3 Storage Lens

You can register a delegated administrator using the Amazon S3 console, the AWS CLI or any of the AWS SDKs. To register a member account as a delegated administrator account for your organization using the Amazon S3 console, see Registering a delegated administrator for S3 Storage Lens in the *Amazon Simple Storage Service User Guide*.

To deregister a delegated administrator for Amazon S3 Storage Lens

You can deregister a delegated administrator using the Amazon S3 console, the AWS CLI or any of the AWS SDKs. To deregister a delegated administrator using the Amazon S3 console, see Deregistering a delegated administrator for S3 Storage Lens in the *Amazon Simple Storage Service User Guide*.

AWS Security Incident Response and AWS Organizations

AWS Security Incident Response is a security service that provides 24/7, live, human-assisted security incident support to help customers respond rapidly to cybersecurity incidents such as credential theft and ransomware attacks. By integrating with Organizations you enable security coverage for your entire organization. For more information, see Managing AWS Security Incident Response accounts with AWS Organizations in the *Security Incident Response User Guide*.

Use the following information to help you integrate AWS Security Incident Response with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked roles are automatically created in your organization's management account when you enable trusted access.

- `AWSServiceRoleForSecurityIncidentResponse` - used for creating Security Incident Response membership - your subscription to the service through AWS Organizations.
- `AWSServiceRoleForSecurityIncidentResponse_Triage` - used only when you enable the triage feature during sign-up.

Service principals used by Security Incident Response

The service-linked roles in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Security Incident Response grant access to the following service principal:

- `security-ir.amazonaws.com`

Enabling trusted access to Security Incident Response

Enabling trusted access to Security Incident Response allows the service to keep track of your organization's structure and ensure that all accounts in the organization have active security incident coverage. It also allows the service to use a service-linked role in member accounts for triaging capabilities when you enable the triage feature.

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can enable trusted access using either the AWS Security Incident Response console or the AWS Organizations console.

> **Important**
>
> We strongly recommend that whenever possible, you use the AWS Security Incident Response console or tools to enable integration with Organizations. This lets AWS Security Incident Response perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by AWS Security Incident Response. For more information, see this note. If you enable trusted access by using the AWS Security Incident Response console or tools then you don't need to complete these steps.

Organizations automatically enables the Organizations trusted access when you use the Security Incident Response console for setup and management. If you use the Security Incident Response CLI/SDK then you have to manually enable trusted access by using the EnableAWSServiceAccess API. To learn how to enable trusted access through the Security Incident Response console, see Enabling trusted access for AWS Account Management in the *Security Incident Response User Guide*.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Security Incident Response** in the list of services.
4. Choose **Enable trusted access**.
5. In the **Enable trusted access for AWS Security Incident Response** dialog box, type **enable** to confirm, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Security Incident Response that they can now enable that service to work with AWS Organizations from the service console .

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable AWS Security Incident Response as a trusted service with Organizations.

```
$ aws organizations enable-aws-service-access \
    --service-principal security-ir.amazonaws.com
```

  This command produces no output when successful.

- AWS API: EnableAWSServiceAccess

Disabling trusted access with Security Incident Response

Only an administrator in the Organizations management account can disable trusted access with Security Incident Response.

You can only disable trusted access using the Organizations tools.

You can disable trusted access by using either the AWS Organizations console, by running an Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Security Incident Response** in the list of services.
4. Choose **Disable trusted access**.

5. In the **Disable trusted access for AWS Security Incident Response** dialog box, type **disable** to confirm, and then choose **Disable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Security Incident Response that they can now disable that service from working with AWS Organizations using the service console or tools .

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS Security Incident Response as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal security-ir.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for Security Incident Response

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for Security Incident Response that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of Security Incident Response. For more information, see Managing AWS Security Incident Response accounts with AWS Organizations in the *Security Incident Response User Guide*.

Minimum permissions

Only a user or role in the Organizations management account can configure a member account as a delegated administrator for Security Incident Response in the organization

To learn how to configure a delegated administrator through the Security Incident Response console, see Designating a delegated Security Incident Response administrator account in the *Security Incident Response User Guide*.

AWS CLI, AWS API

If you want to configure a delegated administrator account using the AWS CLI or one of the AWS SDKs, you can use the following commands:

- AWS CLI:

```
$ aws organizations register-delegated-administrator \
    --account-id 123456789012 \
    --service-principal security-ir.amazonaws.com
```

- AWS SDK: Call the Organizations `RegisterDelegatedAdministrator` operation and the member account's ID number and identify the account service `security-ir.amazonaws.com` as parameters.

Disabling a delegated administrator for Security Incident Response

Important

If membership was created from the delegated administrator account, deregistering the delegated administrator is a destructive action and will cause service disruption. To re-register DA:

1. Sign in to the Security Incident Response console at https://console.aws.amazon.com/security-ir/home#/membership/settings
2. Cancel membership from the service console. Membership remains active until the end of billing cycle.
3. Once membership is cancelled disable service access through the Organizations console, CLI or SDK.

Only an administrator in the Organizations management account can remove a delegated administrator for Security Incident Response. You can remove the delegated administrator using the Organizations `DeregisterDelegatedAdministrator` CLI or SDK operation.

Amazon Security Lake and AWS Organizations

Amazon Security Lake centralizes security data from cloud, on-premises, and custom sources into a data lake that's stored in your account. By integrating with Organizations, you can create a data lake that collects logs and events across your accounts. For more information see Managing multiple accounts with AWS Organizations in the *Amazon Security Lake user guide*.

Use the following information to help you integrate Amazon Security Lake with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you call the RegisterDataLakeDelegatedAdministrator API. This role allows Amazon Security Lake to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Amazon Security Lake and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForSecurityLake`

> ⚠ **Recommendation: Use Security Lake's RegisterDataLakeDelegatedAdministrator API to allow Security Lake access to your Organization and to register Organizations's delegated administrator**
>
> If you use Organizations' APIs to register a delegated administrator, service-linked roles for the Organizations might not be created successfully. To ensure full functionality, use the Security Lake APIs.

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Amazon Security Lake grant access to the following service principals:

- `securitylake.amazonaws.com`

Enabling trusted access with Amazon Security Lake

When you enable trusted access with Security Lake, Security Lake can react automatically to changes in the organization membership. The delegated administrator can enable AWS logs collection from supported services in any organization account. For more information, see Service-linked role for Amazon Security Lake in the *Amazon Security Lake user guide*.

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can only enable trusted access using the Organizations tools.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **Amazon Security Lake** in the list of services.
4. Choose **Enable trusted access**.
5. In the **Enable trusted access for Amazon Security Lake** dialog box, type **enable** to confirm, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of Amazon Security Lake that they can now enable that service to work with AWS Organizations from the service console .

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

Run the following command to enable Amazon Security Lake as a trusted service with Organizations.

```
$ aws organizations enable-aws-service-access \
    --service-principal securitylake.amazonaws.com
```

This command produces no output when successful.

- AWS API: EnableAWSServiceAccess

Disabling trusted access with Amazon Security Lake

Only an administrator in the Organizations management account can disable trusted access with Amazon Security Lake.

You can only disable trusted access using the Organizations tools.

You can disable trusted access by using either the AWS Organizations console, by running an Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **Amazon Security Lake** in the list of services.
4. Choose **Disable trusted access**.
5. In the **Disable trusted access for Amazon Security Lake** dialog box, type **disable** to confirm, and then choose **Disable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of Amazon Security Lake that they can now disable that service from working with AWS Organizations using the service console or tools .

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable Amazon Security Lake as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal securitylake.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for Amazon Security Lake

The Amazon Security Lake delegated administrator adds other accounts in the organization as member accounts. The delegated administrator can enable Amazon Security Lake and configure Amazon Security Lake settings for the member accounts. The delegated administrator can collect logs across an organization in all AWS Regions where Amazon Security Lake is enabled (regardless of which Regional endpoint you're currently using).

You can also set up the delegated administrator to automatically add new accounts in the organization as members. The Amazon Security Lake delegated administrator has access to the logs and events in associated member accounts. Accordingly, you can set up Amazon Security Lake to collect data owned by associated member accounts. You can also grant subscribers permission to consume data owned by associated member accounts.

For more information see Managing multiple accounts with AWS Organizations in the *Amazon Security Lake user guide*.

Minimum permissions

Only an administrator in the Organizations management account can configure a member account as a delegated administrator for Amazon Security Lake in the organization

You can specify a delegated administrator account by using the Amazon Security Lake console, the Amazon Security Lake `CreateDatalakeDelegatedAdmin` API operation, or the `create-datalake-delegated-admin` CLI command. Alternatively, you can use the Organizations `RegisterDelegatedAdministrator` CLI or SDK operation. For instructions about enabling a delegated administrator account for Amazon Security Lake, see Designating the delegated Security Lake administrator and adding member accounts in the *Amazon Security Lake user guide*.

AWS CLI, AWS API

If you want to configure a delegated administrator account using the AWS CLI or one of the AWS SDKs, you can use the following commands:

- AWS CLI:

```
$  aws organizations register-delegated-administrator \
    --account-id 123456789012 \ --service-principal securitylake.amazonaws.com
```

- AWS SDK: Call the Organizations `RegisterDelegatedAdministrator` operation and the member account's ID number and identify the account service principal `account.amazonaws.com` as parameters.

Disabling a delegated administrator for Amazon Security Lake

Only an administrator in either the Organizations management account or the Amazon Security Lake delegated administrator account can remove a delegated administrator account from the organization.

You can remove the delegated administrator account by using the Amazon Security Lake `DeregisterDataLakeDelegatedAdministrator` API operation, the `deregister-data-lake-delegated-administrator` CLI command, or by using the Organizations `DeregisterDelegatedAdministrator` CLI or SDK operation. To remove a delegated administrator using Amazon Security Lake, see Removing the Amazon Security Lake delegated administrator in the *Amazon Security Lake user guide*.

AWS Service Catalog and AWS Organizations

Service Catalog enables you to create and manage catalogs of IT services that are approved for use on AWS.

The integration of Service Catalog with AWS Organizations simplifies the sharing of portfolios and copying of products across an organization. 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 Service Catalog.

For more information about Service Catalog, see the *Service Catalog Administrator Guide*.

Use the following information to help you integrate AWS Service Catalog with AWS Organizations.

Service-linked roles created when you enable integration

AWS Service Catalog doesn't create any service-linked roles as part of enabling trusted access.

Service principals used to grant permissions

To enable trusted access, you must specify the following service principal:

- `servicecatalog.amazonaws.com`

Enabling trusted access with Service Catalog

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can enable trusted access using either the AWS Service Catalog console or the AWS Organizations console.

> **Important**
>
> We strongly recommend that whenever possible, you use the AWS Service Catalog console or tools to enable integration with Organizations. This lets AWS Service Catalog perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by AWS Service Catalog. For more information, see this note.
> If you enable trusted access by using the AWS Service Catalog console or tools then you don't need to complete these steps.

To enable trusted access using the Service Catalog CLI or AWS SDK

Call one of the following commands or operations:

- AWS CLI: aws servicecatalog enable-aws-organizations-access
- AWS SDKs: AWSServiceCatalog::EnableAWSOrganizationsAccess

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Service Catalog** in the list of services.
4. Choose **Enable trusted access**.
5. In the **Enable trusted access for AWS Service Catalog** dialog box, type **enable** to confirm, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Service Catalog that they can now enable that service to work with AWS Organizations from the service console .

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable AWS Service Catalog as a trusted service with Organizations.

```
$ aws organizations enable-aws-service-access \
```

```
    --service-principal servicecatalog.amazonaws.com
```

This command produces no output when successful.

- AWS API: EnableAWSServiceAccess

Disabling trusted access with Service Catalog

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

If you disable trusted access using AWS Organizations while you are using 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.

To disable trusted access using the Service Catalog CLI or AWS SDK

Call one of the following commands or operations:

- AWS CLI: aws servicecatalog disable-aws-organizations-access
- AWS SDKs: DisableAWSOrganizationsAccess

You can disable trusted access by using either the AWS Organizations console, by running an Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Service Catalog** in the list of services.
4. Choose **Disable trusted access**.
5. In the **Disable trusted access for AWS Service Catalog** dialog box, type **disable** to confirm, and then choose **Disable trusted access**.

6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Service Catalog that they can now disable that service from working with AWS Organizations using the service console or tools .

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS Service Catalog as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal servicecatalog.amazonaws.com
  ```

  This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Service Quotas and AWS Organizations

Service Quotas is an AWS service that enables you to view and manage your quotas from a central location. Quotas, also referred to as limits, are the maximum value for your resources, actions, and items in your AWS account.

When Service Quotas is associated with AWS Organizations, you can create a quota request template to automatically request quota increases when accounts are created.

For more information about Service Quotas, see the Service Quotas User Guide.

Use the following information to help you integrate Service Quotas with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Service Quotas to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Service Quotas and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForServiceQuotas`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Service Quotas grant access to the following service principals:

- `servicequotas.amazonaws.com`

Enabling trusted access with Service Quotas

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can only enable trusted access using Service Quotas.

You can enable trusted access using the Service Quotas console, AWS CLI or SDK:

- **To enable trusted access using the Service Quotas console**

  Sign in with your AWS Organizations management account and then configure the template on the Service Quotas console. For more information, see Using the Service Quota Template in the *Service Quotas User Guide*.

- **To enable trusted access using the Service Quotas AWS CLI or SDK**

  Call the following command or operation:

  - AWS CLI: aws service-quotas associate-service-quota-template
  - AWS SDKs: AssociateServiceQuotaTemplate

AWS IAM Identity Center and AWS Organizations

AWS IAM Identity Center provides single sign-on access for all of your AWS accounts and cloud applications. It connects with Microsoft Active Directory through AWS Directory Service to allow users in that directory to sign in to a personalized AWS access portal using their existing Active Directory user names and passwords. From the AWS access portal, users have access to all the AWS accounts and cloud applications that they have permissions for.

For more information about IAM Identity Center, see the AWS IAM Identity Center User Guide.

Use the following information to help you integrate AWS IAM Identity Center with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows IAM Identity Center to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between IAM Identity Center and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForSSO`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by IAM Identity Center grant access to the following service principals:

- `sso.amazonaws.com`

Enabling trusted access with IAM Identity Center

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can enable trusted access using either the AWS IAM Identity Center console or the AWS Organizations console.

> **Important**
>
> We strongly recommend that whenever possible, you use the AWS IAM Identity Center console or tools to enable integration with Organizations. This lets AWS IAM Identity Center perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by AWS IAM Identity Center. For more information, see this note.
> If you enable trusted access by using the AWS IAM Identity Center console or tools then you don't need to complete these steps.

IAM Identity Center requires trusted access with AWS Organizations to function. Trusted access is enabled when you set up IAM Identity Center. For more information, see Getting Started - Step 1: Enable AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS IAM Identity Center** in the list of services.
4. Choose **Enable trusted access**.
5. In the **Enable trusted access for AWS IAM Identity Center** dialog box, type **enable** to confirm, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS IAM Identity Center that they can now enable that service to work with AWS Organizations from the service console .

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable AWS IAM Identity Center as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal sso.amazonaws.com
  ```

  This command produces no output when successful.

- AWS API: EnableAWSServiceAccess

Disabling trusted access with IAM Identity Center

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

IAM Identity Center requires trusted access with AWS Organizations to operate. If you disable trusted access using AWS Organizations while you are using IAM Identity Center, it stops functioning because it can't access the organization. Users can't use IAM Identity Center to access accounts. Any roles that IAM Identity Center creates remain, but the IAM Identity Center service can't access them. The IAM Identity Center service-linked roles remain. If you reenable trusted access, IAM Identity Center continues to operate as before, without the need for you to reconfigure the service.

If you remove an account from your organization, IAM Identity Center automatically cleans up any metadata and resources, such as its service-linked role. A standalone account that is removed from an organization no longer works with IAM Identity Center.

You can only disable trusted access using the Organizations tools.

You can disable trusted access by using either the AWS Organizations console, by running an Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS IAM Identity Center** in the list of services.
4. Choose **Disable trusted access**.
5. In the **Disable trusted access for AWS IAM Identity Center** dialog box, type **disable** to confirm, and then choose **Disable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS IAM Identity Center that they can now disable that service from working with AWS Organizations using the service console or tools .

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS IAM Identity Center as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal sso.amazonaws.com
  ```

  This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for IAM Identity Center

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for IAM Identity Center that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of IAM Identity Center.

> **Minimum permissions**
>
> Only a user or role in the Organizations management account can configure a member account as a delegated administrator for IAM Identity Center in the organization.

For instructions about how to enable a delegated administrator account for IAM Identity Center, see Delegated administration in the *AWS IAM Identity Center 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. The following Systems Manager capabilities work with Organizations across all of the AWS accounts in your organization:

- Systems Manager Explorer, is a customizable operations dashboard that reports information about your AWS resources. You can synchronize operations data across all AWS accounts in your organization by using Organizations and Systems Manager Explorer. For more information, see Systems Manager Explorer in the *AWS Systems Manager User Guide*.
- Systems Manager Change Manager is an enterprise change management framework for requesting, approving, implementing, and reporting on operational changes to your application configuration and infrastructure. For more information, see AWS Systems Manager Change Manager in the *AWS Systems Manager User Guide*.
- Systems Manager OpsCenter provides a central location where operations engineers and IT professionals can view, investigate, and resolve operational work items (OpsItems) related to AWS resources. When you use OpsCenter with Organizations it supports working with OpsItems from a management account (either an Organizations management account or a Systems Manager delegated administrator account) and one other account during a single session. Once configured, users can perform the following types of actions:
  - Create, view, and update OpsItems in another account.

- View detailed information about AWS resources that are specified in OpsItems in another account.
- Start Systems Manager Automation runbooks to remediate issues with AWS resources in another account.

  For more information, see AWS Systems Manager OpsCenter in the *AWS Systems Manager User Guide*.

- Use Quick Setup to quickly configure frequently used AWS services and features with recommended best practices. For more information, see AWS Systems Manager Quick Setup in the *AWS Systems Manager User Guide*.

  When you register an AWS Organizations delegated administrator account for Systems Manager you can create, update, view, and delete Quick Setup configuration managers that target organizational units in an organization. Learn more in Using a delegated administrator for Quick Setup in the *AWS Systems Manager User Guide*.

- When you set up the integrated console for Systems Manager, you enter a delegated administrator account. This account is used to register AWS Organizations delegated administrator accounts with Quick Setup, Explorer, CloudFormation StackSets, and Resource Explorer. Learn more in Setting up Systems Manager integrated console for an organization *AWS Systems Manager User Guide*.

Use the following information to help you integrate AWS Systems Manager with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Systems Manager to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Systems Manager and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForAmazonSSM_AccountDiscovery`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Systems Manager grant access to the following service principals:

- `ssm.amazonaws.com`

Enabling trusted access with Systems Manager

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can only enable trusted access using the Organizations tools.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Systems Manager** in the list of services.
4. Choose **Enable trusted access**.
5. In the **Enable trusted access for AWS Systems Manager** dialog box, type **enable** to confirm, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Systems Manager that they can now enable that service to work with AWS Organizations from the service console .

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable AWS Systems Manager as a trusted service with Organizations.

  ```
  $ aws organizations enable-aws-service-access \
      --service-principal ssm.amazonaws.com
  ```

  This command produces no output when successful.

- AWS API: EnableAWSServiceAccess

Disabling trusted access with Systems Manager

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

Systems Manager requires trusted access with AWS Organizations to synchronize operations data across AWS accounts in your organization. If you disable trusted access, then Systems Manager fails to synchronize operations data and reports an error.

You can only disable trusted access using the Organizations tools.

You can disable trusted access by using either the AWS Organizations console, by running an Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS Management Console

To disable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Systems Manager** in the list of services.
4. Choose **Disable trusted access**.

5. In the **Disable trusted access for AWS Systems Manager** dialog box, type **disable** to confirm, and then choose **Disable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Systems Manager that they can now disable that service from working with AWS Organizations using the service console or tools .

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

You can use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS Systems Manager as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal ssm.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for Systems Manager

When you designate a member account as a delegated administrator for the organization, users and roles from that account can perform administrative actions for Systems Manager that otherwise can be performed only by users or roles in the organization's management account. This helps you to separate management of the organization from management of Systems Manager.

If you use Change Manager across an organization, you use a delegated administrator account. This is the AWS account that has been designated as the account for managing change templates, change requests, change runbooks and approval workflows in Change Manager. The delegated account manages change activities across your organization. When you set up your organization for use with Change Manager, you specify which of your accounts serves in this role. It does not have to be the organization's management account. The delegated administrator account is not required if you use Change Manager with a single account only.

To designate a member account as a delegated administrator see the following topics in the *AWS Systems Manager User Guide*:

- For Explorer and OpsCenter, see Configuring a Delegated Administrator.
- For Change Manager, see Setting up an organization and delegated account for Change Manager.
- For Quick Setup see Register a delegated administrator for Quick Setup .

Disabling a delegated administrator account for Systems Manager

To deregister a delegated administrator see the following topics in the *AWS Systems Manager User Guide*:

- For Explorer and OpsCenter, see Deregister an Explorer delegated administrator .
- For Change Manager, see Setting up an organization and delegated account for Change Manager.
- For Quick Setup see Deregister a delegated administrator for Quick Setup .

AWS User Notifications and AWS Organizations

AWS User Notifications is a central location for your AWS notifications.

After you integrate with AWS Organizations, you can configure and view notifications centrally across accounts in your organization.

Use the following information to help you integrate AWS User Notifications with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows User Notifications to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between User Notifications and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForAWSUserNotifications`

For more information, see Using Service-Linked Roles in the *AWS User Notifications User Guide*.

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by User Notifications grant access to the following service principals:

- `notifications.amazon.com`

Enabling trusted access with User Notifications

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can only enable trusted access using AWS User Notifications.

To enable trusted access using the User Notifications console, see Enabling AWS Organizations in AWS User Notifications in the *User Notifications User Guide*.

Disabling trusted access with User Notifications

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

You can only enable trusted access using AWS User Notifications.

To disable trusted access using the User Notifications console, see Enabling AWS Organizations in AWS User Notifications in the *User Notifications User Guide*.

Enabling a delegated administrator account for User Notifications

The management account administrator can delegate User Notifications administrative permissions to a designated member account known as delegated administrator. To register an account as a delegated administrator for the private marketplace, the management account administrator must ensure that trusted access and the service-linked role are enabled, choose **Register a new administrator**, provide the 12-digit AWS account number, and choose **Submit**.

Management accounts and delegated administrator accounts can perform User Notifications administrative tasks, such as creating experiences, updating branding settings, associating or

disassociating audiences, adding or removing products, and approving or declining pending requests.

To configure a delegated administrator using the User Notifications console, see Registering delegated administrators in AWS User Notifications in the *User Notifications User Guide*.

You can also configure a delegated administrator by using the Organizations `RegisterDelegatedAdministrator` API. For more information, see RegisterDelegatedAdministrator in the *Organizations Command Reference*.

Disabling a delegated administrator for User Notifications

Only an administrator in the organization management account can configure a delegated administrator for User Notifications.

You can remove the delegated administrator using either the User Notifications console or API, or by using the Organizations `DeregisterDelegatedAdministrator` CLI or SDK operation.

To disable the delegated admin User Notifications account using the User Notifications console, see Removing delegated administrators in in AWS User Notifications in the *User Notifications User Guide*.

Tag policies and AWS Organizations

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

Use the following information to help you integrate tag policies with AWS Organizations.

Service principals used by the service-linked roles

Organizations interacts with the tags attached to your resources using the following service principal.

- `tagpolicies.tag.amazonaws.com`

Enabling trusted access for tag policies

You can enable trusted access either by enabling tag policies in the organization, or by using the AWS Organizations console.

> **Important**
>
> We strongly recommend that you enable trusted access by enabling tag policies. This enables Organizations to perform required setup tasks.

You can enable trusted access for tag policies by enabling the tag policy type in the AWS Organizations console. For more information, see Enabling a policy type.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **tag policies** in the list of services.
4. Choose **Enable trusted access**.
5. In the **Enable trusted access for tag policies** dialog box, type **enable** to confirm, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of tag policies that they can now enable that service to work with AWS Organizations from the service console .

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable tag policies as a trusted service with Organizations.

```
$ aws organizations enable-aws-service-access \
    --service-principal tagpolicies.tag.amazonaws.com
```

This command produces no output when successful.

- AWS API: EnableAWSServiceAccess

Disabling trusted access with tag policies

You can disable trusted access for tag policies by disabling the tag policy type in the AWS Organizations console. For more information, see Disabling a policy type.

AWS Trusted Advisor and AWS Organizations

AWS Trusted Advisor inspects your AWS environment and makes recommendations when opportunities exist to save money, to improve system availability and performance, or to help close security gaps. When integrated with Organizations, you can receive Trusted Advisor check results for all of the accounts in your organization and download reports to view the summaries of your checks and any affected resources.

For more information, see Organizational view for AWS Trusted Advisor in the *AWS Support User Guide*.

Use the following information to help you integrate AWS Trusted Advisor with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Trusted Advisor to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Trusted Advisor and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForTrustedAdvisorReporting`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Trusted Advisor grant access to the following service principals:

- `reporting.trustedadvisor.amazonaws.com`

Enabling trusted access with Trusted Advisor

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can only enable trusted access using AWS Trusted Advisor.

To enable trusted access using the Trusted Advisor console

See Enable organizational view in the *AWS Support User Guide*.

Disabling trusted access with Trusted Advisor

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

After you disable this feature, Trusted Advisor stops recording check information for all other accounts in your organization. You can't view or download existing reports or create new reports.

You can disable trusted access using either the AWS Trusted Advisor or the AWS Organizations tools.

> **Important**
>
> We strongly recommend that whenever possible, you use the AWS Trusted Advisor console or tools to disable integration with Organizations. This lets AWS Trusted Advisor perform any clean up that it requires, such as deleting resources or access roles that are no longer needed by the service. Proceed with these steps only if you can’t disable integration using the tools provided by AWS Trusted Advisor.
> If you disable trusted access by using the AWS Trusted Advisor console or tools then you don’t need to complete these steps.

To disable trusted access using the Trusted Advisor console

See Disable organizational view in the *AWS Support User Guide*.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable AWS Trusted Advisor as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal reporting.trustedadvisor.amazonaws.com
  ```

  This command produces no output when successful.
- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for Trusted Advisor

When you designate a member account to be a delegated administrator for the organization, users and roles from the designated account can manage the AWS account metadata for other member accounts in the organization. If you don't enable a delegated admin account, then these tasks can be performed only by the organization's management account. This helps you to separate management of the organization from management of your account details.

Minimum permissions

Only a user or role in the Organizations management account can configure a member account as a delegated administrator for Trusted Advisor in the organization

For instruction about enabling a delegated administrator account for Trusted Advisor, see Register delegated administrators in the *Support User Guide*.

AWS CLI, AWS API

If you want to configure a delegated administrator account using the AWS CLI or one of the AWS SDKs, you can use the following commands:

- AWS CLI:

```
$  aws organizations register-delegated-administrator \
    --account-id 123456789012 \
    --service-principal reporting.trustedadvisor.amazonaws.com
```

- AWS SDK: Call the Organizations `RegisterDelegatedAdministrator` operation and the member account's ID number and identify the account service principal `account.amazonaws.com` as parameters.

Disabling a delegated administrator for Trusted Advisor

You can remove the delegated administrator using either the Trusted Advisor console, or by using the the Organizations `DeregisterDelegatedAdministrator` CLI or SDK operation. For information on how to disable the delegated admin Trusted Advisor account using the Trusted Advisor console, see Deregister delegated administrators in the *Support user guide*.

AWS Well-Architected Tool and AWS Organizations

The AWS Well-Architected Tool helps you document the state of your workloads and compares them to the latest AWS architectural best practices.

Using AWS Well-Architected Tool with Organizations enables both AWS Well-Architected Tool and Organizations customers to simplify the process of sharing AWS Well-Architected Tool resources with other members of their organization.

For more information, see Sharing your AWS Well-Architected Tool resources in the *AWS Well-Architected Tool User Guide*.

Use the following information to help you integrate AWS Well-Architected Tool with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows AWS WA Tool to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between AWS WA Tool and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForWellArchitected`

The service role policy is `AWSWellArchitectedOrganizationsServiceRolePolicy`

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by AWS WA Tool grant access to the following service principals:

- `wellarchitected.amazonaws.com`

Enabling trusted access with AWS WA Tool

Allows the updating of AWS WA Tool to reflect hierarchical changes in an organization.

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

You can enable trusted access using either the AWS Well-Architected Tool console or the AWS Organizations console.

> **Important**
>
> We strongly recommend that whenever possible, you use the AWS Well-Architected Tool console or tools to enable integration with Organizations. This lets AWS Well-Architected Tool perform any configuration that it requires, such as creating resources needed by the service. Proceed with these steps only if you can't enable integration using the tools provided by AWS Well-Architected Tool. For more information, see this note.

If you enable trusted access by using the AWS Well-Architected Tool console or tools then you don't need to complete these steps.

To enable trusted access using the AWS WA Tool console

See Sharing your AWS Well-Architected Tool resources in the *AWS Well-Architected Tool User Guide*.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. In the navigation pane, choose **Services**.
3. Choose **AWS Well-Architected Tool** in the list of services.
4. Choose **Enable trusted access**.
5. In the **Enable trusted access for AWS Well-Architected Tool** dialog box, type **enable** to confirm, and then choose **Enable trusted access**.
6. If you are the administrator of only AWS Organizations, tell the administrator of AWS Well-Architected Tool that they can now enable that service to work with AWS Organizations from the service console .

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

Use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  Run the following command to enable AWS Well-Architected Tool as a trusted service with Organizations.

```
$ aws organizations enable-aws-service-access \
```

```
    --service-principal wellarchitected.amazonaws.com
```

This command produces no output when successful.

- AWS API: EnableAWSServiceAccess

Disabling trusted access with AWS WA Tool

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

You can disable trusted access using either the AWS Well-Architected Tool or the AWS Organizations tools.

> **Important**
>
> We strongly recommend that whenever possible, you use the AWS Well-Architected Tool console or tools to disable integration with Organizations. This lets AWS Well-Architected Tool perform any clean up that it requires, such as deleting resources or access roles that are no longer needed by the service. Proceed with these steps only if you can't disable integration using the tools provided by AWS Well-Architected Tool.
> If you disable trusted access by using the AWS Well-Architected Tool console or tools then you don't need to complete these steps.

To disable trusted access using the AWS WA Tool console

See Sharing your AWS Well-Architected Tool resources in the *AWS Well-Architected Tool User Guide*.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

Run the following command to disable AWS Well-Architected Tool as a trusted service with Organizations.

```
$ aws organizations disable-aws-service-access \
    --service-principal wellarchitected.amazonaws.com
```

This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Amazon VPC IP Address Manager (IPAM) and AWS Organizations

Amazon VPC IP Address Manager (IPAM) is a VPC feature that makes it easier for you to plan, track, and monitor IP addresses for your AWS workloads.

Using AWS Organizations allows you to monitor IP address usage throughout your organization and share IP address pools across member accounts.

For more information, see Integrate IPAM with AWS Organizations in the *Amazon VPC IPAM User Guide*.

Use the following information to help you integrate Amazon VPC IP Address Manager (IPAM) with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account and each member account when you integrate IPAM with AWS Organizations either by using the IPAM console or using IPAM's `EnableIpamOrganizationAdminAccount` API.

- `AWSServiceRoleForIPAM`

For more information, see Service-linked roles for IPAM in the *Amazon VPC IPAM User Guide*.

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by IPAM grant access to the following service principals:

- `ipam.amazonaws.com`

To enable trusted access with IPAM

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

> **Note**
>
> When you designate a delegated administrator for IPAM it automatically enables trusted access for IPAM for your organization.
> IPAM requires trusted access to AWS Organizations before you can designate a member account to be the delegated administrator for this service for your organization.

You can enable trusted access using only Amazon VPC IP Address Manager (IPAM) tools.

If you integrate IPAM with AWS Organizations using the IPAM console or using the IPAM `EnableIpamOrganizationAdminAccount` API, you automatically grant trusted access to IPAM. Granting trusted access creates the service-linked role `AWSServiceRoleForIPAM` in the management account and in all of the member accounts in the organization. IPAM uses the service-linked role to monitor CIDRs associated with EC2 networking resources in your organization and to store metrics related to IPAM in Amazon CloudWatch. For more information, see Service-linked roles for IPAM in the *Amazon VPC IPAM User Guide*.

For instructions about enabling trusted access, see Integrate IPAM with AWS Organizations in the *Amazon VPC IPAM User Guide*.

> **Note**
>
> You can't enable trusted access with IPAM using the AWS Organizations console or with the `EnableAWSServiceAccess` API.

To disable trusted access with IPAM

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

Only an administrator in the AWS Organizations management account can disable trusted access with IPAM using the AWS Organizations `disable-aws-service-access` API.

For information about disabling IPAM account permissions and deleting the service-linked role, see Service-linked roles for IPAM in the *Amazon VPC IPAM User Guide*.

You can disable trusted access by running a Organizations AWS CLI command, or by calling an Organizations API operation in one of the AWS SDKs.

AWS CLI, AWS API

To disable trusted service access using the Organizations CLI/SDK

Use the following AWS CLI commands or API operations to disable trusted service access:

- AWS CLI: disable-aws-service-access

  Run the following command to disable Amazon VPC IP Address Manager (IPAM) as a trusted service with Organizations.

  ```
  $ aws organizations disable-aws-service-access \
      --service-principal ipam.amazonaws.com
  ```

  This command produces no output when successful.

- AWS API: DisableAWSServiceAccess

Enabling a delegated administrator account for IPAM

The delegated administrator account for IPAM is responsible for creating the IPAM and IP address pools, managing and monitoring IP address usage in the organization, and sharing IP address pools across member accounts. For more information, see Integrate IPAM with AWS Organizations in the *Amazon VPC IPAM User Guide*.

Only an administrator in the organization management account can configure a delegated administrator for IPAM.

You can specify a delegated administrator account from the IPAM console, or by using the `enable-ipam-organization-admin-account` API. For more information, see enable-ipam-organization-admin-account in the *AWS AWS CLI Command Reference*.

Minimum permissions

Only a user or role in the Organizations management account can configure a member account as a delegated administrator for IPAM in the organization

To configure a delegated administrator using the IPAM console, see Integrate IPAM with AWS Organizations in the *Amazon VPC IPAM User Guide*.

Disabling a delegated administrator for IPAM

Only an administrator in the organization management account can configure a delegated administrator for IPAM.

To remove a delegated administrator using the AWS AWS CLI, see disable-ipam-organization-admin-account in the *AWS AWS CLI Command Reference*.

To disable the delegated admin IPAM account using the IPAM console, see Integrate IPAM with AWS Organizations in the *Amazon VPC IPAM User Guide*.

Amazon VPC Reachability Analyzer and AWS Organizations

Reachability Analyzer is a configuration analysis tool that enables you to perform connectivity testing between a source resource and a destination resource in your virtual private clouds (VPCs).

Using AWS Organizations with Reachability Analyzer allows you to trace paths across accounts in your organizations.

For more information, see Manage delegated administrator accounts in Reachability Analyzer in the *Reachability Analyzer user guide*.

Use the following information to help you integrate Reachability Analyzer with AWS Organizations.

Service-linked roles created when you enable integration

The following service-linked role is automatically created in your organization's management account when you enable trusted access. This role allows Reachability Analyzer to perform supported operations within your organization's accounts in your organization.

You can delete or modify this role only if you disable trusted access between Reachability Analyzer and Organizations, or if you remove the member account from the organization.

- `AWSServiceRoleForReachabilityAnalyzer`

For more information, see Cross-account analyses for Reachability Analyzer in the *Reachability Analyzer user guide*.

Service principals used by the service-linked roles

The service-linked role in the previous section can be assumed only by the service principals authorized by the trust relationships defined for the role. The service-linked roles used by Reachability Analyzer grant access to the following service principals:

- `reachabilityanalyzer.networkinsights.amazonaws.com`

To enable trusted access with Reachability Analyzer

For information about the permissions needed to enable trusted access, see Permissions required to enable trusted access.

When you designate a delegated administrator for Reachability Analyzer it automatically enables trusted access for Reachability Analyzer for your organization.

Reachability Analyzer requires trusted access to AWS Organizations before you can designate a member account to be the delegated administrator for this service for your organization.

Important

- You can enable trusted access using either the Reachability Analyzer console or the Organizations console. However, we strongly recommend that you use the Reachability Analyzer console or the `EnableMultiAccountAnalysisForAwsOrganization` API to enable integration with Organizations. This lets Reachability Analyzer perform any configuration that it requires, such as creating resources needed by the service.
- Granting trusted access creates the service-linked role `AWSServiceRoleForReachabilityAnalyzer` in the management account and in all of the member accounts in the organization. Reachability Analyzer uses the service-linked role to allow management, and the delegated administrator to run connectivity analyses between any resources in the organization. Reachability Analyzer is able to take snapshots of the networking elements of the accounts in an organization in order to answer connectivity queries.

- For more information, and for instructions on enabling trusted access through Reachability Analyzer, see Cross-account analyses for Reachability Analyzer in the *Reachability Analyzer user guide*.

You can enable trusted access by using either the AWS Organizations console, by running a AWS CLI command, or by calling an API operation in one of the AWS SDKs.

AWS Management Console

To enable trusted service access using the Organizations console

1. Sign in to the AWS Organizations console. You must sign in as an IAM user, assume an IAM role, or sign in as the root user (not recommended) in the organization's management account.
2. On the **Services** page, find the row for **VPC Reachability Analyzer**, choose the service's name, and then choose **Enable trusted access**.
3. In the confirmation dialog box, enable **Show the option to enable trusted access**, enter **`enable`** in the box, and then choose **Enable trusted access**.
4. If you are the administrator of only AWS Organizations, tell the administrator of Reachability Analyzer that they can now enable that service using its console to work with AWS Organizations.

AWS CLI, AWS API

To enable trusted service access using the OrganizationsCLI/SDK

You can use the following AWS CLI commands or API operations to enable trusted service access:

- AWS CLI: enable-aws-service-access

  You can run the following command to enable Reachability Analyzer as a trusted service with Organizations.

```
$ aws organizations enable-aws-service-access \
    --service-principal reachabilityanalyzer.networkinsights.amazonaws.com
```

This command produces no output when successful.

- AWS API: EnableAWSServiceAccess

To disable trusted access with Reachability Analyzer

For information about the permissions needed to disable trusted access, see Permissions required to disable trusted access.

You can disable trusted access using either the Reachability Analyzer console (recommended), or the Organizations console. To disable trusted access using the Reachability Analyzer console, see Cross-account analyses for Reachability Analyzer in the *Reachability Analyzer user guide*.

Enabling a delegated administrator account for Reachability Analyzer

The delegated administrator account is able to run connectivity analyses across any of the resources in the organization. For more information, see Integrate Reachability Analyzer with AWS Organizations in the *Reachability Analyzer user guide*.

Only an administrator in the organization management account can configure a delegated administrator for Reachability Analyzer.

You can specify a delegated administrator account from the Reachability Analyzer console, or by using the `RegisterDelegatedAdministrator` API. For more information, see RegisterDelegatedAdministrator in the *Organizations Command Reference*.

Minimum permissions

Only a user or role in the Organizations management account can configure a member account as a delegated administrator for Reachability Analyzer in the organization

To configure a delegated administrator using the Reachability Analyzer console, see Integrate Reachability Analyzer with AWS Organizations in the *Reachability Analyzer user guide*.

Disabling a delegated administrator for Reachability Analyzer

Only an administrator in the organization management account can configure a delegated administrator for Reachability Analyzer.

You can remove the delegated administrator using either the Reachability Analyzer console or API, or by using the Organizations `DeregisterDelegatedAdministrator` CLI or SDK operation.

To disable the delegated admin Reachability Analyzer account using the Reachability Analyzer console, see Cross-account analyses for Reachability Analyzer in the *Reachability Analyzer user guide*.

Delegated administrator for AWS services that work with Organizations

We recommend that you use the AWS Organizations management account and its users and roles only for tasks that must be performed by that account. We also recommend that you store your AWS resources in other member accounts in the organization and keep them out of the management account. This is because security features like Organizations service control policies (SCPs) do not restrict users or roles in the management account. Separating your resources from your management account can also help you understand the charges on your invoices.

Many AWS services that integrate with Organizations enable you to reduce the usage of the management account. These services enable you to register one or more member accounts as administrators that can manage all of the organization's accounts used in the service. These accounts are called *delegated administrators* for that specific service. By registering a member account as a delegated administrator for an AWS service you enable that account to have some administrative permissions for that service, as well as permissions for Organizations read-only actions.

Before you register an account as a delegated administrator for a service:

- Confirm that the service supports delegated administrators. See the table in AWS services that you can use with AWS Organizations to learn which services support delegated administrators.
- Enable trusted access for that service.

> **Note**
>
> To learn how to enable a delegated administrator a service, reference the table in AWS services that you can use with AWS Organizations and select the **Learn more** link in the **Supports Delegated Administrator** column for that service.

Permissions granted to delegated administrator accounts

Each service-specific delegated administrator account has permissions granted by that service. To learn more, reference the table in AWS services that you can use with AWS Organizations and select the **Learn more** link in the **Supports Delegated Administrator** column for that service.

A delegated administrator account also has these read-only permissions:

- `DescribeAccount`
- `DescribeCreateAccountStatus`
- `DescribeEffectivePolicy`
- `DescribeHandshake`
- `DescribeOrganization`
- `DescribeOrganizationalUnit`
- `DescribePolicy`
- `DescribeResourcePolicy`
- `ListAccounts`
- `ListAccountsForParent`
- `ListAWSServiceAccessForOrganization`
- `ListChildren`
- `ListCreateAccountStatus`
- `ListDelegatedAdministrators`
- `ListDelegatedServicesForAccount`
- `ListHandshakesForAccount`
- `ListHandshakesForOrganization`
- `ListOrganizationalUnitsForParent`
- `ListParents`
- `ListPolicies`
- `ListPoliciesForTarget`
- `ListRoots`
- `ListTagsForResource`
- `ListTargetsForPolicy`

These permissions enable you to view, but not change these console items:

- Organization structure, all accounts and OUs, and organizational policies
- Memberships
- All accounts and OUs.
- Organizational policies

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 PrivateLink for AWS Organizations
- Identity and Access Management for AWS Organizations
- Logging and monitoring in AWS Organizations
- Compliance validation for AWS Organizations
- Resilience in AWS Organizations
- Infrastructure security in AWS Organizations

AWS PrivateLink for AWS Organizations

With AWS PrivateLink for AWS Organizations, you can access the AWS Organizations service from within the Virtual Private Cloud (VPC) without having to cross the public internet.

Amazon VPC lets you launch AWS resources in a custom virtual network. You can use a VPC to control your network settings, such as the IP address range, subnets, route tables, and network gateways. For more information about VPCs, see the *Amazon VPC User Guide*.

To connect your Amazon VPC to AWS Organizations, you must first define an interface VPC endpoint (interface endpoints). Interface endpoints are represented by one or more elastic network interfaces (ENIs) that are assigned private IP addresses from subnets in your VPC. Requests from your VPC to AWS Organizations over interface endpoints stay on the Amazon network.

For general information about interface endpoints, see Access an AWS service using an interface VPC endpoint in the *Amazon VPC User Guide*.

Topics

- Limitations and restrictions of AWS PrivateLink for AWS Organizations
- Creating a VPC endpoint for AWS Organizations
- Creating a VPC endpoint policy for AWS Organizations

Limitations and restrictions of AWS PrivateLink for AWS Organizations

VPC limitations apply to AWS PrivateLink for AWS Organizations. For more information, see Access an AWS service using an interface VPC endpoint and AWS PrivateLink quotas in the *Amazon VPC User Guide*. In addition, the following restrictions apply:

- Only available in the `us-east-1` region
- Doesn't support Transport Layer Security (TLS) 1.1

Creating a VPC endpoint for AWS Organizations

You can create an AWS Organizations endpoint in your VPC using the Amazon VPC Console, the AWS Command Line Interface (AWS CLI) or AWS CloudFormation.

For information about creating and configuring an endpoint using the Amazon VPC console or the AWS CLI, see Create a VPC endpoint in the *Amazon VPC User Guide*. For information about creating and configuring an endpoint using AWS CloudFormation, see the AWS::EC2::VPCEndpoint resource in the *AWS CloudFormation User Guide*.

When you create an AWS Organizations endpoint, use the following as the service name:

```
com.amazonaws.us-east-1.organizations
```

If you require FIPS 140-2 validated cryptographic modules when accessing AWS, use the following AWS Organizations FIPS service name:

```
com.amazonaws.us-east-1.organizations-fips
```

Creating a VPC endpoint policy for AWS Organizations

You can attach an endpoint policy to your VPC endpoint that controls access to Organizations. The policy specifies the following information:

- The principal that can perform actions.
- The actions that can be performed.
- The resources on which actions can be performed.

For more information, see Control access to VPC endpoints using endpoint policies in the *Amazon VPC User Guide*.

Example: VPC endpoint policy for AWS Organizations actions

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "Organizations:DescribeAccount"
         ],
         "Resource":"*"
      }
   ]
}
```

Identity and Access Management for AWS Organizations

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Organizations resources. IAM is an AWS service that you can use with no additional charge.

Topics

- Audience
- Authenticating with identities
- Managing access using policies
- How AWS Organizations works with IAM
- Managing access permissions for an organization with AWS Organizations
- Identity-based policy examples for AWS Organizations
- Resource-based policy examples for AWS Organizations
- AWS managed policies for AWS Organizations
- Attribute-based access control with tags for AWS Organizations
- Troubleshooting AWS Organizations identity and access

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in Organizations.

Service user – If you use the Organizations service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Organizations features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Organizations, see Troubleshooting AWS Organizations identity and access.

Service administrator – If you're in charge of Organizations resources at your company, you probably have full access to Organizations. It's your job to determine which Organizations features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Organizations, see How AWS Organizations works with IAM.

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Organizations. To view example Organizations identity-based policies that you can use in IAM, see Identity-based policy examples for AWS Organizations.

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Multi-factor authentication in the *AWS IAM Identity Center User Guide* and AWS Multi-factor authentication in IAM in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For

the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. To temporarily assume an IAM role in the AWS Management Console, you can switch from a user to an IAM role (console). You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Methods to assume a role in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Create a role for a third-party identity provider (federation) in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

- **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*.
- **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Use an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choose between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – RCPs are JSON policies that you can use to set the maximum available permissions for resources in your accounts without updating the IAM policies attached to each resource that you own. The RCP limits permissions for resources in member accounts and can impact the effective permissions for identities, including the AWS account root user, regardless of whether they belong to your organization. For more information about Organizations and RCPs, including a list of AWS services that support RCPs, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How AWS Organizations works with IAM

Before you use IAM to manage access to Organizations, learn what IAM features are available to use with Organizations.

IAM feature	Organizations support
Identity-based policies	Yes
Resource-based policies	Yes
Policy actions	Yes
Policy resources	Yes
Policy condition keys (service-specific)	Yes
ACLs	No
ABAC (tags in policies)	Yes
Temporary credentials	No
Forward access sessions (FAS)	Yes
Service roles	Yes
Service-linked roles	Yes

To get a high-level view of how Organizations and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for Organizations

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for Organizations

To view examples of Organizations identity-based policies, see Identity-based policy examples for AWS Organizations.

Resource-based policies within Organizations

Supports resource-based policies: Yes

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access

to a principal in the same account, no additional identity-based policy is required. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

The Organizations service supports only one type of resource-based policy called a *resource-based delegation policy*, which specifies which member accounts can perform actions on policies. You can add multiple statements in the policy to denote a different set of permissions to member accounts.

For more information, see Delegated administrator for AWS Organizations.

Resource-based policy examples within Organizations

To view examples of Organizations resource-based policies, see Resource-based policy examples for AWS Organizations,

Policy actions for Organizations

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Organizations actions, see Actions defined by AWS Organizations in the *Service Authorization Reference*.

Policy actions in Organizations use the following prefix before the action:

```
organizations
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
```

```
      "organizations:action1",
      "organizations:action2"
         ]
```

To view examples of Organizations identity-based policies, see Identity-based policy examples for AWS Organizations.

Policy resources for Organizations

Supports policy resources: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of Organizations resource types and their ARNs, see Resources defined by AWS Organizations in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by AWS Organizations.

To view examples of Organizations identity-based policies, see Identity-based policy examples for AWS Organizations.

Policy condition keys for Organizations

Supports service-specific policy condition keys: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of Organizations condition keys, see Condition keys for AWS Organizations in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by AWS Organizations.

To view examples of Organizations identity-based policies, see Identity-based policy examples for AWS Organizations.

ACLs in Organizations

Supports ACLs: No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

ABAC with Organizations

Supports ABAC (tags in policies): Yes

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see Define permissions with ABAC authorization in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using temporary credentials with Organizations

Supports temporary credentials: No

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switch from a user to an IAM role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

Forward access sessions for Organizations

Supports forward access sessions (FAS): Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a

different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

Service roles for Organizations

Supports service roles: Yes

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*.

> **Warning**
>
> Changing the permissions for a service role might break Organizations functionality. Edit service roles only when Organizations provides guidance to do so.

Service-linked roles for Organizations

Supports service-linked roles: Yes

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing service-linked roles, see AWS services that work with IAM. Find a service in the table that includes a Yes in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

Managing access permissions for an organization with AWS Organizations

All AWS resources, including the roots, OUs, accounts, and policies in an organization, are owned by an AWS account, and permissions to create or access a resource are governed by permissions

policies. For an organization, its management account owns all resources. An account administrator can control access to AWS resources by attaching permissions policies to IAM identities (users, groups, and roles).

> **Note**
>
> An *account administrator* (or administrator user) is a user with administrator permissions. For more information, see Security best practices in IAM in the *AWS Account Management Reference Guide*.

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

By default, IAM users, groups, and roles have no permissions. As an administrator in the management account of an organization, you can perform administrative tasks or delegate administrator permissions to other IAM users or roles in the management account. To do this, you attach an IAM permissions policy to an IAM user, group, or role. By default, a user has no permissions at all; this is sometimes called an *implicit deny*. The policy overrides the implicit deny with an *explicit allow* that specifies which actions the user can perform, and which resources they can perform the actions on. If the permissions are granted to a role, users in other accounts in the organization can assume that role.

AWS Organizations resources and operations

This section discusses how AWS Organizations concepts map to their IAM-equivalent concepts.

Resources

In AWS Organizations, you can control access to the following resources:

- The root and the OUs that make up the hierarchical structure of an organization
- The accounts that are members of the organization
- The policies that you attach to the entities in the organization
- The handshakes that you use to change the state of the organization

Each of these resources has a unique Amazon Resource Name (ARN) associated with it. You control access to a resource by specifying its ARN in the `Resource` element of an IAM permission policy.

For a complete list of the ARN formats for resources that are used in AWS Organizations, see Resources types defined by AWS Organizations in the *Service Authorization Reference*.

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 Actions defined by organizations in the *Service Authorization Reference*.

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 in the `Condition` element.

  > **Note**
  >
  > This global condition also applies to the management account of an organization.

  For more information, see the description of `PrincipalOrgID` in AWS global condition context keys in the *IAM User Guide*.
- `aws:PrincipalOrgPaths` – Use this condition key to match members of a specific organization root, an OU, or its children. The `aws:PrincipalOrgPaths` condition key returns true when the principal (root user, IAM user, or role) making the request is in the specified organization path. A path is a text representation of the structure of an AWS Organizations entity. For more information about paths, see Understand the AWS Organizations entity path in the *IAM User*

Guide. For more information about using this condition key, see aws:PrincipalOrgPaths in the *IAM User Guide*.

For example, the following condition element matches for members of either of two OUs in the same organization.

```
            "Condition": {
                "ForAnyValue:StringLike": {
                    "aws:PrincipalOrgPaths": [
                        "o-a1b2c3d4e5/r-f6g7h8i9j0example/ou-def0-awsbbbbb/",
                        "o-a1b2c3d4e5/r-f6g7h8i9j0example/ou-jkl0-awsddddd/"
                    ]
                }
            }
```

- `organizations:PolicyType` – You can use this condition key to restrict the Organizations policy-related API operations to work on only Organizations policies of the specified type. You can apply this condition key to any policy statement that includes an action that interacts with Organizations policies.

  You can use the following values with this condition key:

  - `SERVICE_CONTROL_POLICY`
  - `RESOURCE_CONTROL_POLICY`
  - `DECLARATIVE_POLICY_EC2`
  - `BACKUP_POLICY`
  - `TAG_POLICY`
  - `CHATBOT_POLICY`
  - `AISERVICES_OPT_OUT_POLICY`

  For example, the following example policy allows the user to perform any Organizations operation. However, if the user performs an operation that takes a policy argument, the operation is allowed only if the specified policy is a tagging policy. The operation fails if the user specifies any other type of policy.

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

```
            "Sid": "IfTaggingAPIThenAllowOnOnlyTaggingPolicies",
            "Effect": "Allow",
            "Action": "organizations:*",
            "Resource": "*",
            "Condition": {
                "StringLikeIfExists": {
                    "organizations:PolicyType": [ "TAG_POLICY" ]
                }
            }
        }
    ]
}
```

- `organizations:ServicePrincipal` – Available as a condition if you use the EnableAWSServiceAccess or DisableAWSServiceAccess operations to enable or disable trusted access with other AWS services. You can use `organizations:ServicePrincipal` to restrict requests that those operations make to a list of approved service principal names.

  For example, the following policy allows the user to specify only AWS Firewall Manager when enabling and disabling trusted access with AWS Organizations.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowOnlyAWSFirewallIntegration",
            "Effect": "Allow",
            "Action": [
                "organizations:EnableAWSServiceAccess",
                "organizations:DisableAWSServiceAccess"
            ],
            "Resource": "*",
            "Condition": {
                "StringLikeIfExists": {
                    "organizations:ServicePrincipal": [ "fms.amazonaws.com" ]
                }
            }
        }
    ]
}
```

For a list of all of the AWS Organizations–specific condition keys that can be used as permissions in an IAM policy, see Condition keys for AWS Organizations in the *Service Authorization Reference*.

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 user, an IAM user, or an IAM role) that authenticates the resource creation request. For an organization, that is ***always*** the management account. You can't call most operations that create or access organization resources from the member accounts. The following examples illustrate how this works:

- If you use the root account credentials of your management account to create an OU, your management account is the owner of the resource. (In AWS Organizations, the resource is the OU).
- If you create an IAM user in your management account and grant permissions to create an OU to that user, the user can create an OU. However, the management account, to which the user belongs, owns the OU resource.
- If you create an IAM role in your management account with permissions to create an OU, anyone who can assume the role can create an OU. The management account, to which the role (not the assuming user) belongs, owns the OU resource.

Managing access to resources

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

> **Note**
>
> This section discusses using IAM in the context of AWS Organizations. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see the IAM User Guide. For information about IAM policy syntax and descriptions, see the IAM JSON policy reference in the *IAM User Guide*.

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

Topics

- Identity-based permission policies (IAM policies)

Identity-based permission policies (IAM policies)

You can attach policies to IAM identities to allow those identities to perform operations on AWS resources. For example, you can do the following:

- **Attach a permissions policy to a user or a group in your account** – To grant a user permissions to create an AWS Organizations resource, such as a service control policy (SCP) or an OU, you can attach a permissions policy to a user or a group that the user belongs to. The user or group must be in the organization's management account.
- **Attach a permissions policy to a role (grant cross-account permissions)** – You can attach an identity-based permissions policy to an IAM role to grant cross-account access to an organization. For example, the administrator in the management account can create a role to grant cross-account permissions to a user in a member account as follows:
  1. The management account administrator creates an IAM role and attaches a permissions policy to the role that grants permissions to the organization's resources.
  2. The management account administrator attaches a trust policy to the role that identifies the member account ID as the `Principal` who can assume the role.
  3. The member account administrator can then delegate permissions to assume the role to any users in the member account. Doing this allows users in the member account to create or access resources in the management account and the organization. The principal in the trust policy can also be an AWS service principal if you want to grant permissions to an AWS service to assume the role.

  For more information about using IAM to delegate permissions, see Access Management in the *IAM User Guide*.

The following are examples of policies that allows a user to perform the `CreateAccount` action in your organization.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"Stmt1OrgPermissions",
```

```
         "Effect":"Allow",
         "Action":[
            "organizations:CreateAccount"
         ],
         "Resource":"*"
      }
   ]
}
```

You can also provide a partial ARN in the `Resource` element of the policy to indicate the type of resource.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowCreatingAccountsOnResource",
         "Effect":"Allow",
         "Action":"organizations:CreateAccount",
         "Resource":"arn:aws:organizations::*:account/*"
      }
   ]
}
```

You can also deny the creation of accounts that do not include specific tags to the account being created.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"DenyCreatingAccountsOnResourceBasedOnTag",
         "Effect":"Deny",
         "Action":"organizations:CreateAccount",
         "Resource":"*",
         "Condition":{
            "StringEquals":{
               "aws:ResourceTag/key":"value"
            }
         }
      }
   ]
```

```
}
```

For more information about users, groups, roles, and permissions, see IAM identities (users, user groups, and roles) in the *IAM User Guide*.

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).

To learn more about IAM policy syntax and descriptions, see the IAM JSON policy reference in the *IAM User Guide*.

Identity-based policy examples for AWS Organizations

By default, users and roles don't have permission to create or modify Organizations resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Create IAM policies (console) in the *IAM User Guide*.

For details about actions and resource types defined by Organizations, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for AWS Organizations in the *Service Authorization Reference*.

Topics

- Policy best practices
- Using the Organizations console
- Allow users to view their own permissions
- Granting full admin permissions to a user
- Granting limited access by actions
- Granting access to specific resources
- Granting the ability to enable trusted access to limited service principals

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Organizations resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the Organizations console

To access the AWS Organizations console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Organizations resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

To ensure that users and roles can still use the Organizations console, also attach the Organizations `AWSOrganizationsFullAccess` or `AWSOrganizationsReadOnlyAccess` AWS managed policy to the entities. For more information, see Adding permissions to a user in the *IAM User Guide*.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

Granting full admin permissions to a user

You can create an IAM policy that grants full AWS Organizations administrator permissions to an IAM user in your organization. You can do this using the JSON policy editor in the IAM console.

To use the JSON policy editor to create a policy

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane on the left, choose **Policies**.

   If this is your first time choosing **Policies**, the **Welcome to Managed Policies** page appears. Choose **Get Started**.
3. At the top of the page, choose **Create policy**.
4. In the **Policy editor** section, choose the **JSON** option.
5. Enter the following JSON policy document:

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "organizations:*",
        "Resource": "*"
    }
}
```

6. Choose **Next**.

> **Note**
>
> You can switch between the **Visual** and **JSON** editor options anytime. However, if you make changes or choose **Next** in the **Visual** editor, IAM might restructure your policy to optimize it for the visual editor. For more information, see Policy restructuring in the *IAM User Guide*.

7. On the **Review and create** page, enter a **Policy name** and a **Description** (optional) for the policy that you are creating. Review **Permissions defined in this policy** to see the permissions that are granted by your policy.
8. Choose **Create policy** to save your new policy.

To learn more about creating an IAM policy, see Creating IAM policies in the *IAM User Guide*.

Granting limited access by actions

If you want to grant limited permissions instead of full permissions, you can create a policy that lists individual permissions that you want to allow in the `Action` element of the IAM permissions policy. As shown in the following example, you can use wildcard (*) characters to grant only the `Describe*` and `List*` permissions, essentially providing read-only access to the organization.

> **Note**
>
> In a service control policy (SCP), the wildcard (*) character in an `Action` element can be used only by itself or at the end of the string. It can't appear at the beginning or middle of the string. Therefore, `"servicename:action*"` is valid, but `"servicename:*action"` and `"servicename:some*action"` are both invalid in SCPs.

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

For a list of all the permissions that are available to assign in an IAM policy, see Actions defined by AWS Organizations in the *Service Authorization Reference*.

Granting access to specific resources

In addition to restricting access to specific actions, you can restrict access to specific entities in your organization. The `Resource` elements in the examples in the preceding sections both specify the wildcard character ("*"), which means "any resource that the action can access." Instead, you can replace the "*" with the Amazon Resource Name (ARN) of specific entities to which you want to allow access.

Example: Granting permissions to a single OU

The first statement of the following policy allows an IAM user read access to the entire organization, but the second statement allows the user to perform AWS Organizations administrative actions only within a single, specified organizational unit (OU). This does not extend to any child OUs. No billing access is granted. Note that this doesn't give you administrative access to the AWS accounts in the OU. It grants only permissions to perform AWS Organizations operations on the accounts within the specified OU:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "organizations:Describe*",
        "organizations:List*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "organizations:*",
      "Resource": "arn:aws:organizations::<masterAccountId>:ou/o-<organizationId>/ou-
<organizationalUnitId>"
    }
  ]
}
```

You get the IDs for the OU and the organization from the AWS Organizations console or by calling the `List*` APIs. The user or group that you apply this policy to can perform any action (`"organizations:*"`) on any entity that is directly contained in the specified OU. The OU is identified by the Amazon Resource Name (ARN).

For more information about the ARNs for various resources, see Resources types defined by AWS Organizations in the *Service Authorization Reference*.

Granting the ability to enable trusted access to limited service principals

You can use the `Condition` element of a policy statement to further limit the circumstances where the policy statement matches.

Example: Granting permissions to enable trusted access to one specified service

The following statement shows how you can restrict the ability to enable trusted access to only those services that you specify. If the user tries to call the API with a different service principal than the one for AWS IAM Identity Center, this policy doesn't match and the request is denied:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "organizations:EnableAWSServiceAccess",
            "Resource": "*",
            "Condition": {
                "StringEquals" : {
                    "organizations:ServicePrincipal" : "sso.amazonaws.com"
                }
            }
        }
    ]
}
```

For more information about the ARNs for various resources, see Resources types defined by AWS Organizations in the *Service Authorization Reference*.

Resource-based policy examples for AWS Organizations

The following code examples show how you can use resource-based delegation policies. For more information, see Delegated administrator for AWS Organizations.

Topics

- Example: View organization, OUs, accounts, and policies
- Example: Create, read, update, and delete policies
- Example: Tag and untag policies
- Example: Attach policies to a single OU or account
- Example: Consolidated permissions to manage an organization's backup policies

Example: View organization, OUs, accounts, and policies

Before delegating the management of policies, you must delegate the permissions to navigate the structure of an organization and see the organizational units (OUs), accounts, and the policies attached to them.

This example shows how you might include these permissions in your resource-based delegation policy for the member account, *AccountId*.

> **Important**
>
> It is advisable that you include permissions to only the minimum required actions as shown in the example, although it's possible to delegate any Organizations read-only action using this policy.

This example delegation policy grants the permissions necessary to complete actions programmatically from the AWS API or AWS CLI. To use this delegation policy, replace the AWS placeholder text for *AccountId* with your own information. Then, follow the directions in Delegated administrator for AWS Organizations.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DelegatingNecessaryDescribeListActions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::AccountId:root"
      },
      "Action": [
        "organizations:DescribeOrganization",
        "organizations:DescribeOrganizationalUnit",
```

```
        "organizations:DescribeAccount",
        "organizations:DescribePolicy",
        "organizations:DescribeEffectivePolicy",
        "organizations:ListRoots",
        "organizations:ListOrganizationalUnitsForParent",
        "organizations:ListParents",
        "organizations:ListChildren",
        "organizations:ListAccounts",
        "organizations:ListAccountsForParent",
        "organizations:ListPolicies",
        "organizations:ListPoliciesForTarget",
        "organizations:ListTargetsForPolicy",
        "organizations:ListTagsForResource"
      ],
      "Resource": "*"
    }
  ]
}
```

Example: Create, read, update, and delete policies

You can create a resource-based delegation policy that allows the management account to delegate `create`, `read`, `update`, and `delete` actions for any policy type. This example shows how you might delegate these actions for service control policies to the member account, *`MemberAccountId`*. The two resources shown in the example grant access to customer managed and AWS managed service control policies respectively.

> **Important**
>
> This policy allows delegated administrators to perform specified actions on policies created by any account in the organization, including the management account.
> It doesn't allow delegated administrators to attach or detach policies because it doesn't include the permissions required to perform `organizations:AttachPolicy` and `organizations:DetachPolicy` actions.

This example delegation policy grants the permissions necessary to complete actions programmatically from the AWS API or AWS CLI. Replace the AWS placeholder text for *`MemberAccountId`*, *`ManagementAccountId`*, and *`OrganizationId`* with your own information. Then, follow the directions in Delegated administrator for AWS Organizations.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DelegatingNecessaryDescribeListActions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::MemberAccountId:root"
      },
      "Action": [
        "organizations:DescribeOrganization",
        "organizations:DescribeOrganizationalUnit",
        "organizations:DescribeAccount",
        "organizations:DescribePolicy",
        "organizations:DescribeEffectivePolicy",
        "organizations:ListRoots",
        "organizations:ListOrganizationalUnitsForParent",
        "organizations:ListParents",
        "organizations:ListChildren",
        "organizations:ListAccounts",
        "organizations:ListAccountsForParent",
        "organizations:ListPolicies",
        "organizations:ListPoliciesForTarget",
        "organizations:ListTargetsForPolicy",
        "organizations:ListTagsForResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringLikeIfExists": {
          "organizations:PolicyType": "SERVICE_CONTROL_POLICY"
        }
      }
    },
    {
      "Sid": "DelegatingMinimalActionsForSCPs",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::MemberAccountId:root"
      },
      "Action": [
        "organizations:CreatePolicy",
        "organizations:DescribePolicy",
        "organizations:UpdatePolicy",
```

```
        "organizations:DeletePolicy"
      ],
      "Resource": [
        "arn:aws:organizations::ManagementAccountId:policy/o-OrganizationId/
service_control_policy/*",
        "arn:aws:organizations::aws:policy/service_control_policy/*"
      ]
    }
  ]
}
```

Example: Tag and untag policies

This example shows how you might create a resource-based delegation policy that allows delegated administrators to tag or untag backup policies. It grants the permissions necessary to complete actions programmatically from the AWS API or AWS CLI.

To use this delegation policy, replace the AWS placeholder text for *MemberAccountId*, *ManagementAccountId*, and *OrganizationId* with your own information. Then, follow the directions in Delegated administrator for AWS Organizations.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DelegatingNecessaryDescribeListActions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::MemberAccountId:root"
      },
      "Action": [
        "organizations:DescribeOrganization",
        "organizations:DescribeOrganizationalUnit",
        "organizations:DescribeAccount",
        "organizations:DescribePolicy",
        "organizations:DescribeEffectivePolicy",
        "organizations:ListRoots",
        "organizations:ListOrganizationalUnitsForParent",
        "organizations:ListParents",
        "organizations:ListChildren",
        "organizations:ListAccounts",
        "organizations:ListAccountsForParent",
```

```
        "organizations:ListPolicies",
        "organizations:ListPoliciesForTarget",
        "organizations:ListTargetsForPolicy",
        "organizations:ListTagsForResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringLikeIfExists": {
          "organizations:PolicyType": "BACKUP_POLICY"
        }
      }
    },
    {
      "Sid": "DelegatingTaggingBackupPolicies",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::MemberAccountId:root"
      },
      "Action": [
        "organizations:TagResource",
        "organizations:UntagResource"
      ],
      "Resource": "arn:aws:organizations::ManagementAccountId:policy/o-OrganizationId/
backup_policy/*"
    }
  ]
}
```

Example: Attach policies to a single OU or account

This example shows how you might create a resource-based delegation policy that allows delegated administrators to `attach` or `detach` Organizations policies from a specified organizational unit (OU) or a specified account. Before delegating these actions, you must delegate the permissions to navigate the structure of an organization and see the accounts under it. For details, see Example: View organization, OUs, accounts, and policies

Important

- While this policy allows attaching or detaching policies from the specified OU or account, it excludes child OUs and accounts under child OUs.

- This policy allows delegated administrators to perform the specified actions on policies created by any account in the organization, including the management account.

This example delegation policy grants the permissions necessary to complete actions programmatically from the AWS API or AWS CLI. To use this delegation policy, replace the AWS placeholder text for *MemberAccountId*, *ManagementAccountId*, *OrganizationId*, and *TargetAccountId* with your own information. Then, follow the directions in Delegated administrator for AWS Organizations.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DelegatingNecessaryDescribeListActions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::MemberAccountId:root"
      },
      "Action": [
        "organizations:DescribeOrganization",
        "organizations:DescribeOrganizationalUnit",
        "organizations:DescribeAccount",
        "organizations:DescribePolicy",
        "organizations:DescribeEffectivePolicy",
        "organizations:ListRoots",
        "organizations:ListOrganizationalUnitsForParent",
        "organizations:ListParents",
        "organizations:ListChildren",
        "organizations:ListAccounts",
        "organizations:ListAccountsForParent",
        "organizations:ListPolicies",
        "organizations:ListPoliciesForTarget",
        "organizations:ListTargetsForPolicy",
        "organizations:ListTagsForResource"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AttachDetachPoliciesSpecifiedAccountOU",
      "Effect": "Allow",
      "Principal": {
```

```
        "AWS": "arn:aws:iam::MemberAccountId:root"
      },
      "Action": [
        "organizations:AttachPolicy",
        "organizations:DetachPolicy"
      ],
      "Resource": [
        "arn:aws:organizations::ManagementAccountId:ou/o-OrganizationId/ou-OUId",
        "arn:aws:organizations::ManagementAccountId:account/
o-OrganizationId/TargetAccountId",
        "arn:aws:organizations::ManagementAccountId:policy/o-OrganizationId/
backup_policy/*"
      ]
    }
  ]
}
```

To delegate attaching and detaching policies to any OU or account in the organizations, replace the resource in the previous example with the following resources:

```
"Resource": [
    "arn:aws:organizations::ManagementAccountId:ou/o-OrganizationId/*",
    "arn:aws:organizations::ManagementAccountId:account/o-OrganizationId/*",
    "arn:aws:organizations::ManagementAccountId:policy/o-OrganizationId/backup_policy/
*"
]
```

Example: Consolidated permissions to manage an organization's backup policies

This example shows how you might create a resource-based delegation policy that allows the management account to delegate full permissions necessary to manage backup policies within the organization, including `create`, `read`, `update`, and `delete` actions, as well as `attach` and `detach` policy actions.

> **Important**
>
> This policy allows delegated administrators to perform the specified actions on policies created by any account in the organization, including the management account.

This example delegation policy grants the permissions necessary to complete actions programmatically from the AWS API or AWS CLI. To use this delegation policy, replace the AWS placeholder text for *MemberAccountId*, *ManagementAccountId*, *OrganizationId*, and *RootId* with your own information. Then, follow the directions in Delegated administrator for AWS Organizations.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DelegatingNecessaryDescribeListActions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::MemberAccountId:root"
      },
      "Action": [
            "organizations:DescribeOrganization",
            "organizations:DescribeOrganizationalUnit",
            "organizations:DescribeAccount",
            "organizations:ListRoots",
            "organizations:ListOrganizationalUnitsForParent",
            "organizations:ListParents",
            "organizations:ListChildren",
            "organizations:ListAccounts",
            "organizations:ListAccountsForParent",
            "organizations:ListTagsForResource"
        ],
      "Resource": "*"
    },
    {
      "Sid": "DelegatingNecessaryDescribeListActionsForSpecificPolicyType",
      "Effect": "Allow",
      "Principal": {
            "AWS": "arn:aws:iam::MemberAccountId:root"
      },
      "Action": [
            "organizations:DescribePolicy",
            "organizations:DescribeEffectivePolicy",
            "organizations:ListPolicies",
            "organizations:ListPoliciesForTarget",
            "organizations:ListTargetsForPolicy"
      ],
      "Resource": "*",
```

```
      "Condition": {
            "StringLikeIfExists": {
                "organizations:PolicyType": "BACKUP_POLICY"
            }
      }
    },
    {
      "Sid": "DelegatingAllActionsForBackupPolicies",
      "Effect": "Allow",
      "Principal": {
      "AWS": "arn:aws:iam::MemberAccountId:root"
    },
      "Action": [
            "organizations:CreatePolicy",
            "organizations:UpdatePolicy",
            "organizations:DeletePolicy",
            "organizations:AttachPolicy",
            "organizations:DetachPolicy",
            "organizations:EnablePolicyType",
            "organizations:DisablePolicyType"
      ],
      "Resource": [
            "arn:aws:organizations::ManagementAccountId:root/o-OrganizationId/
r-RootId",
            "arn:aws:organizations::ManagementAccountId:ou/o-OrganizationId/*",
            "arn:aws:organizations::ManagementAccountId:account/o-OrganizationId/*",
            "arn:aws:organizations::ManagementAccountId:policy/o-OrganizationId/
backup_policy/*"
      ],
      "Condition": {
            "StringLikeIfExists": {
                "organizations:PolicyType": "BACKUP_POLICY"
            }
      }
    }
  ]
}
```

AWS managed policies for AWS Organizations

This section identifies the AWS-managed policies provided for your use to manage your organization. You can't modify or delete an AWS managed policy, but you can attach or detach them to entities in your organization as needed.

AWS Organizations managed policies for use with AWS Identity and Access Management (IAM)

An IAM managed policy is provided and maintained by AWS. A managed policy provides permissions for common tasks that you can assign to your users by attaching the managed policy to the appropriate IAM user or role object. You don't have to write the policy yourself, and when AWS updates the policy as appropriate to support new services, you automatically and immediately get the benefit of the update.

You can see the list of AWS managed policies in Policies page on the IAM console. Use the **Filter policies** drop-down to select **AWS managed**.

You can use the following managed policies to grant permissions to users in your organization.

AWS managed policy: AWSOrganizationsFullAccess

Provides all of the permissions required to create and fully administer an organization.

View the policy: `AWSOrganizationsFullAccess`.

AWS managed policy: AWSOrganizationsReadOnlyAccess

Provides read only access to information about the organization. It doesn't permit the user to make any changes.

View the policy: `AWSOrganizationsReadOnlyAccess`.

AWS managed policy: DeclarativePoliciesEC2Report

This policy is used by the AWSServiceRoleForDeclarativePoliciesEC2Report service-linked role to enable it to describe account attribute states for member accounts.

View the policy: DeclarativePoliciesEC2Report.

Updates to Organizations AWS managed policies

The following table details updates to AWS managed policies since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the Document History page.

Change	Description	Date
AWSOrganizationsFullAccess – updated to allow account API permissions required to view or modify an account name via the Organizations console.	Added the `account:GetAccountInformation` action to enable access to view the account name of any account in an organization and the `account:PutAccountName` action to enable access to modify any account name in an organization.	April 22, 2025
DeclarativePoliciesEC2Report – New managed policy	Added the `DeclarativePoliciesEC2Report` policy to enable the functionality of the `AWSServiceRoleForDeclarativePoliciesEC2Report` service-linked role.	November 22, 2024
AWSOrganizationsReadOnlyAccess – updated to allow account API permissions required to view a root user email address.	Added the `account:GetPrimaryEmail` action to enable access to view the root user email address for any member account in an organization and the `account:GetRegionOptStatus` action to enable access to view the enabled Regions for any member account in an organization.	June 6, 2024
AWSOrganizationsFullAccess – updated to include `Sid` elements that describe the policy statement.	Added `Sid` elements for the `AWSOrganizationsFullAccess` managed policy.	February 6, 2024
AWSOrganizationsReadOnlyAccess – updated to include `Sid` elements that describe the policy statement.	Added `Sid` elements for the `AWSOrganizationsReadOnlyAccess` managed policy.	February 6, 2024

Change	Description	Date
AWSOrganizationsFullAccess – updated to allow account API permissions required to enable or disable AWS Regions via the Organizations console.	Added the `account:ListRegions` , `account:EnableRegion` and `account:DisableRegion` action to the policy to enable write access to enable or disable Regions for an account.	December 22, 2022
AWSOrganizationsReadOnlyAccess – updated to allow account API permissions required to list AWS Regions via the Organizations console.	Added the `account:ListRegions` action to the policy to enable access to view Regions for an account.	December 22, 2022
AWSOrganizationsFullAccess – updated to allow account API permissions required to add or edit account contacts via the Organizations console.	Added the `account:GetContactInformation` and `account:PutContactInformation` action to the policy to enable write access to modify contacts for an account.	October 21, 2022
AWSOrganizationsReadOnlyAccess – updated to allow account API permissions required to view account contacts via the Organizations console.	Added the `account:GetContactInformation` action to the policy to enable access to view contacts for an account.	October 21, 2022
AWSOrganizationsFullAccess – updated to allow creating an organization.	Added the `CreateServiceLinkedRole` permission to the policy to enable creating the service linked role required to create an organization. The permission is restricted to creating a role that can be used only by the `organizations.amazonaws.com` service.	August 24, 2022

Change	Description	Date
AWSOrganizationsFullAccess – updated to allow account API permissions required to add, edit, or delete account alternate contacts via the Organizations console.	Added the `account:GetAlternateContact` , `account:DeleteAlternateContact` , `account:PutAlternateContact` actions to the policy to enable write access to modify alternate contacts for an account.	February 7, 2022
AWSOrganizationsReadOnlyAccess – updated to allow account API permissions required to view account alternate contacts via the Organizations console.	Added the `account:GetAlternateContact` action to the policy to enable access to view alternate contacts for an account.	February 7, 2022

AWS managed authorization policies

Authorization policies are similar to IAM permission policies, but are a feature of AWS Organizations rather than IAM. You use authorization policies to centrally configure and manage access for principals and resources in your member accounts.

You can see the list of policies in your organization on the Policies page on the Organizations console.

Policy name	Description	ARN
FullAWSAccess	Allows access to every operation.	arn:aws:organizations::aws:policy/service_control_policy/p-FullAWSAccess
RCPFullAWSAccess	Allows access to every resource.	arn:aws:organizations::aws:policy/resource_control_policy/p-RCPFullAWSAccess

Attribute-based access control with tags for AWS Organizations

Attribute-based access control let you use administrator-managed attributes such as tags attached to both AWS resources and AWS identities to control access to those resources. For example, you can specify that a user can access a resource when both the user and the resource have the same value for a certain tag.

AWS Organizations taggable resources include AWS accounts, the organization's root, organizational units (OUs), or policies. When you attach tags to Organizations resources, you can then use those tags to control who can access those resources. You do this by adding `Condition` elements to your AWS Identity and Access Management (IAM) permissions policy statements that check whether certain tag keys and values are present before allowing the action. This enables you to create an IAM policy that effectively says "Allow the user to manage only those OUs that have a tag with a key X and a value Y" or "Allow the user to manage only those OUs that are tagged with a key Z that has the same value as the user's attached tag key Z."

You can base your `Condition` tests on different types of tag references in an IAM policy.

- Checking the tags that are attached to resources specified in the request
- Checking the tags that are attached to the IAM user or role who is making the request
- Check the tags that are included as parameters in the request

For more information about using tags for access control in policies, see Controlling access to and for IAM users and roles using resource tags. For complete syntax of IAM permission policies, see the IAM JSON Policy Reference

Checking the tags that are attached to resources specified in the request

When you make a request by using the AWS Management Console, the AWS Command Line Interface (AWS CLI), or one of the AWS SDKs, you specify what resources you want to access with that request. Whether you are trying to list available resources of a given type, read a resource, or write to, modify, or update a resource, you specify the resource to access as a parameter in the request. Such requests are controlled by IAM permissions policies that you attach to your users and roles. In these policies, you can compare the tags attached to the requested resource and choose to allow or deny access based on the keys and values of those tags.

To check a tag that is attached to the resource, you reference the tag in a `Condition` element by prefacing the tag key name with the following string: `aws:ResourceTag/`

For example, the following sample policy allows the user or role to perform any AWS Organizations operation ***unless*** that resource has a tag with the key `department` and the value `security`. If that key and value is present, then the policy explicitly denies the `UntagResource` operation.

```
{
    "Version" : "2012-10-17",
    "Statement" : [
        {
            "Effect" : "Allow",
            "Action" : "organizations:*",
            "Resource" : "*"

        },
        {
            "Effect" : "Deny",
            "Action" : "organizations:UntagResource",
            "Resource" : "*",
            "Condition" : {
                "StringEquals" : {
                    "aws:ResourceTag/department" : "security"
                }
            }
        }
    ]
}
```

For more information about how to use this element, see Controlling access to resource and aws:ResourceTag in the *IAM User Guide*.

Checking the tags that are attached to the IAM user or role who is making the request

You can control what the person making the request (the principal) is allowed to do based on the tags that are attached to that person's IAM user or role. To do this, use the `aws:PrincipalTag/`*`key-name`* condition key to specify which tag and value must be attached to the calling user or role.

The following example shows how to allow an action only when the specified tag (`cost-center`) has the same value on both the principal calling the operation, and the resource being accessed by the operation. In this example, the calling user can start and stop an Amazon EC2 instance only if the instance is tagged with the same `cost-center` value as the user.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "ec2:startInstances",
            "ec2:stopInstances"
        ],
        "Resource": "*",
        "Condition": {"StringEquals":
            {"ec2:ResourceTag/cost-center": "${aws:PrincipalTag/cost-center}"}}
    }
}
```

For more information about how to use this element, see Controlling access for IAM principals and `aws:PrincipalTag` in the *IAM User Guide*.

Check the tags that are included as parameters in the request

Several operations enable you to specify tags as part of the request. For example, when you create a resource you can specify the tags that are attached to the new resource. You can specify a `Condition` element that uses `aws:TagKeys` to allow or deny the operation based on whether a specific tag key, or a set of keys, is included in the request. This comparison operator doesn't care what value the tag contains. It only checks whether a tag with the specified key is present.

To check the tag key, or a list of keys, specify a `Condition` element with the following syntax:

```
"aws:TagKeys": [ "tag-key-1", "tag-key-2", ... , "tag-key-n" ]
```

You can use `ForAllValues:` to preface the comparison operator to ensure that all of the keys in the request must match one of the keys specified in the policy. For example, the following sample policy allows any Organizations operation only if all tags present in the request are a***subset of the three*** tags in this policy.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "organizations:*",
        "Resource": "*",
```

```
        "Condition": {
            "ForAllValues:StringEquals": {
                "aws:TagKeys": [
                    "department",
                    "costcenter",
                    "manager"
                ]
            }
        }
    }
}
```

Alternatively, you can use `ForAnyValue:` to preface a comparison operator to ensure that at least one of the keys in the request must match one of the keys specified in the policy. For example, the following policy allows an Organizations operation only if ***at least one*** of the specified tag keys is present in the request.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "organizations:*",
        "Resource": "*",
        "Condition": {
            "ForAnyValue:StringEquals": {
                "aws:TagKeys": [
                    "stage",
                    "region",
                    "domain"
                ]
            }
        }
    }
}
```

Several operations enable you to specify tags in the request. For example, when you create a resource you can specify the tags that are attached to the new resource. You can compare a tag key-value pair in the policy with a key-value pair that is included with the request. To do this, reference the tag in a `Condition` element by prefacing the tag key name with the following string: `aws:RequestTag/`*`key-name`* and then specify the tag value that must be present.

For example, the following sample policy denies any request by the user or role to create an AWS account where the request is either missing the `costcenter` tag, or provides that tag with a value other than 1, 2, or 3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "organizations:CreateAccount",
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:RequestTag/costcenter": "true"
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": "organizations:CreateAccount",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringNotEquals": {
                    "aws:RequestTag/costcenter": [
                        "1",
                        "2",
                        "3"
                    ]
                }
            }
        }
    ]
}
```

For more information about how to use these elements, see aws:TagKeys and aws:RequestTag in the *IAM User Guide*.

Troubleshooting AWS Organizations identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Organizations and IAM.

Topics

- I am not authorized to perform an action in Organizations
- I am not authorized to perform iam:PassRole
- I want to allow people outside of my AWS account to access my Organizations resources

I am not authorized to perform an action in Organizations

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `organizations:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 organizations:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `organizations:`*`GetWidget`* action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to Organizations.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Organizations. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my Organizations resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Organizations supports these features, see How AWS Organizations works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM 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 API calls with AWS CloudTrail for AWS Organizations
- Amazon EventBridge and AWS Organizations

Logging API calls with AWS CloudTrail for AWS Organizations

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 user or IAM user credentials
- Whether the request was made with temporary security credentials for an IAM role or a federated user
- Whether the request was made by another AWS service

For more information, see the CloudTrail userIdentity Element.

Understanding AWS Organizations log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

Example log entries: CloseAccount

The following example shows a CloudTrail log entry for a sample `CloseAccount` call that is generated when the API is called and the workflow to close the account starts processing in the background.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAMVNPBQA3EXAMPLE:my-admin-role",
        "arn": "arn:aws:sts::111122223333:assumed-role/my-admin-role/my-session-id",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
```

```
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDAMVNPBQA3EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/my-admin-role",
                "accountId": "111122223333",
                "userName": "my-session-id"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2022-03-18T18:17:06Z"
            }
        }
    },
    "eventTime": "2022-03-18T18:17:06Z",
    "eventSource": "organizations.amazonaws.com",
    "eventName": "CloseAccount",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.168.0.1",
    "userAgent":  "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": {
        "accountId": "555555555555"
    },
    "responseElements": null,
    "requestID": "e28932f8-d5da-4d7a-8238-ef74f3d5c09a",
    "eventID": "19fe4c10-f57e-4cb7-a2bc-6b5c30233592",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

The following example shows a CloudTrail log entry for a `CloseAccountResult` call after the background workflow to close the account successfully completes.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "accountId": "111122223333",
    "invokedBy": "organizations.amazonaws.com"
  },
```

```
  "eventTime": "2022-03-18T18:17:06Z",
  "eventSource": "organizations.amazonaws.com",
  "eventName": "CloseAccountResult",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "organizations.amazonaws.com",
  "userAgent": "organizations.amazonaws.com",
  "requestParameters": null,
  "responseElements": null,
  "eventID": "EXAMPLE8-90ab-cdef-fedc-ba987EXAMPLE",
  "readOnly": false,
  "eventType": "AwsServiceEvent",
  "readOnly": false,
  "eventType": "AwsServiceEvent",
  "managementEvent": true,
  "recipientAccountId": "111122223333",
  "serviceEventDetails": {
    "closeAccountStatus": {
      "accountId": "555555555555",
      "state": "SUCCEEDED",
      "requestedTimestamp": "Mar 18, 2022 6:16:58 PM",
      "completedTimestamp": "Mar 18, 2022 6:16:58 PM"
    }
   },
   "eventCategory": "Management"
}
```

Example log entries: CreateAccount

The following example shows a CloudTrail log entry for a sample `CreateAccount` call that is generated when the API is called and the workflow to create the account starts processing in the background.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAMVNPBQA3EXAMPLE:my-admin-role",
        "arn": "arn:aws:sts::111122223333:assumed-role/my-admin-role/my-session-id",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
```

```
                "principalId": "AIDAMVNPBQA3EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/my-admin-role",
                "accountId": "111122223333",
                "userName": "my-session-id"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-09-16T21:16:45Z"
            }
        }
    },
    "eventTime": "2018-06-21T22:06:27Z",
    "eventSource": "organizations.amazonaws.com",
    "eventName": "CreateAccount",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.168.0.1",
    "userAgent":  "Mozilla/5.0 (Windows NT 10.0; Win64; x64)...",
    "requestParameters": {
        "tags": [],
        "email": "****",
        "accountName": "****"
    },
    "responseElements": {
        "createAccountStatus": {
            "accountName": "****",
            "state": "IN_PROGRESS",
            "id": "car-examplecreateaccountrequestid111",
            "requestedTimestamp": "Sep 16, 2020 9:20:50 PM"
        }
    },
    "requestID": "EXAMPLE8-90ab-cdef-fedc-ba987EXAMPLE",
    "eventID": "EXAMPLE8-90ab-cdef-fedc-ba987EXAMPLE",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111111111111"
}
```

The following example shows a CloudTrail log entry for a `CreateAccount` call after the background workflow to create the account successfully completes.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
```

```
    "accountId": "111122223333",
    "invokedBy": "..."
  },
  "eventTime": "2020-09-16T21:20:53Z",
  "eventSource": "organizations.amazonaws.com",
  "eventName": "CreateAccountResult",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "....",
  "requestParameters": null,
  "responseElements": null,
  "eventID": "EXAMPLE8-90ab-cdef-fedc-ba987EXAMPLE",
  "readOnly": false,
  "eventType": "AwsServiceEvent",
  "recipientAccountId": "111122223333",
  "serviceEventDetails": {
    "createAccountStatus": {
      "id": "car-examplecreateaccountrequestid111",
      "state": "SUCCEEDED",
      "accountName": "****",
      "accountId": "444455556666",
      "requestedTimestamp": "Sep 16, 2020 9:20:50 PM",
      "completedTimestamp": "Sep 16, 2020 9:20:53 PM"
    }
  }
}
```

The following example shows a CloudTrail log entry that is generated after a `CreateAccount` background workflow fails to create the account.

```
  {
  "eventVersion": "1.06",
  "userIdentity": {
    "accountId": "111122223333",
    "invokedBy": "AWS Internal"
  },
  "eventTime": "2018-06-21T22:06:27Z",
  "eventSource": "organizations.amazonaws.com",
  "eventName": "CreateAccountResult",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "AWS Internal",
  "userAgent": "AWS Internal",
```

```
  "requestParameters": null,
  "responseElements": null,
  "eventID": "EXAMPLE8-90ab-cdef-fedc-ba987EXAMPLE",
  "readOnly": false,
  "eventType": "AwsServiceEvent",
  "recipientAccountId": "111122223333",
  "serviceEventDetails": {
    "createAccountStatus": {
      "id": "car-examplecreateaccountrequestid111",
      "state": "FAILED",
      "accountName": "****",
      "failureReason": "EMAIL_ALREADY_EXISTS",
      "requestedTimestamp": Jun 21, 2018 10:06:27 PM,
      "completedTimestamp": Jun 21, 2018 10:07:15 PM
    }
  }
}
```

Example log entry: CreateOrganizationalUnit

The following example shows a CloudTrail log entry for a sample `CreateOrganizationalUnit` call.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAMVNPBQA3EXAMPLE",
        "arn": "arn:aws:iam::111111111111:user/diego",
        "accountId": "111111111111",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "diego"
    },
    "eventTime": "2017-01-18T21:40:11Z",
    "eventSource": "organizations.amazonaws.com",
    "eventName": "CreateOrganizationalUnit",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/55.0.2883.95 Safari/537.36",
    "requestParameters": {
        "name": "OU-Developers-1",
        "parentId": "r-a1b2"
```

```
    },
    "responseElements": {
        "organizationalUnit": {
            "arn": "arn:aws:organizations::111111111111:ou/o-aa111bb222/ou-
examplerootid111-exampleouid111",
            "id": "ou-examplerootid111-exampleouid111",
            "name": "test-cloud-trail"
        }
    },
    "requestID": "EXAMPLE8-90ab-cdef-fedc-ba987EXAMPLE",
    "eventID": "EXAMPLE8-90ab-cdef-fedc-ba987EXAMPLE",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111111111111"
}
```

Example log entry: InviteAccountToOrganization

The following example shows a CloudTrail log entry for a sample `InviteAccountToOrganization` call.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAMVNPBQA3EXAMPLE",
        "arn": "arn:aws:iam::111111111111:user/diego",
        "accountId": "111111111111",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "diego"
    },
    "eventTime": "2017-01-18T21:41:17Z",
    "eventSource": "organizations.amazonaws.com",
    "eventName": "InviteAccountToOrganization",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/55.0.2883.95 Safari/537.36",
    "requestParameters": {
        "notes": "This is a request for Mary's account to join Diego's organization.",
        "target": {
            "type": "ACCOUNT",
            "id": "111111111111"
        }
```

```
    },
    "responseElements": {
        "handshake": {
            "requestedTimestamp": "Jan 18, 2017 9:41:16 PM",
            "state": "OPEN",
            "arn": "arn:aws:organizations::111111111111:handshake/o-aa111bb222/invite/
h-examplehandshakeid111",
            "id": "h-examplehandshakeid111",
            "parties": [
                {
                    "type": "ORGANIZATION",
                    "id": "o-aa111bb222"
                },
                {
                    "type": "ACCOUNT",
                    "id": "222222222222"
                }
            ],
            "action": "invite",
            "expirationTimestamp": "Feb 2, 2017 9:41:16 PM",
            "resources": [
                {
                    "resources": [
                        {
                            "type": "MASTER_EMAIL",
                            "value": "diego@example.com"
                        },
                        {
                            "type": "MASTER_NAME",
                            "value": "Management account for organization"
                        },
                        {
                            "type": "ORGANIZATION_FEATURE_SET",
                            "value": "ALL"
                        }
                    ],
                    "type": "ORGANIZATION",
                    "value": "o-aa111bb222"
                },
                {
                    "type": "ACCOUNT",
                    "value": "222222222222"
                },
                {
```

```
                    "type": "NOTES",
                    "value": "This is a request for Mary's account to join Diego's
 organization."
                }
            ]
        }
    },
    "requestID": "EXAMPLE8-90ab-cdef-fedc-ba987EXAMPLE",
    "eventID": "EXAMPLE8-90ab-cdef-fedc-ba987EXAMPLE",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111111111111"
}
```

Example log entry: AttachPolicy

The following example shows a CloudTrail log entry for a sample `AttachPolicy` call. The response indicates that the call failed because the requested policy type isn't enabled in the root where the request to attach was attempted.

```
{
    "eventVersion": "1.06",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAMVNPBQA3EXAMPLE",
        "arn": "arn:aws:iam::111111111111:user/diego",
        "accountId": "111111111111",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "diego"
    },
    "eventTime": "2017-01-18T21:42:44Z",
    "eventSource": "organizations.amazonaws.com",
    "eventName": "AttachPolicy",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/55.0.2883.95 Safari/537.36",
    "errorCode": "PolicyTypeNotEnabledException",
    "errorMessage": "The given policy type ServiceControlPolicy is not enabled on the
 current view",
    "requestParameters": {
        "policyId": "p-examplepolicyid111",
        "targetId": "ou-examplerootid111-exampleouid111"
    },
```

```
    "responseElements": null,
    "requestID": "EXAMPLE8-90ab-cdef-fedc-ba987EXAMPLE",
    "eventID": "EXAMPLE8-90ab-cdef-fedc-ba987EXAMPLE",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111111111111"
}
```

Amazon EventBridge and AWS Organizations

AWS Organizations can work with Amazon EventBridge, formerly Amazon 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 EventBridge 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 EventBridge to monitor key activity in your organization, see Tutorial: Monitor important changes to your organization with Amazon EventBridge.

> **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.

To learn more about EventBridge, including how to configure and enable it, see the *Amazon EventBridge User Guide*.

Compliance validation for AWS Organizations

To learn whether an AWS service is within the scope of specific compliance programs, see AWS services in Scope by Compliance Program and choose the compliance program that you are interested in. For general information, see AWS Compliance Programs.

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

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

- Security Compliance & Governance – These solution implementation guides discuss architectural considerations and provide steps for deploying security and compliance features.
- HIPAA Eligible Services Reference – Lists HIPAA eligible services. Not all AWS services are HIPAA eligible.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Customer Compliance Guides – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see Security Hub controls reference.
- Amazon GuardDuty – This AWS service detects potential threats to your AWS accounts, workloads, containers, and data by monitoring your environment for suspicious and malicious activities. GuardDuty can help you address various compliance requirements, like PCI DSS, by meeting intrusion detection requirements mandated by certain compliance frameworks.
- AWS Audit Manager – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

Resilience in AWS Organizations

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability

Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

Infrastructure security in AWS Organizations

As a managed service, AWS Organizations is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access Organizations through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). 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.

If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

Troubleshooting AWS Organizations

If you encounter issues when working with AWS Organizations, consult the topics in this section.

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
- I get an "access denied" message when I make a request with temporary security credentials
- I get an "access denied" message when I try to leave an organization as a member account or remove a member account as the management account
- I get a "quota exceeded" message when I try to add an account to my organization
- I get a "this operation requires a wait period" message while adding or removing accounts
- I get an "organization is still initializing" message when I try to add an account to my organization
- I get an "Invitations are disabled" message when I try to invite an account to my organization.
- Changes that I make aren't always immediately visible

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 user, group, or role. 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 a 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 user or role that you are using to make the request has the correct permissions. Permissions for temporary security credentials are derived from an user or role, so the permissions are limited to those granted to the user or role. For more information about how permissions for temporary security credentials are determined, see Controlling Permissions for Temporary Security Credentials in the *IAM User Guide*.
- Verify that your requests are being signed correctly and that the request is well formed. For details, see the toolkit documentation for your chosen SDK or Using Temporary Security Credentials to Request Access to AWS Resources in the *IAM User Guide*.
- Verify that your temporary security credentials haven't expired. For more information, see Requesting Temporary Security Credentials in the *IAM User Guide*.

I get an "access denied" message when I try to leave an organization as a member account or remove a member account as the management account

- You can remove a member account only after you enable IAM user access to billing in the member account. For more information, see Activating Access to the Billing and Cost Management Console in the *AWS Billing User Guide*.
- You can remove an account from your organization only if the account has the information required for it to operate as a standalone account. When you create an account in an organization using the AWS Organizations console, API, or AWS CLI commands, that information isn't automatically collected. For an account that you want to make standalone, you must accept the AWS Customer Agreement, choose a support plan, provide and verify the required contact information, and provide a current payment method. AWS uses the payment method to charge for any billable (not AWS Free Tier) AWS activity that occurs while the account isn't attached to an organization. For more information, see Leaving an organization from a member account with AWS Organizations.

I get a "quota exceeded" message when I try to add an account to my organization

There is a maximum number of accounts that you can have in an organization. Deleted or closed accounts continue to count against this quota.

An invitation to join counts against the maximum number of accounts in your organization. The count is returned if the invited account declines, the management account cancels the invitation, or the invitation expires.

- Before you close or delete an AWS account, remove it from your organization so that it doesn't continue to count against your quota.
- See Maximum and minimum values for information about how to request a quota increase.

I get a "this operation requires a wait period" message while adding or removing accounts

Some actions require a wait period due to account quotas. For example, you can't immediately remove newly created accounts. Try the action again in a few days.

For issues with adding accounts, see the quota Default maximum number of accounts. For issues with removing accounts, see the quota Number of accounts you can close within a 30-day period.

I get an "organization is still initializing" message when I try to add an account to my organization

If you receive this error and it's been over an hour since you created the organization, contact AWS Support.

I get an "Invitations are disabled" message when I try to invite an account to my organization.

This happens when you enable all features in your organization. This operation can take some time and requires that all member accounts respond. Until the operation is completed, you can't invite new accounts to join the organization.

Changes that I make aren't always immediately visible

As a service that is accessed through computers in data centers around the world, AWS Organizations uses a distributed computing model called eventual consistency. Any change that you make in AWS Organizations takes time to become visible from all possible endpoints. Some of the delay results from the time it takes to send the data from server to server or from replication zone to replication zone. AWS Organizations also uses caching to improve performance, but in some cases this can add time. The change might not be visible until the previously cached data times out.

Design your global applications to account for these potential delays and ensure that they work as expected, even when a change made in one location isn't instantly visible at another.

For more information about how some other AWS services are affected by this, consult the following resources:

- Managing Data Consistency in the *Amazon Redshift Database Developer Guide*
- Amazon S3 Data Consistency Model in the *Amazon Simple Storage Service User Guide*
- Ensuring Consistency When Using Amazon S3 and Amazon Elastic MapReduce for ETL Workflows in the AWS Big Data Blog
- EC2 Eventual Consistency in the *Amazon EC2 API Reference*.

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
- HTTPS required
- Signing AWS Organizations API requests

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 Regional 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 account root user credentials for everyday work with AWS Organizations. You can use the credentials for a user or role.

To sign your API requests, you must use AWS Signature Version 4. For information about using Signature Version 4, see Signing AWS API requests in the *IAM User Guide*.

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.
- Security best practices in IAM – Offers suggestions for using the IAM service to help secure your AWS resources, including those in AWS Organizations.
- Temporary security credentials in IAM – Describes how to create and use temporary security credentials.

Code examples for Organizations using AWS SDKs

The following code examples show how to use Organizations with an AWS software development kit (SDK).

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Organizations with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Code examples

- Basic examples for Organizations using AWS SDKs
  - Actions for Organizations using AWS SDKs
    - Use AttachPolicy with an AWS SDK or CLI
    - Use CreateAccount with an AWS SDK or CLI
    - Use CreateOrganization with an AWS SDK or CLI
    - Use CreateOrganizationalUnit with an AWS SDK or CLI
    - Use CreatePolicy with an AWS SDK or CLI
    - Use DeleteOrganization with an AWS SDK or CLI
    - Use DeleteOrganizationalUnit with an AWS SDK or CLI
    - Use DeletePolicy with an AWS SDK or CLI
    - Use DescribePolicy with an AWS SDK or CLI
    - Use DetachPolicy with an AWS SDK or CLI
    - Use ListAccounts with an AWS SDK or CLI
    - Use ListOrganizationalUnitsForParent with an AWS SDK or CLI
    - Use ListPolicies with an AWS SDK or CLI

Basic examples for Organizations using AWS SDKs

The following code examples show how to use the basics of AWS Organizations with AWS SDKs.

Examples

- Actions for Organizations using AWS SDKs
  - Use AttachPolicy with an AWS SDK or CLI
  - Use CreateAccount with an AWS SDK or CLI
  - Use CreateOrganization with an AWS SDK or CLI
  - Use CreateOrganizationalUnit with an AWS SDK or CLI
  - Use CreatePolicy with an AWS SDK or CLI
  - Use DeleteOrganization with an AWS SDK or CLI
  - Use DeleteOrganizationalUnit with an AWS SDK or CLI
  - Use DeletePolicy with an AWS SDK or CLI
  - Use DescribePolicy with an AWS SDK or CLI
  - Use DetachPolicy with an AWS SDK or CLI
  - Use ListAccounts with an AWS SDK or CLI
  - Use ListOrganizationalUnitsForParent with an AWS SDK or CLI
  - Use ListPolicies with an AWS SDK or CLI

Actions for Organizations using AWS SDKs

The following code examples demonstrate how to perform individual Organizations actions with AWS SDKs. Each example includes a link to GitHub, where you can find instructions for setting up and running the code.

The following examples include only the most commonly used actions. For a complete list, see the AWS Organizations API Reference.

Examples

- Use AttachPolicy with an AWS SDK or CLI
- Use CreateAccount with an AWS SDK or CLI
- Use CreateOrganization with an AWS SDK or CLI
- Use CreateOrganizationalUnit with an AWS SDK or CLI
- Use CreatePolicy with an AWS SDK or CLI
- Use DeleteOrganization with an AWS SDK or CLI
- Use DeleteOrganizationalUnit with an AWS SDK or CLI
- Use DeletePolicy with an AWS SDK or CLI

- Use DescribePolicy with an AWS SDK or CLI
- Use DetachPolicy with an AWS SDK or CLI
- Use ListAccounts with an AWS SDK or CLI
- Use ListOrganizationalUnitsForParent with an AWS SDK or CLI
- Use ListPolicies with an AWS SDK or CLI

Use `AttachPolicy` with an AWS SDK or CLI

The following code examples show how to use `AttachPolicy`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to attach an AWS Organizations policy to an organization,
/// an organizational unit, or an account.
/// </summary>
public class AttachPolicy
{
    /// <summary>
    /// Initializes the Organizations client object and then calls the
    /// AttachPolicyAsync method to attach the policy to the root
    /// organization.
    /// </summary>
    public static async Task Main()
    {
        IAmazonOrganizations client = new AmazonOrganizationsClient();
```

```
            var policyId = "p-00000000";
            var targetId = "r-0000";

            var request = new AttachPolicyRequest
            {
                PolicyId = policyId,
                TargetId = targetId,
            };

            var response = await client.AttachPolicyAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully attached Policy ID {policyId} to
 Target ID: {targetId}.");
            }
            else
            {
                Console.WriteLine("Was not successful in attaching the policy.");
            }
        }
    }
```

- For API details, see AttachPolicy in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

To attach a policy to a root, OU, or account

Example 1

The following example shows how to attach a service control policy (SCP) to an OU:

```
aws organizations attach-policy
                --policy-id p-examplepolicyid111
                --target-id ou-examplerootid111-exampleouid111
```

Example 2

The following example shows how to attach a service control policy directly to an account:

```
aws organizations attach-policy
                --policy-id p-examplepolicyid111
                --target-id 333333333333
```

- For API details, see AttachPolicy in *AWS CLI Command Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
def attach_policy(policy_id, target_id, orgs_client):
    """
    Attaches a policy to a target. The target is an organization root, account,
 or
    organizational unit.

    :param policy_id: The ID of the policy to attach.
    :param target_id: The ID of the resources to attach the policy to.
    :param orgs_client: The Boto3 Organizations client.
    """
    try:
        orgs_client.attach_policy(PolicyId=policy_id, TargetId=target_id)
        logger.info("Attached policy %s to target %s.", policy_id, target_id)
    except ClientError:
        logger.exception(
            "Couldn't attach policy %s to target %s.", policy_id, target_id
        )
        raise
```

- For API details, see AttachPolicy in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Organizations with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `CreateAccount` with an AWS SDK or CLI

The following code examples show how to use `CreateAccount`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Creates a new AWS Organizations account.
/// </summary>
public class CreateAccount
{
    /// <summary>
    /// Initializes an Organizations client object and uses it to create
    /// the new account with the name specified in accountName.
    /// </summary>
    public static async Task Main()
    {
        IAmazonOrganizations client = new AmazonOrganizationsClient();
        var accountName = "ExampleAccount";
        var email = "someone@example.com";

        var request = new CreateAccountRequest
        {
            AccountName = accountName,
            Email = email,
```

```
            };

            var response = await client.CreateAccountAsync(request);
            var status = response.CreateAccountStatus;

            Console.WriteLine($"The staus of {status.AccountName} is
 {status.State}.");
        }
    }
```

- For API details, see CreateAccount in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

To create a member account that is automatically part of the organization

The following example shows how to create a member account in an organization. The member account is configured with the name Production Account and the email address of susan@example.com. Organizations automatically creates an IAM role using the default name of OrganizationAccountAccessRole because the roleName parameter is not specified. Also, the setting that allows IAM users or roles with sufficient permissions to access account billing data is set to the default value of ALLOW because the IamUserAccessToBilling parameter is not specified. Organizations automatically sends Susan a "Welcome to AWS" email:

```
aws organizations create-account --email susan@example.com --account-
name "Production Account"
```

The output includes a request object that shows that the status is now IN_PROGRESS:

```
{
        "CreateAccountStatus": {
                "State": "IN_PROGRESS",
                "Id": "car-examplecreateaccountrequestid111"
        }
}
```

You can later query the current status of the request by providing the Id response value to the describe-create-account-status command as the value for the create-account-request-id parameter.

For more information, see Creating an AWS Account in Your Organization in the *AWS Organizations Users Guide*.

- For API details, see CreateAccount in *AWS CLI Command Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Organizations with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `CreateOrganization` with an AWS SDK or CLI

The following code examples show how to use `CreateOrganization`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Creates an organization in AWS Organizations.
/// </summary>
public class CreateOrganization
{
    /// <summary>
    /// Creates an Organizations client object and then uses it to create
    /// a new organization with the default user as the administrator, and
    /// then displays information about the new organization.
```

```
        /// </summary>
        public static async Task Main()
        {
            IAmazonOrganizations client = new AmazonOrganizationsClient();

            var response = await client.CreateOrganizationAsync(new
CreateOrganizationRequest
            {
                FeatureSet = "ALL",
            });

            Organization newOrg = response.Organization;

            Console.WriteLine($"Organization: {newOrg.Id} Main Accoount:
{newOrg.MasterAccountId}");
        }
    }
```

- For API details, see CreateOrganization in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

Example 1: To create a new organization

Bill wants to create an organization using credentials from account 111111111111. The following example shows that the account becomes the master account in the new organization. Because he does not specify a features set, the new organization defaults to all features enabled and service control policies are enabled on the root.

```
aws organizations create-organization
```

The output includes an organization object with details about the new organization:

```
{
        "Organization": {
                "AvailablePolicyTypes": [
                        {
```

```
                                "Status": "ENABLED",
                                "Type": "SERVICE_CONTROL_POLICY"
                        }
                ],
                "MasterAccountId": "111111111111",
                "MasterAccountArn": "arn:aws:organizations::111111111111:account/
o-exampleorgid/111111111111",
                "MasterAccountEmail": "bill@example.com",
                "FeatureSet": "ALL",
                "Id": "o-exampleorgid",
                "Arn": "arn:aws:organizations::111111111111:organization/o-
exampleorgid"
        }
}
```

Example 2: To create a new organization with only consolidated billing features enabled

The following example creates an organization that supports only the consolidated billing features:

```
aws organizations create-organization --feature-set CONSOLIDATED_BILLING
```

The output includes an organization object with details about the new organization:

```
{
        "Organization": {
                "Arn": "arn:aws:organizations::111111111111:organization/o-
exampleorgid",
                "AvailablePolicyTypes": [],
                "Id": "o-exampleorgid",
                "MasterAccountArn": "arn:aws:organizations::111111111111:account/
o-exampleorgid/111111111111",
                "MasterAccountEmail": "bill@example.com",
                "MasterAccountId": "111111111111",
                "FeatureSet": "CONSOLIDATED_BILLING"
        }
}
```

For more information, see Creating an Organization in the *AWS Organizations Users Guide*.

- For API details, see CreateOrganization in *AWS CLI Command Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Organizations with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `CreateOrganizationalUnit` with an AWS SDK or CLI

The following code examples show how to use `CreateOrganizationalUnit`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Creates a new organizational unit in AWS Organizations.
/// </summary>
public class CreateOrganizationalUnit
{
    /// <summary>
    /// Initializes an Organizations client object and then uses it to call
    /// the CreateOrganizationalUnit method. If the call succeeds, it
    /// displays information about the new organizational unit.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var orgUnitName = "ProductDevelopmentUnit";

        var request = new CreateOrganizationalUnitRequest
        {
```

```
                Name = orgUnitName,
                ParentId = "r-0000",
            };

            var response = await client.CreateOrganizationalUnitAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully created organizational unit:
 {orgUnitName}.");
                Console.WriteLine($"Organizational unit {orgUnitName} Details");
                Console.WriteLine($"ARN: {response.OrganizationalUnit.Arn} Id:
 {response.OrganizationalUnit.Id}");
            }
            else
            {
                Console.WriteLine("Could not create new organizational unit.");
            }
        }
    }
```

- For API details, see CreateOrganizationalUnit in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

To create an OU in a root or parent OU

The following example shows how to create an OU that is named AccountingOU:

```
aws organizations create-organizational-unit --parent-id r-examplerootid111 --
name AccountingOU
```

The output includes an organizationalUnit object with details about the new OU:

```
{
        "OrganizationalUnit": {
                "Id": "ou-examplerootid111-exampleouid111",
                "Arn": "arn:aws:organizations::111111111111:ou/o-exampleorgid/ou-
examplerootid111-exampleouid111",
```

```
                "Name": "AccountingOU"
        }
}
```

- For API details, see CreateOrganizationalUnit in *AWS CLI Command Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Organizations with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `CreatePolicy` with an AWS SDK or CLI

The following code examples show how to use `CreatePolicy`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Creates a new AWS Organizations Policy.
/// </summary>
public class CreatePolicy
{
    /// <summary>
    /// Initializes the AWS Organizations client object, uses it to
    /// create a new Organizations Policy, and then displays information
    /// about the newly created Policy.
    /// </summary>
    public static async Task Main()
    {
```

```
            IAmazonOrganizations client = new AmazonOrganizationsClient();
            var policyContent = "{" +
                "   \"Version\": \"2012-10-17\"," +
                " \"Statement\" : [{" +
                    " \"Action\" : [\"s3:*\"]," +
                    " \"Effect\" : \"Allow\"," +
                    " \"Resource\" : \"*\"" +
                "}]" +
            "}";

            try
            {
                var response = await client.CreatePolicyAsync(new
CreatePolicyRequest
                {
                    Content = policyContent,
                    Description = "Enables admins of attached accounts to
delegate all Amazon S3 permissions",
                    Name = "AllowAllS3Actions",
                    Type = "SERVICE_CONTROL_POLICY",
                });

                Policy policy = response.Policy;
                Console.WriteLine($"{policy.PolicySummary.Name} has the following
content: {policy.Content}");
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

- For API details, see CreatePolicy in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

Example 1: To create a policy with a text source file for the JSON policy

The following example shows you how to create an service control policy (SCP) named `AllowAllS3Actions`. The policy contents are taken from a file on the local computer called `policy.json`.

```
aws organizations create-policy --content file://policy.json --
name AllowAllS3Actions, --type SERVICE_CONTROL_POLICY --description "Allows
 delegation of all S3 actions"
```

The output includes a policy object with details about the new policy:

```
{
        "Policy": {
                "Content": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect
\":\"Allow\",\"Action\":[\"s3:*\"],\"Resource\":[\"*\"]}]}",
                "PolicySummary": {
                        "Arn": "arn:aws:organizations::o-exampleorgid:policy/
service_control_policy/p-examplepolicyid111",
                        "Description": "Allows delegation of all S3 actions",
                        "Name": "AllowAllS3Actions",
                        "Type":"SERVICE_CONTROL_POLICY"
                }
        }
}
```

Example 2: To create a policy with a JSON policy as a parameter

The following example shows you how to create the same SCP, this time by embedding the policy contents as a JSON string in the parameter. The string must be escaped with backslashes before the double quotes to ensure that they are treated as literals in the parameter, which itself is surrounded by double quotes:

```
aws organizations create-policy --content "{\"Version\":\"2012-10-17\",
\"Statement\":[{\"Effect\":\"Allow\",\"Action\":[\"s3:*\"],\"Resource
\":[\"*\"]}]}" --name AllowAllS3Actions --type SERVICE_CONTROL_POLICY --
description "Allows delegation of all S3 actions"
```

For more information about creating and using policies in your organization, see Managing Organization Policies in the *AWS Organizations User Guide*.

- For API details, see CreatePolicy in *AWS CLI Command Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
def create_policy(name, description, content, policy_type, orgs_client):
    """
    Creates a policy.

    :param name: The name of the policy.
    :param description: The description of the policy.
    :param content: The policy content as a dict. This is converted to JSON
 before
                    it is sent to AWS. The specific format depends on the policy
 type.
    :param policy_type: The type of the policy.
    :param orgs_client: The Boto3 Organizations client.
    :return: The newly created policy.
    """
    try:
        response = orgs_client.create_policy(
            Name=name,
            Description=description,
            Content=json.dumps(content),
            Type=policy_type,
        )
        policy = response["Policy"]
        logger.info("Created policy %s.", name)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy
```

- For API details, see CreatePolicy in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Organizations with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DeleteOrganization` with an AWS SDK or CLI

The following code examples show how to use `DeleteOrganization`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to delete an existing organization using the AWS
/// Organizations Service.
/// </summary>
public class DeleteOrganization
{
    /// <summary>
    /// Initializes the Organizations client and then calls
    /// DeleteOrganizationAsync to delete the organization.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();
```

```
            var response = await client.DeleteOrganizationAsync(new
DeleteOrganizationRequest());

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine("Successfully deleted organization.");
            }
            else
            {
                Console.WriteLine("Could not delete organization.");
            }
        }
    }
```

- For API details, see DeleteOrganization in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

To delete an organization

The following example shows how to delete an organization. To perform this operation, you must be an admin of the master account in the organization. The example assumes that you previously removed all the member accounts, OUs, and policies from the organization:

```
aws organizations delete-organization
```

- For API details, see DeleteOrganization in *AWS CLI Command Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Organizations with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DeleteOrganizationalUnit` with an AWS SDK or CLI

The following code examples show how to use `DeleteOrganizationalUnit`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to delete an existing AWS Organizations organizational unit.
/// </summary>
public class DeleteOrganizationalUnit
{
    /// <summary>
    /// Initializes the Organizations client object and calls
    /// DeleteOrganizationalUnitAsync to delete the organizational unit
    /// with the selected ID.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var orgUnitId = "ou-0000-00000000";

        var request = new DeleteOrganizationalUnitRequest
        {
            OrganizationalUnitId = orgUnitId,
        };

        var response = await client.DeleteOrganizationalUnitAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
```

```
                Console.WriteLine($"Successfully deleted the organizational unit
with ID: {orgUnitId}.");
            }
            else
            {
                Console.WriteLine($"Could not delete the organizational unit with
ID: {orgUnitId}.");
            }
        }
    }
```

- For API details, see DeleteOrganizationalUnit in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

To delete an OU

The following example shows how to delete an OU. The example assumes that you previously removed all accounts and other OUs from the OU:

```
aws organizations delete-organizational-unit --organizational-unit-id ou-
examplerootid111-exampleouid111
```

- For API details, see DeleteOrganizationalUnit in *AWS CLI Command Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Organizations with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DeletePolicy` with an AWS SDK or CLI

The following code examples show how to use `DeletePolicy`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Deletes an existing AWS Organizations policy.
/// </summary>
public class DeletePolicy
{
    /// <summary>
    /// Initializes the Organizations client object and then uses it to
    /// delete the policy with the specified policyId.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var policyId = "p-00000000";

        var request = new DeletePolicyRequest
        {
            PolicyId = policyId,
        };

        var response = await client.DeletePolicyAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully deleted Policy: {policyId}.");
        }
```

```
            else
            {
                Console.WriteLine($"Could not delete Policy: {policyId}.");
            }
        }
    }
```

- For API details, see DeletePolicy in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

To delete a policy

The following example shows how to delete a policy from an organization. The example assumes that you previously detached the policy from all entities:

```
aws organizations delete-policy --policy-id p-examplepolicyid111
```

- For API details, see DeletePolicy in *AWS CLI Command Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
def delete_policy(policy_id, orgs_client):
    """
    Deletes a policy.

    :param policy_id: The ID of the policy to delete.
    :param orgs_client: The Boto3 Organizations client.
```

```
    """
    try:
        orgs_client.delete_policy(PolicyId=policy_id)
        logger.info("Deleted policy %s.", policy_id)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_id)
        raise
```

- For API details, see DeletePolicy in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Organizations with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribePolicy` with an AWS SDK or CLI

The following code examples show how to use `DescribePolicy`.

CLI

AWS CLI

To get information about a policy

The following example shows how to request information about a policy:

```
aws organizations describe-policy --policy-id p-examplepolicyid111
```

The output includes a policy object that contains details about the policy:

```
{
        "Policy": {
                "Content": "{\n  \"Version\": \"2012-10-17\",\n  \"Statement
\": [\n    {\n      \"Effect\": \"Allow\",\n      \"Action\": \"*\",\n
 \"Resource\": \"*\"\n    }\n  ]\n}",
                "PolicySummary": {
                        "Arn": "arn:aws:organizations::111111111111:policy/o-
exampleorgid/service_control_policy/p-examplepolicyid111",
                        "Type": "SERVICE_CONTROL_POLICY",
```

```
                        "Id": "p-examplepolicyid111",
                        "AwsManaged": false,
                        "Name": "AllowAllS3Actions",
                        "Description": "Enables admins to delegate S3
 permissions"
                }
        }
}
```

- For API details, see DescribePolicy in *AWS CLI Command Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
def describe_policy(policy_id, orgs_client):
    """
    Describes a policy.

    :param policy_id: The ID of the policy to describe.
    :param orgs_client: The Boto3 Organizations client.
    :return: The description of the policy.
    """
    try:
        response = orgs_client.describe_policy(PolicyId=policy_id)
        policy = response["Policy"]
        logger.info("Got policy %s.", policy_id)
    except ClientError:
        logger.exception("Couldn't get policy %s.", policy_id)
        raise
    else:
        return policy
```

- For API details, see DescribePolicy in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Organizations with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DetachPolicy` with an AWS SDK or CLI

The following code examples show how to use `DetachPolicy`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to detach a policy from an AWS Organizations organization,
/// organizational unit, or account.
/// </summary>
public class DetachPolicy
{
    /// <summary>
    /// Initializes the Organizations client object and uses it to call
    /// DetachPolicyAsync to detach the policy.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var policyId = "p-00000000";
```

```
            var targetId = "r-0000";

            var request = new DetachPolicyRequest
            {
                PolicyId = policyId,
                TargetId = targetId,
            };

            var response = await client.DetachPolicyAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully detached policy with Policy Id:
{policyId}.");
            }
            else
            {
                Console.WriteLine("Could not detach the policy.");
            }
        }
    }
```

- For API details, see DetachPolicy in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

To detach a policy from a root, OU, or account

The following example shows how to detach a policy from an OU:

```
aws organizations  detach-policy  --target-id ou-examplerootid111-exampleouid111
 --policy-id p-examplepolicyid111
```

- For API details, see DetachPolicy in *AWS CLI Command Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
def detach_policy(policy_id, target_id, orgs_client):
    """
    Detaches a policy from a target.

    :param policy_id: The ID of the policy to detach.
    :param target_id: The ID of the resource where the policy is currently
 attached.
    :param orgs_client: The Boto3 Organizations client.
    """
    try:
        orgs_client.detach_policy(PolicyId=policy_id, TargetId=target_id)
        logger.info("Detached policy %s from target %s.", policy_id, target_id)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from target %s.", policy_id, target_id
        )
        raise
```

- For API details, see DetachPolicy in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Organizations with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListAccounts` with an AWS SDK or CLI

The following code examples show how to use `ListAccounts`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Uses the AWS Organizations service to list the accounts associated
/// with the default account.
/// </summary>
public class ListAccounts
{
    /// <summary>
    /// Creates the Organizations client and then calls its
    /// ListAccountsAsync method.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var request = new ListAccountsRequest
        {
            MaxResults = 5,
        };

        var response = new ListAccountsResponse();
        try
        {
            do
            {
                response = await client.ListAccountsAsync(request);
                response.Accounts.ForEach(a => DisplayAccounts(a));
```

```
                    if (response.NextToken is not null)
                    {
                        request.NextToken = response.NextToken;
                    }
                }
                while (response.NextToken is not null);
            }
            catch (AWSOrganizationsNotInUseException ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        /// <summary>
        /// Displays information about an Organizations account.
        /// </summary>
        /// <param name="account">An Organizations account for which to display
        /// information on the console.</param>
        private static void DisplayAccounts(Account account)
        {
            string accountInfo = $"{account.Id}
{account.Name}\t{account.Status}";

            Console.WriteLine(accountInfo);
        }
    }
```

- For API details, see ListAccounts in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

To retrieve a list of all of the accounts in an organization

The following example shows you how to request a list of the accounts in an organization:

```
aws organizations list-accounts
```

The output includes a list of account summary objects.

```
{
        "Accounts": [
                {
                        "Arn": "arn:aws:organizations::111111111111:account/o-
exampleorgid/111111111111",
                        "JoinedMethod": "INVITED",
                        "JoinedTimestamp": 1481830215.45,
                        "Id": "111111111111",
                        "Name": "Master Account",
                        "Email": "bill@example.com",
                        "Status": "ACTIVE"
                },
                {
                        "Arn": "arn:aws:organizations::111111111111:account/o-
exampleorgid/222222222222",
                        "JoinedMethod": "INVITED",
                        "JoinedTimestamp": 1481835741.044,
                        "Id": "222222222222",
                        "Name": "Production Account",
                        "Email": "alice@example.com",
                        "Status": "ACTIVE"
                },
                {
                        "Arn": "arn:aws:organizations::111111111111:account/o-
exampleorgid/333333333333",
                        "JoinedMethod": "INVITED",
                        "JoinedTimestamp": 1481835795.536,
                        "Id": "333333333333",
                        "Name": "Development Account",
                        "Email": "juan@example.com",
                        "Status": "ACTIVE"
                },
                {
                        "Arn": "arn:aws:organizations::111111111111:account/o-
exampleorgid/444444444444",
                        "JoinedMethod": "INVITED",
                        "JoinedTimestamp": 1481835812.143,
                        "Id": "444444444444",
                        "Name": "Test Account",
                        "Email": "anika@example.com",
                        "Status": "ACTIVE"
                }
        ]
```

```
}
```

- For API details, see ListAccounts in *AWS CLI Command Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Organizations with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListOrganizationalUnitsForParent` with an AWS SDK or CLI

The following code examples show how to use `ListOrganizationalUnitsForParent`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Lists the AWS Organizations organizational units that belong to an
/// organization.
/// </summary>
public class ListOrganizationalUnitsForParent
{
    /// <summary>
    /// Initializes the Organizations client object and then uses it to
    /// call the ListOrganizationalUnitsForParentAsync method to retrieve
    /// the list of organizational units.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
```

```
            IAmazonOrganizations client = new AmazonOrganizationsClient();

            var parentId = "r-0000";

            var request = new ListOrganizationalUnitsForParentRequest
            {
                ParentId = parentId,
                MaxResults = 5,
            };

            var response = new ListOrganizationalUnitsForParentResponse();
            try
            {
                do
                {
                    response = await
client.ListOrganizationalUnitsForParentAsync(request);
                    response.OrganizationalUnits.ForEach(u =>
DisplayOrganizationalUnit(u));
                    if (response.NextToken is not null)
                    {
                        request.NextToken = response.NextToken;
                    }
                }
                while (response.NextToken is not null);
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        /// <summary>
        /// Displays information about an Organizations organizational unit.
        /// </summary>
        /// <param name="unit">The OrganizationalUnit for which to display
        /// information.</param>
        public static void DisplayOrganizationalUnit(OrganizationalUnit unit)
        {
            string accountInfo = $"{unit.Id} {unit.Name}\t{unit.Arn}";

            Console.WriteLine(accountInfo);
        }
    }
```

- For API details, see ListOrganizationalUnitsForParent in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

To retrieve a list of the OUs in a parent OU or root

The following example shows you how to get a list of OUs in a specified root:

```
aws organizations list-organizational-units-for-parent --parent-id r-
examplerootid111
```

The output shows that the specified root contains two OUs and shows details of each:

```
{
        "OrganizationalUnits": [
                {
                        "Name": "AccountingDepartment",
                        "Arn": "arn:aws:organizations::o-exampleorgid:ou/r-
examplerootid111/ou-examplerootid111-exampleouid111"
                },
                {
                        "Name": "ProductionDepartment",
                        "Arn": "arn:aws:organizations::o-exampleorgid:ou/r-
examplerootid111/ou-examplerootid111-exampleouid222"
                }
        ]
}
```

- For API details, see ListOrganizationalUnitsForParent in *AWS CLI Command Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Organizations with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListPolicies` with an AWS SDK or CLI

The following code examples show how to use `ListPolicies`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to list the AWS Organizations policies associated with an
/// organization.
/// </summary>
public class ListPolicies
{
    /// <summary>
    /// Initializes an Organizations client object, and then calls its
    /// ListPoliciesAsync method.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        // The value for the Filter parameter is required and must must be
        // one of the following:
        //     AISERVICES_OPT_OUT_POLICY
        //     BACKUP_POLICY
        //     SERVICE_CONTROL_POLICY
        //     TAG_POLICY
        var request = new ListPoliciesRequest
        {
```

```
                Filter = "SERVICE_CONTROL_POLICY",
                MaxResults = 5,
            };

            var response = new ListPoliciesResponse();
            try
            {
                do
                {
                    response = await client.ListPoliciesAsync(request);
                    response.Policies.ForEach(p => DisplayPolicies(p));
                    if (response.NextToken is not null)
                    {
                        request.NextToken = response.NextToken;
                    }
                }
                while (response.NextToken is not null);
            }
            catch (AWSOrganizationsNotInUseException ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        /// <summary>
        /// Displays information about the Organizations policies associated
        /// with an organization.
        /// </summary>
        /// <param name="policy">An Organizations policy summary to display
        /// information on the console.</param>
        private static void DisplayPolicies(PolicySummary policy)
        {
            string policyInfo = $"{policy.Id}
{policy.Name}\t{policy.Description}";

            Console.WriteLine(policyInfo);
        }
    }
```

- For API details, see ListPolicies in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

To retrieve a list of all policies in an organization of a certain type

The following example shows you how to get a list of SCPs, as specified by the filter parameter:

```
aws organizations list-policies --filter SERVICE_CONTROL_POLICY
```

The output includes a list of policies with summary information:

```
{
        "Policies": [
                {
                        "Type": "SERVICE_CONTROL_POLICY",
                        "Name": "AllowAllS3Actions",
                        "AwsManaged": false,
                        "Id": "p-examplepolicyid111",
                        "Arn": "arn:aws:organizations::111111111111:policy/
service_control_policy/p-examplepolicyid111",
                        "Description": "Enables account admins to delegate
 permissions for any S3 actions to users and roles in their accounts."
                },
                {
                        "Type": "SERVICE_CONTROL_POLICY",
                        "Name": "AllowAllEC2Actions",
                        "AwsManaged": false,
                        "Id": "p-examplepolicyid222",
                        "Arn": "arn:aws:organizations::111111111111:policy/
service_control_policy/p-examplepolicyid222",
                        "Description": "Enables account admins to delegate
 permissions for any EC2 actions to users and roles in their accounts."
                },
                {
                        "AwsManaged": true,
                        "Description": "Allows access to every operation",
                        "Type": "SERVICE_CONTROL_POLICY",
                        "Id": "p-FullAWSAccess",
                        "Arn": "arn:aws:organizations::aws:policy/
service_control_policy/p-FullAWSAccess",
                        "Name": "FullAWSAccess"
```

```
                }
        ]
}
```

- For API details, see ListPolicies in *AWS CLI Command Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
def list_policies(policy_filter, orgs_client):
    """
    Lists the policies for the account, limited to the specified filter.

    :param policy_filter: The kind of policies to return.
    :param orgs_client: The Boto3 Organizations client.
    :return: The list of policies found.
    """
    try:
        response = orgs_client.list_policies(Filter=policy_filter)
        policies = response["Policies"]
        logger.info("Found %s %s policies.", len(policies), policy_filter)
    except ClientError:
        logger.exception("Couldn't get %s policies.", policy_filter)
        raise
    else:
        return policies
```

- For API details, see ListPolicies in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Organizations with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Document history for AWS Organizations

The following table describes major documentation updates for AWS Organizations.

- **API version: 2016-11-28**
- **Latest documentation update**: April 22, 2025

Change	Description	Date
Updated the AWSOrganizationsFullAccess managed policy	Added the `account:GetAccountInformation` action to enable access to view the account name of any account in an organization and the `account:PutAccountName` action to enable access to modify any account name in an organization.	April 22, 2025
Organizations integration with AWS User Notifications	You can integrate User Notifications with AWS Organizations to configure and view notifications centrally across accounts in your organization.	January 24, 2025
Organizations integration with AWS Managed Services (AMS) Self-Service Reporting (SSR)	You can integrate AMS SSR with AWS Organizations to enable Aggregated self-service reporting (SSR). This is an AMS feature that allows Advanced and Accelerate customers to view their existing Self-service reports	January 21, 2025

	aggregated at the organizat ion level, cross-account.	
Added declarative policies	You can use declarative policies to centrally declare and enforce desired configura tions for a given AWS service at scale across an organizat ion. Once attached, the configuration is always maintained when the service adds new features or APIs.	December 1, 2024
New AWS managed policy	Added the `Declarati vePoliciesEC2Report` policy to enable the functiona lity of the declarative-polici es-ec2.amazonaws.com service-linked role.	November 22, 2024
Updated Backup policies	AWS Backup policies updated the `selections` policy key to include a `condition s` policy key and added a new `resources` policy key to the schema. With the new schema, you have more flexibility in resource selection for your backup policies.	November 14, 2024

Centrally manage root access for member accounts	You can now manage privileged root user credentials across member accounts in AWS Organizations with centralized root access. Centrally secure the root user credentials of your AWS accounts managed using AWS Organizations to remove and prevent root user credential recovery and access at scale.	November 14, 2024
Added resource control policies (RCPs)	You can use resource control policies (RCPs) to control the maximum available permissions for resources in an organization.	November 13, 2024
Added chat applications policies	You can use chat applications policies to control access to your organization's accounts from chat applications such as Slack and Microsoft Teams.	September 26, 2024
Scenario-driven content updates	The AWS Organizations documentation was updated to be more scenario-driven throughout the entire guide and content was reorganized to improve readability and discovery. If you have feedback on these changes, use the **Provide feedback** button at the bottom of a page.	September 4, 2024

New opt out from all AI services topic	Added documentation about how opt out from all supported AWS AI services.	August 16, 2024
Organizations now supports 10,000 accounts in an organization	You can now manage up to 10,000 member accounts in an organization, doubling the previous limit of 5,000 accounts. If you have a valid requirement and business need, you can request and be approved for a 10,000 account quota without service limit checks from Organizations or other integrated AWS services.	August 14, 2024
New account migration topic	Added documentation about how to migrate an account from one organization to another.	August 1, 2024
Updated Backup policies	AWS Backup policies now support Amazon Elastic Block Store (Amazon EBS) snapshot archives. For updated examples, see Updating a backup policy and Backup policy syntax and examples.	July 9, 2024

Updated the AWSOrganizationsReadOnlyAccess managed policy	Added the `account:GetPrimaryEmail` action to the AWSOrganizationsReadOnlyAccess policy which enables access to view the root user email address for any member account in an organization and added the `account:GetRegionOptStatus` action to enable access to view the enabled Regions for any member account in an organization.	June 6, 2024
New update root user email address topic	Organizations now provides the capability to centrally update the root user email address for any member account in an organization.	June 6, 2024
Updated policy statements	Added new `Sid` elements to the AWS Organizations managed policy statements.	February 6, 2024
New close management account topic	Added links to considerations and detailed steps that walk through how to close a management account.	February 1, 2024
Updated best practices	Added new information to the best practices section to help align with IAM best practices.	June 12, 2023

Updated the AWSOrganizationsFullAccess and AWSOrganizationsReadOnlyAccess managed policies	Both managed policies were updated to enable write or read access to contacts for accounts.	October 21, 2022
Updated the AWSOrganizationsFullAccess managed policy	The managed policy was updated to allow creating an organization by adding the permission required to create the service linked role needed by a new organization.	August 24, 2022
Organizations close account capability from the AWS Organizations console	Principals in the management account can close member accounts from the AWS Organizations console, and protect member accounts from accidental closure by using IAM policies.	March 29, 2022
Updated announcement to update alternate contacts with AWS Organizations console	Organizations now provides ability to update alternate contacts for accounts within your organization using the AWS Organizations console. Announce new capability and points to Account Management Reference for instructions.	February 8, 2022

Organizations managed policy updates - Update to an existing policy	Updated the AWSOrganizationsFullAccess and AWSOrganizationsReadOnlyAccess managed policies to allow account API permissions required to update or view account alternate contacts via the AWS Organizations console.	February 7, 2022
Organizations integration with Amazon DevOps Guru	You can integrate Amazon DevOps Guru with AWS Organizations to monitor application health holistically across all of your organization accounts and gain insights.	January 3, 2022
Organizations integration with Amazon Detective	You can integrate Amazon Detective with AWS Organizations to ensure that your Detective behavior graph provides visibility into the activity for all of your organization accounts.	December 16, 2021
Organizations integration with AWS Config now supports multi-account multi-region data aggregation.	You can use a delegated administrator account to aggregate resource configuration and compliance data from all of the member accounts your organization. For more information, see Multi-account multi-region data aggregationin the *AWS Config Developer Guide*.	June 16, 2021

Organizations integration with AWS Firewall Manager now includes support for a delegated administrator	You can now designate a member account in your organization to be the Firewall Manager administrator for the entire organization. This allows for better separation of permissions from the organization's management account.	April 30, 2021
Organizations backup policies now support continuous backup	You can use the AWS Backup continuous backups feature with your organization's backup policies.	March 10, 2021
Organizations integration with AWS CloudFormation StackSets now includes support for a delegated administrator	You can now designate a member account in your organization to be the AWS CloudFormation StackSets administrator for the entire organization. This allows for better separation of permissions from the organization's management account.	February 18, 2021
Continue inviting accounts while you enable all features	AWS updated the process to enable all features in an organization. You can now continue to invite new accounts to join your organization while you wait for existing accounts to respond to their invitations.	February 3, 2021

Introduces version 2.0 of the AWS Organizations console	AWS introduced a new version of the AWS console. All of the documentation has been updated to reflect the new way of performing tasks.	January 21, 2021
Organizations now supports integration with AWS Marketplace	You can now enable AWS Marketplace to more easily share your software licenses across all of the accounts in your organization.	December 3, 2020
Organizations now supports integration with Amazon S3 Lens	Amazon S3 Lens supports both trusted access and delegated administrator with Organizations. For details, see Amazon S3 Storage Lens in the *Amazon Simple Storage Service User Guide*.	November 18, 2020
Cross-account backup copies	When you use backup policies to backup the resources in your organization, you can now store copies of your backup in other AWS accounts in the organization.	November 18, 2020
AWS Regions in China now support AWS Resource Access Manager as an Organizations trusted service	You can now use AWS RAM features that integrate with Organizations as a trusted service when you use Organizations and AWS RAM in China.	November 18, 2020

Organizations now supports integration with AWS Security Hub	You can enable Security Hub across all of the accounts in your organization, and designate one of your organization's member accounts as the delegated administrator account for Security Hub.	November 12, 2020
Renamed the master account	AWS Organizations changed the name of the “master account” to “manageme nt account”. This is a name change only, and there is no change in functionality.	October 20, 2020
New Best Practices section and topics	Added a new section for best practices for AWS Organizat ions. The new section includes topics that discuss best practices for the management account and member account root users and password management.	October 6, 2020
Added new best practices section and first two pages	There is a new section for topics that describe best practices for AWS Organizat ions. This update includes a topic for best practices for an organization's managemen t account and a topic for best practices for member accounts.	October 2, 2020

Organizations backup policies now support application-consistent backups on Windows EC2 instances by using VSS (Volume Shadow Copy Service)	Backup policies support a new `advanced_backup_settings` " section. The first entry in this new section is an `ec2` setting called `WindowsVSS` that you can enable or disable. For details, see Creating a VSS-Enabled Windows Backup in the *AWS Backup Developer Guide*.	September 24, 2020
Organizations supports tag-on-create and tag-based access control	You can add tags to Organizations resources when you create them. You can use tag policies to standardize tag usage on Organizations resources. You can use IAM policies to restrict access to only resources that have specified tag keys and values.	September 15, 2020
Added AWS Health as a trusted service	You can aggregate AWS Health events across accounts in your organization.	August 4, 2020
Artificial Intelligence (AI) services opt-out policies	You can use AI services opt-out policies to control whether AWS AI services may store and use customer content processed by those services (AI content) for the development and continuous improvement of AWS AI services and technologies.	July 8, 2020

Added backup policies and integration with AWS Backup	You can use backup policies to create and enforce backup policies across all of the accounts in your organization.	June 24, 2020
Support delegated administration for IAM Access Analyzer	Enables you to delegate administrative access for Access Analyzer in your organization to a designated member account.	March 30, 2020
Integration with AWS CloudFormation StackSets	You can create a service-managed stack set to deploy stack instances to accounts managed by AWS Organizations.	February 11, 2020
Integration with Compute Optimizer	Compute Optimizer was added as a service that can work with accounts in your organization.	February 4, 2020
Tag policies	You can use tag policies to help standardize tags across resources in your organization's accounts.	November 26, 2019
Integration with Systems Manager	You can synchronize operations data across all AWS accounts in your organization in Systems Manager Explorer.	November 26, 2019
aws:PrincipalOrgPaths	New global condition key checks the AWS Organizations path for the IAM user, IAM role, or AWS account root user who is making the request.	November 20, 2019

Integration with AWS Config rules	You can use AWS Config API operations to manage AWS Config rules across all AWS accounts in your organization.	July 8, 2019
New service for trusted access	Service Quotas added as a service that can work with the accounts in your organization.	June 24, 2019
Integration with AWS Control Tower	AWS Control Tower added as a service that can work with the accounts in your organizat ion.	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 Service Catalog added as services that can work with the accounts in your organizat ion.	December 21, 2018
New services for trusted access	AWS CloudTrail and AWS RAM added as services that can work with the accounts in your organization.	December 4, 2018
New service for trusted access	AWS Directory Service added as a service that can work with the accounts in your organization.	September 25, 2018
Email address verification	You must verify that you own the email address that is associated with the management account before you can invite existing accounts to your organization.	September 20, 2018
CreateAccount notifications	`CreateAccount` notificat ions are published to the management account's CloudTrail logs.	June 28, 2018
New service for trusted access	AWS Artifact added as a service that can work with the accounts in your organization.	June 20, 2018
New services for trusted access	AWS Config and AWS Firewall Manager added as services that can work with the accounts in your organization.	April 18, 2018

Trusted service access	You can now enable or disable access for select AWS services to work in the accounts in your organization. IAM Identity Center is the initial supported trusted service.	March 29, 2018
Account removal is now self-service	You can now remove accounts that were created from within AWS Organizations without contacting AWS Support.	December 19, 2017
Added support for new service AWS IAM Identity Center	AWS Organizations now supports integration with AWS IAM Identity Center (IAM Identity Center).	December 7, 2017
AWS added a service-linked role to all organization accounts	A service-linked role named `AWSServiceRoleFor0rganizations` 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	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